SEMA SIMAI Springer Series 5

Simona Perotto
Luca Formaggia *Editors*

# New Challenges in Grid Generation and Adaptivity for Scientific Computing

SēMA SIMAI SOCIETÀ ITALIANA DI MATEMATICA APPLICATA E INDUSTRIALE

Springer

# SEMA SIMAI Springer Series

Volume 5

Simona Perotto • Luca Formaggia
Editors

# New Challenges in Grid Generation and Adaptivity for Scientific Computing

Springer

*Editors*

Simona Perotto
MOX Dipartimento di Matematica
Politecnico di Milano
Milano
Italy

Luca Formaggia
MOX Dipartimento di Matematica
Politecnico di Milano
Milano
Italy

ISSN 2199-3041 ISSN 2199-305X (electronic)
SEMA SIMAI Springer Series
ISBN 978-3-319-06052-1 ISBN 978-3-319-06053-8 (eBook)
DOI 10.1007/978-3-319-06053-8

Library of Congress Control Number: 2015937357

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

# Preface

This volume presents selected contributions from the Fourth Tetrahedron Workshop on Grid Generation for Numerical Computation, which was held in Verbania, Italy, in July 2013. The previous editions of this workshop have been hosted by the Weierstrass Institute in Berlin, Germany, in 2005, by INRIA Rocquencourt, France, in 2007, and by Swansea University, United Kingdom, in 2010.

The goal of this book is to present recent developments in mesh generation and adaptation, with emphasis on applications to different fields of interest in science and engineering. Mesh generation is a crucial aspect of numerical simulation of problems governed by partial differential equations. Almost all discretization methods for this large class of problems rely on partitioning the computational domain into a set of elements that form a tessellation of the domain of interest. These elements are used either to define the support of the basis of the approximating space (in finite element or spectral element formulations) or as the basic unit for the setup of the discrete problem (as in a finite volume framework).

As a consequence, their shape and distribution may considerably affect the quality and accuracy of the numerical solution. One of the issues tackled in this book is how to efficiently generate a mesh that ensures a certain bound of the error between the exact solution and a corresponding discretization. This is normally accomplished by resorting to either *a priori* or *a posteriori* bounds relating the discretization error to the element size, shape, and orientation, often through the definition of a solution-dependent metric.

Things are even more difficult when dealing with realistic three-dimensional domains, whose boundary can be extremely complex and non-planar. In such cases, the control of the error induced by the discretization of the physical boundary is also demanded. Thus, the generation of good surface meshes and the control of mesh quality near the domain boundary become crucial tasks.

Another more practical issue is how to generate or adapt the mesh in an automatic and computationally efficient manner. Since mesh generation is often one of the most time-consuming issues in simulations applied to engineering problems, research on this topic is of great importance for real-life applications and deserves methodical investigation.

With the contributions in this book, we cover different, though related, aspects in the field of mesh generation and adaptation: the generation of quality grid for complex three-dimensional geometries, with some contributions on parallel techniques; mesh adaptation, addressing both theoretical and implementation aspects; and mesh generation and adaptation on surface — all with an interesting mix of numerical analysis, computer science, and strongly applicative problems.

It was the intention of the editors and organizers of the workshop to bring together mathematicians, engineers, and industrial researchers. This explains the variety of the contributions, which, in our opinion, gives added value to this work. The book is thus addressed to the numerical analysis and scientific computing community as well as to industrial researchers or software engineers who wish to keep abreast of the state of the art in the field of mesh generation and adaptation.

We wish to acknowledge the institutions and companies that have supported the workshop and thus made possible the production of this monograph: the Comune di Verbania, the Istituto Nazionale di Alta Matematica "F. Severi" (INDAM), the Department of Mathematics of Politecnico di Milano and, in particular, its laboratory for Modeling and Scientific Computing (MOX), Springer Italy, MOXOFF S.r.L., and Beta CAE System. The workshop was held under the auspices of Società Italiana di Matematica Applicata e Industriale (SIMAI).

Special thanks are also due to Laura Guarino and Anna Rho of the EventiMate staff of the Department of Mathematics of Politecnico di Milano for their precious help in organizing the event.

Milan, Italy
September 2014

Simona Perotto
Luca Formaggia

# Contents

**Implicit Boundary and Adaptive Anisotropic Meshing** ..................... 1
Thierry Coupez, Luisa Silva, and Elie Hachem

**A Curvature-Adapted Anisotropic Surface Re-meshing Method** .......... 19
Franco Dassi and Hang Si

**The Benefits of Anisotropic Mesh Adaptation for Brittle Fractures Under Plane-Strain Conditions** .................................. 43
Marco Artina, Massimo Fornasier, Stefano Micheletti, and Simona Perotto

**Deforming Surface Meshes** ............................................. 69
Siu-Wing Cheng and Jiongxin Jin

**An Optimization Based Method for the Construction of 2D Parameterizations for Isogeometric Analysis with T-Splines** .............. 91
José Iván López, Marina Brovka, José María Escobar, José Manuel Cascón, and Rafael Montenegro

**Thread-Parallel Anisotropic Mesh Adaptation** ............................ 113
Gerard J. Gorman, Georgios Rokos, James Southern, and Paul H.J. Kelly

**Immersed NURBS for CFD Applications** .................................. 139
Jeremy Veysset, Ghina Jannoun, Thierry Coupez, and Elie Hachem

**Strategies for Generating Well Centered Tetrahedral Meshes on Industrial Geometries** ................................................ 161
Sean Walton, Oubay Hassan, and Kenneth Morgan

**Enhanced Viscous Mesh Generation with Metric-Based Blending** ........ 181
David Marcum and Frédéric Alauzet

**On the Generation of Curvilinear Meshes Through Subdivision of Isoparametric Elements** ........ 203
David Moxey, Mashy D. Green, Spencer J. Sherwin, and Joaquim Peiró

**Anisotropic, Adaptive Finite Elements for a Thin 3D Plate** ........ 217
Marco Picasso and Adrien Loseille

**Anisotropic Mesh and Time Step Adaptivity for Solute Transport Modeling in Porous Media** ........ 231
Bahman Esfandiar, Giovanni Porta, Simona Perotto, and Alberto Guadagnini

**A 2D Topology-Adaptive Mesh Deformation Framework for Mesh Warping** ........ 261
Jibum Kim, David McLaurin, and Suzanne M. Shontz

**On Shape Deformation Techniques for Simulation-Based Design Optimization** ........ 281
Daniel Sieger, Stefan Menzel, and Mario Botsch

**Creating Free-Surface Flow Grids with Automatic Grid Refinement** ...... 305
Jeroen Wackers, Ganbo Deng, Emmanuel Guilmineau, Alban Leroyer, Patrick Queutey, and Michel Visonneau

# Implicit Boundary and Adaptive Anisotropic Meshing

**Thierry Coupez, Luisa Silva, and Elie Hachem**

**Abstract** Implicit boundary means that the boundaries and/or interfaces between domains are not anymore defined by an explicit boundary mesh but rather by an implicit function. It is the case with embedded boundary methods or immersed boundary methods. Here we consider a filtered level set methods and meshing is then performed using an anisotropic mesh adaptation framework applied to the level sel interpolation. The interpolation error estimate is driving the adaptive process giving rise to a new way of boundary recovery. The accuracy of the recovery process depends then on the user given parameter, an arbitrary thickness of the interface. The thickness is normally related to the mesh size, but it is shown that adaptive meshing enables to reverse this condition: fixing the thickness parameter and accounting for the adaptation process to fulfill the mesh size condition. Several examples are given to demonstrate the potential of this approach.

## 1 Introduction

Numerical simulation is still strongly depending on the meshing capabilities of complex geometry and especially in fmulti-domain, multi-physics, fluid structure interaction and multiphase flow applications. One of the drawback of the common usage in multi-domain meshing is to remain on the constraint of enforcement of the boundary or interface meshes in the volume mesh. This task becomes more and more difficult in computational mechanics when one wants to use a posteriori adaptive meshing or/and dealing with boundary layers.

The alternative proposed in this paper is to consider an implicit representation of the boundary. It means the boundary is not given anymore by a surface mesh or any explicit representation but implicitly by a scalar field which value can be accessed anywhere in the domain of calculation. For instance it can be the distance

T. Coupez (✉) • L. Silva • E. Hachem
Institut du Calcul Intensif, Ecole Centrale de Nantes, 1 rue de la Noë, BP 92101, 44321 Nantes, France
e-mail: thierry.coupez@ec-nantes.fr; luisa.rocha-da-silva@ec-nantes.fr; elie.hachem@mines-paristech.fr

S. Perotto, L. Formaggia (eds.), *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, SEMA SIMAI Springer Series 5,
DOI 10.1007/978-3-319-06053-8_1

to the boundary and therefore the zero-value of this distance function is defining the position of the boundary.

In fact, we can vary the choice of the implicit function. In this paper, we propose to use a composition of the distance function with an hyperbolic tangent. The latter enables to filter in a smooth way the distance function to a narrow band around the targeted boundary. The thickness of the narrow band is central to the paper, giving rise to the proposed theory of implicit boundary as the meshing counterpart of the embedded or immersed volume method.

The second fundamental idea of the proposed approach, is to interpolate the implicit function within meshes and to rely to an adaptive process based on an a posteriori interpolation error framework to improve its representation. Anisotropic meshing is a particularly powerful tool to do this task in an very efficient way. The result is an optimal capture of the interfaces within the mesh, whatever is the complexity of the geometrical configuration. However, surface meshes do not exist anymore, and numerical methods (flow or solid solvers…) need to account for a new paradigm.

Eventually, this approach becomes universal once the flow solver (as well as the structure solver) can afford with anisotropic meshes and implicit boundary representation as it is the case with the immersed volume method.

Immersed methods for Fluid Structure Interaction (FSI) are gaining popularity in many scientific and engineering applications. Different approaches can be found such as the Embedded Boundary method [1], the Immersed Boundary method [2], the fictitious domain [3], the Immersed Volume method [4–6] and the Cartesian method [7]. All these methods are attractive because they simplify a number of issues in Fluid-Structure applications, such as meshing the fluid domain, using a fully Eulerian algorithm, problems involving large structural motion and deformation [8] or topological changes [9].

However, they use non-body fitted grids which require special interface treatments. Indeed, recent developments are focusing on issues related to the immersion of a surface mesh for complex 3D geometries: the detection and the intersection algorithms for the interface and finally the transmission of boundary conditions between the solid and the fluid regions. In particular, these methods appear to be limited by the quality and the accuracy of the surface mesh description of a given immersed solid.

It is claimed in this paper that these methods can be as accurate as classical explicit boundary methods, once the thickness parameter is small enough. Moreover, the thickness parameter can be chosen small a priori in order to meet the accuracy threshold because anisotropic adaptation enables to build the required meshes able to render interfaces which thickness remains as small as desired.

The proposed technique is the present achievement of research started earlier. Our meshing engine is based on local modification and a minimal volume principle first proposed in [10–12] and fully proven latter in [13]. The first idea about meshing geometries with multiple interfaces by using an anisotropic adaptive framework has been proposed in [14]. However, the technique proposed in this work was quite complex because of the VOF like method used to define implicitly the geometries.

Nowadays, we rely on the level set method to represent the various domains, the solid ones as the moving and deformable ones, all together in a single mesh [15, 16]. Combined with anisotropic mesh adaptation, it provides an attractive immersed framework.

The anisotropic techniques behind the proposed work are based on our recent theoretical framework with a metric field constructed directly at the node of the mesh without any direct information from the element, neither considering any underlying interpolation [17]. It introduced the basic idea of length distribution function approximated by a second order tensor constructed by gathering the edges at the node. The error is only calculated along and in the direction of each edge. The mesh construction is controlled by the total number of mesh nodes aimed in the simulation and a robust improvement has been done on that matter in [18], where extension to vector field and multi-components objective can be also found in [19] with a first application to high Reynolds flow and a first clue on possible boundary layers automatic capturing.

The paper is structured as follows. Section 2 presents the details of the immersed technique. Section 3 describes the used error estimator for anisotropic mesh adaptation. In Section 4 several numerical examples are used to highlight the capability of the approach. Finally, conclusions and perspectives are given in Sect. 5.

## 2 Implicit Boundary

### *2.1 Immersed Subdomain and Regularisation*

Let us consider a domain $\omega$ totally included in the larger one, $\Omega$. It will be said that $\omega$ is immersed into $\Omega$. However, it is not a limitation, but it simplifies the following presentation.

More precisely, if $\Gamma = \delta\omega$ is its boundary, immersion means that $\omega \cap \Omega = \omega$ and $\Gamma \cap \Omega = \Gamma$.

From a geometrical view point, $\omega$ is fully defined by the complete knowledge of its boundary. The implicit view point is to represent the domain by an implicit function. For instance, the immediate candidate is the signed distance function:

$$\alpha = \bar{d}(x, \Gamma) = \begin{cases} d(x, \Gamma) \text{ if } x \in \omega \\ -d(x, \Gamma) \text{ if } x \notin \omega \end{cases}. \quad (1)$$

Reciprocally, the interface is completely defined by the zero value of the distance function and thus, to know the domain $\omega$ is equivalent to know the scalar field spanned by the distance function that is defined everywhere in the larger domain $\Omega$.

From $\alpha$, we can calculate the Heaviside function by:

$$H(\alpha) = \begin{cases} 1 \text{ if } \alpha > 0 \\ 0 \text{ if } \alpha < 0 \end{cases}, \tag{2}$$

and thus, $H(\alpha(x)) = 1_\omega(x)$, the characteristic function associated with $\omega$.

The Heaviside function enables to separate strictly the domains, but the counterpart is to reintroduce the discontinuity which has been left by the use of an implicit function. It is why an important ingredient of the proposed approach is to introduce a regularization process. Let us consider the following implicit function:

$$u_\epsilon = u(\alpha, \epsilon) = \epsilon \tanh(\alpha/\epsilon). \tag{3}$$

It enables to introduce a thick interface which thickness is related to the parameter $\epsilon$. Moreover, $u_\epsilon$ gives us a way to define a smooth truncation of the distance function allowing to introduce a narrow band of thickness $\epsilon$ and in which $u_\epsilon$ is approximating the distance function when $\alpha$ is small (i.e.: $\alpha << \epsilon$ and beyond it takes a constant value $\epsilon$).

The smoothed Heaviside function can be introduced as well by:

$$H_\epsilon = \frac{1}{2}(1 + \frac{u_\epsilon}{\epsilon}). \tag{4}$$

In fact, the regularized Heaviside function is about the distance function in the vicinity of the interface. Indeed, we have the following:

$$u'_\epsilon(\alpha) = 1 - \tanh(\frac{\alpha}{\epsilon})^2 = 1 - (\frac{u_\epsilon}{\epsilon})^2 \text{ and } u_\epsilon(\alpha) = u_\epsilon(0) + u'_\epsilon(0)\alpha + O(\alpha^2)$$
$$\text{therefore } u_\epsilon(\alpha) \approx \alpha \text{ when } \alpha << \epsilon. \tag{5}$$

*Remark 1* $\lim_{\epsilon \to 0} H_\epsilon \longrightarrow H$, and it means that, in the case of the true modeling, it is well in the limit of the regularized model, the basic idea is now to design a method in which the regularization parameter can be small enough to attain a satisfying accuracy.

*Remark 2* The zero-value of the distance function and those of the implicit scalar $u_\epsilon$ are strictly the same whatever the value of $\epsilon$ is. Thus, varying the value of $\epsilon$ will not change the background geometrical representation but only its vicinity.

*Remark 3* Knowing the distance function, it easy to build the scalar field $u_\epsilon$. But, in fact, knowing only the zero value (the interface position), it is possible to rebuild the scalar field everywhere by solving an hyperbolic equation. Indeed, because $u'_\epsilon(\alpha) =$

$1-(\frac{u_\epsilon}{\epsilon})^2$, then $u_\epsilon$ is the solution of following equation:

$$\begin{cases} \dfrac{\partial u}{\partial \tau} = s(u)(|\nabla u| - (1-(\dfrac{u}{\epsilon})^2)) \\ u(\tau = 0) = 0 \text{ on } \Gamma \end{cases} \tag{6}$$

where $s(u)$ denotes the sign of $u$.

## 2.2 Interface Thickness

The smoothed Heaviside function is useful for dealing with the multiphase modeling required by the implicit boundary approach. Let us introduce the material scalar field $\eta$ from the material parameters $\eta_1$ and $\eta_2$ occupying two domains sharing an interface $\Gamma$ implicitly defined by the signed distance function $\alpha$.

$$\eta = \eta_1 H(\alpha) + \eta_2(1 - H(\alpha)). \tag{7}$$

This represents a strict discontinuous variation of the global scalar field and it is not tractable by any standard numerical method without care and adjustment. But if we consider the regularized version of this mixture law, we get a smooth transition which is controlled by the thickness parameter, for instance:

$$\eta_\epsilon = \eta_1(1 - u_\epsilon)/2 + \eta_2(1 + u_\epsilon)/2. \tag{8}$$

In fact, any kind of mixture is possible under the condition that $\lim_{\epsilon\to 0}\eta_\epsilon \longrightarrow \eta$. The regularization can be done for all material parameters replaced by material fields varying everywhere in the global domain. We will admit in the sequel that the regularized model gives solutions which are close approximations of the solution of the original discontinuous model.

## 2.3 Smoothness and Interpolation Error

$u_\epsilon$ is a smooth function, at least a continuous function which represents exactly the background geometry since:

$$\Gamma = \{x, u_\epsilon(x) = 0\}, \forall \epsilon \geq 0$$

$\Gamma$ does not depend on $\epsilon$. Thus, $\Gamma$ can be retrieved from $u_\epsilon$ for any $\epsilon$. Let us consider an unstructured mesh of $\Omega$ and a simple $P^1$ Lagrange interpolation $u_\epsilon^h$ of $u_\epsilon$.

$$U^i = u_\epsilon(X^i) = u_\epsilon^h(X^i)$$

where $\{X^i, i \in N\}$ are the vertexes of the triangulation and $u_\epsilon^h|^K \in P^1(K)$, $K$ being any element of the mesh.

The main idea of this paper is to construct the geometrical mesh of the sub-domain $\omega$ as the adapted mesh of its associated interpolated implicit function, $u_\epsilon^h$ by using the interpolation error theory. The boundary is still given by the zero value of the approximated function and the accuracy of the geometrical representation is of the same order of one given by a surface triangulation (it is also an interpolation, the nodes being chosen on the boundary).

Figures 1 and 2 show the obtained function $u_\epsilon^h$ and the adapted meshes for different values of $\epsilon$, the captured interface being the corner depicted by the centered plotted contour. It is clear that $\Gamma$ does not change from case to case. Note that in the

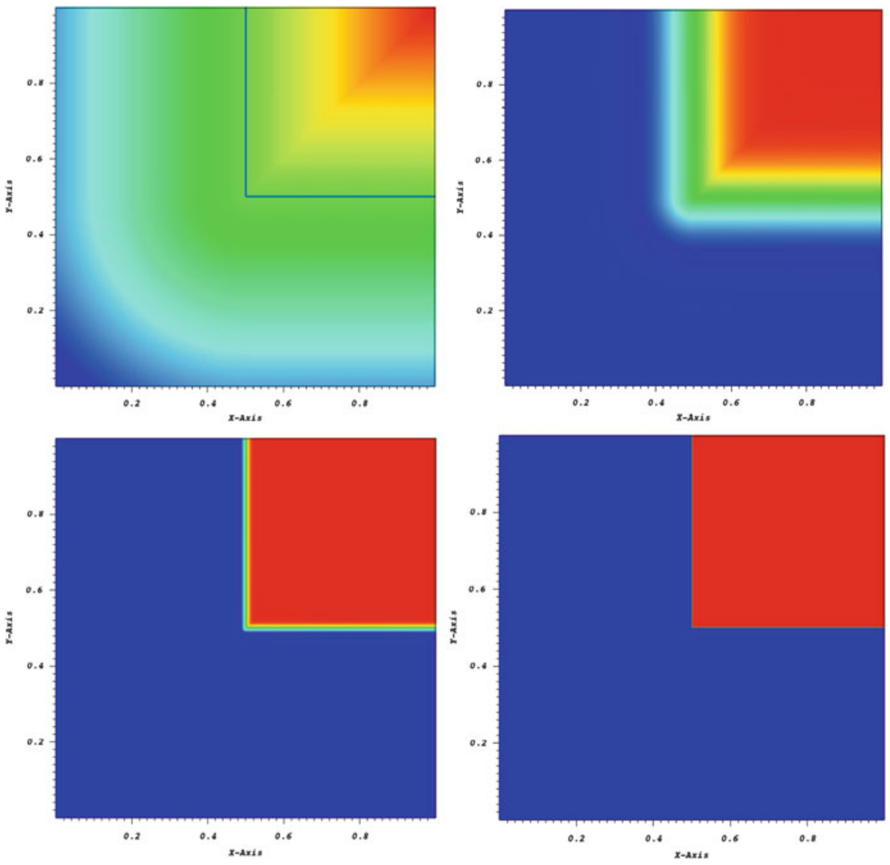


**Fig. 1** *From top left to bottom right*: $u_\epsilon^h$ for different values of the thickness $\epsilon \in \{\infty, 0,01, 0,001, 0,0001\}$

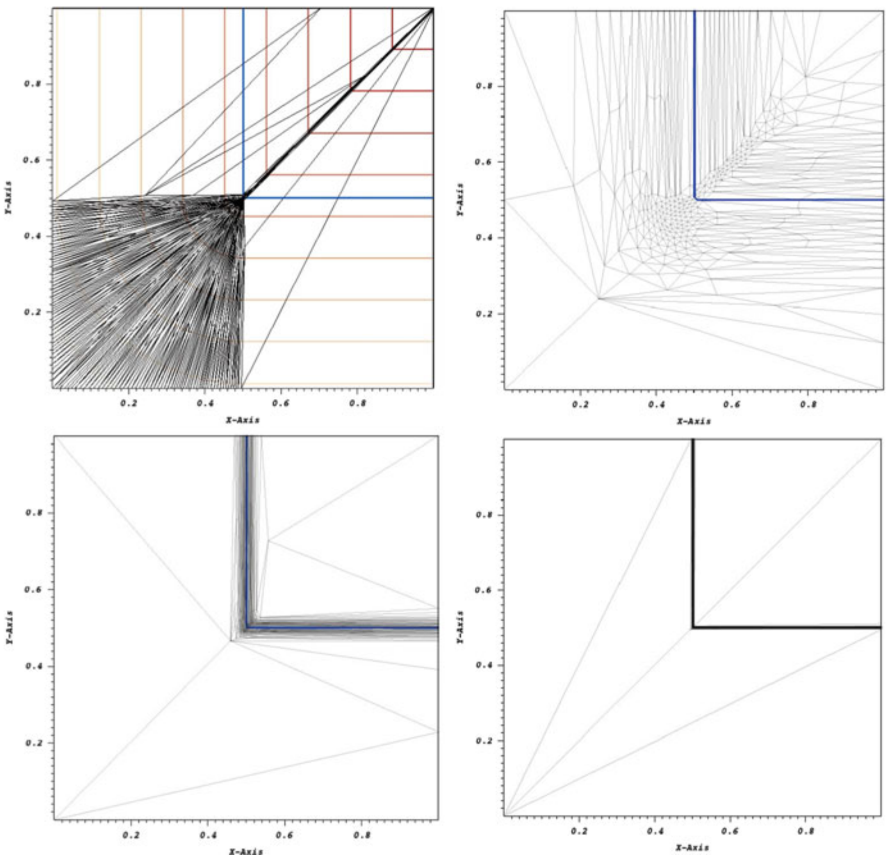


**Fig. 2** The optimal anisotropic meshes at a constant number of nodes for different values of the thickness $\epsilon \in \{\infty, 0, 01, 0, 001, 0, 0001\}$

first sub figure, for ($\epsilon = \infty$), we have the exact level set function and the mesh is totally changing with it. Therefore, the smaller is the thickness parameter, the closer is $u_\epsilon^h$ to the characteristic function of $\omega$. However, the smoothness near the interface is still there but enclosed in a narrow band and becomes non visible.

An important remark appearing clearly from these figures is that such a singularity (the inner corner) is well regularized by the use of a distance function outside the domain, the function being infinitely derivable, but the singularity is propagated inside, since only continuity is preserved. In fact, it depends on the convex, concave local properties of the contour. Consequently, the use of a small thickness parameter enables to constrain the geometrical singularity in such a thin region.

## 3 Anisotropic Mesh Adaptation

In this section, we will see how to make a mesh by using the interpolation error analysis, and thus doing a geometrical mesh by adaptation on the implicit scalar field of the previous section. We recall first, the main features of the anisotropic meshing approach using a posteriori estimates relying on the length distribution tensor approach and the associated edge based error analysis as developed in [17].

### *3.1 Edge Based Error Estimation*

We consider $u \in \mathscr{C}^2(\Omega) = \mathscr{V}$ and $\mathscr{V}_h$ a simple $P^1$ finite element approximation space:

$$\mathscr{V}_h = \left\{ w_h \in \mathscr{C}^0(\Omega), w_h|_K \in P^1(K), K \in \mathscr{K} \right\}$$

where $\Omega = \bigcup_{K \in \mathscr{K}} K$ and $K$ is a simplex (segment, triangle, tetrahedron, …).

We define $\mathbf{X} = \left\{ \mathbf{X}^i \in \mathbb{R}^d,\ i = 1, \cdots, N \right\}$ as the set of node coordinates of the mesh and we denote by $U^i$ the nodal value of $u$ at $\mathbf{X}^i$ and define $u_h$ in unique way by the Lagrange interpolation operator $\Pi_h$ from $\mathscr{V}$ to $\mathscr{V}_h$ as:

$$u_h(\mathbf{X}^i) = (\Pi_h u(\mathbf{X}^i) = u(\mathbf{X}^i) = U^i\ ,\ \forall i = 1, \cdots, N.$$

The set of nodes connected to node $i$ is denoted by

$$\Gamma(i) = \{ j\ ,\ \exists K \in \mathscr{K}\ ,\ i, j \in K \}\,.$$

We introduce the edge vector by: $\mathbf{X}^{ij} = \mathbf{X}^j - \mathbf{X}^i$ and the variation of $u$ within the edge by $U^{ij}$, as plotted on Fig. 3.

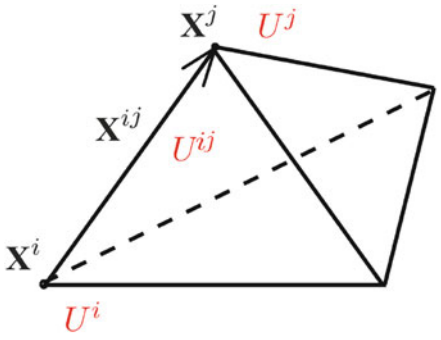


**Fig. 3** Edge vector $\mathbf{X}^{ij}$ joining nodes $i$ to $j$. $U^{ij} = U^j - U^i$, the variation of $u$ over the edge

We first remark that:

$$\nabla u_h \cdot \mathbf{X}^{ij} = U^{ij}\,, \tag{9}$$

and using the analysis carried in [17], we can set the following results:

$$||\underbrace{\nabla u_h \cdot \mathbf{X}^{ij}}_{U^{ij}} - \nabla u(X^i) \cdot \mathbf{X}^{ij}|| \leq \max_{Y \in [X^i, X^j]} |\mathbb{H}(u)(Y)\mathbf{X}^{ij} \cdot \mathbf{X}^{ij}|\,, \tag{10}$$

where $\mathbb{H}(u) = \nabla^{(2)}u$ is the associated Hessian of $u$. Recall that taking $u \in \mathscr{C}^2(\Omega)$, we obtain $\nabla u \in \mathscr{C}^1(\Omega)$.

Applying the interpolation operator on $\nabla u$ and using (9), we obtain a definition of the projected second derivative of $u$ in terms of only the values of the gradient at the extremities of the edge:

$$\nabla g_h \mathbf{X}^{ij} \cdot \mathbf{X}^{ij} = g^{ij} \cdot \mathbf{X}^{ij} \tag{11}$$

where $\nabla g_h = \Pi_h \nabla u$, $g^i = \nabla u(\mathbf{X}^i)$ and $g^{ij} = g^j - g^i$.

Using a mean value argument, we set that:

$$\exists y \in [x^i, x^j] | g^{ij} \cdot \mathbf{X}^{ij} = \mathbb{H}(u)(Y)\mathbf{X}^{ij} \cdot \mathbf{X}^{ij}\,.$$

We use this projection as an expression of the error along the edge:

$$e_{ij} = g^{ij} \cdot \mathbf{X}^{ij}. \tag{12}$$

However, this equation cannot be evaluated exactly as it requires that the gradient of $u$ be known and continuous at the nodes of the mesh. For that reason, we resort to a gradient recovery procedure.

## 3.2 Gradient Recovery

Based on an optimization analysis, the recovery gradient operator proposed in [17] is defined by:

$$G^i = (\mathbb{X}^i)^{-1} \sum_{j \in \Gamma(i)} U^{ij}\mathbf{X}^{ij} \tag{13}$$

where $\mathbb{X}^i = \frac{d}{|\Gamma(i)|} \sum_{j \in \Gamma(i)} \mathbf{X}^{ij} \otimes \mathbf{X}^{ij}$ is the length distribution tensor at node $\mathbf{X}^i$. Note that such a reconstruction preserves the second order:

$$\left|\left(G^i - g^i\right) \cdot \mathbf{X}^{ij}\right| \sim \left(\mathbb{H}(u)\mathbf{X}^{ij} \cdot \mathbf{X}^{ij}\right)$$

where $G^i$ is the recovered gradient at node $i$ [given by (13)] and $g^i$ is the exact value of the gradient at node $i$.

The error is evaluated by substituting $G$ by $g$ in (12):

$$e_{ij} = G^{ij} \cdot \mathbf{X}^{ij}.$$

## 3.3 Metric Construction from the Edge Distribution Tensor

Taking into account this error analysis, we construct the metric for the unit mesh as follows:

$$\mathbb{M}^i = \left( \frac{d}{|\Gamma(i)|} \sum_{j \in \Gamma(i)} \mathbf{X}^{ij} \otimes \mathbf{X}^{ij} \right)^{-1} .$$

For a complete justification of this result, the reader is referred to [17].

## 3.4 Error Behavior Due to Varying the Edge Length

In this section, we introduce a new way to enforce the number of nodes $N$ and we propose a novel approach to compute the stretching factor without using the dimensional parameter $p$ as was proposed in [17]. First, we start by examining how the error behaves when we change the length of the edges by stretching coefficients

$$\mathscr{S} = \left\{ s_{ij} \in \mathbb{R}^+ , \, i = 1, \cdots , N \, , \, j = 1, \cdots , N \, , \, \Gamma(i) \cap \Gamma(j) \neq \phi \right\} .$$

In order to obtain a new metric depending on the error analysis, one has to calculate first a new length for each edge and then use it to rebuild the length distribution tensor. An interesting way of linking the error variations to the changes in edge lengths is by introducing a stretching factor $s \in \mathbb{R}$ such that

$$\begin{cases} \widetilde{\mathbf{X}_{ij}} = s \mathbf{X}_{ij} \\ ||\widetilde{e_{ij}}|| = s^2 ||e_{ij}|| = s^2 ||G^{ij} \cdot \mathbf{X}_{ij}|| \end{cases} \tag{14}$$

where $\widetilde{e_{ij}}$ and $\widetilde{\mathbf{X}_{ij}}$ are the target error at edge $ij$ and its associated edge length.

Following the lines of [17], we can simply define the metric associated with $S$ by:

$$\widetilde{\mathbb{M}^i} = \frac{|\Gamma(i)|}{d} \left( \widetilde{\mathbb{X}^i} \right)^{-1} \tag{15}$$

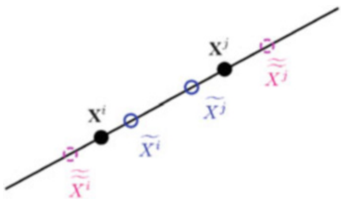


**Fig. 4** Varying the edge in its own direction

where

$$\widetilde{\mathbb{X}^i} = \frac{d}{|\Gamma(i)|} \sum_{j \in \Gamma(i)} s_{ij}^2 \mathbf{X}^{ij} \otimes \mathbf{X}^{ij}$$

is the length distribution tensor. Let $n_{ij}$ be the number of created nodes in relation with the stretching factor $s_{ij}$ and along the edge $ij$. When scaling the edges by a factor $s_{ij}$, the error changes quadratically so that the number of created nodes (number of sub-edges as shown in Fig. 4) along the edge $ij$ is given by:

$$n_{ij} = \left(\frac{\widetilde{e_{ij}}}{e_{ij}}\right)^{\frac{1}{2}} = s_{ij}^{-1}.$$

Here, $\widetilde{e_{ij}}$ denotes the induced error for edge $\widetilde{\mathbf{X}^{ij}}$.

Giving the number of nodes (or sub-edges) created along the current edge, it is possible now to build a tensor of distribution of number of nodes in all directions at node $i$, $\mathbb{N}^i$, by solving the following optimization problem:

$$\min_i \sum_{j \in \Gamma(i)} |\mathbb{N}^i \cdot \mathbf{X}^{ij} - n_{ij} \mathbf{X}^{ij}|^2$$

and from which we can calculate the number of nodes around it by:

$$n^i = \det(\mathbb{N}^i) = \det\left((\mathbb{X}^i)^{-1} \sum_{j \in \Gamma(i)} n_{ij} \mathbf{X}^{ij} \otimes \mathbf{X}^{ij}\right).$$

By considering the averaging process of the number of nodes distribution function, the total number of nodes in the adapted mesh is given by:

$$N = \sum_i n^i.$$

Assuming a uniform totally balanced error along the edge, $\widetilde{e_{ij}} = e$ is constant, we get a direct relation between $N$ and $e$ as follows:

$$n_{ij}(e) = s_{ij}^{-1}(e) = \left(\frac{\widetilde{e_{ij}}}{e_{ij}}\right)^{-\frac{1}{2}}.$$

For a node $i$ we have

$$n^i(e) = \det\left((\mathbb{X}^i)^{-1} \sum_{j \in \Gamma(i)} n_{ij}(e)\right)$$

with

$$n^i(e) = e^{\frac{2}{d}} n^i(1)$$

so that

$$N = e^{\frac{2}{d}} \sum_i n^i(1).$$

Hence, the global induced error for a given total number of nodes $N$ can be determined by:

$$e(N) = \left(\frac{N}{\sum_i n^i(1)}\right)^{\frac{2}{d}}. \tag{16}$$

Therefore, the corresponding stretching factors under the constraint of a fixed number of nodes $N$ are given by:

$$s_{ij} = \left(\frac{\widetilde{e_{ij}}}{e(N)}\right)^{-\frac{1}{2}}.$$

### 3.4.1 Extension to Multi-Component Field

Here, we propose to construct a unique metric directly from a multi-component vector field containing, for instance, all the components of the velocity field and/or different levelset functions of the immersed solids. Consequently, we do not need to intersect several metrics but construct it using the following error vector: $\mathbf{e}_{ij} = \left\{e_{ij}^1, e_{ij}^2, \cdots, e_{ij}^n\right\}$.

Let us introduce $u = \{u_1, u_2, \cdots, u_n\}$,

$$\mathbb{Z} = \mathscr{V} \times \mathscr{V} \times \cdots \times \mathscr{V}$$

and

$$\mathbb{Z}_h = \mathcal{V}_h \times \mathcal{V}_h \times \cdots \times \mathcal{V}_h.$$

In the view of constructing a unique metric, we choose to apply the above theory for each component of $u$. It comes out immediately that the error is now a vector given by the following expression:

$$\overrightarrow{e_{ij}} = \{e^1_{ij}, e^2_{ij}, \cdots, e^n_{ij}\}$$

and then

$$s_{ij} = \left(\frac{||\widetilde{e_{ij}}||}{||\overrightarrow{e_{ij}}||}\right)^{\frac{1}{2}}.$$

Here, the norm can be $L_2$, $L_1$ or $L_\infty$. In the following numerical experiments, we used the $L_2$ case to compute the error.

## 3.5 *Application to the Velocity Field and to the Levelset Function*

Let $v_h(\mathbf{X}^i) = V^i \in \mathbb{R}^d, d = 2, 3$ be the finite element solution of the Navier–Stokes equations. We introduce the vector field $\mathcal{Y} = \left(\frac{v}{|v|}, \frac{|v|}{\max(|v|)}, \frac{u_\epsilon}{\epsilon}\right)$ made of $d+1$ component vector fields. Recall that $\alpha$ is the level set function used to localize an immersed body. We obtain then for every node $i$,

$$\Pi_h \mathcal{Y}(\mathbf{X}^i) = \left\{\frac{V^i}{|V^i|}, \frac{|V^i|}{\max(|V^i|)}, \frac{U^i_\epsilon}{\epsilon}\right\} = \mathcal{Y}^i.$$

Obviously, the case $|v| = 0$ must be accounted for by using $\frac{V^i}{\max(|v^i|, \varepsilon)}$ with $\varepsilon \approx 10^{-6}$ chosen as a small value so that $\mathcal{Y}^i_k = 0$ when $|v^i| = 0$.

Using the vector $\mathcal{Y}^i$, the adaptivity will now take into account, using one unique metric, the variations in the velocity directions, the velocity norm and the levelset functions. Indeed, the adaptivity will focus mainly on the change of direction rather than the intensity of the velocity. Consequently, and as presented by the numerical results in the following section, even the small vortices developed by the solution will be very well captured. What is even more interesting is the capability of the method to automatically detect the boundary layers at the fluid-solid interfaces due to the anisotropically adapted mesh exhibiting highly stretched elements. Finally, we recall that we use a mesh technique (MTC) based on the local modification and the conformity control through the theorem for minimal volume preserving. This was introduced in [20] and extended to anisotropic mesh adaptation in [17, 21].

## 4 Applications

The performance of the implicit boundary method will be assessed using several 2D and 3D examples. First we show that combining the new immersed method with anisotropic mesh adaptation can lead to a novel, efficient and flexible immersed framework able to handle simple and very complex geometries. Then, we combine it with flow solvers based on a stabilized finite element method [22] to simulate complex multiphase flows and fluid structure interaction problems. Moreover, to calculate the interface motion and deformation, we use the convected levelset method proposed earlier in [23].

### *4.1 Flow Past a Cylinder and a Sphere*

First, we revisit two well-known examples using the new implicit boundary method: the flow past a cylinder and a sphere. We immerse the cylinder and the sphere inside a unit fluid domain. To represent the immersed geometries, we compute the distance functions for the circle and the sphere using simple analytical functions. We set the thickness to $\epsilon = 0.0001$ and we use the smoothed Heaviside function given in (2) to mix the physical properties between the solid and the surrounding fluid.

Figure 5 shows the evolution of the implicit representation of the circle. Indeed, by simulating the flow past the circle and applying the anisotropic mesh adaptation, the implicit boundary representation evolves automatically and improves. The mesh is adapted dynamically and the whole process converges to the exact representation of the circle. The smaller the thickness is, the more the regularized interface will be close to the exact representation. Moreover, Fig. 6 shows the flow characteristics past the cylinder and the sphere. All the boundary layers at the interface and the

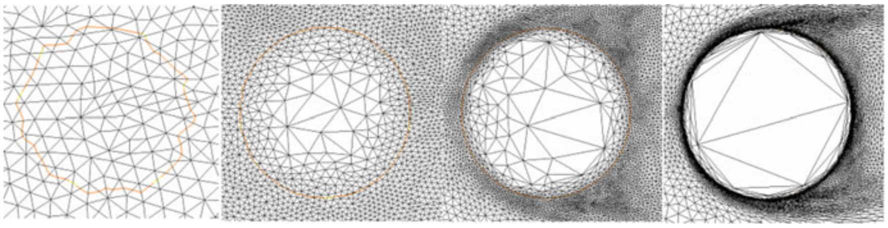


**Fig. 5** The distance function of the immersed circle and the obtained adapted mesh at different time steps

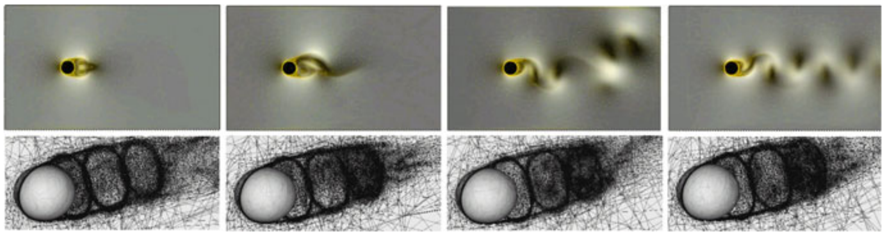


**Fig. 6** Snapshots of the adapted mesh for the flow past a circle (*up*) and a sphere (*bottom*)

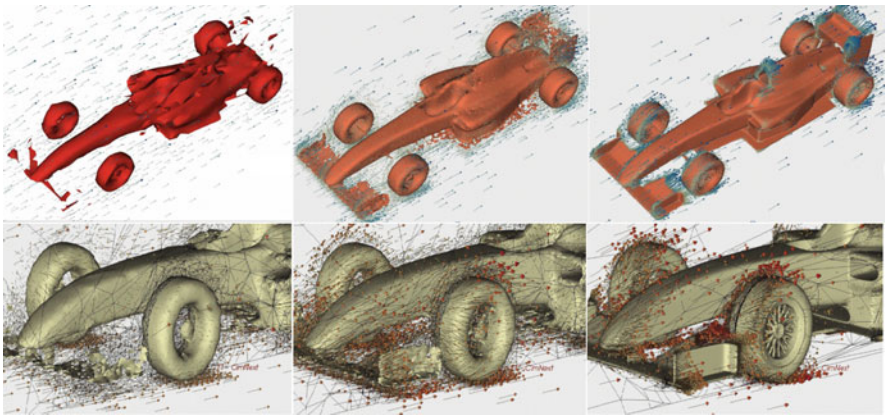


**Fig. 7** Snapshots of the adapted mesh around a F1 car

flow detachments in the wake are captured automatically. This again highlights that combining both the implicit representation with anisotropic mesh adaptation leads to an attractive immersed framework (Fig. 6).

## 4.2 *Flow Past a 3D Formula One Car*

The objective of the second example is to show the performance of the anisotropic meshing on real applications. Indeed, combined with flow solvers it allows to easily and accurately deal with complex fluid structure interaction problems. Therefore, we consider a turbulent flow past an immersed F1 car. This 3D computations have been obtained using 64 2.4 GHz Opteron cores.

Figure 7 shows the evolution of both the implicit representation of the immersed car and the surrounding velocity fields. Note the concentration of the resolution along all the boundary layers. The zoomed snapshots, in particular close to the wheels, highlight the capability of the proposed implicit boundary method. This reflects well the anisotropy of the solution caused by the discontinuity of the boundary conditions and the nature of the flow. Taking a closer look at the mesh near the interfaces, we can detect the good orientation of the elements with the stretching in the right direction. This demonstrates the ability of the algorithm to work under the constraint of a fixed number of nodes and to effectively control the element sizes, orientations and locations.

The elements far from the immersed solid are mostly isotropic and increase in size as the velocity gradient decreases. Again, this reflects and explains why, for a controlled number of nodes, the mesh is naturally and automatically coarsened in that region with the goal of reducing the mesh size around the boundaries and in the wake regions.

## *4.3 3D Multiphase Flow*

In this section, we test the implicit immersed method on a complex multiphase flow problem: the falling drop simulation. The simulation of a droplet of water falling on a flat surface of water is commonly used to demonstrate the capability of a multiphase flow solver. Indeed, handling the true viscosities of water and air and the high ratio of the densities, determine the ability of the implemented Navier–Stokes solver.

Figure 8 shows clearly the splash-back resulting from the water droplet impacting on the flat water surface. A very important feature of this simulation is the physical effect of the surface tension. To take it into account in the proposed framework, it was necessary to introduce a Dirac force depending on the local curvature of the implicit air-water interface. A smooth Dirac function is then given directly by the

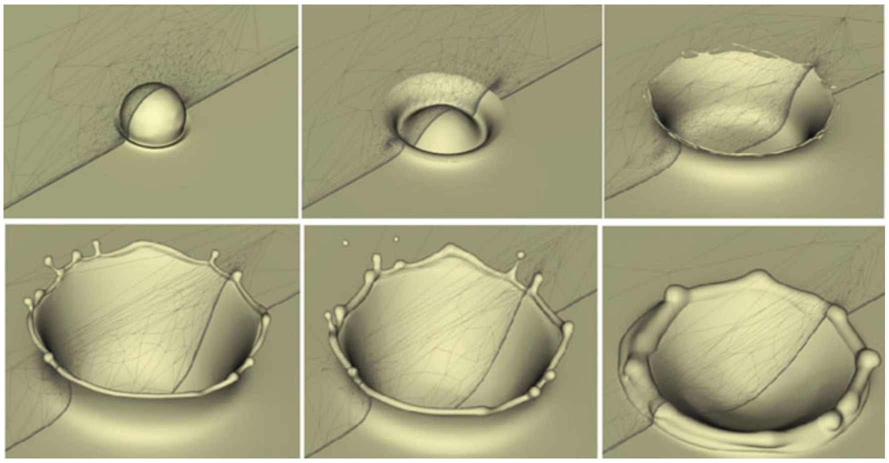


**Fig. 8** The impact of a falling droplet into water

derivative of the regularized Heaviside function given in (4) and the curvature is obtained by using the recovered gradient operator of $u_\epsilon^h$ described in (13).

As expected, the use of anisotropic adapted meshing is important to capture all the pattern formations. A closer look at the interface shows the right orientation and deformation of the mesh elements (longest edges parallel to the boundary). This yields a great reduction of the number of triangles and consequently a reduction in the computational costs. Once again the results prove that the implemented method works well and shows that combining the new immersed method with anisotropic mesh adaptation lead to a very practical tool for immersed methods.

## 5 Conclusions

One of the drawback in automatic meshing and even more in the adaptive meshing is the treatment of the surface boundary in a volume mesh. Explicit boundary meshing is referring to methods for which the boundary must be given as a surface mesh which constrains the volume mesh construction, i.e.: surface faces must be face of volume elements. The alternative which has been proposed in this paper is to consider an implicit representation of the boundary. The boundary is given with the help of an implicit function and a composition of a hyperbolic tangent and the distance function. This implicit function enabled to introduce a scalar field which is defined everywhere in the domain of calculation. Its zero-value defines the position of the boundary. This is a common way to define interfaces in embedded boundary methods or immersed volume ones. Several 2D and 3D examples were given and show that using the interpolation error analysis and the anisotropic meshing machinery based on the edge error analysis and length distribution tensor metrics were able to solve various adaptive meshing multi-domain problems with complex geometrical representations. In such a strategy, both the geometrical representation and the flow pattern are solved in a unique loop of adaptation leading to a very new way of making complex applications without user intervention in the entire process of meshing.

## References

1. Kreiss, H., Petersson, A.: A second-order accurate embedded boundary method for the wave equation with dirichlet data. SIAM J. Sci. Comput. **27**, 1141–1167 (2006)
2. Peskin, C.: Flow patterns around heart valves: a numerical method. J. Comput. Phys. **10**, 252–271 (1972)
3. Glowinski, R., Pan, T., Kearsley, A., Periaux, J.: Numerical simulation and optimal shape for viscous flow by a fictitious domain method. Int. J. Numer. Methods Fluids **20**, 695–711 (2005)
4. Hachem, E., Digonnet, H., Massoni, E., Coupez, T.: Immersed volume method for solving natural convection, conduction and radiation of a hat-shaped disk inside a 3d enclosure. Int. J. Numer. Methods Heat Fluid Flow **22**, 718–741 (2012)

5. Hachem, E., Kloczko, T., Digonnet, H., Coupez, T.: Stabilized finite element solution to handle complex heat and fluid flows in industrial furnace using the immersed volume method. Int. J. Numer. Methods Fluids **68**, 99–121 (2012)
6. Hachem, E., Jannoun, G., Veysset, J., Henri, M., Pierrot, R., Poitrault, I., Massoni, E., Coupez, T.: Modeling of heat transfer and turbulent flows inside industrial furnaces. Simul. Model. Pract. Theory **30**, 35–53 (2013)
7. Johansen, H., Colella, P.: A cartesian grid embedded boundary method for poisson's equation on irregular domains. J. Comput. Phys. **147**, 60–85 (1998)
8. Farhat, C., Rallu, A., Wang, K., Belytschko, T.: Robust and provably second-order explicit-explicit and implicit-explicit staggered time-integratorsfor highly nonlinear fluid-structure interaction problems. Int. J. Numer. Methods Eng. **84**(1), 73–107 (2010)
9. Farhat, C., Maute, K., Argrow, B., Nikbay, M.: A shape optimization methodology for reducing the sonic boom initial pressure rise. AIAA J. Aircr. **45**, 1007–1018 (2007)
10. Coupez, T., Soyris, N., Chenot, J.-L.: 3-d finite element modelling of the forging process with automatic remeshing. J. Mater. Process. Technol. **27**(1–3), 119–133 (1991)
11. Coupez, T., Chenot, J.: Large deformation and automatic remeshing. In: Computational Plasticity (COMPLAS III), pp. 1077–1087. Pineridge Press, Swansea (1992)
12. Coupez, T., Chenot, J.: Mesh topology for mesh generation problems-application to three-dimensional remeshing. In: Numerical Methods in Industrial Forming Processes, pp. 237–242. AA Balkema, Rotterdam (1992)
13. Coupez, T.: Generation de maillage et adaptation de maillage par optimisation locale. Revue Europeenne des elements finis **9**(4), 403–423 (2000)
14. Gruau, C., Coupez, T.: 3D tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric. Comput. Methods Appl. Mech. Eng. **194**, 4951–4976 (2005)
15. Hachem, E., Feghali, S., Codina, R., Coupez, T.: Anisotropic adaptive meshing and monolithic variational multiscale method for fluid-structure interaction. Comput. Struct. **122**, 88–100 (2013)
16. Hachem, E., Feghali, S., Codina, R., Coupez, T.: Immersed stress method for fluid structure interaction. Int. J. Numer. Methods Eng. **94**, 805–825 (2013)
17. Coupez, T.: Metric construction by length distribution tensor and edge based error for anisotropic adaptive meshing. J. Comput. Phys. **230**, 2391–2405 (2011)
18. Coupez, T., Jannoun, G., Nassif, N., Nguyen, H., Digonnet, H., Hachem, E.: Adaptive time-step with anisotropic meshing for incompressible flows. J. Comput. Phys. **241**, 195–211 (2013)
19. Coupez, T., Hachem, E.: Solution of high-Reynolds incompressible flow with stabilized finite element and adaptive anisotropic meshing. Comput. Methods Appl. Mech. Eng. **267**, 65–85 (2013)
20. Coupez, T.: A mesh improvement method for 3D automatic remeshing. In: Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, pp. 615–626. Pineridge Press, Swansea (1994)
21. Gruau, C., Coupez, T.: 3D tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric. Comput. Methods Appl. Mech. Eng. **194**, 4951–4976 (2005)
22. Hachem, E., Rivaux, B., Kloczko, T., Digonnet, H., Coupez, T.: Stabilized finite element method for incompressible flows with high Reynolds number. J. Comput. Phys. **229**, 8643–8665 (2010)
23. Ville, L., Silva, L., Coupez, T.: Convected level set method for the numerical simulation of fluid buckling. Int. J. Numer. Methods Fluids. **66**(3), 324–344 (2011)

# A Curvature-Adapted Anisotropic Surface Re-meshing Method

**Franco Dassi and Hang Si**

**Abstract** We present a method for re-meshing surfaces in order to follow the intrinsic anisotropy of the surfaces. In particular, we use the information related to the normals to the surfaces, and embed the surfaces into a higher dimensional space (here we embed the surfaces in a six-dimensional space). This allows us to settle an isotropic mesh optimization problem in this embedded space: starting from an initial mesh of a surface, we optimize the mesh by improving the mesh quality measured in the embedded space. The mesh is optimized by properly combining common local mesh operations, i.e., edge flipping, edge contraction, vertex smoothing, and vertex insertion. All operations are applied directly on the three-dimensional surface mesh and the resulting mesh is curvature adapted. This new method improves the approach proposed by Lévy and Bonneel (Variational anisotropic surface meshing with Voronoi parallel linear enumeration. In: Proceedings of the 21st International Meshing Roundtable, pp. 349–366. Springer, New York, 2013), by allowing to preserve sharp features. The reliability and robustness of the proposed re-meshing technique is shown via a number of examples.

## 1 Introduction

Surface mesh generation is a central topic in computer visualization, geometry processing, and numerical simulation. Many computational applications involve triangulation of a complex surface geometry. The main challenge is to automatically generate a surface mesh which satisfies various criteria with respect to geometry approximation such as mesh size and mesh quality.

The goal of the current work is to generate a surface mesh which well approximates the geometry of the surface and with a number of elements as small

F. Dassi (✉)
Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy
e-mail: franco.dassi@polimi.it

H. Si
Weierstrass Institute, Mohrenstr. 39, 10117 Berlin, Germany
e-mail: hang.si@wias-berlin.de

S. Perotto, L. Formaggia (eds.), *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, SEMA SIMAI Springer Series 5,
DOI 10.1007/978-3-319-06053-8_2

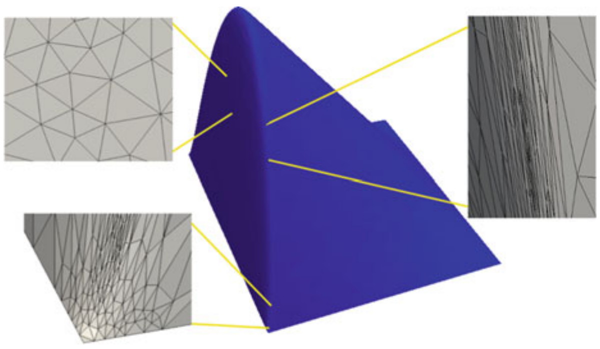


**Fig. 1** The best approximation of a surface (shown in the *middle*) should consist of mesh elements of different size, shape, and orientation that respect the principle curvatures of that surface

as possible. For this purpose, we aim at modifying the mesh in order to get a new mesh whose elements are adapted according to the curvature of the surface. Intuitively, more curved regions of the surface will contain small elements and a dense vertex sampling, while almost flat regions will have large elements with more sparse vertices. Using only isotropic elements may be far from optimal for these purposes. However, an anisotropic mesh, i.e., a mesh with stretched elements, could offer a better "number of elements vs geometry fitting" behavior. Figure 1 shows an example of an anisotropic mesh.

In this paper, we propose a new method for re-meshing 3d surfaces based on the idea of higher dimensional embedding [8, 32, 34]. We use the information related to the normals to the surface, and we embed the surface into $\mathbb{R}^6$. Our method directly optimizes a two-dimensional triangular mesh of the surface embedded in $\mathbb{R}^6$ in such a way that its triangles are as uniform as possible in $\mathbb{R}^6$. Thus, the resulting mesh will be a curvature-adapted anisotropic surface mesh. This method has the following properties:

- The core operation of this method is a uniform re-meshing of a surface. It fits the well-developed mesh adaptation strategies. It is possible to use any optimization-based isotropic surface re-meshing method.
- It can handle arbitrary complicated geometries and topologies, as well as a very strong anisotropy.
- It automatically preserves sharp features, corner and edge singularities.
- It is robust. For instance, the initial mesh can be very coarse or crude in geometry approximation.
- It is easy to implement.

This paper is organized as follows: Sect. 2 presents the background about anisotropic meshes and reviews the related works on anisotropic surface re-meshing. In Sect. 3, the main idea of the embedding space is introduced. The proposed method is described into the details in Sect. 4. Some experimental results of re-meshing surfaces from implicit functions and on a surface containing sharp features are shown in Sect. 5. Finally, a summary of the proposed method and an outlook of future works are given in Sect. 6.

## 2 Background and Related Work

In this section we describe the basic idea behind anisotropic mesh adaptation, then we provide the state of art about surface re-meshing.

### *2.1 Anisotropic Meshes*

Many physical problems exhibit anisotropic features, i.e., their solutions change more significantly in one direction than others. Examples include in particular convection-dominated problems whose solutions have, e.g., layers, shocks, or corner and edge singularities. Anisotropic meshes have great importance in numerical methods to solve partial differential equations. They improve the accuracy of the solution and decrease the computational cost.

Anisotropy denotes the way distances and angles are distorted. It is naturally related to approximation theory and it is important in function interpolation [10, 21, 39, 42, 43]. For a smooth function, the anisotropy is best characterized by the Hessian of that function. In practice, a central question is how to efficiently distinguish the anisotropy of a given problem. Another important question is how to characterize the anisotropy in a such a way that an optimal mesh for a given problem can be defined. These are all difficult questions and are active research subjects. In practice, a Riemannian metric tensor field (either provided or automatically derived) is used to guide the generation of anisotropic meshes [19, 27, 37, 38]. Mesh adaptation has been proven an efficient way to capture the anisotropy, see, e.g., [14, 21, 22].

### *2.2 Related Works*

Surface re-meshing has been an active research subject for nearly two decades, see a nice survey given by  Alliez et al. [3]. Most of the early methods work either in a 2d parameterization space or directly in the 3d space, and their focus is to create isotropic 3d surface meshes. Many recent methods have been developed for creating anisotropic surface meshes. In the following, we give an overview of works which are more related to ours. More complete surveys are available [6, 44].

### 2.2.1 Curvature Evaluation and Reconstruction

Heckbert and Garland [24] proposed a quadric-based metric tensor for surface simplification. It is defined for each vertex of the mesh and uses the face normals around it. They showed that this metric is directly related to surface curvature. Jiao et al. [28] used this metric to derive a Riemannian metric field for anisotropic surface mesh adaptation. The method of Alliez et al. [1] first constructs a curvature tensor field from a given polygonal surface mesh by estimating the curvature tensor at every vertex. Then, they trace lines along the principle curvature directions from which an anisotropic mesh can be obtained. These methods, which rely on a reconstructed metric tensor field, are very suitable for re-meshing polygonal meshes whose original geometry is not available. As we will show in Sect. 3, for surfaces whose geometry is accessible, such as implicit functions and CAD models, there is a natural way to perform a surface re-meshing that respects the curvature **without** using a reconstructed metric tensor field.

### 2.2.2 Delaunay Refinement Based Methods

Voronoi diagrams and their dual Delaunay triangulations are fundamental data structures and have several applications [4]. The so-called *restricted Voronoi diagram* (RVD) and *restricted Delaunay triangulation* (RDT) are their generalization to surfaces [18]. Cheng et al. proposed to generate an anisotropic RDT from a 3d anisotropic RVD [11, 31]. Although their concept is valid in theory, it remains a big challenge in practice to ensure that the dual of an anisotropic RVD admits a valid triangulation. Boissonnat et al. [5] proposed the notion of locally uniform anisotropic Delaunay meshes, and proposed a practical Delaunay refinement algorithm to generate an anisotropic RDT of a surface with respect to a given metric tensor field [6]. By using the curvature tensor of the surface, this algorithm is able to produce a curvature-adapted anisotropic mesh of the surface. However, a fundamental difficulty of Delaunay refinement based algorithms is that it does not respect sharp features. Cheng et al. [12] showed that sharp features can be respected by using weighted restricted Delaunay triangulations. However, to efficiently obtain the appropriate weights remains a challenging problem.

### 2.2.3 Centroidal Voronoi Based Methods

A Centroidal Voronoi Tessellation (CVT) is a particular type of Voronoi diagram such that its generating points coincide with the centroids (center of mass) of its Voronoi cells. It has a highly regular structure such that the RDT obtained from it will have well-shaped triangles. It has been applied in many applications including surface mesh generation [2, 16, 35]. Efficient algorithms are proposed to generate CVTs [36]. It has been further generalized to anisotropic CVT with respect to a Riemannian metric tensor field [15], and an anisotropic RVD on surfaces can be defined. However, a more theoretical analysis is needed in understanding anisotropic CVTs, and it remains a challenge to efficiently generate them. Lévy and Bonneel [34] proposed a novel approach to compute CVT in higher dimensions. They used it to generate anisotropic curvature-adapted surface meshes. However, it does not preserve sharp features. Our method is inspired by the idea of Lévy and Bonneel in [34], and it can easily preserve sharp features.

## 3 Surface Embedding in $\mathbb{R}^6$

The re-meshing method proposed in this paper is inspired by the method of Lévy and Bonneel [34]. The basic idea is pioneered by Cañas and Gortler and Lai et al [8, 32] and is originated in the application of feature characterization [41] from image processing [29]. It treats the anisotropy by increasing the dimensions such that it becomes an isotropy in a higher dimensional space: *an isotropic mesh in a higher dimensional space will correspond to an anisotropic mesh in the lower dimensional space* (see Fig. 2). This concept has been successfully applied in generating curvature-adapted surface meshes [30, 34].

For a smooth surface, it is natural to consider the unit normals defined on surface points as the components of the codimension, i.e., using the components of the Gaussian map of the surface. Given a surface $\Omega$ in $\mathbb{R}^3$, one can embed it into $\mathbb{R}^6$ by using the embedding: $\Phi : \Omega \to \mathbb{R}^6$ defined by: $\Phi(\mathbf{x}) = [x,\ y,\ z,\ s\,n_x,\ s\,n_y,\ s\,n_z]^t$, where $(n_x, n_y, n_z)^t$ denotes the unit normal to $\Omega$ at $\mathbf{x}$, and $s \in [0, +\infty)$ is a user-specified constant. This embedding $\Phi$ allows us to approximate the geodesic edge lengths in $\Omega$ by the Euclidean edge lengths in $\Phi(\Omega)$. Each edge length in $\Phi(\Omega)$ is determined by two parts: its Euclidean length in $\mathbb{R}^3$ and the variation of the normals at its endpoints, scaled by the parameter.

By this transformation, in flat regions of $\Omega$, the lengths of edges remain the same in $\Phi(\Omega)$. While in regions which have high curvatures, the lengths of edges in $\Phi(\Omega)$ become much larger than theirs in $\mathbb{R}^3$. Since the distance in $\mathbb{R}^6$ are affected by the

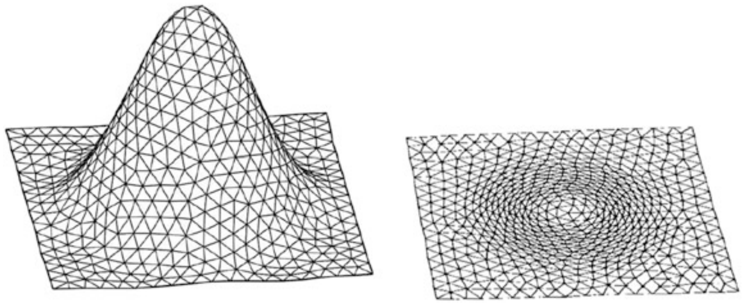


**Fig. 2** An isotropic mesh in $\mathbb{R}^3$ (*left*) and the corresponding anisotropic mesh in $\mathbb{R}^2$ (*right*). This is a very representative picture of the idea behind the higher dimension embedding proposed in [34]

normals, an isotropic mesh of the surface $\Phi(\Omega)$ in $\mathbb{R}^6$, when transformed back into $\mathbb{R}^3$, will become a curvature-adapted anisotropic mesh of $\Omega$.

By embedding a surface in higher dimensions motivates the new problem: *How to generate an isotropic good quality surface mesh in this embedding space?* In principle, a direct generalization of available methods in 3d is possible. But this will be impractical due to the $d!$ cost of memory requirement.

Lévy and Bonneel [34] overcome this difficulty by using their *Vorpaline* (Voronoi Parallel Linear Enumeration) technique to compute a restricted centroidal Voronoi diagram (CVT) embedded in 6d. It directly compute the 6d Voronoi cells by iterative half-space clipping. It requires only the nearest neighbor information for a point set in $\mathbb{R}^6$.

It is reported by Lévy and Bonneel [34] that this method may produce flipped (self-intersected) triangles, in particular in regions where the anisotropy varies too fast. This fact implies that if the normals between the neighbor vertices are varying too big, their method may not work correctly. One possible way to resolve this problem is to insert new vertices between these neighboring vertices. However, the method of Lévy and Bonneel [34] does not support inserting new vertices dynamically.

Another well-known problem in RVD- and CVT-based methods is that sharp features or details of the surfaces may be smoothed or missing in the resulting mesh, see the examples in [34]. Although a theoretical solution has been proposed in [12], its efficiency is still a challenge in practice.

Due to these problems, we propose a new method in the next section. In particular, we show that a common mesh optimization framework for isotropic surface re-meshing may be applied in re-meshing surfaces embedded in a higher dimensional space. The primary difference between previous methods and ours is the use of lengths and angles evaluated in the embedded space.

## 4 The Re-meshing Approach

Consider a surface $\Omega$ in $\mathbb{R}^3$. For simplicity, we assume that $\Omega$ is a smooth surface, i.e., it contains no corner and edge singularities, then we consider the non-smooth case. In this section, we propose an optimization-based method for re-meshing $\Omega$.

### *4.1 Preliminaries*

#### 4.1.1 Assumptions

Our method assumes that the following two functions are provided: (1) given a point $\mathbf{p} \in \Omega$, the function returns the normal to $\Omega$ at $\mathbf{p}$; (2) given a point $\mathbf{p} \in \mathbb{R}^3$, the function returns the closest point $\mathbf{q} \in \Omega$. If $\Omega$ is represented by an implicit function or it is a parameterized surface (of CAD models), the exact normals and the closest points of $\Omega$ are provided. If $\Omega$ is given as a polygonal mesh, these two functions must be approximated from the input data, see [1, 9, 23, 40].

Our method assumes that an initial triangular surface mesh $\mathscr{T}_{init}$ of $\Omega$ is provided. Moreover, all triangles in $\mathscr{T}_{init}$ are consistently oriented. For example, when the edge **ab** is the common edge of two triangles **abc** and **bad** in $\mathscr{T}_{init}$, we understand that both normals to **abc** and **bad** are pointing outwards the surface.

### 4.1.2 Geometry Approximation

We use a heuristic condition to justify geometric approximation. Intuitively, if a triangle is used to approximate a patch of a surface, the normal of the triangle should not vary too much with respect to the normals of any point in this patch. Let $f$ be a face in the surface mesh. We say that $f$ is *inverted* if the angle between two vectors $\mathbf{n}_f$ and $\mathbf{n_c}$ is less than a given threshold, where $\mathbf{n}_f$ is the outwards normal to $f$ and $\mathbf{n_c}$ is the normal to the surface at the closest point of the centroid $\mathbf{c}$ of $f$. In other words, the angle between $\mathbf{n}_f$ and $\mathbf{n_c}$ determines if the face is inverted or not. An inverted face is considered as a bad approximation of the geometry. It is crucial to use an appropriate threshold in order to achieve best mesh quality. In our experiments in Sect. 5, the threshold we used is 90°.

### 4.1.3 Controlling Anisotropy

The basic idea behind the anisotropic mesh adaptation is to distort the distance. In this framework we are going to achieve this goal moving from the standard scalar product in the embedded space. We consider a surface $\Gamma$ and two points $A, B \in \Gamma$, we apply the map $\Phi$ and we have: $\Phi(A) = (x_A, y_A, z_A, sn_A, sv_A, sw_A)^t$, and $\Phi(B) = (x_B, y_B, z_B, sn_B, sv_B, sw_B)^t$, where $x_A, y_A, z_A$ and $x_B, y_B, z_B$ are the $\mathbb{R}^3$ coordinates of $A$ and $B$, respectively, and $n_A, v_A, w_A$ and $n_B, v_B, w_B$ are the components of the unit normal vectors to $\Gamma$ at $A$ and $B$, respectively. Then, we define a scalar product by $(A, B)_{6d} := x_A x_B + y_A y_B + z_A z_B + s^2\big(n_A n_B + v_A v_B + w_A w_B\big)$. Then the length of a segment $AB$ in the embedded space is $l_{AB} = \sqrt{(A - B, A - B)_{6d}}$ and the angle could be computed via the cosine

$$\cos(ACB) := \frac{(A - C, B - C)_{6d}}{(A - C, A - C)_{6d}(B - C, B - C)_{6d}},$$

where $C$ is another point of the surface $\Gamma$. At this level, we find it is better to reinterpret the constant $s$ in $\Phi$. We consider a surface $\Gamma$ and two points $A, B \in \Gamma$. The 6d vector scalar product between these two points is

$$(A, B)_{6d} = \underbrace{x_A x_B + y_A y_B + z_A z_B}_{I} + s^2\big(\underbrace{n_A n_B + v_A v_B + w_A w_B}_{II}\big).$$

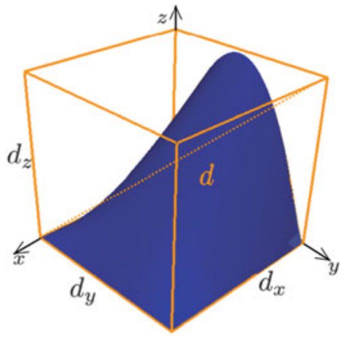


**Fig. 3** Bounding box of the surface $\Gamma$

Since the coordinates of both $A$ and $B$ varies in the bounding box of the surface $\Gamma$, we can say that $I \in [-d^2, d^2]$ where $d$ is the diagonal of the bounding box of $\Gamma$. Moreover, we have $n_A^2 + v_A^2 + w_A^2 = 1$ and $n_B^2 + v_B^2 + w_B^2 = 1$. We can see that the quantity $II \in [-1, 1]$. The parameter $s$ is introduced to give more or less importance to the normals, $II$, on the whole value of $(A, B)_{6d}$. But, since $I \in [-d^2, d^2]$ and $II \in [-1, 1]$, the contribution of $I$ and $II$ is unbalanced because it depends on the dimension of the bounding box. To make the quantity $I$ and $II$ almost comparable, we decide to modify the 6d scalar product in $(A, B)_{6d} = x_A x_B + y_A y_B + z_A z_B + (h_\Gamma s)^2 \big(n_A n_B + v_A v_B + w_A w_B\big)$, where $h_\Gamma = (d_x + d_y + d_z)/3$ and $d_x$, $d_y$ and $d_z$ are the dimension of the bounding box of $\Gamma$, see Fig. 3. In this way the quantity $I$ and $II$ are at most comparable and the parameter $s$ have effectively the effect to give more or less importance to the normals.

## 4.2 Overview of the Approach

The inputs of the algorithm are: an initial triangular mesh $\mathscr{T}_{init}$ of the surface $\Omega$, a user-specified edge length $L_{des}$, a user-specified minimum face angle $\theta_{min}$, and a parameter $s$ that specifies the desired amount of anisotropy.

We initialize a mesh $\mathscr{T} := \mathscr{T}_{init}$. Then we use the map $\Phi$ to transform $\mathscr{T}$ into a surface mesh $\mathscr{T}_\Phi$ in $\mathbb{R}^6$. Our goal is to remesh $\mathscr{T}$ such that $\mathscr{T}_\Phi$ is an uniform isotropic triangular mesh in $\mathbb{R}^6$. For simplicity, we assume the map $\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}^6$ is bijective. Therefore, we still work in $\mathbb{R}^3$, **but** we use calculated quantities (edge lengths and angles) from $\mathbb{R}^6$.

Our method works in two phases: (i) sampling, we split edges in $\mathscr{T}$ whose lengths (measured in $\mathbb{R}^6$) are too long with respect to the given parameter $L_{min}$ (Sect. 4.4); (ii) optimizing, we maximize the smallest face angle (measured in $\mathbb{R}^6$) such that they are not smaller than $\theta_{min}$ (Sect. 4.5). The result is a curvature-adapted anisotropic triangular mesh of the surface. This method is detailed in the following subsections.

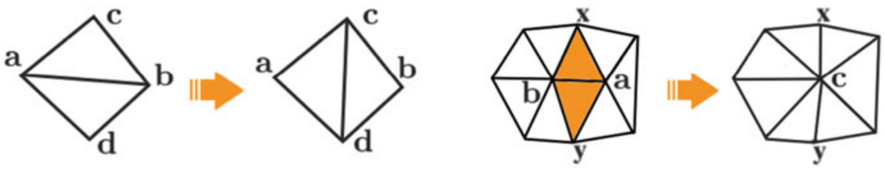


**Fig. 4** Edge-flip (*left*); edge-collapse (*right*)

## 4.3 Local Mesh Modifications

Our algorithm applies a series of local surface mesh modifications directly on the mesh $\mathscr{T}$. The most well-known and commonly used local modifications are: edge-flip, edge-collapse, vertex insertion, and vertex smoothing. These operations are already extensively discussed, see [13, 23, 25, 26]. In this section, we describe how they are realized in our method (Fig. 4).

### 4.3.1 An Edge-Flip Algorithm

Edge-flip is the most efficient and effective local operation to improve simultaneously the geometry approximation and the quality of the surface mesh. Therefore, it should be applied whenever it is possible. For this purpose, we developed an edge-flip algorithm. It is inspired by the well-known Lawson's flip algorithm [33] for constructing 2d Delaunay triangulations.

An *edge-flip* on **ab** will remove the two faces **abc** and **bad** and replace them with two new faces **cdb** and **dca**. As a result, edge **ab** is replaced by edge **cd**. In our method, we want to flip **ab** if the new triangles are "better" than the old ones with regard to either geometry approximation or mesh quality.

Given an edge **ab** in $\mathscr{T}$, we check if **ab** needs to be flipped if one of the following two conditions are met: (a) geometric approximation, either face **abc** or face **bad** is inverted; (b) mesh quality both **abc** and **bad** are not inverted and the smallest 6d-angle of the two new faces (**cdb** and **dca**) is larger than the smallest 6d-angle of **abc** and **bad**.

If an edge **ab** satisfies either (a) or (b), it implies that either face **abc** or face **bad** is bad (or both of them), it (or they) should be replaced by a better face(s). However, it is not always possible to do an edge-flip. An edge **ab** is *flippable*, if both of the following two conditions are satisfied: (c) topology validation, the edge **cd** does not exist in $\mathscr{T}$; (d) geometry validation, neither **cdb** nor **dca** is inverted. Condition (c) guarantees that the topology of the surface will not be changed after the flip, and (d) ensures that the new created faces are not bad approximation of the geometry.

The edge-flip algorithm is shown in Algorithm 1. This algorithm uses two stacks $S$ and $S_1$. Initially, the stack $S$ keeps all edges to be checked and flipped, and the stack $S_1$ is empty, and it will keep edges which are not flippable. Edges in $S_1$ are

**Algorithm 1** The edge-flip algorithm

```
FLIPEDGES($S$, $S_1$)
Data: $S$ is a stack of edges to be checked and flipped; and
      $S_1$ is another stack which is empty on input.
 1: while $S$ is non-empty do
 2:    count := 0;
 3:    while $S$ is non-empty do
 4:       pop ab from $S$;
 5:       if ab meets condition (a) or (b) then
 6:          if ab meets conditions (c) and (d) then
 7:             flip ab to cd;
 8:             for xy $\in$ {ac, cb, bd, da} do
 9:                push xy on $S$;
10:             end for
11:             count := count + 1;
12:          else
13:             push ab on $S_1$;
14:          end if
15:       end if
16:    end while
17:    if $S_1$ is non-empty and count > 0 then
18:       swap $S$ and $S_1$;
19:    end if
20: end while
```

tried again if any flip has been done in the inner loop (lines 3–16). Once an edge is get flipped, we push the boundary edges of the new triangles into stack (lines 8–10). Hence flips may propagate to the neighboring edges. On return, if $S_1$ is not empty, it means there are unflippable edges or even inverted faces.

### 4.3.2 Edge Collapse

Edge collapse is a common operation for simplifying meshes. An edge collapse unifies the two endpoints of the edge and two adjacent faces of this edge vanish. The unification of the two endpoints can be either one of the endpoints or a new vertex inside the cavity of adjacent faces of this edge.

For simplicity, we simply choose one of the endpoints as the new vertex by checking if there will be no inverted face at this endpoint after the edge collapse. Then, we push all the linked edges of this vertex into a stack, and the routine FLIPEDGES() is called to locally improve the mesh.

### 4.3.3 Vertex Insertion

Vertex insertion is a common operation for refining meshes. Two well-known approaches for vertex insertion are the Bowyer-Watson algorithm and the

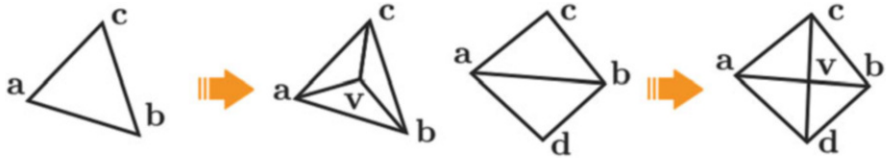


**Fig. 5** Vertex insertion. Face-split (*left*); edge-split (*right*)

incremental flip algorithm [17]. They are equivalent in generating Delaunay triangulations. We use the incremental flip algorithm for our vertex insertion in 3d surfaces.

Let **v** be a new vertex (in the surface) to be inserted. It is first located in the mesh, i.e., either a triangle **abc** or an edge **ab** of the mesh is declared to contain this vertex to be inserted. We then replace the face **abc** by three new faces, **abv**, **bcv**, **cav** (see Fig. 5, left); or split the edge **ab** by replacing two faces **abc**, **bad** by four faces **avc**, **vbc**, **bvd**, **vad** (see Fig. 5, right). Then we put all linked edges of **v** into a stack, and the routine FLIPEDGES() is called to improve the mesh.

*Remark 1* Since the surface may be curved, the vertex (in the surface) is not necessary in a face or edge in current mesh. Vertex location in a 3d surface mesh is a difficult problem. We avoid this problem in our method by always choosing the midpoint of an edge, and then a new vertex is obtained by snapping this point to the surface. Hence, we can declare that the edge to be split contain this new vertex.

### 4.3.4 Vertex Smoothing

For a given vertex **v**, the vertex smoothing operation finds a new location for this vertex such that the local mesh quality is improved **without** changing the mesh topology.

A generic smoothing method moves a point **v** in a new location **v**′ given by the formula

$$\mathbf{v}' = \mathbf{v} + \alpha \sum_{\mathbf{v}_i \in \omega_{\mathbf{v}}} f(d(\mathbf{v}, \mathbf{v}_i))\mathbf{u}_i \ , \tag{1}$$

here $\alpha$ is a constant, $f$ is a function $f : \mathbb{R} \to \mathbb{R}$, $\omega_{\mathbf{v}}$ is the set of vertices that are connected to **v** and $\mathbf{u}_i$ are the unit vectors that identifies the direction from $\mathbf{v}_i$ to **v**, see Fig. 6. Finally, $d$ is the distance between **v** and $\mathbf{v}_i$, normalized with a desired edge length of the mesh, i.e. $d(\mathbf{v}, \mathbf{v}_i) = ||\mathbf{v} - \mathbf{v}_i||/L_{\text{des}}$, where $|| \cdot ||$ is the standard euclidean norm in $\mathbb{R}^3$ and $L_{\text{des}}$ is the desired edge length.

Different smoothing methods are characterized by different choices of the parameter $\alpha$ and the function $f$ in Equation (1). For example the classical Laplacian smoothing, [20], is defined by $\alpha = 1/\#\omega_{\mathbf{v}}$ and $f(d) = -d$, where $\#\omega_{\mathbf{v}}$ is the cardinality of the set $\omega_{\mathbf{v}}$.

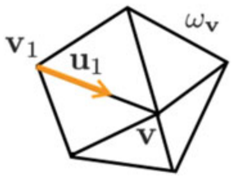


**Fig. 6** Patch $\omega_{\mathbf{v}}$

$$\begin{tikzcd}
 & \phi(\Gamma_h) \arrow[r, "\text{sampling}"] & \phi(\Gamma_h') \arrow[r, "\text{optimizing}"] & \phi(\Gamma_h'') \arrow[d, "\phi^{-1}"'] \\
\Gamma \arrow[r, "\text{mesh}"] & \Gamma_h \arrow[u, "\phi"] & & \Gamma_h''
\end{tikzcd}$$

**Fig. 7** Scheme of the re-meshing process

It is interesting to notice that, according to the sign of the function $f$, the vertex $\mathbf{v}$ is $\mathbf{v}$ is attracted or repelled by the vertex $\mathbf{v}_i$, i.e., if $f(d(\mathbf{v}, \mathbf{v}_i)) > 0$, $\mathbf{v}$ is repelled by $\mathbf{v}_i$, while if $f(d(\mathbf{v}, \mathbf{v}_i)) < 0$ it is attracted by $\mathbf{v}_i$. Bossen and Heckbert [7] implement a vertex smoothing method that exploits this attraction/repulsion behavior via the function $f_{\mathrm{BH}}(d) := (1-d^4)e^{-d^4}$. Since their mesh generation procedure is metric-based, their desired edge length 1. Thus if the vertex $\mathbf{v}$ is too close to $\mathbf{v}_i$, $0 < d < 1$, it is repelled by $\mathbf{v}_i$, then if it is far from $\mathbf{v}_i$, $1 < d < 1.7$, it is attracted. Finally, if it is too far, $d \geq 1.7$, or if it is exactly at the right distance, $d = 1$, $\mathbf{v}_i$ does **not** influence the position of $\mathbf{v}$.

In this paper we use the vertex smoothing function proposed by Bossen and Heckbert [7]. But we use the distance from the embedded space. We the vertex $\mathbf{v}$ $\mathbf{v}$ and the set of vertexes $\omega_{\mathbf{v}} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_n\}$, the desired location of the vertex $\mathbf{v}$, will be the one such that $||\mathbf{v}-\mathbf{v}_i||_{6\mathrm{d}} = L_{\mathrm{des}} \quad \forall i = 1, 2, \ldots n$. To move the point in this optimal location, we use the following normalized distance function $d_{6\mathrm{d}}(\mathbf{v}, \mathbf{v}_i) = ||\mathbf{v}-\mathbf{v}_i||_{6\mathrm{d}}/l_{\mathrm{mean}}$, where $l_{\mathrm{mean}}$ is the mean 6d-length of the edges $\mathbf{v}\mathbf{v}_i$ in the patch $\omega_{\mathbf{v}}$. The smoothing formula we propose is $\mathbf{v}' = \mathbf{v} + \alpha\mathbf{w}/||\mathbf{w}||$, where $\mathbf{w} = \sum_{\mathbf{v}_i \in \omega_{\mathbf{v}}} f_{\mathrm{BH}}(d_{6\mathrm{d}}(\mathbf{v}, \mathbf{v}_i))\,\mathbf{u}_i$ , and the parameter $\alpha$ is chosen in such a way $\alpha := \max\left\{c_1 l_{\omega_{\mathbf{v}}}, c_2 L_{\min}\right\}$ , where $L_{\min}$ is the minimum valid edge length and $L_{\omega_{\mathbf{v}}} := \min_{\mathbf{v}_i \in \omega_{\mathbf{v}}} ||\mathbf{v}-\mathbf{v}_i||$ , and $c_1$ and $c_2$ are constant that adjust these two length, in this work we choose $c_1 = 0.01$ and $c_2 = 10$. After we have found the location $\mathbf{v}'$, we project the point on the surface $\Gamma$, we push all the link edges at $\mathbf{v}$ into a stack and we call the routine FLIPEDGES() to locally improve the quality of the mesh.

The vertex smoothing is the most time consuming step in the proposed surface re-meshing method. To increase the speed of this process we decide to move not **all** the vertices of the mesh, but only the ones that are more far away from their desired positions. We first sort the vertices according to the distances to their desired locations. Then, we move a percentage of the vertices which are far away from their desired locations.

A sketch of the whole embedding procedure is provided in Fig. 7.

**Algorithm 2** The sampling algorithm

```
SAMPLING($\Omega$, $\mathscr{T}$, $Q$, $L_{min}$)
Data: $Q$ is a queue of triangles in $\mathscr{T}$.
1: while $Q$ is non-empty do
2:     pop a face $f$ from $Q$;
3:     Let $e$ be the longest edge of $f$;
4:     if $\|e\|_{6d} > 1.5\ L_{min}$, then
5:         split $e$ by adding $v \in \Omega$ into $\mathscr{T}$;
6:         update $Q$;
7:     end if
8: end while
```

## 4.4 Sampling

The purpose of sampling is to achieve the desired mesh size with respect to the given 6d-length parameter $L_{min}$. Our strategy is straightforward, splitting edges of $\mathscr{T}$ whose 6d-lengths is too long compared to $L_{min}$. For simplicity, each long edge is split by adding the point in the surface which is closet to the midpoint of the edge. Also, we do not collapse short edges in this phase, it will be applied in the optimization phase.

The sampling algorithm is shown in Algorithm 2. It initializes a queue $Q$ which contains all triangles in $\mathscr{T}$. It then works in a loop until $Q$ is empty. On each face $f$ popped from $Q$, it checks the longest edge $e$ of $f$ and split it if the 6d-length of $e$ is too long (lines 4–7). Then $Q$ is updated (line 6) by removing old faces and adding new faces.

Inserting a new vertex into a 3d surface mesh may deform it very much. This is particularly the case when the surface mesh is only a very crude approximation of the original geometry. Recall that our vertex insertion routine will automatically improve the local mesh by FLIPEDGES(). Our experiments (shown in Sect. 5) showed that this edge flip algorithm is very effective in improving both of the geometry approximation and the mesh quality in this phase.

## 4.5 Optimizing

Since the sampling phase has removed long edges, the goal of the optimizing phase is to maximize the smallest 6d-angle of triangles in the mesh $\mathscr{T}$.

Mesh optimization is performed by iteratively combining a series of local modifications, which are, edge-flips, vertex smooth, edge-collapse, and vertex insertion: vertex smoothing iteratively moves the positions of vertices (stay on surface) such that the minimum 6d-angle is improved around each vertex; edge collapse is used to remove the edges opposite to small angles; edge split is used to split the edges opposite to large angles by inserting a vertex close to the midpoints of

**Algorithm 3** The optimizing algorithm

```
OPTMIZING($\Omega$, $\mathcal{T}$, $L_{min}$, $\theta_{min}$, $I$, $J$, $K$)
Data: $I$, $J$, and $K$ are user-specified iterations.
 1: $\theta_{max} := 180^o - 2 * \theta_{min}$;
 2: Collapse too short edges with respect to $L_{min}$;
 3: for $i \in \{1, \dots, \mathrm{I}\}$ do
 4:     for $j \in \{1, \dots, \mathrm{J}\}$ do
 5:         for $k \in \{1, \dots, \mathrm{K}\}$ do
 6:             Smooth all vertices;
 7:         end for
 8:         Collapse edges for removing angles $< \theta_{min}$;
 9:     end for
10:     Split edges for removing angles $>= \theta_{max}$;
11: end for
```

the edges, iteratively. We call the routine FLIPEDGES() within each local operations to improve local mesh quality.

### 4.5.1 Sharp Features

Our method can be easily adapted to mesh non-smooth surfaces, i.e., surfaces containing edges and corner singularities and sharp features. These features are commonly present in complicated geometries. Sharp features can be preserved as follows. We consider a surface $\Gamma$, for example a geometry coming from a CAD model. The whole geometry is the sum of a finite set of patches that are joint together along their common boundaries, i.e.,

$$\Gamma = \bigcup_{i=1}^{n} \Gamma_i \,, \quad \text{and} \quad \Gamma_i \cap \Gamma_j = \begin{cases} \emptyset \\ \gamma_{ij} \end{cases} , \tag{2}$$

where $\Gamma_i$ are the patches and $\gamma_{ij}$ is the common line between $\Gamma_i$ and $\Gamma_j$, if it exists. The proposed method will re-mesh each sub-surface $\Gamma_i$ separately. To preserve sharp features in the routine FLIPEDGES() we add the condition (e) sharp features, an edge that belongs to a sharp feature will never be flipped.

## 5 Examples

In this section, several examples are presented to demonstrate the reliability of the proposed method. Firstly, we re-mesh some very simple cases to ensure that this method works. In particular, we consider surface in which we could predict where and how the triangles will be stretched, and we experimentally verify the behavior of the proposed re-meshing method. Then, we use several examples which

**Table 1** Statistics of examples

| | Examples | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 1. | $L_{min}$ | 0.1 | 0.1 | 0.1 | 0.1 |
| 2. | # Vertices in $\mathscr{T}_{init}$ | 11,518 | 49,761 | 12,985 | 5,246 |
| 3. | # Triangles in $\mathscr{T}_{init}$ | 23,032 | 98,560 | 25,543 | 10,488 |
| 4. | # Vertices in $\mathscr{T}_{output}$ | 3,697 | 20,659 | 4,937 | 2,357 |
| 5. | # Triangles in $\mathscr{T}_{output}$ | 7,390 | 40,790 | 9,582 | 4,710 |
| 6. | Sampling time (s) | 7 | 28 | 8 | 2 |
| 7. | Optimizing time (s) | 20 | 141 | 85 | 49 |
| 8. | Minimum aspect ratio | 3.841e−02 | 2.383e−01 | 8.414e−04 | 3.249e−01 |

The Optimizing time (in row 7) was the time for one outer loop in the optimizing algorithm (Algorithm 3), i.e. $I = 1$. The number of inner loops is $J = K = 4$

contain strong anisotropy and sharp features. The statistical information about these examples and the CPU times are given in Table 1. To evaluate the level of anisotropy of the resulting mesh, we consider the so-called **aspect ratio**:

$$q(T) := 2\,\frac{r_T}{R_T}\,, \tag{3}$$

here $T$ is a generic triangle of the mesh, $r_T$ and $R_T$ are the radii of the inscribed and circumscribed circle, respectively. If $q(T) \approx 0$, the triangle $T$ is stretched, while, for triangles close to the equilateral one, we have $q(T) \approx 1$.

From Table 1, we observe that the meshes present really stretched elements, in fact the minimum value of the aspect ratio is close to 0. Moreover, we see that in all these examples the final anisotropic mesh have fewer number of elements and nodes than the initial one.

*Example 1* We consider the disk, defined by the zero level set of the function $f_1(x\,,y\,,z) := \left(\frac{x}{0.8}\right)^2 + \left(\frac{y}{0.8}\right)^2 + \left(\frac{z}{0.4}\right)^2 - 1$. In Fig. 8, we highlight some zones and we indicate how the triangles will look like after the re-meshing procedure.

Since there is a very big change of curvature along one direction $\overrightarrow{v}$, see zone $A$ in Fig. 8, we expect that the triangles will be stretched along $\overrightarrow{w}$, i.e. the perpendicular direction that lies on the tangent plane to the surface. Moreover we expect equilateral triangles in the zone $B$, where the surface is smooth and flat. The resulting mesh of this example is shown in Fig. 9. The shapes of triangles in the resulting mesh behave as expected.

*Example 2* We consider the sinusoidal surface, defined by the zero level set of the function $f_2(x\,,y\,,z) := \sin(\pi x)\sin(\pi y) - z$. In Fig. 10 we show the whole surface together some zones of interest. In particular we notice that there are a lot of regions where the mesh has to be isotropic, see the zones $A$ and $B$ in Fig. 10. The resulting mesh of this example is shown in Fig. 11.

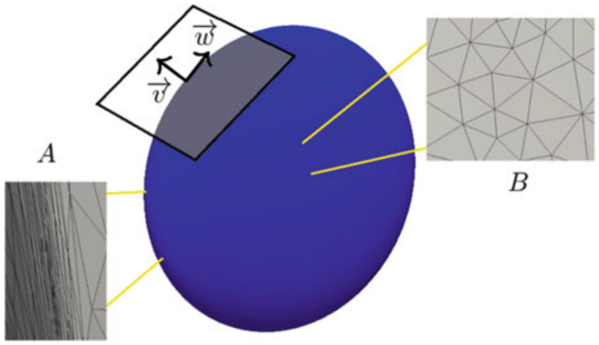


**Fig. 8** The geometry of **Example 1**. The zero level set of the function $f_1$, some zones are *highlighted* to show how the adapted mesh will look like

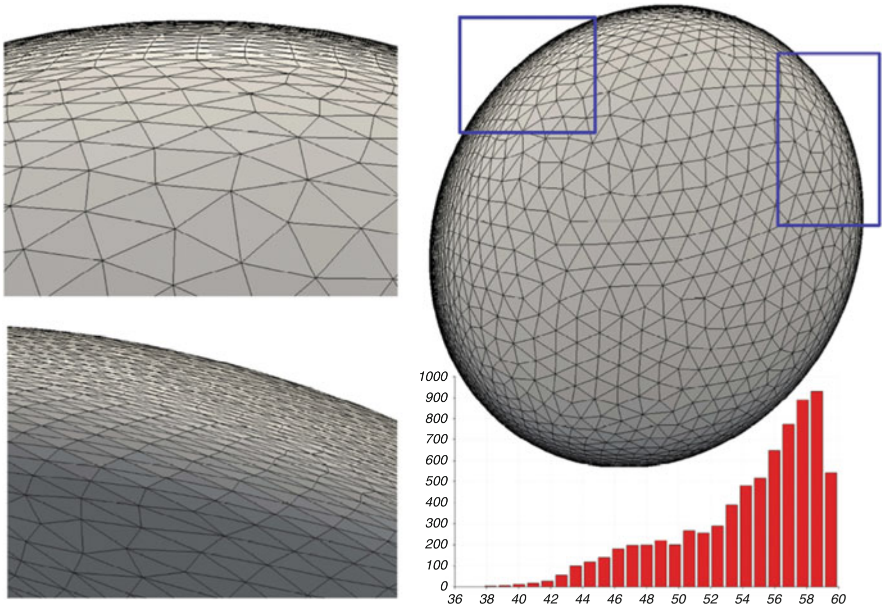


**Fig. 9** The optimized mesh of **Example 1**. The mesh quality (the smallest 6d angles in triangles) histogram (*bottom-right*)

*Example 3* In this example, we consider the function, $f_3(x\ ,y\ ,z) := \tanh(20x) - \tanh(20(x-y)\ -10) - z$. The zero level set of such a function is a surface that presents a smart change of curvature and it is flat in others. We stat from a very bad approximating mesh obtained by a marching cube algorithm, see Fig. 12 on the left. This initial mesh does not sharply approximate the surface: the triangles are not oriented in the right way, there is an over sampling and the triangles are

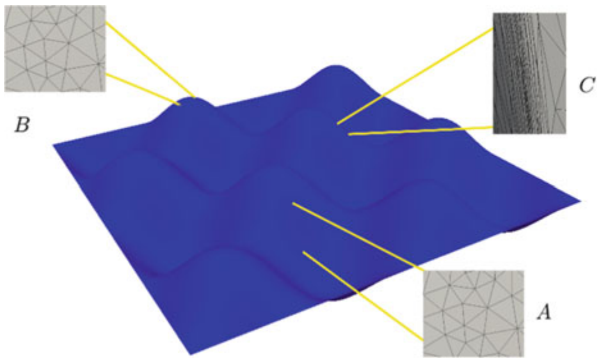


**Fig. 10** The geometry of the **Example 2**. The zero level set of the function $f_2$, some zones are *highlighted* to show how the adapted mesh will look like

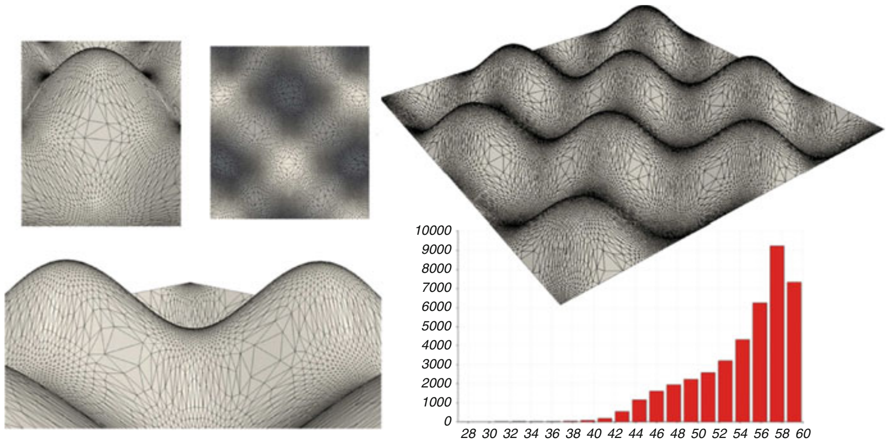


**Fig. 11** The resulting mesh of **Example 2**. The mesh quality (the smallest 6d angles in triangles) histogram (*bottom-right*)

somewhere really far away from the real geometry. This experiment will challenge the robustness of the proposed method.

Figure 12, right shows the resulting mesh of this example. We may appreciate that the geometry is really well fitted by the resulting mesh. In particular, the initial mesh has been entirely changed. Both the **shape** and **orientation** of the elements are suited to fit correctly the geometry at hand.

*Example 4* In this example, we constructed a surface containing sharp features. The purpose is to illustrate that the sharp features are well preserved by our method. The input of this example consists of three surface patches. They join together along their common boundary, see Fig. 13. The resulting mesh shown in Fig. 13 well preserved the sharp features.

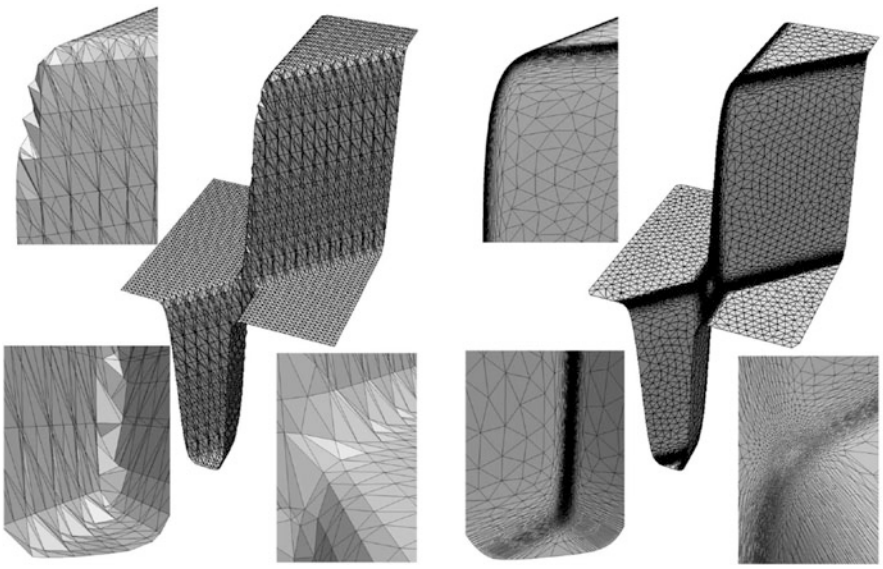


**Fig. 12** The initial mesh of **Example 3** (*left*) and the resulting mesh (*right*) with some zooms

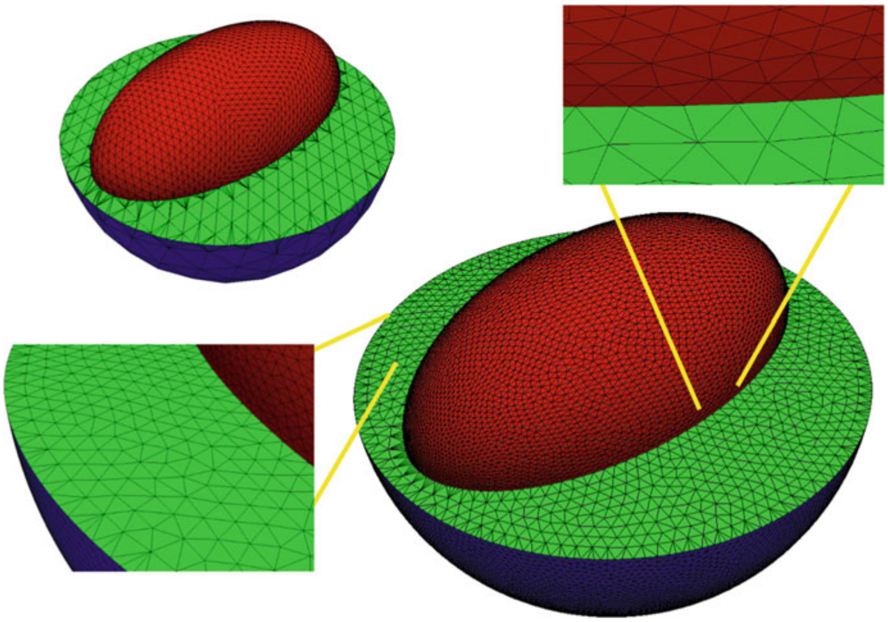


**Fig. 13** **Example 4**. The initial mesh (*top-left*) and the resulting mesh and some of its details (*bottom-right*)

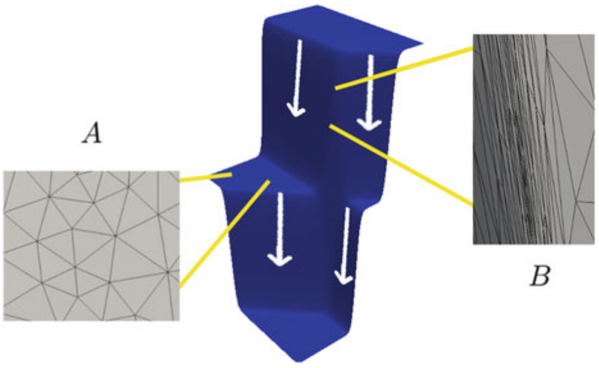


**Fig. 14** The geometry of example in subsection 5.1. The zero level set of the function $f_4$, some zones are *highlighted* to show how the adapted mesh will look like

## *5.1 Numerical Test of the s Parameter*

In all the previous examples we have considered $s = 1$ to proceed with the re-meshing algorithm. Now we try to change this value and see what is the effect on the resulting mesh. We define the function, $f_4(x, y, z) := \tanh(20y) - \tanh(20(x - y) - 10) - z$, whose zero level set is the surface represented in Fig. 14. This surface presents some flat regions, like the zone $A$ in Fig. 14, and a series of very deep jumps, see the arrows in Fig. 14.

We **set** a desired 6d-length and we consider these values of the parameter $s = 0.1, 1., 5., 25.$

Increasing the factor $s$, will increase the length of the edges of the mesh, **but** this fact it is not completely true. The **real** effect of increasing the parameter $s$ is to **emphasize** the variation of the normal, i.e., the variation of curvature. In fact, where the surface is flat, the size of the mesh elements is the same for each values of $s$. Instead, where it presents a variation of the curvature, it is more and more refined. When we are dealing with big values of $s$, small variation of the normals will correspond to large variation of the edge length, so the sampling procedure may refine these edges. The resulting meshes are shown in Figs. 15 and 16. In Table 2 we report the minimum value of the aspect ratio for each mesh. From these data we observe that higher values of $s$ increase the stretching of triangles and the number of elements.

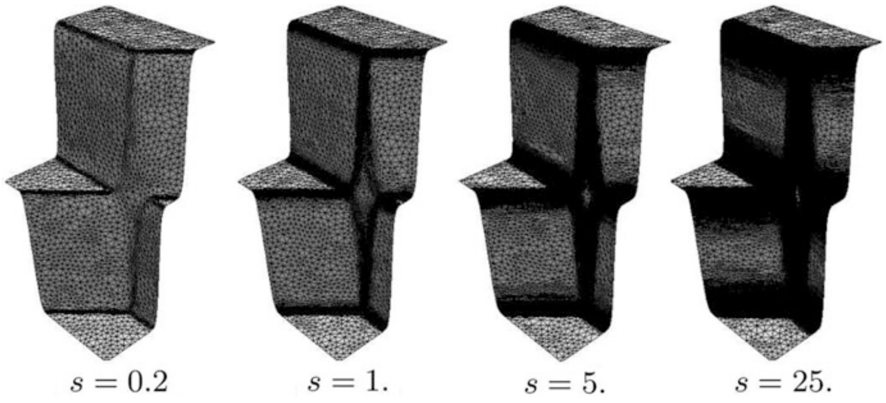


**Fig. 15** Resulting meshes for different values of the parameter *s*

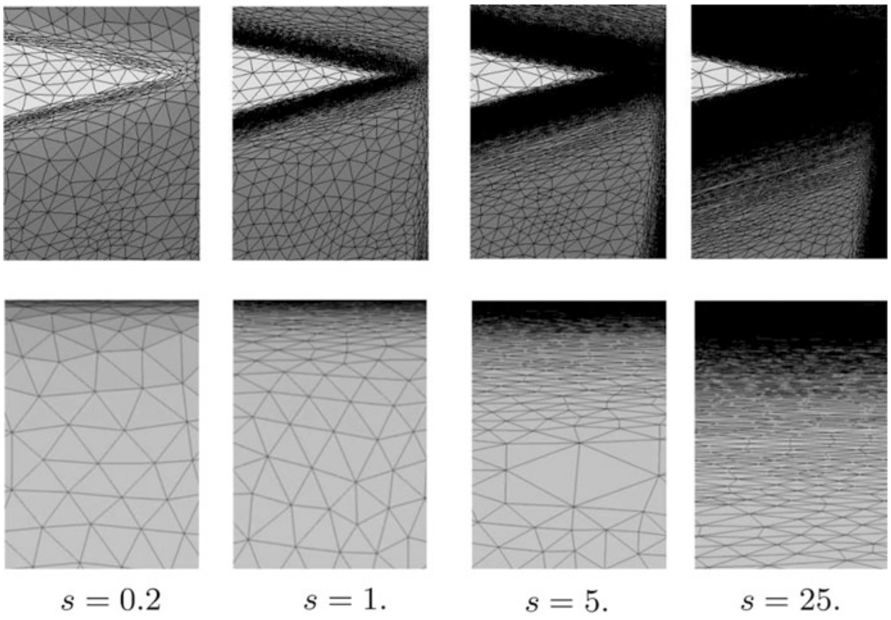


**Fig. 16** Details of the optimized meshes in Fig. 15

**Table 2** Minimum value of the aspect ratio with different value of *s*

| | $s = 0.2$ | $s = 1.$ | $s = 5.$ | $s = 25.$ |
|---|---|---|---|---|
| Minimum aspect ratio | 1.472e−01 | 3.173e−04 | 8.414e−05 | 4.162e−07 |
| # Vertices in $\mathscr{T}_{output}$ | 2,652 | 8,583 | 54,001 | 409,444 |
| # Triangles in $\mathscr{T}_{output}$ | 5,067 | 16,696 | 107,239 | 817,605 |

## 6 Conclusions

In this paper, we presented a curvature-adapted anisotropic surface re-meshing method, based on a high-dimensional embedding of surfaces. This method is, in principle, simple. It fits the well-developed mesh adaptation framework, while it has several advantages. For instances, sharp features are always respected; it is robust in handling strong anisotropy, and it is easy to implement. Our experimental results showed that this method is able to produce good meshes for various 3d surfaces which may have an arbitrarily complicated geometry and may contain strong anisotropy.

There are many questions which are still open. A very important theoretical question is: How well may this mapping $\Phi$ approximate the geodesic distances in 3d surfaces? Are there upper or lower bounds on distance variations by this mapping? A theoretical study of these issues could lead to more efficient methods, and meshes with fewer elements.

The edge-flip algorithm we described seems very useful in improving both geometry approximation and mesh quality. However, its termination is not yet proved. We found that the selection of the threshold angle for checking inverted faces is very crucial. A good value will give edge-flip more freedom and may produce highly stretched triangles.

In practice, many surfaces are given as a polygonal mesh, i.e., the original geometry is not available. A good recovery and estimation of the surface normals are necessary in order to achieve good results. We plan to implement such feature into our code.

The running time of our implementation is far from optimal. There are many possibilities to improve it.

## References

1. Alliez, P., Cohen-Steiner, D., Devillers, O., Lévy, B., Desbrun, M.: Anisotropic polygonal remeshing. ACM Trans. Graph. **22**(3), 485–493 (2003)
2. Alliez, P., De Verdire, E., Devillers, O., Isenburg, M.: Isotropic surface remeshing. In: Shape Modeling International, 2003, pp. 49–58. IEEE (2003)
3. Alliez, P., Ucelli, G., Gotsman, C., Attene, M.: Recent advances in remeshing of surfaces. In: Shape Analysis and Structuring, pp. 53–82. Springer, New York (2008)
4. Aurenhammer, F.: Voronoi diagrams a survey of a fundamental geometric data structure. ACM Comp. Surv. **23**(3), 345–405 (1991)
5. Boissonnat, J.D., Wormser, C., Yvinec, M.: Locally uniform anisotropic meshing. In: Proceedings of the Twenty-Fourth Annual Symposium on Computational geometry, pp. 270–277. ACM, New York (2008)
6. Boissonnat, J.D., Shi, K.-L., Tournois, J., Yvinec, M.: Anisotropic Delaunay meshes of surfaces. ACM Trans. Graph. p. 10. http://doi.acm.org/10.1145/2721895, 2012
7. Bossen, F.J., Heckbert, P.S.: A pliant method for anisotropic mesh generation. In: 5th International Meshing Roundtable, pp. 63–74. Citeseer (1996)

8. Cañas, G.D., Gortler, S.J.: Surface remeshing in arbitrary codimensions. Vis. Comput. **22**(9–11), 885–895 (2006)
9. Canas, G.D., Gortler, S.J.: Shape operator metric for surface normal approximation. In: Proceedings of the 18th International Meshing Roundtable, pp. 447–461. Springer, New York (2009)
10. Chen, L., Sun, P., Xu, J.: Optimal anisotropic meshes for minimizing interpolation errors in $L^p$-norm. Math. Comput. **76**(257), 179–204 (2007)
11. Cheng, S.W., Dey, T.K., Ramos, E.A., Wenger, R.: Anisotropic surface meshing. In: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm, pp. 202–211. ACM, New York (2006)
12. Cheng, S.W., Dey, T.K., Levine, J.A.: A practical Delaunay meshing algorithm for a large class of domains. In: Proceedings of the 16th International Meshing Roundtable, pp. 477–494. Springer, New York (2008)
13. de Cougny, H.L., Shephard, M.S.: Surface meshing using vertex insertion. In: Proceedings of the 5th International Meshing Roundtable, pp. 243–256. Citeseer (1996)
14. Dobrzynski, C., Frey, P.: Anisotropic Delaunay mesh adaptation for unsteady simulations. In: Proceedings of the 17th International Meshing Roundtable, pp. 177–194. Springer, New York (2008)
15. Du, Q., Wang, D.: Anisotropic centroidal Voronoi tessellations and their applications. SIAM J. Sci. Comput. **26**(3), 737–761 (2005)
16. Du, Q., Faber, V., Gunzburger, M.: Centroidal Voronoi tessellations: applications and algorithms. SIAM Rev. **41**(4), 637–676 (1999)
17. Edelsbrunner, H.: Geometry and Topology for Mesh Generation. Cambridge University Press, Cambridge (2001)
18. Edelsbrunner, H., Shah, N.R.: Triangulating topological spaces. Int. J. Comput. Geo. Appl. **7**(04), 365–378 (1997)
19. El-Hamalawi, A.: Mesh generation—application to finite elements. Eng. Constr. Archit. Manag. **8**(3), 234–235 (2001)
20. Field, D.A.: Laplacian smoothing and Delaunay triangulations. Commun. Appl. Numer. Methods **4**(6), 709–712 (1988)
21. Formaggia, L., Perotto, S.: New anisotropic a priori error estimates. Numer. Math. **89**(4), 641–667 (2001)
22. Frey, P.J., Alauzet, F.: Anisotropic mesh adaptation for CFD computations. Comput. Methods Appl. Mech. Eng. **194**(48), 5068–5082 (2005)
23. Frey, P.J., Borouchaki, H.: Geometric surface mesh optimization. Comput. Vis. Sci. **1**(3), 113–121 (1998)
24. Heckbert, P.S., Garland, M.: Optimal triangulation and quadric-based surface simplification. Comput. Geol. **14**(1), 49–65 (1999)
25. Hoppe, H.: Progressive meshes. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp. 99–108. ACM, New York (1996)
26. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Mesh optimization. In: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, pp. 19–26. ACM, New York (1993)
27. Huang, W.: Metric tensors for anisotropic mesh generation. J. Comput. Phys. **204**(2), 633–665 (2005)
28. Jiao, X., Colombi, A., Ni, X., Hart, J.: Anisotropic mesh adaptation for evolving triangulated surfaces. Eng. Comput. **26**(4), 363–376 (2010)
29. Kimmel, R., Malladi, R., Sochen, N.: Images as embedded maps and minimal surfaces: movies, color, texture, and volumetric medical images. Int. J. Comput. Vis. **39**(2), 111–129 (2000)
30. Kovacs, D., Myles, A., Zorin, D.: Anisotropic quadrangulation. Comput. Aid. Geom. Des. **28**(8), 449–462 (2011)

31. Labelle, F., Shewchuk, J.R.: Anisotropic Voronoi diagrams and guaranteed-quality anisotropic mesh generation. In: Proceedings of the Nineteenth Annual Symposium on Computational Geometry, pp. 191–200. ACM, New York (2003)
32. Lai, Y.K., Zhou, Q.-Y., Hu, S.-M., Wallner, J., Pottmann, D., et al.: Robust feature classification and editing. IEEE Trans. Vis. Comput. Graph. **13**(1), 34–45 (2007)
33. Lawson, C.L.: Software for $C^1$ surface interpolation. In: Mathematical Software III, pp. 164–191. Academic, New York (1977)
34. Lévy, B., Bonneel, N.: Variational anisotropic surface meshing with Voronoi parallel linear enumeration. In: Proceedings of the 21st International Meshing Roundtable, pp. 349–366. Springer, New York (2013)
35. Lévy, B., Liu, Y.: $l_p$ centroidal Voronoi tessellation and its applications. ACM Trans. Graph. **29**(4), 119 (2010)
36. Liu, Y., Wang, W., Lévy, B., Sun, F., Yan, D.-M., Lu, L., Yang, C.: On centroidal Voronoi tessellation energy smoothness and fast computation. ACM Trans. Graph. **28**(4), 101 (2009)
37. Loseille, A., Alauzet, F.: Continuous mesh framework part I: well-posed continuous interpolation error. SIAM J. Numer. Anal. **49**(1), 38–60 (2011)
38. Loseille, A., Alauzet, F.: Continuous mesh framework part II: validations and applications. SIAM J. Numer. Anal. **49**(1), 61–86 (2011)
39. Mirebeau, J.M.: Optimally adapted meshes for finite elements of arbitrary order and $W^{1,p}$ norms. Numer. Math. **120**, 271–305 (2012)
40. Owen, S.J., White, D.R., Tautges, T.J.: Facet-based surfaces for 3d mesh generation. In: IMR, pp. 297–311 (2002)
41. Pottmann, H., T. Steiner, T., Hofer, M., Haider, C., Hanbury, A.: The Isophotic Metric and its Application to Feature Sensitive Morphology on Surfaces. Springer, New York (2004)
42. Rippa, S.: Long and thin triangles can be good for linear interpolation. SIAM J. Numer. Anal. **29**(1), 257–270 (1992)
43. Shewchuk, J.R.: What is a good linear element? Interpolation, conditioning, and quality measures. In: Proceedings of 11th International Meshing Roundtable, pp. 115–126 (2002)
44. Zhong, Z., Guo, X., Wang, W., Lévy, B., Sun, F., Liu, Y., Mao, W.: Particle-based anisotropic surface meshing. ACM Trans. Graph. **32**(4), 99 (2013)

# The Benefits of Anisotropic Mesh Adaptation for Brittle Fractures Under Plane-Strain Conditions

**Marco Artina, Massimo Fornasier, Stefano Micheletti, and Simona Perotto**

**Abstract** We develop a reliable a posteriori anisotropic first order estimator for the numerical simulation of the Frankfort and Marigo model of brittle fracture, after its approximation by means of the Ambrosio-Tortorelli variational model. We show that an adaptive algorithm based on this estimator reproduces all the previously obtained well-known benchmarks on fracture development with particular attention to the fracture directionality. Additionally, we explain why our method, based on an extremely careful tuning of the anisotropic adaptation, has the potential of outperforming significantly in terms of numerical complexity the ones used to achieve similar degrees of accuracy in previous studies.

## 1 Introduction

A variational formulation for the evolution of the fracture surface in a brittle, linearly elastic solid was proposed by Frankfort and Marigo in [19]. The main feature of this model is that there is no predefined crack, i.e., the crack is able to propagate in the material without any constraint, driven only by elastic forces. Bourdin et al. [7] addressed the numerical approximation of the solution of the fracture model by Frankfort and Marigo by first approximating it via the Ambrosio-Tortorelli variational model. Then, an extremely fine discretization is considered to be able to capture the fracture path and its expected directional developments, independently of the intrinsic anisotropies of the a priori prescribed mesh. This technique proved to be very stable not only in the case of anti-plane shear, but also in the more challenging situation where plane-strain is considered, capturing the physically

M. Artina (✉) • M. Fornasier
Faculty of Mathematics, Technische Universität München, Boltzmannstrasse 3, 85748 Garching, Germany
e-mail: marco.artina@ma.tum.de; massimo.fornasier@ma.tum.de

S. Micheletti • S. Perotto
MOX - Dipartimento di Matematica “F.Brioschi”, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy
e-mail: stefano.micheletti@polimi.it; simona.perotto@polimi.it

S. Perotto, L. Formaggia (eds.), *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, SEMA SIMAI Springer Series 5,
DOI 10.1007/978-3-319-06053-8_3

expected crack paths and directionalities. However, the cost of an extremely fine discretization to render the material numerically homogenous is enormous, leading to the quest for possible alternative techniques based on adaptive strategies, which can break the ambiguity of *"the crack following the mesh or the mesh following the crack"*. In the work of Chambolle et al. [10], an anisotropic adaptive finite element method was presented for the simulation of the model of Frankfort and Marigo in the anti-plane shear case. The adaptive re-meshing is, however, based on a local approximation of the Hessian of the solution, which, unfortunately, may lack the expected regularity. In the approach of Süli et al. [8], the adaptivity is driven exclusively by an a posteriori first order estimator, but only isotropic mesh refinement was considered. In our recent work [5], we tried to combine these two previous approaches, designing an appropriate a posteriori anisotropic first order estimator, leading to mesh coarsening far from the fracture and fine mesh elements exclusively very close to the crack path. Again this new method resulted being very efficient and effective, producing stable and realistic results for some test cases where the force applied to the domain is orthogonal to the domain itself. In this work, we study and present numerical results in the case the fracture is induced by a plane-strain. These tests play a key role in validating the reliability and the applicability of anisotropic mesh adaptation in the context of quasi-static crack path detection. Indeed, for assessing the quality of our results we can count on previous precise studies of the behaviour of the fracture, both from numerical and physical viewpoints [2, 7].

The numerical experiments in Sect. 4 show that the proposed method is very stable and it allows us to reproduce all the previously obtained predictions on fracture development, in particular its directionality features. Additionally, we expect that our method, based on an extremely careful tuning of the anisotropic adaptation, outperforms significantly the ones used to achieve similar degrees of accuracy in previous studies. Unfortunately, the only reference with which we can compare the computational burden is Süli et al. [8], while for Bourdin et al., Chambolle et al., Del Piero et al. [7, 10, 16] we are obliged to extrapolate our positive expectation from the very fine meshes showed in the corresponding numerical sections.

The paper is organized as follows. In Sect. 1.1, we describe the model, in Sect. 2, we introduce the discrete setting and the anisotropic error estimator which drives the mesh adaptation. In Sect. 3, we provide the algorithm for the minimization of the energy functional, while in Sect. 4, we address the numerical results on the benchmark tests, comparing them with the expected ones from the literature.

## 1.1 The Mathematical Model of Plane-Strain Fracture

The considered model extends the anti-plane case proposed in [1] and, following [7], we introduce an isotropic linearly elastic constitutive law, i.e., the Plane-strain

Ambrosio-Tortorelli functional

$$J^{PAT}(\mathbf{u}, v) = \frac{1}{2}\int_{\Omega} (v^2 + \eta)\sigma(\mathbf{u}) : E(\mathbf{u})\, d\mathbf{x} + \frac{1}{2}\int_{\Omega} \Big[\alpha(1-v)^2 + \varepsilon|\nabla v|^2\Big]\, d\mathbf{x}, \tag{1}$$

where $\Omega \subset \mathbb{R}^2$, the fields $\mathbf{u} : \Omega \to \mathbb{R}^2$ and $v : \Omega \to [0,1]$ are the displacement and a smoothed crack path indicator, $0 < \eta \ll \varepsilon \ll 1$ and $\alpha = 1/(4\varepsilon)$ are suitable regularizing constants, $\boldsymbol{\sigma}(\mathbf{u}) = \lambda\ \mathrm{tr}\,(E(\mathbf{u}))\, I + 2\mu\, E(\mathbf{u})$, is the Cauchy stress tensor, with $\lambda$ and $\mu$ the Lamé constants, and, where, for every $\mathbf{d} : \Omega \to \mathbb{R}^2$,

$$E(\mathbf{d}) = \begin{bmatrix} \frac{\partial d_1}{\partial x_1} & \frac{1}{2}\left(\frac{\partial d_1}{\partial x_2} + \frac{\partial d_2}{\partial x_1}\right) \\ \frac{1}{2}\left(\frac{\partial d_1}{\partial x_2} + \frac{\partial d_2}{\partial x_1}\right) & \frac{\partial d_2}{\partial x_2} \end{bmatrix}$$

is the symmetric gradient tensor, $T_1 : T_2$ denoting the tensor product between $T_1, T_2 : \Omega \to \mathbb{R}^{2\times 2}$, and $\mathbf{x} = (x_1, x_2)^T \in \Omega$. In practice, $v$, with $0 \le v \le 1$, can be considered as a phase field for the crack interface [6, 28]. The first integral in (1) represents the elastic energy of the material, while the second integral models the energy associated with the crack propagation inside the material. The case $v = 1$ is the crack-free configuration, since the last integral vanishes. On the contrary, the regions where $v = 0$ identify the cracked area.

Let $0 = t_0 < \ldots < t_F = T$ be a partition of the time window $[0, T]$. Let $\mathbf{g} : \Omega \times [0, T] \to \mathbb{R}^2$ be an displacement assigned over a subset $\Omega_D \subset \Omega$ which drives the fracture onset, i.e.,

$$\mathbf{g}(\mathbf{x}, t) = \begin{cases} \mathbf{g}_D(t) & \text{if } \mathbf{x} \in \Omega_D, \\ \mathbf{0} & \text{elsewhere .} \end{cases}$$

Notice that, with a view to the numerical test cases, function $\mathbf{g}_D$ is assumed to be constant in space. We denote by $\mathscr{A}_k(\mathbf{g}) = \{\mathbf{u} \in [H^1(\Omega)]^2 : \mathbf{u}(\mathbf{x}) = \mathbf{g}(\mathbf{x}, t_k)\ \forall \mathbf{x} \in \Omega_D\}$ the space of the admissible solutions, i.e., the fields which coincide with $\mathbf{g}$ on $\Omega_D$ at $t = t_k$. According to a quasi-static approximation [19], the minimization of the functional $J$ in (1) at the time level $t_k$ consists of finding the pair $(\mathbf{u}(t_k), v(t_k))$, with $k = 0, \ldots, F$, such that

$$(\mathbf{u}(t_k), v(t_k)) \in \underset{\substack{\mathbf{u} \in \mathscr{A}_k(\mathbf{g}) \\ v \in H^1(\Omega; [0,1]),\, v|_{CR_{k-1}} = 0}}{\mathrm{argmin}} J(\mathbf{u}, v), \tag{2}$$

where $CR_{k-1} = \{\mathbf{x} \in \overline{\Omega} : v(t_{k-1}) < \texttt{CRTOL}\}$, with $\texttt{CRTOL}$ a tolerance used to enforce the irreversibility of the crack. For simplicity we denote hereafter $\mathbf{g}(\mathbf{x}, t)$ with $\mathbf{g}(t)$. Moreover, standard notation is understood to denote Sobolev spaces and their norms [23].

Following [5], we relax the constraint in (2) with two penalization terms which lead us to rewrite the Plane-strain Ambrosio-Tortorelli elasticity functional as

$$J^{PAT}(\mathbf{u}, v) = \frac{1}{2}\int_{\Omega} (v^2 + \eta)\sigma(\mathbf{u}) : E(\mathbf{u})\, d\mathbf{x} + \frac{1}{2}\int_{\Omega} \left[\alpha(v-1)^2 + \varepsilon|\nabla v|^2\right] d\mathbf{x}$$
$$+ \frac{1}{2\gamma_A}\int_{\Omega_D} |\mathbf{u} - \mathbf{g}(t_k)|^2\, d\mathbf{x} + \frac{1}{2\gamma_B}\int_{CR_{k-1}} v^2\, d\mathbf{x}, \tag{3}$$

where $\gamma_A$ and $\gamma_B$ are the penalty constants. Henceforth we always deal with this functional instead of (1). We are dealing now with an unconstrained minimization process. At each time level, we seek the pair $(\mathbf{u}(t_k), v(t_k))$ such that

$$(\mathbf{u}(t_k), v(t_k)) \in \underset{(\mathbf{u},v)\in[H^1(\Omega)]^2\times H^1(\Omega;[0,1])}{\operatorname{argmin}} J^{PAT}(\mathbf{u}, v). \tag{4}$$

Since the penalized constraints are clearly continuous, convex, and always non-negative, the proof of the convergence of the minimizers of (4) to the minimizers of (2), for $\gamma_A, \gamma_B \rightarrow 0$, follows from $\Gamma$-convergence arguments (see [14]). Moreover, we are interested in local minimizers for two reasons. On the one side, the search for global minimizers is an NP-hard problem; on the other side, one can expect that the fracture moves along critical points of the physical energy. Therefore, it is not only (numerically) impossible to compute global minimizers with some guarantees, but it may also not be a meaningful choice from a physical viewpoint.

Mimicking the proof in [8] for the anti-plane case, we can prove that the functional $J^{PAT}$ is Fréchet-differentiable in $[H^1(\Omega)]^2 \times (H^1(\Omega) \cap L^\infty(\Omega))$. In particular, the Fréchet derivative of $J^{PAT}$ along direction $(\mathbf{w}, z)$ is

$$\left(J^{PAT}(\mathbf{u}, v; \mathbf{w}, z)\right)' = \underbrace{\int_{\Omega} (v^2 + \eta)\sigma(\mathbf{u}) : E(\mathbf{w})\, d\mathbf{x} + \frac{1}{\gamma_A}\int_{\Omega_D} (\mathbf{u} - \mathbf{g}(t_k)) \cdot \mathbf{w}\, d\mathbf{x}}_{=a(v;\mathbf{u},\mathbf{w})}$$
$$+ \underbrace{\int_{\Omega} \left[v\, z\sigma(\mathbf{u}) : E(\mathbf{u}) + \alpha(v-1)z + \varepsilon\nabla v \cdot \nabla z\right] d\mathbf{x} + \frac{1}{\gamma_B}\int_{CR_{k-1}} v\, z\, d\mathbf{x}.}_{=b(\mathbf{u};v,z)} \tag{5}$$

Accordingly, we recall the definition of critical points of $J^{PAT}$:

**Definition 1** The pair $(\mathbf{u}, v) \in [H^1(\Omega)]^2 \times (H^1(\Omega) \cap L^\infty(\Omega))$ is a *critical point* of $J^{PAT}$ if $\left(J^{PAT}(\mathbf{u}, v; \mathbf{w}, z)\right)' = 0$ for all $\mathbf{w} \in [H^1(\Omega)]^2$ and for all $z \in (H^1(\Omega) \cap L^\infty(\Omega))$.

Following Proposition 2.2 in [5], we can prove that condition $0 \leq v \leq 1$ is automatically guaranteed for any critical point.

## 2 Anisotropic Error Analysis

This section collects the main developments of this paper. After providing the discrete approximation of the functional $J^{PAT}$, we introduce the main tools of the anisotropic background, and we derive the theoretical result used to drive the anisotropic mesh adaptation procedure.

### 2.1 Discretization of $J^{PAT}$

We introduce the discrete counterpart of the minimization problem (4) in a finite element setting. Thus, we denote by $\{\mathcal{T}_h\}_{h>0}$ a family of conforming meshes of $\overline{\Omega}$, and let $N_h$ be the index set of the vertices of $\mathcal{T}_h$, and $\mathcal{E}_h$ the skeleton of $\mathcal{T}_h$. Henceforth, we assume that the boundary of $\Omega_D$ coincides with the union of consecutive edges in $\mathcal{E}_h$. We associate with $\mathcal{T}_h$ the space $X_h$ of continuous piecewise linear finite elements [11].
We denote by $J_h^{PAT}(\mathbf{u}_h, v_h)$ the discrete correspondent of $J^{PAT}(\mathbf{u}, v)$ in (3), with $\mathbf{u}_h = (u_{h,1}, u_{h,2})^T \in [X_h]^2$ and $v_h \in X_h$, given by

$$
\begin{aligned}
J_h^{PAT}(\mathbf{u}_h, v_h) = \frac{1}{2}\int_{\Omega} \Big[ \big(P_h(v_h^2) + \eta\big)\, \sigma(\mathbf{u}_h) : E(\mathbf{u}_h) \\
+ \alpha P_h((v_h-1)^2) + \varepsilon |\nabla v_h|^2 \Big]\, d\mathbf{x} \\
+ \frac{1}{2\gamma_A}\sum_{i=1}^{2}\int_{\Omega_D} P_h\big((u_{h,i} - g_{h,i}(t_k))^2\big)\, d\mathbf{x} + \frac{1}{2\gamma_B}\int_{CR_{k-1}} P_h\big(v_h^2\big)\, d\mathbf{x},
\end{aligned}
\tag{6}
$$

where $P_h : C^0(\overline{\Omega}) \to X_h$ is the Lagrangian interpolant onto the space $X_h$, with $\mathbf{g}_h(t_k) = (g_{h,1}(t_k), g_{h,2}(t_k))^T \in [X_h]^2$ a suitable discrete approximation of $\mathbf{g}(t_k)$. In particular, we pick $\mathbf{g}_h(t_k)$ such that

$$
\int_{\Omega_D} \mathbf{g}_h(t_k)\cdot \mathbf{w}_h\, d\mathbf{x} = \int_{\Omega_D} \mathbf{g}(t_k)\cdot \mathbf{w}_h\, d\mathbf{x} \qquad \forall \mathbf{w}_h \in [X_h]^2, \tag{7}
$$

i.e., $\mathbf{g}_h(t_k)$ is the $L^2(\Omega_D)$-projection of $\mathbf{g}(t_k)$ onto $[X_h]^2$. The action of the operator $P_h$ is equivalent to a mass lumping [30].

The discrete analogue to (4) consists of finding the pair $(\mathbf{u}_h(t_k), v_h(t_k))$ such that

$$
(\mathbf{u}_h(t_k), v_h(t_k)) \in \underset{(\mathbf{u}_h, v_h)\in [X_h]^2\times X_h}{\operatorname{argmin}} J_h^{PAT}(\mathbf{u}_h, v_h).
$$

Definition 1 can be also provided in the discrete case.

**Definition 2** The pair $(\mathbf{u}_h, v_h) \in [X_h]^2 \times X_h$ is a *critical point* of $J_h^{PAT}$ if, for all $(\mathbf{w}_h, z_h) \in [X_h]^2 \times X_h$, $\big(J_h^{PAT}(\mathbf{u}_h, v_h; \mathbf{w}_h, z_h)\big)' = 0$, where

$$\big(J_h^{PAT}(\mathbf{u}_h, v_h; \mathbf{w}_h, z_h)\big)' =$$

$$\underbrace{\int_\Omega (P_h(v_h^2) + \eta)\sigma(\mathbf{u}_h) : E(\mathbf{w}_h)\, d\mathbf{x} + \frac{1}{\gamma_A}\sum_{i=1}^{2}\int_{\Omega_D} P_h\left((u_{h,i} - g_{h,i}(t_k))w_{h,i}\right)\, d\mathbf{x}}_{=a_h(v_h;\mathbf{u}_h,\mathbf{w}_h)}$$

$$+ \underbrace{\begin{array}{c}\displaystyle\int_\Omega \Big[P_h(v_h z_h)\sigma(\mathbf{u}_h) : E(\mathbf{u}_h) + \alpha P_h\big((v_h - 1)z_h\big) + \varepsilon \nabla v_h \cdot \nabla z_h\Big] d\mathbf{x} \\ \displaystyle + \frac{1}{\gamma_B}\int_{CR_{k-1}} P_h(v_h z_h) d\mathbf{x}\end{array}}_{=b_h(\mathbf{u}_h;v_h,z_h)}$$

is the Fréchet derivative of $J_h^{PAT}$.

Thanks to the mass lumping associated with $P_h$ and to the assumption

$$k_{ij} = \int_\Omega \nabla\xi_i \cdot \nabla\xi_j \, d\mathbf{x} \le 0 \qquad \forall i \ne j \in N_h,$$

about the stiffness matrix $K$, with $\{\xi_l\}_{l=1}^{\#N_h}$ the basis of $X_h$, the property $0 \le v_h \le 1$, related to the discrete maximum principle (see, e.g., [12, 22, 29]), can be assessed for any critical point $v_h$ of (6).

## 2.2 The Anisotropic Setting

Following [15, 25], we recover the anisotropic information from the spectral properties of the affine map $T_K : \hat{K} \to K$, with $\mathbf{x} = T_K(\hat{\mathbf{x}}) = M_K\hat{\mathbf{x}} + \mathbf{b}_K$, from the equilateral reference triangle $\hat{K}$ with vertices $(-\sqrt{3}/2, -1/2)$, $(\sqrt{3}/2, -1/2)$, $(0, 1)$, inscribed in the unit circle, to the generic triangle $K$ of $\mathscr{T}_h$, with $M_K \in \mathbb{R}^{2\times 2}$, $\mathbf{b}_K \in \mathbb{R}^2$, $\mathbf{x} \in K$ and $\hat{\mathbf{x}} \in \hat{K}$.
In particular, we apply the polar decomposition to the Jacobian $M_K$, i.e., $M_K = B_K Z_K$, where $B_K, Z_K \in \mathbb{R}^{2\times 2}$ are a symmetric positive definite and an orthogonal matrix, respectively. Matrix $B_K$ deforms $K$, while $Z_K$ turns it about the origin. Then, we consider the spectral decomposition of $B_K$, i.e., $B_K = R_K^T \Lambda_K R_K$, with $R_K^T = [\mathbf{r}_{1,K}, \mathbf{r}_{2,K}]$ and $\Lambda_K = \mathrm{diag}(\lambda_{1,K}, \lambda_{2,K})$, with $\lambda_{1,K} \ge \lambda_{2,K}$. The eigenvectors $\mathbf{r}_{i,K}$ identify the directions of the semi-axes of the ellipse circumscribed to $K$, while the eigenvalues $\lambda_{i,K}$ provide the length of these semi-axes (see Fig. 1). We also define the aspect ratio of the element $K$ by $s_K = \lambda_{1,K}/\lambda_{2,K}$. The value $s_K = 1$ corresponds to the isotropic case.

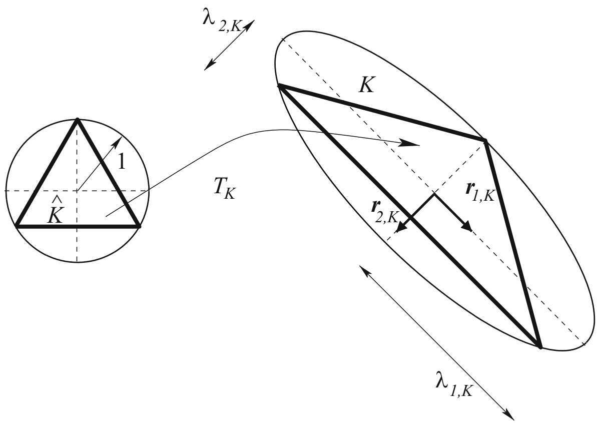


**Fig. 1** Anisotropic geometric quantities associated with the map $T_K$

To derive the a posteriori error estimator, we introduce anisotropic error estimates for the quasi-interpolant Clément operator $\mathscr{C}_h : L^2(\Omega) \to X_h$ [13].

**Lemma 1** *Let* $w \in H^1(\Omega)$*. If* $\#\Delta_K \leq \mathscr{N}$ *for some* $\mathscr{N} \in \mathbb{N}$*, and* $\mathrm{diam}(T_K^{-1}(\Delta_K)) \leq C_\Delta \simeq O(1)$*, where* $\Delta_K = \{T \in \mathscr{T}_h : T \cap K \neq \emptyset\}$*, then there exist constants* $C_s = C_s(\mathscr{N}, C_\Delta)$*, with* $s = 1, 2, 3$*, such that, for any* $K \in \mathscr{T}_h$*, it holds*

$$\|w - \mathscr{C}_h(w)\|_{L^2(K)} \leq C_1 \Big[ \sum_{j=1}^{2} \lambda_{j,K}^2 (\mathbf{r}_{j,K}^T G_{\Delta_K}(w) \mathbf{r}_{j,K}) \Big]^{1/2},$$

$$|w - \mathscr{C}_h(w)|_{H^1(K)} \leq C_2 \frac{1}{\lambda_{2,K}} \Big[ \sum_{j=1}^{2} \lambda_{j,K}^2 (\mathbf{r}_{j,K}^T G_{\Delta_K}(w) \mathbf{r}_{j,K}) \Big]^{1/2}, \tag{8}$$

$$\|w - \mathscr{C}_h(w)\|_{L^2(\partial K)} \leq C_3 \left( \frac{h_K}{\lambda_{1,K}\lambda_{2,K}} \right)^{1/2} \left[ \sum_{j=1}^{2} \lambda_{j,K}^2 (\mathbf{r}_{j,K}^T G_{\Delta_K}(w) \mathbf{r}_{j,K}) \right]^{1/2},$$

*where* $h_K = \mathrm{diam}(K)$*, while* $G_{\Delta_K}(w)$ *is the symmetric positive semi-definite matrix*

$$G_{\Delta_K}(w) = \sum_{T \in \Delta_K} \begin{bmatrix} \displaystyle\int_T \left(\frac{\partial w}{\partial x_1}\right)^2 d\mathbf{x} & \displaystyle\int_T \frac{\partial w}{\partial x_1}\frac{\partial w}{\partial x_2}\, d\mathbf{x} \\ \displaystyle\int_T \frac{\partial w}{\partial x_1}\frac{\partial w}{\partial x_2}\, d\mathbf{x} & \displaystyle\int_T \left(\frac{\partial w}{\partial x_2}\right)^2 d\mathbf{x} \end{bmatrix}. \tag{9}$$

We refer to [17, 18] for the proof.

*Remark 1* The geometric hypotheses in Lemma 1 do not limit the anisotropic features of the elements, but ensure that the variation of these features is smooth over $\Delta_K$ [27].

An equivalence result between the $H^1(\Delta_K)$-seminorm and a corresponding anisotropic version is also useful for the a posteriori analysis.

**Lemma 2** *Let $w \in H^1(\Omega)$ and $K \in \mathcal{T}_h$. For any $\beta_1, \beta_2 > 0$, it holds*

$$\min\{\beta_1, \beta_2\} \leq \frac{\beta_1(\mathbf{r}_{1,K}^T G_{\Delta_K}(w)\mathbf{r}_{1,K}) + \beta_2(\mathbf{r}_{2,K}^T G_{\Delta_K}(w)\mathbf{r}_{2,K})}{|w|^2_{H^1(\Delta_K)}} \leq \max\{\beta_1, \beta_2\},$$

*where $G_{\Delta_K}(\cdot)$ is defined as in (9).*

The proof of this result can be found in [24].

We have now all the theoretical tools required for tackling the anisotropic a posteriori analysis.

## 2.3 The A Posteriori Error Estimator

The following proposition states the main result of the paper and provides a variant on the anti-plane case addressed in [5].

**Proposition 1** *Let $(\mathbf{u}_h, v_h) \in [X_h]^2 \times X_h$ be a critical point of $J_h^{PAT}$ according to Definition 2. Then, for any pair of functions $(\mathbf{w}, z) \in [H^1(\Omega)]^2 \times H^1(\Omega)$, with $\mathbf{w} = (w_1, w_2)^T$, it holds*

$$\left|\left(J^{PAT}(\mathbf{u}_h, v_h; \mathbf{w}, z)\right)'\right| \leq C \sum_{K\in\mathcal{T}_h} \left\{ \sum_{i=1}^{2} \rho_{i,K}^A(v_h, \mathbf{u}_h)\,\omega_K(w_i) + \rho_K^B(\mathbf{u}_h, v_h)\,\omega_K(z) \right\}, \tag{10}$$

*where $C = C(\mathcal{N}, C_\Delta)$, the residuals $\rho_{i,K}^A(v_h, \mathbf{u}_h)$ and $\rho_K^B(\mathbf{u}_h, v_h)$ are*

$$\rho_{i,K}^A(v_h, \mathbf{u}_h) = \|2 v_h \sigma_i(\mathbf{u}_h) \cdot \nabla v_h\|_{L^2(K)} + \tfrac{1}{\lambda_{2,K}} \|v_h^2 - P_h(v_h^2)\|_{L^\infty(K)}\, \|\sigma_i(\mathbf{u}_h)\|_{L^2(K)}$$

$$+\tfrac{1}{2}\|[\![\sigma_i(\mathbf{u}_h)]\!]\|_{L^\infty(\partial K)}\, \|v_h^2 + \eta\|_{L^2(\partial K)} \left(\tfrac{h_K}{\lambda_{1,K}\lambda_{2,K}}\right)^{1/2} + \tfrac{|K|^{1/2} h_K^2}{\lambda_{2,K}\,\gamma_A} |u_{h,i} - g_{h,i}(t_k)|_{W^{1,\infty}(K)}$$

$$+\tfrac{\delta_{K,\Omega_D}}{\gamma_A}\left(\|u_{h,i} - g_{h,i}(t_k)\|_{L^2(K)} + \|g_{h,i}(t_k) - g_i(t_k)\|_{L^2(K)}\right),$$

$$\rho_K^B(\mathbf{u}_h, v_h) = \|(\sigma(\mathbf{u}_h) : E(\mathbf{u}_h) + \alpha)v_h - \alpha\|_{L^2(K)} + \tfrac{\varepsilon}{2}\, \|[\![\nabla v_h]\!]\|_{L^2(\partial K)} \left(\tfrac{h_K}{\lambda_{1,K}\lambda_{2,K}}\right)^{1/2}$$
$$+ \tfrac{\delta_{K,CR_{k-1}}}{\gamma_B}\, \|v_h\|_{L^2(K)} + \tfrac{h_K^2}{\lambda_{2,K}} \Big[\|\sigma(\mathbf{u}_h) : E(\mathbf{u}_h) + \alpha\|_{L^2(K)}$$
$$+ \tfrac{|K|^{1/2}\delta_{K,CR_{k-1}}}{\gamma_B}\Big]|v_h|_{W^{1,\infty}(K)},$$

*with* $\mathbf{u}_h = (u_{h,1}, u_{h,2})^T$*, the weights are*

$$\omega_K(\xi) = \Big[\sum_{i=j}^{2} \lambda_{j,K}^2 (\mathbf{r}_{j,K}^T G_{\Delta_K}(\xi)\mathbf{r}_{j,K})\Big]^{1/2} \qquad \forall \xi \in H^1(\Omega),$$

*where*

$$[\![\sigma_i(\mathbf{u}_h)]\!] = \begin{cases} [\sigma_i(\mathbf{u}_h)\cdot \mathbf{n}]_e & e \in \mathscr{E}_h \cap \Omega \\ 2(\sigma_i(\mathbf{u}_h)\cdot\mathbf{n})|_e & e \in \mathscr{E}_h \cap \partial\Omega \end{cases}, \quad [\![\nabla v_h]\!] = \begin{cases} [\nabla v_h \cdot \mathbf{n}]_e & e \in \mathscr{E}_h \cap \Omega \\ 2(\nabla v_h \cdot \mathbf{n})|_e & e \in \mathscr{E}_h \cap \partial\Omega \end{cases} \tag{11}$$

*denote the generalized jump of the* $i$*-th component of the normal Cauchy stress tensor and of the normal derivative of* $v_h$*, respectively, with* $[\cdot]_e$ *the standard jump across* $e$*,* $\mathbf{n}$ *the unit normal vector to the generic edge in* $\mathscr{E}_h$*,* $\sigma_i(\mathbf{u}_h)$ *the* $i$*-th column of* $\sigma$*,* $\mathbf{g}_h$ *is chosen as in (7), and* $\delta_{K,\varpi} = 1$ *if* $K \cap \varpi \neq \emptyset$ *and* $\delta_{K,\varpi} = 0$ *otherwise, with* $\varpi \subset \Omega$*.*

*Proof* Since $(\mathbf{u}_h, v_h)$ is a critical point of $J_h^{PAT}$, we have that

$$a_h(v_h; \mathbf{u}_h, \mathbf{w}_h) = 0 \quad \forall \mathbf{w}_h \in [X_h]^2, \qquad b_h(\mathbf{u}_h; v_h, z_h) = 0 \quad \forall z_h \in X_h. \tag{12}$$

Moreover, from (5), for any pair $(\mathbf{w}, z) \in [H^1(\Omega)]^2 \times H^1(\Omega)$, it holds

$$|\big(J^{PAT}(\mathbf{u}_h, v_h; \mathbf{w}, z)\big)'| \le |a(v_h; \mathbf{u}_h, \mathbf{w})| + |b(\mathbf{u}_h; v_h, z)|. \tag{13}$$

Now, we analyze the two terms in (13) separately, starting from $|a(v_h; \mathbf{u}_h, \mathbf{w})|$. Thanks to (12), for any $\mathbf{w} \in [H^1(\Omega)]^2$ and $\mathbf{w}_h \in [X_h]^2$, we have that

$$|a(v_h; \mathbf{u}_h, \mathbf{w})| \le |a(v_h; \mathbf{u}_h, \mathbf{w} - \mathbf{w}_h)| + |a(v_h; \mathbf{u}_h, \mathbf{w}_h) - a_h(v_h; \mathbf{u}_h, \mathbf{w}_h)|. \tag{14}$$

Let us focus on the first term on the right-hand side of (14). After splitting the integrals on the mesh elements, and by exploiting integration by parts, we get

$$\begin{aligned}
&\big|a(v_h;\mathbf{u}_h,\mathbf{w}-\mathbf{w}_h)\big| = \Big|\sum_{K\in\mathscr{T}_h}\Big\{\int_K (v_h^2+\eta)\sigma(\mathbf{u}_h) : E(\mathbf{w}-\mathbf{w}_h)\,d\mathbf{x}\\
&\quad+\frac{1}{\gamma_A}\int_K (\mathbf{u}_h-\mathbf{g}(t_k))\cdot(\mathbf{w}-\mathbf{w}_h)\,\chi_{\Omega_D}\,d\mathbf{x}\Big\}\Big|\\
&= \Big|\sum_{K\in\mathscr{T}_h}\Big\{\int_K -2v_h\,\sigma(\mathbf{u}_h)\,(\mathbf{w}-\mathbf{w}_h)\cdot\nabla v_h\,d\mathbf{x}+\int_{\partial K}(v_h^2+\eta)\sigma(\mathbf{u}_h)\,(\mathbf{w}-\mathbf{w}_h)\cdot\mathbf{n}\,ds\\
&\quad+\frac{1}{\gamma_A}\int_K\big[(\mathbf{u}_h-\mathbf{g}_h(t_k))+(\mathbf{g}_h(t_k)-\mathbf{g}(t_k))\big]\cdot(\mathbf{w}-\mathbf{w}_h)\,\chi_{\Omega_D}\,d\mathbf{x}\Big\}\Big|,
\end{aligned}$$

where $\chi_\varpi$ denotes the characteristic function of the generic set $\varpi\subset\Omega$. To preserve the directional information, we now deal with the terms on the right-hand side componentwise. For this purpose, we define

$$a(v_h;\mathbf{u}_h,\mathbf{w}-\mathbf{w}_h)=\sum_{i=1}^{2}a_i(v_h;\mathbf{u}_h,w_i-w_{h,i}),$$

with $\mathbf{w}_h=(w_{h,1},w_{h,2})^T$, and

$$\begin{aligned}
&a_i(v_h;\mathbf{u}_h,w_i-w_{h,i})=\\
&\sum_{K\in\mathscr{T}_h}\Big\{\int_K -2v_h\sigma_i(\mathbf{u}_h)\cdot\nabla v_h(w_i-w_{h,i})\,d\mathbf{x}+\int_{\partial K}(v_h^2+\eta)\sigma_i(\mathbf{u}_h)\cdot\mathbf{n}(w_i-w_{h,i})\,ds\\
&+\frac{1}{\gamma_A}\int_K\big[(u_{h,i}-g_{h,i}(t_k))+(g_{h,i}(t_k)-g_i(t_k))\big](w_i-w_{h,i})\,\chi_{\Omega_D}\,d\mathbf{x}\Big\}.
\end{aligned}$$

Thanks to Hölder and Cauchy–Schwarz inequalities and definition (11), we obtain

$$\begin{aligned}
&\big|a_i(v_h;\mathbf{u}_h,w_i-w_{h,i})\big|\le\sum_{K\in\mathscr{T}_h}\Big\{\|2v_h\sigma_i(\mathbf{u}_h)\cdot\nabla v_h\|_{L^2(K)}\,\|w_i-w_{h,i}\|_{L^2(K)}\\
&+\tfrac{1}{2}\|[\![\sigma_i(\mathbf{u}_h)]\!]\|_{L^\infty(\partial K)}\|v_h^2+\eta\|_{L^2(\partial K)}\,\|w_i-w_{h,i}\|_{L^2(\partial K)}\\
&+\tfrac{1}{\gamma_A}\|(w_i-w_{h,i})\,\chi_{\Omega_D}\|_{L^2(K)}\\
&\Big(\|(u_{h,i}-g_{h,i}(t_k))\chi_{\Omega_D}\|_{L^2(K)}+\|(g_{h,i}(t_k)-g_i(t_k))\chi_{\Omega_D}\|_{L^2(K)}\Big)\Big\}.
\end{aligned}$$

Picking $w_{h,i} = \mathscr{C}_h(w_i)$ and thanks to Lemma 1, we obtain

$$\begin{aligned}
&\big|a_i(v_h; \mathbf{u}_h, w_i - w_{h,i})\big| \le C \sum_{K\in\mathscr{T}_h} \Big\{ \|2v_h\sigma_i(\mathbf{u}_h)\cdot\nabla v_h\|_{L^2(K)} \\
&+\tfrac{1}{2}\|[\![\sigma_i(\mathbf{u}_h)]\!]\|_{L^\infty(\partial K)}\|v_h^2 + \eta\|_{L^2(\partial K)} \left(\tfrac{h_K}{\lambda_{1,K}\lambda_{2,K}}\right)^{1/2} \\
&+\tfrac{\delta_{K,\Omega_D}}{\gamma_A}\big(\|u_{h,i} - g_{h,i}(t_k)\|_{L^2(K)} + \|g_{h,i}(t_k) - g_i(t_k)\|_{L^2(K)}\big)\Big\} \\
&\left[\sum_{j=1}^{2}\lambda_{j,K}^2(\mathbf{r}_{j,K}^T G_{\Delta_K}(w_i)\mathbf{r}_{j,K})\right]^{1/2}.
\end{aligned} \tag{15}$$

Now we deal with the second term on the right-hand side of (14), that we bound as

$$\begin{aligned}
&|a(v_h; \mathbf{u}_h, \mathbf{w}_h) - a_h(v_h; \mathbf{u}_h, \mathbf{w}_h)| \le \left|\int_\Omega \big[v_h^2 - P_h(v_h^2)\big]\sigma(\mathbf{u}_h) : E(\mathbf{w}_h)\, d\mathbf{x}\right| \\
&+\frac{1}{\gamma_A}\left|\int_{\Omega_D}(I - P_h)\big((\mathbf{u}_h - \mathbf{g}_h(t_k))\cdot\mathbf{w}_h\big)\, d\mathbf{x}\right| + \frac{1}{\gamma_A}\left|\int_{\Omega_D}\big(\mathbf{g}_h(t_k) - \mathbf{g}(t_k)\big)\cdot\mathbf{w}_h\, d\mathbf{x}\right|.
\end{aligned} \tag{16}$$

We anticipate the auxiliary result based on the equivalence of norms on a finite-dimensional space,

$$|\varphi_h\psi_h|_{H^2(K)} \le 4\,|\varphi_h|_{W^{1,\infty}(K)}\,\|\nabla\psi_h\|_{L^2(K)} \qquad \forall\varphi_h, \psi_h \in X_h, \quad \forall K \in \mathscr{T}_h, \tag{17}$$

which follows by straightforward calculus. Using the definition (7) of $\mathbf{g}_h(t_k)$, the last term in (16) turns out to be zero. Considering again (16) componentwise, employing Hölder and Cauchy–Schwarz inequalities together with the standard isotropic estimate for the $L^2$-norm of the interpolation error associated with $P_h$, we get

$$\begin{aligned}
&|a_i(v_h; \mathbf{u}_h, \mathbf{w}_h) - a_{i,h}(v_h; \mathbf{u}_h, \mathbf{w}_h)| \le C \sum_{K\in\mathscr{T}_h} \Big\{ \tfrac{|K|^{1/2}h_K^2}{\gamma_A}\,|(u_{h,i} - g_{h,i}(t_k))w_{h,i}|_{H^2(K)} \\
&+\|v_h^2 - P_h(v_h^2)\|_{L^\infty(K)}\,\|\sigma_i(\mathbf{u}_h)\|_{L^2(K)}\,\|\nabla w_{h,i}\|_{L^2(K)}\Big\},
\end{aligned}$$

where the constant $C$ does not depend on the aspect ratio $s_K$ of $K$. Then, we employ (17) together with estimate (8) and Lemma 2 with $\beta_1 = \lambda_{1,K}^2$, $\beta_2 = \lambda_{2,K}^2$, to obtain

$$
\begin{aligned}
&|a_i(v_h; u_h, \mathbf{w}_h) - a_{i,h}(v_h; u_h, \mathbf{w}_h)| \leq C \sum_{K\in\mathscr{T}_h} \Big\{ \Big( \tfrac{|K|^{1/2}\, h_K^2}{\gamma_A} \, |u_{h,i} - g_{h,i}(t_k)|_{W^{1,\infty}(K)} \\
&\quad + \|v_h^2 - P_h(v_h^2)\|_{L^\infty(K)} \, \|\sigma_i(\mathbf{u}_h)\|_{L^2(K)} \Big) \|\nabla w_{h,i}\|_{L^2(K)} \Big\} \\
&\leq C \sum_{K\in\mathscr{T}_h} \Big\{ \Big( \tfrac{|K|^{1/2}\, h_K^2}{\gamma_A} \, |u_{h,i} - g_{h,i}(t_k)|_{W^{1,\infty}(K)} \\
&\quad + \|v_h^2 - P_h(v_h^2)\|_{L^\infty(K)} \, \|\sigma_i(\mathbf{u}_h)\|_{L^2(K)} \Big) \Big( \|\nabla w_{h,i} - \nabla w_i\|_{L^2(K)} + \|\nabla w_i\|_{L^2(K)} \Big) \Big\} \\
&\leq C \sum_{K\in\mathscr{T}_h} \Big\{ \Big( \tfrac{|K|^{1/2}\, h_K^2}{\gamma_A} \, |u_{h,i} - g_{h,i}(t_k)|_{W^{1,\infty}(K)} \\
&\quad + \|v_h^2 - P_h(v_h^2)\|_{L^\infty(K)} \, \|\sigma_i(\mathbf{u}_h)\|_{L^2(K)} \Big) \tfrac{1}{\lambda_{2,K}} \Big[ \sum_{j=1}^{2} \lambda_{j,K}^2 (\mathbf{r}_{j,K}^T G_{\Delta_K}(w_i) \mathbf{r}_{j,K}) \Big]^{1/2} \Big\}.
\end{aligned} \tag{18}
$$

Therefore, collecting (15) and (18), we are able to bound componentwise the first term on the right-hand side of (13), as

$$
|a(v_h; \mathbf{u}_h, \mathbf{w})| \leq C \sum_{K\in\mathscr{T}_h} \sum_{i=1}^{2} \rho_{i,K}^A(v_h, \mathbf{u}_h)\, \omega_K^A(w_i).
$$

The estimate of the second term on the right-hand side of (13) can be carried out exactly as the corresponding one in the proof of Proposition 3.3 in [5], after replacing $|\nabla u_h|^2$ with $\sigma(\mathbf{u}_h) : E(\mathbf{u}_h)$. This yields

$$
4|b(\mathbf{u}_h; v_h, z)| \leq C \sum_{K\in\mathscr{T}_h} \rho_K^B(\mathbf{u}_h, v_h)\, \omega_K(z).
$$

□

To make estimate (10) useful in practice, we have to pick the pair of functions $(\mathbf{w}, z)$. Mimicking the considerations in [5], we choose $\mathbf{w} = \mathbf{u}_h$ and $z = v_h$. This leads us to define the error estimator

$$
\eta = \sum_{K\in\mathscr{T}_h} \eta_K(\mathbf{u}_h, v_h),
$$

where the local estimator is

$$
\eta_K(\mathbf{u}_h, v_h) = \sum_{i=1}^{2} \rho_{i,K}^A(v_h, \mathbf{u}_h)\, \omega_K(u_{h,i}) + \rho_K^B(\mathbf{u}_h, v_h)\, \omega_K(v_h). \tag{19}
$$

*Remark 2* Although in this work we deal with a specific case of linear elasticity constitutive law, we do believe that it is possible to extend the a posteriori analysis to a more general model, for instance, the one recently introduced in [9].

## 3 The Numerical Anisotropic Procedure

The numerical minimization of (6) is not a trivial task since it is a nonconvex functional due to the presence of the term $P_h(v_h^2)\sigma(\mathbf{u}_h) : E(\mathbf{u}_h)$. In particular, the methods available in the literature do not guarantee, in general, the convergence to global minimizers (see, e.g., [3]) but only to local minima.

In the first part of this section, we introduce the procedure exploited to convert the anisotropic estimator (19) into an actual anisotropic tool. In the second part of this section, we merge this approach with a suitable minimization algorithm, extending the method in [5].

### 3.1 A Metric-Driven Approach

Following [15, 25], we use a metric-based mesh adaptive approach (see, e.g., [20]). In particular, we predict the mesh with the least number of elements ensuring a given accuracy on the global estimator $\eta$.

There exists a tight relation between metric and mesh. Actually, with an assigned mesh $\mathcal{T}_h$, we can associate a corresponding piecewise constant metric given by $M_K = R_K^T \Lambda_K^{-2} R_K$, for any $K \in \mathcal{T}_h$, where matrices $R_K$ and $\Lambda_K$ are exactly the same as in Sect. 2.2. Likewise, for a given metric field $\mathcal{M} : \Omega \to \mathbb{R}^{2\times 2}$, we can build a mesh, say $\mathcal{T}_{\mathcal{M}}$, such that $\mathcal{M}_K = \mathcal{M}|_K$ coincides with $M_K$, for any $K \in \mathcal{T}_{\mathcal{M}}$.

To build the new adapted mesh, we adopt a two-step procedure. First, we derive a metric $\mathcal{M}$ out of the error estimator (19). Then, we generate the new mesh induced by this metric using the metric-based mesh generator in `FreeFem++` [21].

To obtain $\mathcal{M}$, we resort to an iterative procedure. At each iteration, say $l$, we deal with three quantities:

(i) The actual mesh $\mathcal{T}_h^{(l)}$.
(ii) The new metric $\mathcal{M}^{(l+1)}$ computed on $\mathcal{T}_h^{(l)}$.
(iii) The updated mesh $\mathcal{T}_h^{(l+1)}$ induced by $\mathcal{M}^{(l+1)}$.

The new metric is predicted by suitably rewriting the local estimator $\eta_K(\mathbf{u}_h, v_h)$ to single out the geometric information and then by applying an error equidistribution criterion combined with the minimization of the number of elements. The re-arranged local estimator is

$$\eta_K(\mathbf{u}_h, v_h) = \mu_K \left\{ \sum_{i=1}^{2} \overline{\rho}_{i,K}^{A}(v_h, \mathbf{u}_h)\, \overline{\omega}_K(u_{h,i}) + \overline{\rho}_K^{B}(\mathbf{u}_h, v_h)\, \overline{\omega}_K(v_h) \right\}, \tag{20}$$

where $\mu_K = |\hat{K}| \left(\lambda_{1,K}\lambda_{2,K}\right)^{3/2}$ lumps all the area $|K|$ information,

$$\overline{\rho}_{i,K}^{A}(v_h, \mathbf{u}_h) = \frac{\rho_{i,K}^{A}(v_h, \mathbf{u}_h)}{\left(|\hat{K}|\lambda_{1,K}\lambda_{2,K}\right)^{1/2}}, \quad \overline{\rho}_K^{B}(\mathbf{u}_h, v_h) = \frac{\rho_K^{B}(\mathbf{u}_h, v_h)}{\left(|\hat{K}|\lambda_{1,K}\lambda_{2,K}\right)^{1/2}},$$

with $i = 1, 2$, are approximately pointwise values (at least for a sufficiently fine mesh), while the anisotropic information associated with $K$ is collected in the scaled weights

$$\overline{\omega}_K(\xi_h) = \left[ s_K\, \mathbf{r}_{1,K}^T\, \overline{G}_{\Delta_K}(\xi_h)\, \mathbf{r}_{1,K} + \frac{1}{s_K}\, \mathbf{r}_{2,K}^T\, \overline{G}_{\Delta_K}(\xi_h)\, \mathbf{r}_{2,K} \right]^{1/2} \text{ with } \quad \xi_h = u_{h,1}, u_{h,2}, v_h,$$

with $\overline{G}_{\Delta_K}(\cdot) = G_{\Delta_K}(\cdot)/(|\hat{K}|\, \lambda_{1,K}\lambda_{2,K})$. In principle, each term in (20) provides a metric. For practical reasons, however, we merge this information to obtain a single metric, thus avoiding metric intersection. To do this, we follow the approach in Sect. 4 of [26], which allows us to rewrite (20) as $\eta_K(\mathbf{u}_h, v_h) = \mu_K \Upsilon_K$ with

$$\Upsilon_K = \left[ s_K\, \mathbf{r}_{1,K}^T\, \Gamma_K\, \mathbf{r}_{1,K} + \frac{1}{s_K}\, \mathbf{r}_{2,K}^T\, \Gamma_K\, \mathbf{r}_{2,K} \right]^{1/2}, \tag{21}$$

where the local matrix

$$\Gamma_K = \sum_{i=1}^{2} \left[\overline{\rho}_{i,K}^{A}(v_h, \mathbf{u}_h)\right]^2 \overline{G}_{\Delta_K}(u_{h,i}) + \left[\overline{\rho}_K^{B}(\mathbf{u}_h, v_h)\right]^2 \overline{G}_{\Delta_K}(v_h) \tag{22}$$

gathers the anisotropic information provided by $\mathbf{u}_h$ and $v_h$, suitably weighted via the local residuals.

We minimize now the number of mesh elements by maximizing the area of each element $K$ with an error *equidistribution* constraint, i.e., we enforce that, for each element $K \in \mathscr{T}_h^{(l+1)}$, $\eta_K(\mathbf{u}_h, v_h) = \mu_K \Upsilon_K = \mathtt{TOL}/\#\mathscr{T}_h^{(l)}$, where $\mathtt{TOL}$ and $\#\mathscr{T}_h^{(l)}$ are the user-defined global tolerance and the number of mesh elements in $\mathscr{T}_h^{(l)}$, respectively. The constant value $\mathtt{TOL}/\#\mathscr{T}_h^{(l)}$ is ensured with an element of maximal area only if $\Upsilon_K$ is minimized with respect to $s_K$ and $\mathbf{r}_{1,K}$, i.e., we solve elementwise the constrained minimization problem

$$\min_{s_K \geq 1, \mathbf{r}_{m,K} \cdot \mathbf{r}_{n,K} = \delta_{mn}} \Upsilon_K(\mathbf{r}_{1,K}, s_K), \tag{23}$$

$\delta_{mn}$ being the Kronecker symbol. For computational convenience, all the quantities appearing in (22) are evaluated on the background grid $\mathcal{T}_h^{(l)}$. On the other hand, the aspect ratio $s_K$ and the unit vector $\mathbf{r}_{1,K}$ in (21) represent our actual unknowns. According to Proposition 4.2 in [26], we can state the desired minimization result.

**Proposition 2** *Let $\{\boldsymbol{\gamma}_{i,K}, g_{i,K}\}$ be the eigenvector-eigenvalue pair of $\Gamma_K$ with $g_{1,K} \geq g_{2,K} > 0$. Then, the minimum (23) is obtained for the choices*

$$\mathbf{r}_{1,K} = \boldsymbol{\gamma}_{2,K} \quad \textit{and} \quad s_K = \left(\frac{g_{1,K}}{g_{2,K}}\right)^{1/2}, \tag{24}$$

*yielding the value $\left(2\sqrt{g_{1,K}g_{2,K}}\right)^{1/2}$ for $\Upsilon_K$.*

The minimization problem (23) can be solved analytically via (24) without resorting to any numerical optimization tool.

Finally, the optimal metric $\mathcal{M}^{(l+1)}$ is generated by exploiting again the equidistribution constraint, i.e., by solving the equations

$$|\hat{K}| \left(\lambda_{1,K}\lambda_{2,K}\right)^{3/2} \left(2\sqrt{g_{1,K}g_{2,K}}\right)^{1/2} = \frac{\mathtt{TOL}}{\#\mathcal{T}_h^{(l)}} \quad \text{and} \quad \frac{\lambda_{1,K}}{\lambda_{2,K}} = s_K = \left(\frac{g_{1,K}}{g_{2,K}}\right)^{1/2}. \tag{25}$$

System (25) provides us with the distinct values

$$\begin{aligned} \lambda_{1,K} &= \left(\frac{1}{|\hat{K}|\sqrt{2}}\left(\frac{g_{1,K}}{g_{2,K}^2}\right)^{1/2} \frac{\mathtt{TOL}}{\#\mathcal{T}_h^{(l)}}\right)^{1/3}, \\ \lambda_{2,K} &= \left(\frac{1}{|\hat{K}|\sqrt{2}}\left(\frac{g_{2,K}}{g_{1,K}^2}\right)^{1/2} \frac{\mathtt{TOL}}{\#\mathcal{T}_h^{(l)}}\right)^{1/3}. \end{aligned} \tag{26}$$

Eventually, the optimal metric $\mathcal{M}^{(l+1)}$ is characterized by $\mathbf{r}_{1,K}$ in (24), $\lambda_{1,K}$ and $\lambda_{2,K}$ in (26), with $\mathbf{r}_{2,K} \perp \mathbf{r}_{1,K}$.

## *3.2 The Whole Adaptive Procedure*

In this section we propose a numerical algorithm which combines a suitable minimization method for the nonconvex functional $J_h^{PAT}$ with the mesh adaptation procedure of the previous section.

The algorithm is a generalization of the Algorithms 2 and 3 proposed in [5]. In practice, we switch from mesh adaptation, driven by the tolerance $\mathtt{TOL} = \mathtt{REFTOL} \ll 1$, to minimization of $J_h^{PAT}$, until both the mesh and the functional stagnate to within given thresholds, $\mathtt{MESHTOL} \ll 1$ and $\mathtt{VTOL} \ll 1$, respectively.

The minimization of the functional exploits the alternate minimization algorithm proposed in [8] for dealing with nonconvex functionals, relying on the convexity only along the directions identified by $\mathbf{u}_h$ and $v_h$. In particular, our new algorithm carries out mesh adaptation after a maximum number, `nMIN`, of minimization steps. Given an initial mesh, $\mathscr{T}_h^{(0)}$, we proceed as follows:

**Algorithm 1** Optimize(`nMIN`)-while-adapt

```
1:  Set $k = 0, l = 0$;
2:  If $k = 0$, set $v_h^1 = 1$; else $v_h^1 = v_h(t_{k-1})$;
3:  Set $l = 0$; errmesh= 1; err= 1;
4:  while errmesh $\geq$ MESHTOL | err $\geq$ VTOL do
5:      Set $i = 1$; err=1;
6:      while err $\geq$ VTOL & $i \leq$ nMIN do
7:          $\mathbf{u}_h^i = \underset{\mathbf{z}_h \in [X_h^{(l)}]^2}{\text{argmin}}\ J_h^{PAT}(\mathbf{z}_h, v_h^i)$;
8:          $v_h^{i+1} = \underset{z_h \in X_h^{(l)}}{\text{argmin}}\ J_h^{PAT}(\mathbf{u}_h^i, z_h)$;
9:          err $= \|v_h^{i+1} - v_h^i\|_{L^\infty(\Omega)}$;
10:         $i \leftarrow i + 1$;
11:     end while
12:     Compute the new metric $\mathscr{M}^{(l+1)}$ based on $\mathbf{u}_h^{i-1}$ and $v_h^i$;
13:     Build the adapted mesh $\mathscr{T}_h^{(l+1)}$;
14:     errmesh $= |\#\mathscr{T}_h^{(l+1)} - \#\mathscr{T}_h^{(l)}|/\#\mathscr{T}_h^{(l)}$;
15:     Set $v_h^1 = \Pi_{l\to l+1}(v_h^i)$;
16:     $l \leftarrow l + 1$;
17: end while
18: $\mathbf{u}_h(t_k) = \Pi_{l-1\to l}(\mathbf{u}_h^{i-1})$; $v_h(t_k) = \Pi_{l-1\to l}(v_h^i)$; $\mathscr{T}_h^k = \mathscr{T}_h^{(l)}$;
19: Set $\mathscr{T}_h^{(0)} = \mathscr{T}_h^k$;
20: $k \leftarrow k + 1$;
21: if $k > F$, stop; else goto 2.
```

The minimization of the functional with respect to $\mathbf{u}_h$ and $v_h$ is performed by solving the corresponding Euler-Lagrange equations, since the functional is actually (strictly) convex with respect to the individual variables. In both cases, the equations are standard linear elliptic problems.

The interpolation operator $\Pi_{n\to n+1}(z_h)$ is used to map the finite element function $z_h$ defined on $\mathscr{T}_h^n$ onto the new mesh $\mathscr{T}_h^{n+1}$, before restarting any new optimization or time loop.

The convergence of the mesh adaptivity is assessed by checking the relative variation of the number of elements. The main novelty with respect to the algorithms in [5] is that, through `nMIN`, the functional $J_h^{PAT}$ is not necessarily exactly minimized after the inner **while** loop. Algorithms 2 and 3 represent particular cases of the algorithm above. Selecting `nMIN` $= \infty$, we recover Algorithm 2, which is suited to deal with slowly advancing fractures, because the coupling between optimization and adaptation is not so tight. Setting `nMIN` $= 1$, we get back Algorithm 3, which alternates optimization and mesh adaptation more closely.

However, in such a case, the crack evolution may be biased by the mesh which is adapted to nonoptimal fields, $\mathbf{u}_h, v_h$. These values of `nMIN` represent two extreme choices. In general, we may pick any intermediate value, e.g., `nMIN` $= 7$ in the section below.

## 4 Numerical Assessment

We verify Algorithm 1 on two numerical tests inspired by [7, 16]. The second test case turns out to be particularly challenging.

### 4.1 Traction of a Fiber-Reinforced Matrix

We consider the rectangular domain $\Omega = (0, 3) \times (0, 3.5)$ in Fig. 2 left, comprising a nonelastic circular fiber of radius 0.5 centered at $(1.5, 1.5)$, for $t \in [0, 0.5]$, uniformly partitioned with a total number of $F = 50$ time steps. On the subdomain $\Omega_D = (0, 3) \times (3, 3.5)$ we enforce the load $\mathbf{g}$, with $\mathbf{g}_D = (0, t)^T$. The fiber is held fixed while a uniform vertical displacement is induced by $\mathbf{g}_D$ on the top side of the matrix. The other sides of the domain are traction-free. As a function of time, at the beginning the matrix behaves elastically; then, an asymmetric crack suddenly

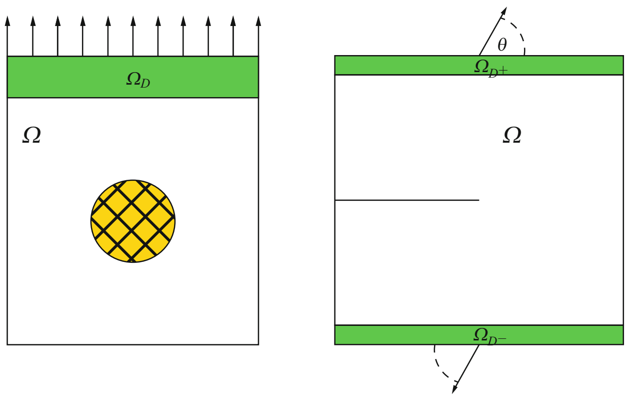


**Fig. 2** Geometric configurations for the traction of a fiber-reinforced matrix (*left*) and for the crack branching test (*right*)

develops and eventually cuts the matrix in two parts. The parameters involved in (3) are set to

$$\varepsilon = 10^{-1}, \quad \eta = 10^{-3}, \quad \gamma_A = \gamma_B = 10^{-7},$$
$$\lambda = \frac{Yp}{(1+p)(1-2p)} \quad \mu = \frac{Y}{2(1+p)},$$

where $Y = 30$ is Young's modulus and $p = 0.18$ is the Poisson coefficient. The values of the tolerances required by Algorithm 1 are

$$\mathtt{VTOL} = 5 \cdot 10^{-3}, \quad \mathtt{CRTOL} = \mathtt{REFTOL} = 10^{-3}, \quad \mathtt{MESHTOL} = 10^{-2}.$$

Figure 3 shows the $v_h$-field at three time levels as well as the associated anisotropic adapted mesh. At time $t = 0.25$ a crack on top of the fiber is created and starts propagating slowly and symmetrically with respect to the fiber. At time $t = 0.35$ the symmetry is broken and the crack splits the matrix on one side only. Afterwards, at time $t = 0.39$, the domain is thoroughly split into two parts. This behavior is not essentially affected by $\varepsilon$. Actually, a reduction of this parameter by one order of magnitude yields the results in Fig. 4, which share the same pattern as in Fig. 3, although with a sharper crack. In all cases, the adapted meshes are very fine close to the fracture and in the area of higher stress. Moreover, the correct path of the crack is detected in a very efficient way, i.e., with quite few elements. In particular, in Figs. 3 and 4 (bottom-right), the meshes consist only of 1,810 and 12,381 elements, respectively. The maximum aspect ratio of the three meshes in Fig. 3 is 16, 32 and 109. Figure 5 shows the time evolution of the energy. The elastic energy (dashed line) is associated with the first term in the integral over $\Omega$ in (6), while the fictitious crack energy (dash-dotted line) represents the second term. The black line is the sum of these two contributions. Theoretically, we expect the elastic energy to disappear after the collapse of the domain. On the contrary, a residual energy remains, due to the regularization parameter $\eta$ in the model. Moreover, three sudden increases of the crack energy occur: the first at time $t = 0.24$, when a finite-length crack appears on top of the fiber; the second at time $t = 0.37$, when the domain breaks on one side; and the last takes place when the domain breaks down, at $t = 0.39$. This behavior is qualitatively comparable with the ones in Fig. 4 in [16] and in Fig. 3 in [7]. This corroborates the fact that anisotropic meshes do not affect the crack dynamics.

## 4.2 Crack Branching

The domain for the second test case is the cracked rectangular elastic sample shown in Fig. 2, right. The initial crack is horizontal and parallel to the upper end lower sides of the sample, while a displacement field of increasing magnitude and fixed orientation, $\theta$, to the $x_1$-axis, is applied to the horizontal sides. The later crack

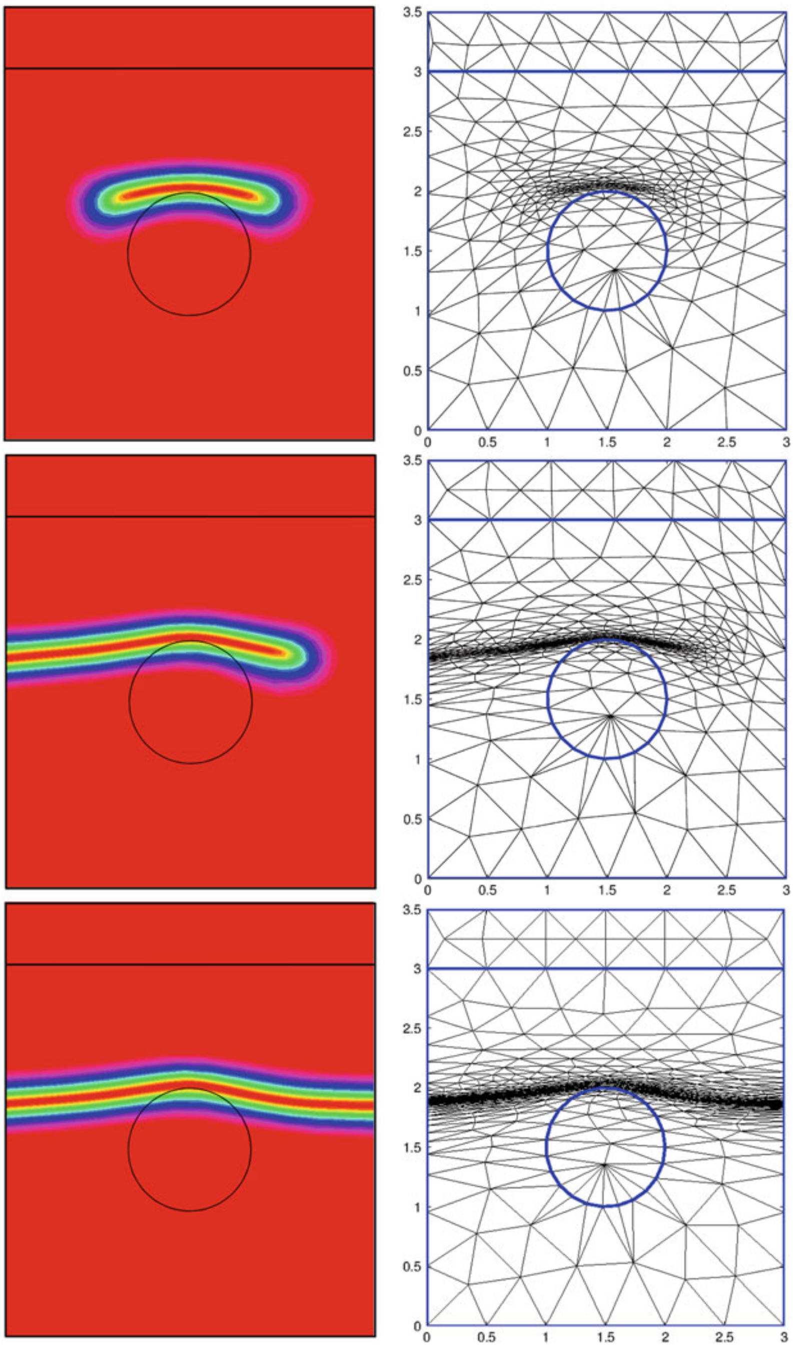


**Fig. 3** Traction of a fiber-reinforced matrix. Time evolution of the $v_h$-field (*left*): $t = 0.25$ (*top*), $t = 0.35$ (*center*), and $t = 0.39$ (*bottom*); corresponding adapted meshes (*right*) with $\varepsilon = 10^{-1}$

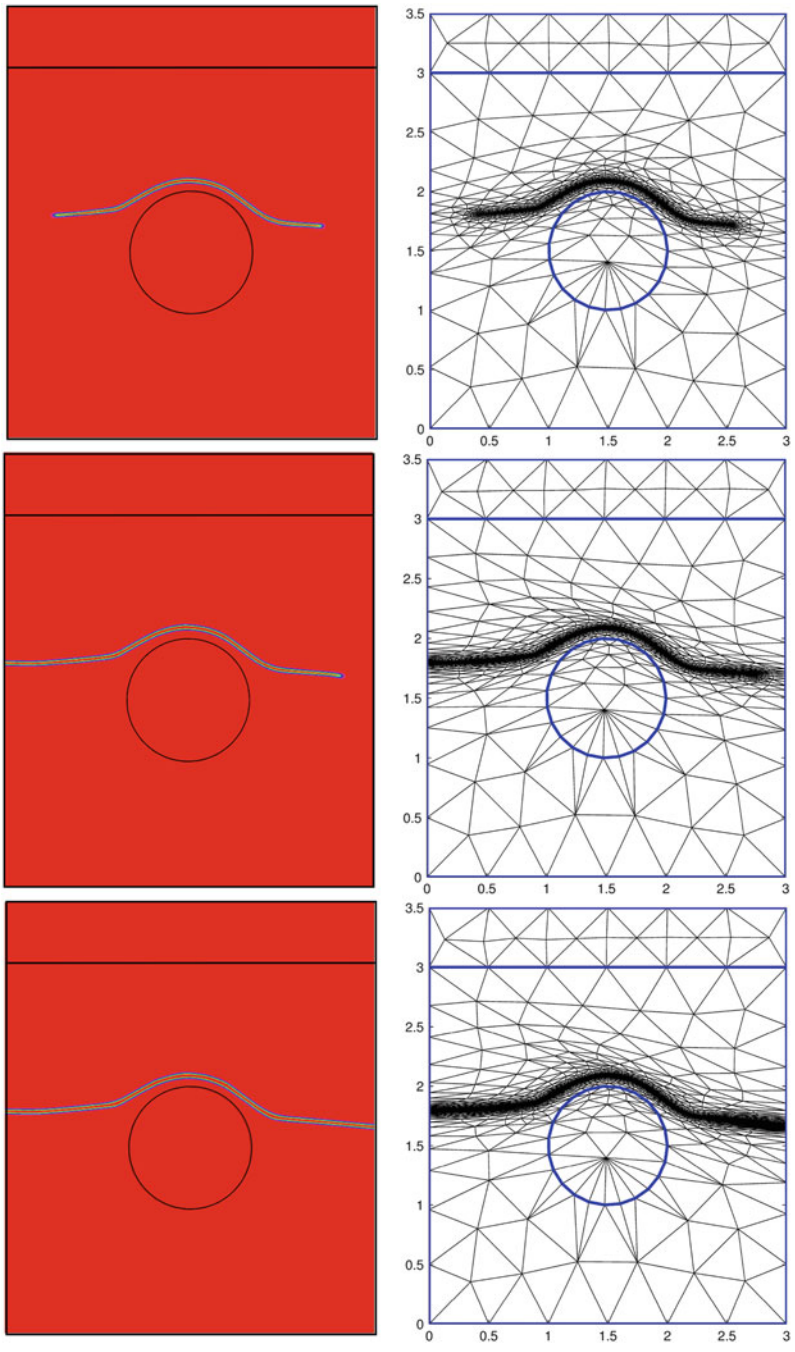


**Fig. 4** Traction of a fiber-reinforced matrix. Time evolution of the $v_h$-field (*left*): $t = 0.30$ (*top*), $t = 0.38$ (*center*), and $t = 0.40$ (*bottom*); corresponding adapted meshes (*right*) with $\varepsilon = 10^{-2}$

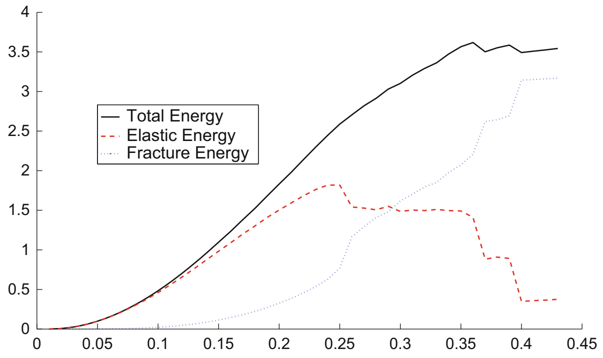


**Fig. 5** Traction of a fiber-reinforced matrix. Time evolution of the energy

evolution is monitored for several values of $\theta$. The final time is set to $T = 0.2$, and the total number of uniform time steps is $F = 20$. The final time is chosen when the crack is about to turn towards the bottom right corner of the domain. The key issues of this problem is the correct prediction of the actual branching angle of the crack, in particular when the applied displacement field is not orthogonal to the domain border. For this purpose, we resort to a suitable mesh adaptation strategy. In particular, we identify $\Omega$ with the square domain $(-1.5, 1.5)^2$, $\Omega_D = \Omega_{D^-} \cup \Omega_{D^+}$ with $\Omega_{D^-} = (-1.5, 1.5) \times (-1.5, -1.3)$ and $\Omega_{D^+} = (-1.5, 1.5) \times (1.3, 1.5)$, $\mathbf{g}_D$ is

$$\mathbf{g}_D(t) = \begin{cases} (t\cos(\theta), t\sin(\theta)) & \text{on } \Omega_{D^+} \\ (-t\cos(\theta), -t\sin(\theta)) & \text{on } \Omega_{D^-} \end{cases} \tag{27}$$

and the model parameters are

$$\varepsilon = 10^{-2}, \quad \eta = 10^{-5}, \quad \gamma_A = \gamma_B = 10^{-5},$$
$$\lambda = \frac{Yp}{(1+p)(1-2p)}, \quad \mu = \frac{Y}{2(1+p)},$$

with $Y = 45$ and $p = 0.18$. The tolerances of Algorithm 1 are

$$\texttt{VTOL} = 10^{-4}, \quad \texttt{CRTOL} = 3 \cdot 10^{-4}, \quad \texttt{REFTOL} = 10^{-3}, \quad \texttt{MESHTOL} = 10^{-2}.$$

Figure 6 gathers the $v_h$-field and the corresponding anisotropic adapted mesh at the final time, for several orientations $\theta$. The cardinality of the meshes in Fig. 6 is 2,941, 1,268, 1,652, 1,302, 1,570, 3,804, in top-down order. Notice that the mesh adaptive procedure identifies the configurations associated with $\theta = \pi/2$ and $\theta = 0$ as being the most challenging. In all cases, the mesh closely matches the crack path, with a very thin thickness of the adapted area. The anisotropic features of the meshes are highlighted by the values of the maximum aspect ratio, which varies between 28, for $\theta = \pi/20$, and 384, for $\theta = 0$. Moreover, when $\theta = 0$, in contrast to [7], where it appears an unphysical symmetric crack branching, we obtain a crack

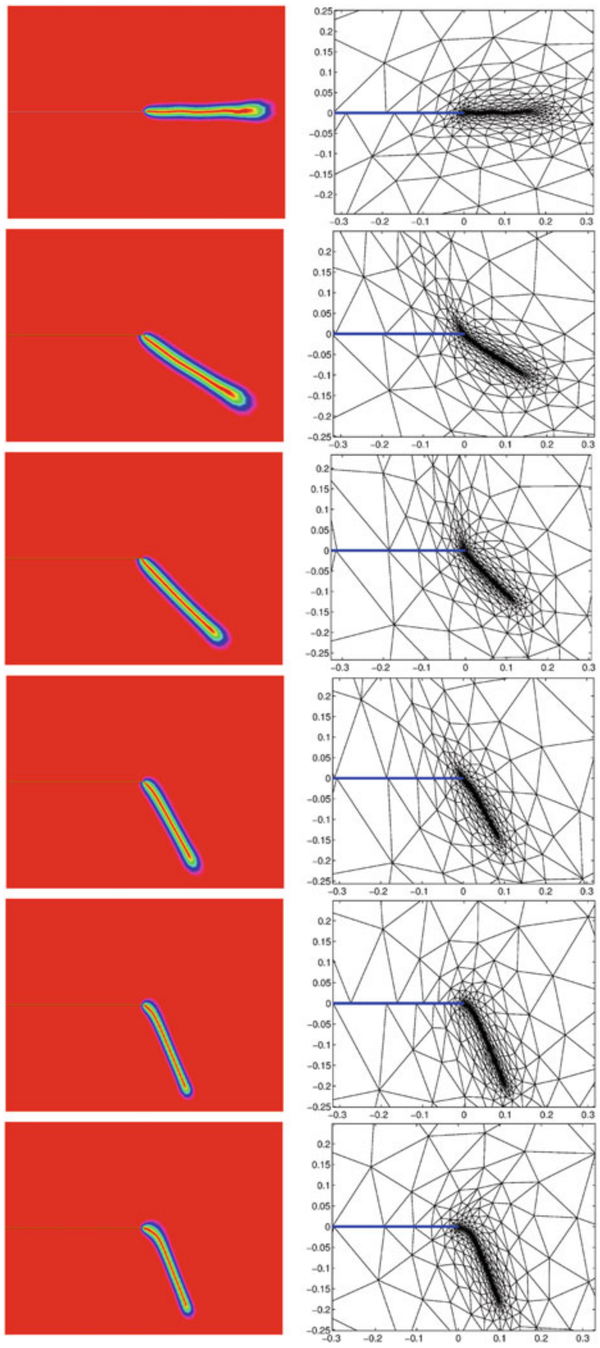


**Fig. 6** Crack branching. Distribution of the $v_h$-field around the tip of the initial crack (*left*) and final adapted mesh (*right*) for $\theta = \pi/2, \pi/4, \pi/6, \pi/20, \pi/60, 0$ (*top-down*)

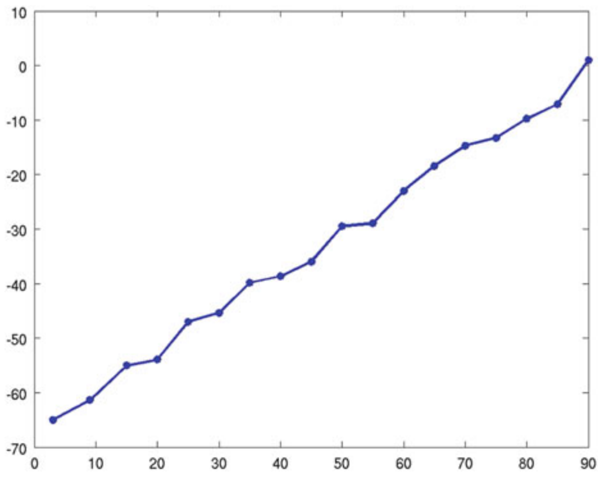


**Fig. 7** Crack branching. Branching angle as a function of the impressed displacement orientation

which moves straight a very short distance, before turning downwards but with a slightly smaller angle than expected. In practice, we are able to predict reliably the crack branching for $\theta \gtrsim 3°$. Figure 7 shows the branching angle as a function of the orientation $\theta$. This angle has been computed by picking the angle at which the distribution of the unit vectors, $\mathbf{r}_{1,K}$, gathered in bins of 20 angles each, over the rectangle $[0, 0.08] \times [-0.08, 0]$ is a maximum. On comparing our results with the ones in [7], we observe a good agreement, with the additional capability of correctly simulating the physical behavior for $3° \lesssim \theta \lesssim 7°$, by enlarging the range of reliability of the numerical tool in [7] where $\theta \gtrsim 7°$.

## 5 Conclusions

We have extended the anisotropic approach provided in [5] for the anti-plane case to the more challenging plane-strain framework. This implies moving from a scalar to a vector elastic problem. The proposed Algorithm 1 has been shown to correctly identifying the physical crack path, under reasonable choices of the physical and algorithmic parameters, aware also of the theoretical limits of the adopted mechanical model. In particular, in the crack branching test case, the proposed procedure allowed us to broaden the range of applicability of this model, with respect to what studied in [7]. Another interesting issue to be investigated is a proper tuning of the modeling parameters, such as $\varepsilon$, $\eta$, and also of the physical parameters $\lambda$ and $\mu$. In Sect. 4.1, we tackle to a some extent the sensitivity to $\varepsilon$ by highlighting the actual influence of $\varepsilon$ on the crack thickness. A more thorough investigation has been carried out in [4] in the anti-plane case. We have also introduced a generalized version of the algorithm proposed in [5]. In particular, Algorithm 1 employs the new parameter, namely nMIN, through which we can adjust in a more precise way the interplay between the minimization of the functional and the adaptation of the mesh. In future developments, we shall be concerned with the study of more general mathematical models, such as the ones introduced in [9], for a possible comparison with actual experimental tests.

## References

1. Ambrosio, L., Tortorelli, V.M.: Approximation of functionals depending on jumps by elliptic functionals via $\gamma$-convergence. Commun. Pure Appl. Math. **43**(8), 999–1036 (1990)
2. Amestoy, M.: Propagations de Fissures en Élasticité Plane. Thèse d'Etat, Paris (1987)
3. Artina, M., Fornasier, M., Solombrino, F.: Linearly constrained nonsmooth and nonconvex minimization. SIAM J. Optim. **23**(3), 1904–1937 (2013)
4. Artina, M., Fornasier, M., Micheletti, S., Perotto, S.: Anisotropic adaptive meshes for brittle fractures: parameter sensitivity. Numerical Mathematics and Advanced Applications ENUMATH 2013. Springer International Publishing 293–301 (2015)
5. Artina, M., Fornasier, M., Micheletti, S., Perotto, S.: Anisotropic mesh adaptation for crack detection in brittle materials. MOX Report 20/2014 (2014)
6. Border, M., Verhoosel, C., Scott, M., Hughes, T., Landis, C.: A phase-field description of dynamic brittle fracture. Comput. Methods Appl. Mech. Eng. **217–220**, 77–95 (2012)
7. Bourdin, B., Francfort, G., Marigo, J.J.: Numerical experiments in revisited brittle fracture. J. Mech. Phys. Solids **48**(4), 797–826 (2000)
8. Burke, S., Ortner, C., Süli, E.: An adaptive finite element approximation of a variational model of brittle fracture. SIAM J. Numer. Anal. **48**(3), 980–1012 (2010)
9. Burke, S., Ortner, C., Süli, E.: An adaptive finite element approximation of a generalized Ambrosio-Tortorelli functional. Math. Models Methods Appl. Sci. **23**(9), 1663–1697 (2013)
10. Chambolle, A., Dal Maso, G.: Discrete approximation of the Mumford-Shah functional in dimension two. M2AN Math. Model. Numer. Anal. **33**(4), 651–672 (1999)
11. Ciarlet, P.G.: The Finite Element Method for Elliptic Problems. North-Holland, Amsterdam (1978)
12. Ciarlet, P.G., Raviart, P.A.: Maximum principle and uniform convergence for the finite element method. Comput. Methods Appl. Mech. Eng. **2**(1), 17–31 (1973)
13. Clément, P.: Approximation by finite element functions using local regularization. RAIRO Anal. Numér. **2**, 77–84 (1975)
14. Dal Maso, G.: An Introduction to $\Gamma$-Convergence. Birkhäuser, Basel (1993)
15. Dedè, L., Micheletti, S., Perotto, S.: Anisotropic error control for environmental applications. Appl. Numer. Math. **58**(9), 1320–1339 (2008)
16. Del Piero, G., Lancioni, G., March, R.: A variational model for fracture mechanics: numerical experiments. J. Mech. Phys. Solids **55**, 2513–2537 (2007)
17. Formaggia, L., Perotto, S.: New anisotropic a priori error estimates. Numer. Math. **89**(4), 641–667 (2001)
18. Formaggia, L., Perotto, S.: Anisotropic error estimates for elliptic problems. Numer. Math. **94**, 67–92 (2003)
19. Francfort, G., Marigo, J.J.: Revisiting brittle fracture as an energy minimization problem. J. Mech. Phys. Solids **46**(8), 1319–1342 (1998)
20. George, P.L., Borouchaki, H.: Delaunay Triangulation and Meshing. Application to Finite Elements. Edition Hermès, Paris (1998)
21. Hecht, F.: New developments in Freefem++. J. Numer. Math. **20**(3–4), 251–265 (2012)

22. Korotov, S., Křížek, M., Neittaanmäki, P.: Weakened acute type condition for tetrahedral triangulations and the discrete maximum principle. Math. Comput. **70**(233), 107–119 (2001)
23. Lions, J.L., Magenes, E.: Non-homogeneous Boundary Value Problems and Applications, vol. I. Springer, Berlin (1972)
24. Micheletti, S., Perotto, S.: Reliability and efficiency of an anisotropic Zienkiewicz–Zhu error estimator. Comput. Methods Appl. Mech. Eng. **195**(9), 799–835 (2006)
25. Micheletti, S., Perotto, S.: Output functional control for nonlinear equations driven by anisotropic mesh adaption: the Navier-Stokes equations. SIAM J. Sci. Comput. **30**(6), 2817–2854 (2008)
26. Micheletti, S., Perotto, S.: The effect of anisotropic mesh adaptation on PDE-constrained optimal control problems. SIAM J. Control. Optim. **49**(4), 1793–1828 (2011)
27. Micheletti, S., Perotto, S., Picasso, M.: Stabilized finite elements on anisotropic meshes: a priori error estimates for the advection–diffusion and the Stokes problems. SIAM J. Numer. Anal. **41**(3), 1131–1162 (2003)
28. Miehe, C., Hofacker, M., Fabian, W.: A phase field model rate-independent crack propagation: robust algorithmic implementation based on operator splits. Comput. Methods Appl. Mech. Eng. **199**(45–48), 2765–2778 (2010)
29. Strang, G., Fix, G.J.: An Analysis of the Finite Element Method, vol. 212. Prentice-Hall, Englewood Cliffs (1973)
30. Thomée, V.: Galerkin Finite Element Methods for Parabolic Problems, vol. 25. Springer, Berlin (1997)

# Deforming Surface Meshes

**Siu-Wing Cheng and Jiongxin Jin**

**Abstract** We study the problem of maintaining a deforming surface mesh, specified only by a dense sample of $n$ points that move with the surface. We propose a motion model under which the class of $(\varepsilon, \alpha)$-meshes can be efficiently maintained by a combination of edge flips and insertion and deletion of vertices. We can enforce bounded aspect ratios and a small approximation error throughout the deformation.

## 1 Introduction

### *1.1 Background*

The simulation of deforming surfaces appears in various settings such as the interface between fluids [14, 18, 21, 28], boundary element methods [5, 15], moving cloth [4, 29], and surgery simulation [10, 19]. In this paper, we consider the simulation of a surface that deforms without changing its topology. The surface is specified only by a set of sample points dense with respect to the local feature size (LFS) and the goal is to approximate the surface by a mesh with vertices chosen from the sample points and triangles of bounded aspect ratio, the latter being a desirable feature for numerical simulation. The deformation may make the angles in the mesh smaller between two successive time steps. Our problem is to restore the mesh quality at the next time step before resuming the deformation. The left two images in Fig. 1 shows snapshots of a twisting cylinder output by our algorithm.

Some notation is needed to state our results. The Euclidean distance between two points $x$ and $y$ is denoted by $d(x, y)$. For all $Y \subseteq \mathbb{R}^3$, $d(x, Y) = \inf_{y \in Y} d(x, y)$. For every pair of vectors $\mathbf{u}$ and $\mathbf{v}$, $\angle(\mathbf{u}, \mathbf{v})$ denotes the angle between them which lies

S.-W. Cheng (✉)
Department of Computer Science & Engineering, HKUST, Clear Water Bay, Kowloon, Hong Kong
e-mail: scheng@cse.ust.hk

J. Jin
Google Inc., Mountain View, CA, USA
e-mail: jamesjjx@google.com

S. Perotto, L. Formaggia (eds.), *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, SEMA SIMAI Springer Series 5,
DOI 10.1007/978-3-319-06053-8_4

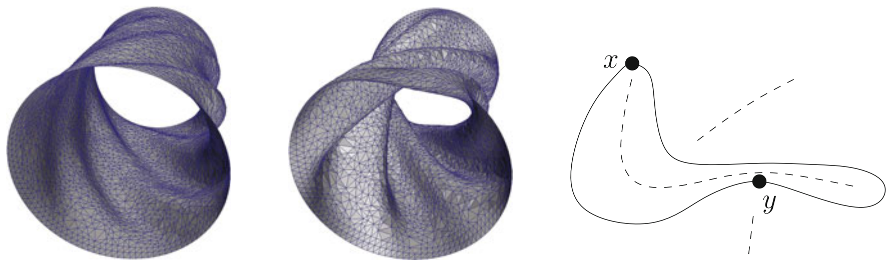


**Fig. 1** The *left two images* show a twisting cylinder. On the *right*, the *dashed skeleton* is the medial axis. The local feature sizes are small at $x$ and $y$ because the curvature is high at $x$ and $y$ is near a subcurve that is far way from $y$ along the curve

in the range $[0, \pi]$. Given three points $a$, $b$ and $c$, we use $\angle abc$ to denote $\angle(a - b, c - b)$. Let $h$ and $h'$ be two linear objects such as vectors, segments, lines, polygons, and planes. We use $\angle_a(h, h')$ to denote the nonobtuse angle between the affine subspaces spanned by $h$ and $h'$. $B(x, r)$ denotes the ball with center $x$ and radius $r$. Given a ball $B$, we use $\partial B$ to denote its boundary. Given a triangle $\tau$, $c_\tau$ denotes its circumcenter, $\gamma_\tau$ denotes its circumradius, $B_\tau$ denotes the *diametric ball* $B(c_\tau, \gamma_\tau)$ of $\tau$, and $\mathbf{n}_\tau$ denotes a unit vector orthogonal to $\mathrm{aff}(\tau)$.

A *triangulated polygonal surface* $T$ is a set of vertices, edges and triangles such that the intersection of every pair of elements in $T$ is either empty or an element in $T$, and for every vertex of $T$, its incident triangles form a topological disk. The union of the vertices, edge and triangles form the *underlying space* $|T|$ of $T$. The *star* of a vertex $p \in T$, denoted $\mathrm{star}(p)$, is the set of edges and triangles in $T$ that are incident to $p$.

Let $\Sigma \subset \mathbb{R}^3$ be a closed connected $C^2$-smooth surface throughout this paper. For every point $x \in \Sigma$, a *medial ball* $B$ at $x$ is a maximal ball tangent to $\Sigma$ at $x$ such that the interior of $B$ does not intersect $\Sigma$. The *medial axis* $\mathscr{M}$ of $\Sigma$ is the set of centers of medial balls at points in $\Sigma$. The *local feature size* of a point $x \in \Sigma$ is $f(x) = d(x, \mathscr{M})$. The local feature size function $f$ is 1-Lipschitz, i.e., $f(x) \leq f(y) + d(x, y)$ [11].

A finite point set $P \subset \Sigma$ is an *$\varepsilon$-sample* of $\Sigma$ for some $\varepsilon \in (0, 1)$ if $d(x, P) \leq \varepsilon f(x)$ for every point $x \in \Sigma$. The local feature size is mall at a point $x$ if the curvature is high at $x$ or if $x$ is near a point in $\Sigma$ whose geodesic distance from $x$ is much larger. The image on the right in Fig. 1 illustrates these two cases in 2D. In such cases, a higher sampling density is needed around $x$ for the reconstruction to be faithful. The local feature size must be nonzero for an $\varepsilon$-sample to be well defined. Thus, $\Sigma$ is required to be $C^2$-smooth (no sharp feature or junction). Boundary is also not allowed, although it seems not difficult to handle boundaries in practice as shown in our experiments. Suppose that one has the primitive to compute intersections between $\Sigma$ and lines, and the primitive to check for every axes-aligned cube $C$, whether every pair of surface normals in $\Sigma \cap C$ deviate by a small angle. Then, one can recursively refine an octree partition until the intersections between the leaf cell edges and $\Sigma$ define a dense sample of $\Sigma$ [25].

The *nearest point map* $\varphi$ maps a point $x \in \mathbb{R}^3 \setminus \mathcal{M}$ to the point $\varphi(x) \in \Sigma$ closest to $x$. We use $\mathbf{n}_x$ to denote the outward unit surface normal at a point $x \in \Sigma$.

A *mesh of* $\Sigma$ is a triangulated polygonal surface $T$ such that the vertices of $T$ are points in $\Sigma$ and $|T|$ is homeomorphic to $\Sigma$.

## 1.2 Motion Model

Consider the simulation of a deforming surface that progresses in unit time steps. The surface is specified by $n$ moving sample points on it. At time $t$, $\Sigma_t$ denotes the surface, $f_t$ denotes the LFS function of $\Sigma_t$, $P_t$ denotes the set of moving sample points, and $\Delta_t(x)$ denotes the speed of a point $x \in \Sigma_t$. For any vertex $v$ of a mesh $T$ of $\Sigma_t$, $n_T(v)$ denotes the distance from $v$ to the nearest vertex in $T$ and $R_T(v)$ denotes the largest circumradius of the triangles incident to $v$.

Suppose that $P_t$ is an $\varepsilon$-sample of $\Sigma_t$ at all times and, at any time step $t$, the velocities of the $n$ sample points are returned by the numerical procedure that drives the simulation. Not all points in $P_t$ can be used as mesh vertices in order that no triangle angle is too small. Theoretically, since an $\varepsilon$-sample can be arbitrarily dense locally, it is impossible to prove that some constant fraction of $P_t$ must appear as mesh vertices. However, it is unlikely that $P_t$ is arbitrarily dense anywhere in practice, and we have observed in our experiments that more that 50 % of the sample points appear as mesh vertices. We assume that the deformation is *smooth* as defined below.

**Definition 1** We say that deformation is *smooth* if the following conditions are satisfied at every time step $t$:

(i) For every point $x \in \Sigma_t$, $\Delta_t(x)$ is at most $0.005\varepsilon_0 \sin\alpha_0$ times the LFS of $x$ at $t$, and $\Delta_{t+1}(x) = O(\Delta_t(x))$.
(ii) For every pair of sample points $p$ and $q$, if $d(p,q) \leq \Delta_t(p)$, then $\Delta_t(p) = O(\Delta_t(q))$.
(iii) At time $t$, the displaced mesh vertices from the previous time step form an $\varepsilon_1$-sample, where $\varepsilon_1 \leq \kappa\varepsilon$ for some constant $\kappa$.

We are interested in the deformation of a particular class of surfaces meshes defined as follows [7, 8].

**Definition 2** For every $\varepsilon \in (0,1)$ and every constant $\alpha \in (0,\pi/3]$, an $(\varepsilon,\alpha)$-mesh of $\Sigma$ is a triangulation $T$ that satisfies the following conditions.

- The vertices of $T$ form an $\varepsilon$-sample of $\Sigma$.
- The angles of every triangle in $T$ are at least $\alpha$.

- There exists a triangle $\tau$ in $T$ and a vertex $p$ of $\tau$ such that $\angle_a(\mathbf{n}_p, \mathbf{n}_\tau) \leq \arcsin\left(\frac{0.8}{1+2\csc(\alpha/2)}\right)$.
- $\varphi$ restricted to $|T|$ is a homeomorphism between $|T|$ and $\Sigma$.

### *1.3 Main Result*

We prove that there exist constants $\varepsilon_0 \in (0,1)$ and $\alpha_0 \in (0,\pi/6)$ such that an $(\varepsilon_0,\alpha_0)$-mesh can be constructed before the simulation begins and, at each subsequent time step, an $(\varepsilon_0,\alpha_0)$-mesh can be restored via edge flips and insertions and deletions of vertices. Theoretically, $\alpha_0$ can be made close to $\pi/6$, but the sampling density would need to be extremely high. Our experiments suggest that $\alpha_0$ can be made greater than $10°$ in practice. The asymptotic running time can be made $O(n)$ [7, 17]. In our experiments (Sect. 6), 90 % of angles are in the range $[30°, 120°]$, only less than 0.02 % of angles are less than $15°$, and no angle is smaller than $11°$. Our theoretical framework does not allow for topological changes, boundaries, or sharp features. We cater for boundaries in our experiments by keeping all input sample points on the boundaries as mesh vertices and the edges connecting such adjacent sample points as mesh edges.

Level set methods [12, 23, 27] and point-based methods [1, 22, 24] are popular methods to model deforming objects with topological changes, but an explicit mesh is not maintained. Our focus is on fast maintenance of a mesh with theoretical guarantees on its quality instead of producing the deformation. We avoid reconstruction from scratch in order to improve efficiency. Thus, our result is most similar to the prior work on tracking a deforming mesh without any topological change [14, 16, 28], but these prior work do not offer any guarantee. Several techniques have been developed to track and modify a mesh in order to produce topological changes [6, 26, 30]. It may be possible to combine them with our algorithm to allow topological changes and preserve sharp features.

## 2 High Level Strategy

Our strategy is to maintain an $(\varepsilon_0,\alpha_0)$-mesh $M_t$ with vertices from $P_t$ that satisfies the following conditions C1–C3. Let $\ell$ and $\lambda$ be two constants such that $\ell$ is sufficiently large and $\lambda$ is less than 1. The setting of $\ell$ and $\lambda$ will be explained later. By C2, $\alpha_0$ can be set to be $\arcsin(\lambda/2)$.

C1: For every vertex $v$ of $M_t$, $n_{M_t}(v) \geq 20(\sin\alpha_0)^{-1}\Delta_t(v)$.

C2: For every vertex $v$ and triangle $\tau$ in $M_t$, if $v \in B(c_\tau, \ell\gamma_\tau)$, then $n_{M_t}(v) \geq \lambda\gamma_\tau$.

C3: Some points in $P_t$ may not appear as vertices in $M_t$. Such a point $p$ is stored in some list points($v$), where $v$ is a vertex of $M_t$ and $d(p,v) \leq 2R_{M_t}(v)$.

Property C1 ensures that a mesh edge turns an angle less than $\alpha_0/10$ from time $t$ to $t+1$. Property C2 ensures that the triangle circumradii vary smoothly. By C3, every sample point that is not a vertex is stored at some vertex nearby. Let $K_{t+1}$ denote the deformed $M_t$ at time $t+1$, which has the same connectivity as $M_t$. Based on C1–C3 and the smoothness of the deformation, we can show that the deformed mesh $K_{t+1}$ is an $(\varepsilon_1, \alpha_1)$-mesh for some $\alpha_1 \in [\frac{4}{5}\alpha_0, \alpha_0]$ that satisfies the following conditions $\hat{\text{C}}$1–$\hat{\text{C}}$3, which are degraded versions of C1–C3. The proof is given in Sect. 3.

$\hat{\text{C}}$1: For any vertex $v$ of $K_{t+1}$, $n_{K_{t+1}}(v) \geq 18(\sin\alpha_0)^{-1}\Delta_t(v)$.

$\hat{\text{C}}$2: For every vertex $v$ and triangle $\tau$ in $K_{t+1}$, if $v \in B(c_\tau, \frac{1}{2}\ell\gamma_\tau)$, then $n_{K_{t+1}}(v) \geq \frac{1}{2}\lambda\gamma_\tau$.

$\hat{\text{C}}$3: For every vertex $v$ in $K_{t+1}$ and every point $p \in \text{points}(v)$, $d(p, v) = O(R_{K_{t+1}}(v))$. The big-Oh constant depends on that the constant in assumption (ii) of our smooth deformation model.

Our problem is to compute an $(\varepsilon_0, \alpha_0)$-mesh $M_{t+1}$ from $K_{t+1}$ that satisfies C1–C3 so that the simulation can continue to the next time step and so on.

The initial $(\varepsilon_0, \alpha_0)$-mesh can be obtained by pruning the sample to an $\varepsilon_0$-sample that is sparse with respect to their initial velocities and the local feature sizes, followed by running a surface reconstruction algorithm (e.g. [3, 9]) on the pruned sample. The output mesh is an $(\varepsilon_0, \alpha_0)$-mesh satisfying C1–C3.

**Lemma 1** *One can compute an* $(\varepsilon_0, \alpha_0)$*-mesh from an* $\varepsilon$*-sample that satisfies* C1, C2 *and* C3.

*Proof* We first compute a surface mesh $M'$ from the $\varepsilon$-sample using a surface reconstruction algorithm. Then for every vertex $v$, we define its decimation radius $\delta_v = \max\left\{4\lambda\gamma_\tau - \frac{\lambda}{\ell}d(v, c_\tau) : \text{triangle } \tau \in M'\right\}$. Initially, all vertices are unmarked. Take an unprocessed vertex $v$ that is unmarked. Mark all vertices in $B(v, \max\{\delta_v, 20(\sin\alpha_0)^{-1}\Delta_0(v)\}) \setminus \{v\}$. Repeat it until all vertices are processed. Run reconstruction again on the unmarked vertices to compute our initial mesh $M$. We enforce condition C3 by storing each non-vertex sample point in the point list of its nearest vertex in $M$. We can show that $M$ is an $(\varepsilon_0, \alpha_0)$-mesh that satisfies C1–C3 by using the same arguments as in the proofs of Lemmas 17 and 18. □

At time $t+1$, we call UPDATE($K_{t+1}$) to compute $M_{t+1}$. UPDATE iterates two phases, REFINE and DECIMATE. In the first iteration, REFINE inserts some sample points as vertices to make the vertex set an $O(\lambda\varepsilon_1)$-sample of $\Sigma_{t+1}$. Refinement alone cannot restore the angle lower bound $\alpha_0$, because the vertices may become very crowded in some region, where the inter-vertex distance is much less than $\varepsilon$ times the local features sizes, and we run out of sample points to refine its neighborhood. In the second phase, DECIMATE deletes some vertices to increase the inter-vertex distances while keeping a good sampling density. At the end of the first iteration, the vertex set forms an $O(\lambda\varepsilon_1)$-sample. Another iteration of the two phases makes the vertex set an $O(\lambda^2\varepsilon_1)$-sample. So we obtain $M_{t+1}$ in $O(\log(1/\varepsilon_0)) = O(1)$ iterations.

## 3 Mesh Deterioration

Inductively, the mesh at time $t$ satisfies conditions C1–C3. We want to show that after the deformation, the mesh at time $t+1$ satisfies $\hat{\text{C}}1$–$\hat{\text{C}}3$, a set of similar conditions with some degraded constants. The next lemma can be proved by a straightforward trigonometric argument.

**Lemma 2** *Let $\tau' = x'y'x'$ be a triangle at time $t$ with minimum angle larger than $\alpha_0$, and it deforms to $\tau = xyz$ at time $t+1$. Assume that the displacement of each vertex is at most $\kappa^{-1}\sin\alpha_0$ times the length of any of its incident edges, where $\kappa$ is a constant greater than 14. Then $(1-6/\kappa)\gamma_{\tau'} \le \gamma_\tau \le (1+14/\kappa)\gamma_{\tau'}$.*

**Lemma 3** *Let $M_t$ be the mesh at time $t$ satisfying* C1–C3*. Let $K_{t+1}$ be the deformed mesh at time $t+1$. Assume $\ell \ge 54$. Then $K_{t+1}$ satisfies* $\hat{\text{C}}1$–$\hat{\text{C}}3$.

*Proof* Consider $\hat{\text{C}}1$. Let $u$ be the nearest vertex of $v$ in $K_{t+1}$. Suppose $u$ and $v$ are at $u'$ and $v'$, respectively, at time $t$. By C1, both $\Delta_t(v')$ and $\Delta_t(u')$ are less than $d(u',v')/20$. Therefore, $n_{K_{t+1}}(v) = d(u,v) \ge d(u',v') - \Delta_t(v') - \Delta_t(u') \ge 0.9d(u',v') \ge 0.9n_{M_t}(v')$. which is at least $18(\sin\alpha_0)^{-1}\Delta_t(v)$ by C1.

$\hat{\text{C}}2$ requires that for every triangle $\tau$ and every vertex $v$ in $B(c_\tau, \frac{1}{2}\ell\gamma_\tau)$, $n_{K_{t+1}}(v)$ is at least $\frac{1}{2}\lambda\gamma_\tau$. Let $u$ be the vertex of $\tau$. Let $u'$, $v'$ and $\tau'$ be the counterparts of $u$, $v$ and $\tau$ in $M_t$. $d(v',u') \le d(v,c_\tau) + d(u,c_\tau) + \Delta_t(u') + \Delta_t(v') \le (\ell/2+1)\gamma_\tau + d(v',u')/10$. Rearranging the terms, we obtain $d(v',u') \le \frac{10}{9}\left(\frac{1}{2}\ell+1\right)\gamma_\tau < (\ell-1)\gamma_{\tau'}$ by Lemma 2 and the fact that $\ell \ge 54$. Since $d(v',c_{\tau'}) \le d(v',u') + \gamma_{\tau'} \le \ell\gamma_{\tau'}$, C2 implies that $n_{M_t}(v') \ge \lambda\gamma_{\tau'}$ and hence $n_{K_{t+1}}(v) \ge 0.9n_{M_t}(v') \ge 0.9\lambda\gamma_{\tau'}$. Lemma 2 further implies that $n_{K_{t+1}}(v) \ge 0.9\lambda\gamma_{\tau'} \ge \frac{0.9\lambda\gamma_\tau}{1+14/20} > \frac{1}{2}\lambda\gamma_\tau$, establishing $\hat{\text{C}}2$.

Consider $\hat{\text{C}}3$. Let $v$ be a vertex in $K_{t+1}$, and $p$ a sample point in points$(v)$. Suppose they are at $v'$ and $p'$ at time $t$, respectively. By C1 and C3, $d(p,v) \le d(p',v') + \Delta_t(p') + \Delta_t(v') \le 2R_{M_t}(v') + \Delta_t(p') + n_{M_t}(v')/20 \le 2.1R_{M_t}(v') + \Delta_t(p')$. If $\Delta_t(p') \le d(p',v')$, $\Delta_t(p') \le d(p',v') \le 2R_{M_t}(v')$. Otherwise, by our assumption on the smoothness of the deformation, $\Delta_t(p') \le c\Delta_t(v')$ for some constant $c$. Therefore, $\Delta_t(p') \le c\Delta_t(v') \le cn_{M_t}(v')/20 \le cR_{M_t}(v')/10$. In both cases, $d(p,v) < (5+c/10)R_{M_t}(v') = O(R_{K_{t+1}}(v))$ by Lemma 2. □

## 4 Refinement

In the first iteration, the input to Refine is $K_{t+1}$. In remaining iterations, we feed it with the output of Decimate. Inductively, the input mesh $X$ of Refine is an $(\varepsilon_X, \alpha_0)$-mesh satisfying $\hat{\text{C}}1$–$\hat{\text{C}}3$ for some $\varepsilon_X \le \varepsilon_1$. Our goal is to improve the sampling condition from $\varepsilon_X$ to $O(\lambda\varepsilon_X)$.

First, we update the point lists of every vertex, so that a non-vertex sample point is stored in the point list of its nearest vertex. As a result, for every non-vertex sample point $p$ and the vertex $u$ such that $p \in \text{points}(u)$, $d(p,u) \leq 2R_X(u)$. Then, initialize an intermediate mesh $T$ to be $X$ and incrementally inserts points. We go through the points in $P_{t+1}$ that are not vertices and insert those that are far away from existing vertices. Let $p$ be the current non-vertex point being processed. Let $u$ be the vertex of $X$ such that $p$ is stored in points($u$). Let $w$ be the vertex of the current mesh $T$ nearest to $p$, which must lie in $B(u, 4R_X(u))$ because $d(u,w) \leq d(u,p)+d(p,w) \leq 2d(p,u) \leq 4R_X(u)$. We find $w$ by searching $T$ within $B(u, 4R_X(u))$. If the distance between $p$ and $w$ is less than $\lambda R_X(u)$ or $20(\sin\alpha_0)^{-1}\Delta_{t+1}(p)$, skip $p$; otherwise, insert $p$ as explained below.

Suppose that $p$ is a candidate point to be inserted. We apply a result to be introduced shortly, Theorem 1(iii), to the vertices whose incident triangles intersect $B(u, 4R_X(u))$. In the end, all the triangles intersecting $B(u, 4R_X(u))$ have almost empty diametric balls, and so do their neighboring triangles. The common edge $ab$ between two triangles $abc$ and $abd$ is *flippable* if and only if the diametric ball of $abc$ contains $d$ in its interior and the diametric ball of $abd$ contains $c$ in its interior.

In particular, the triangle $v_1v_2v_3$ nearest to $p$ has an almost empty diametric ball, because the distance between $u$ and the nearest triangle $v_1v_2v_3$ is at most $d(u,p) + d(p, v_1v_2v_3) \leq 2d(u,p) \leq 4R_X(u)$. We insert $p$ by calling ADD($T$, $p$, $v_1v_2v_3$). It works as follows. Compute the point $\tilde{p}$ in $v_1v_2v_3$ nearest to $p$. Split $v_1v_2v_3$ using $\tilde{p}$. That is, we replace $v_1v_2v_3$ by three triangles $\tilde{p}v_1v_2$, $\tilde{p}v_2v_3$ and $\tilde{p}v_3v_1$. Flip the edges $v_1v_2$, $v_2v_3$ and $v_1v_3$ if they are flippable. Finally, for each triangle incident to $\tilde{p}$, replace its vertex $\tilde{p}$ by $p$. This adds $p$ as a vertex to $T$.

After all the insertions, we flip the flippable edges in $T$, and migrate all non-vertex sample points to the point lists of their nearest vertices in the current mesh.

## 4.1 Mesh Properties

To analyze REFINE, we need some surface sampling results in the literature and some properties of $(\varepsilon,\alpha)$-meshes that we established in another paper [8]. Some preliminary analogous results can be found in [7].

**Lemma 4 ([2, 11, 13])** *Let $p$, $q$ and $r$ be any three points on $\Sigma$.*

(i) *For every $\varepsilon \leq 1/3$, if $d(p,q) \leq \varepsilon f(p)$, then $\angle(\mathbf{n}_p, pq) \geq \pi/2 - \varepsilon$ and $\angle(\mathbf{n}_p, \mathbf{n}_q) \leq 2\varepsilon$.*

(ii) *For every $\varepsilon \leq 1/10$, if $\gamma_{pqr} \leq \varepsilon f(p)$, then $\angle(\mathbf{n}_p, \mathbf{n}_{pqr}) \leq 10\varepsilon$.*

(iii) *For every $\varepsilon \leq 1/4$ and every point $z$ on tangent plane at $p$, if $d(p,z) \leq \varepsilon f(p)$, then $d(z,\Sigma) \leq \varepsilon d(p,z)$.*

**Lemma 5 ([8])** *Let $p$, $q$ and $r$ be any three points on $\Sigma$. Let $c$ be any positive constant. Suppose that $\gamma_{pqr} \leq c\varepsilon f(p)$ for some $\varepsilon < \min\{1, 1/(72c)\}$. Then, for every point $x$ in the circumdisk of $pqr$, $d(x,\varphi(x)) \leq 10c\varepsilon\, d(p,x) \leq 20c^2\varepsilon^2 f(p)$ and $d(p,\varphi(x)) \leq (2c\varepsilon + 20c^2\varepsilon^2) f(p)$.*

**Lemma 6 ([8])** *Let $\mu = 2(\csc\alpha)^{4\pi/\alpha+1}$. There exists an $\varepsilon_0 \in (0,1)$ depending on $\alpha$ such that for every $\varepsilon \in (0,\varepsilon_0]$, if If $T$ is an $(\varepsilon,\alpha)$-mesh of $\Sigma$, then*

(i) *For each vertex $p \in T$ and every triangle $\tau \in \mathrm{star}(p)$, $\gamma_\tau \le \mu\varepsilon f(p)$ and $\angle_a(\mathbf{n}_p,\mathbf{n}_\tau) < 6\mu\varepsilon$.*
(ii) *For every pair of triangles $\sigma,\tau \in T$ that share an edge, the dihedral angle at $\sigma \cap \tau$ is greater than $\pi - 12\mu\varepsilon$.*

**Lemma 7 ([8, 17])** *For all $c > 1$, if $T$ is an $(\varepsilon,\alpha)$-mesh of $\Sigma$ for a small enough $\varepsilon$, then for every vertex $p \in T$, $|T| \cap B(p, c\mu\varepsilon f(p))$ is connected and it projects injectively onto any plane that makes an angle at least $\pi/3$ with $\mathbf{n}_p$.*

**Theorem 1 ([7, 8])** *For every constant $c \in (0,0.5)$ and every constant $\alpha \in [0,\pi/3]$, there exists $\varepsilon_0 \in (0,1)$ depending on $c$ and $\alpha$ such that for every $\varepsilon \in (0,\varepsilon_0]$, if $T$ is an $(\varepsilon,\alpha)$-mesh of a connected closed smooth surface, then the following properties are satisfied. The common edge $ab$ between two triangles $abc$ and $abd$ is* flippable *if and only if the diametric ball of $abc$ contains $d$ in its interior and the diametric ball of $abd$ contains $c$ in its interior.*

(i) *We can flip flippable edges in $T$ until no edge is flippable in time linear in the number of vertices in $T$. An $(\varepsilon,\alpha)$-mesh $T'$ is produced in the end and for every triangle $\tau \in T'$, $B(c_\tau,(1-\varepsilon^c)\gamma_\tau)$ does not contain any vertex.*
(ii) *For every vertex $p \in T$ and every triangle $\tau \in \mathrm{star}(p)$, if $B(c_\tau,(1-\varepsilon^c)\gamma_\tau)$ does not contain any vertex, then $\gamma_\tau \le (\varepsilon + O(\varepsilon^{1+c}))f(p)$.*
(iii) *Given any subset $V$ of vertices of $T$, we can flip flippable edges in $O(|V|)$ time to produce an $(\varepsilon,\alpha)$-mesh $T'$ so that for every triangle $\tau \in T'$ that is incident to a vertex in $V$ or a neighbor of a vertex in $V$, $B(c_\tau,(1-\varepsilon^c)\gamma_\tau)$ does not contain any vertex.*

## 4.2 Analysis of Refinement

We apply the mesh properties to analyze the effects of refinement. The proofs of the next two technical lemmas are straightforward and omitted.

**Lemma 8** *Let $T$ be an $(\varepsilon,\alpha)$-mesh of $\Sigma$ for a sufficiently small $\varepsilon$.*

(i) *For every point $x$ in the circumdisk of a triangle $\tau \in T$, $d(x,\varphi(x)) \le 10\mu\varepsilon\gamma_\tau \le 10\mu^2\varepsilon^2 f(\varphi(x))$.*
(ii) *For every point $y \in \Sigma$, the distance between $y$ and its nearest point $z \in |T|$ is at most $10\mu^2\varepsilon^2 f(y)$. For any triangle $\sigma \in T$ that contains $z$, $d(y,z) < 24\mu\varepsilon\gamma_\sigma$.*

**Lemma 9** *Let $T_1$ and $T_2$ be a $(\xi_1,\theta_1)$-mesh and a $(\xi_2,\theta_2)$-mesh of $\Sigma$, respectively, possibly with different vertex sets. Let $\tau$ be a triangle in $T_1$. Let $\sigma$ be the triangle in $T_2$ nearest to $\varphi(c_\tau)$.*

(i) $d(c_\tau,\sigma) \le 10\mu\xi_1\gamma_\tau + 24\mu\xi_2\gamma_\sigma$.

(ii) *Let $\beta$ be any constant in the range* $(0, 1]$. *Assume that* $\xi_1$ *and* $\xi_2$ *are sufficiently small. If* $B(c_\tau, \beta\gamma_\tau)$ *does not contain any vertex of* $T_2$*, then* $\gamma_\sigma \geq 0.9\beta\gamma_\tau$.

Throughout the procedure, we need to maintain certain invariants so that the local edge-flip algorithm works (which requires a constant lower bound on angles).

**Lemma 10** *Let $X$ be an $(\varepsilon_X, \alpha_0)$-mesh satisfying* $\hat{C}1$–$\hat{C}3$. *During the execution of* REFINE$(X)$*, the following invariants on the intermediate mesh $T$ are maintained.*

(i) *For every vertex $a$ in $X$ and every vertex $b$ in $T$, if $d(a,b) \leq (\ell/4 - 3)R_X(a)$, then $n_T(b) \geq \frac{1}{4}\lambda^2 R_X(a)$.*
(ii) *$T$ is an $(\varepsilon_X, \theta)$-mesh for some constant $\theta > 0$.*

*Proof* Consider invariant (i). Let $T$ be the mesh just after we add a vertex $p$. Let $u$ be the vertex of $X$ nearest to $p$. It is also the vertex whose point list contains $p$ at the beginning of the main loop.

We first show that invariant (i) holds for $b = p$. Let $a$ be any vertex in $X$ such that $d(a, p) \leq (\ell/4 - 3)R_X(a)$. If $R_X(a) \leq R_X(u)$, then since $n_T(p) \geq \lambda R_X(u)$ for us to decide to insert $p$, we have $n_T(p) \geq \lambda R_X(a)$ as desired. Assume that $R_X(a) > R_X(u)$. We have $d(a,u) \leq d(a,p) + d(p,u) \leq (\ell/4 - 3)R_X(a) + 2R_X(u) < (\ell/4 - 1)R_X(a)$. Let $\tau$ be the triangle incident to $a$ in $X$ with the largest circumradius, i.e., $\gamma_\tau = R_X(a)$. Then, $d(u, c_\tau) \leq d(a,u) + d(a, c_\tau) \leq (\ell/4)\gamma_\tau$. So $\hat{C}2$ applies and implies that $R_X(u) \geq \frac{1}{2}n_X(u) \geq \frac{1}{4}\lambda\gamma_\tau = \frac{1}{4}\lambda R_X(a)$. Since $n_X(p) \geq \lambda R_X(u)$ for us to insert $p$, we conclude that $n_X(p) \geq \frac{1}{4}\lambda^2 R_X(a)$. This proves invariant (i) for the new vertex $p$.

Now consider a vertex $b$ of $T$ other than $p$. If the nearest vertex to $b$ is not changed by the insertion of $p$, then invariant (i) holds for $b$ inductively. Assume that the nearest vertex of $b$ becomes $p$. Let $a$ be any vertex of $X$ such that $d(a,b) \leq (\ell/4 - 3)R_X(a)$. If $R_X(a) \leq R_X(u)$, then $n_T(b) = d(p,b) \geq n_T(p) \geq \lambda R_X(u) \geq \lambda R_X(a)$. So invariant (i) holds for $b$ in this case. Assume that $R_X(a) > R_X(u)$. Since $p$ is the nearest vertex of $b$, we have $d(p,b) \leq d(a,b) \leq (\ell/4 - 3)R_X(a)$. This implies that $d(a,u) \leq d(a,b) + d(p,b) + d(p,u) \leq (\ell/2 - 4)R_X(a)$. Then, we can invoke the same analysis as in the previous paragraph to show that $R_X(u) \geq \frac{1}{4}\lambda R_X(a)$. It follows that $n_X(b) = d(p,b) \geq n_X(p) \geq \frac{1}{4}\lambda^2 R_X(a)$. This proves invariant (i).

Consider invariant (ii). The mesh density cannot decrease because vertices are being inserted. We show that any new angle in $X$ is at least some constant $\theta$ after inserting a point $p$. We omit the argument for establishing the bound on the triangle circumradii and the nearest point map being a homeomorphism as required by the definition of an $(\varepsilon_X, \theta)$-mesh, which is similar to the proof of Theorem 1(i) [8].

Suppose that a point $p$ is inserted into the triangle $v_1v_2v_3$. That is, $v_1v_2v_3$ is the closest triangle to $p$ in the current mesh and we split $v_1v_2v_3$ into three smaller triangles by connecting the three vertices to the projection $\tilde{p}$ of $p$ on $v_1v_2v_3$. Let $qv_1v_2$ be the triangle that shares $v_1v_2$ with $\tilde{p}v_1v_2$.

We first bound the angles in the triangle $qv_1v_2$ from below. Applying Theorem 1(iii) makes the diametric ball of $qv_1v_2$ almost empty. Thus,

$B(c_{qv_1v_2}, 0.8\gamma_{qv_1v_2})$ does not contain any vertex of $X$. Then by Lemma 9, we can find a triangle $\sigma$ in $X$ such that $\gamma_\sigma > 0.7\gamma_{qv_1v_2}$, and $d(c_{qv_1v_2}, \sigma) < 0.1\gamma_{qv_1v_2} + 0.1\gamma_\sigma$. Take the vertex $a$ of $\sigma$ nearest to $c_{qv_1v_2}$. The distance between $a$ and any vertex of $qv_1v_2$ is at most

$$d(a, c_{qv_1v_2}) + \gamma_{qv_1v_2} < \gamma_\sigma + (0.1\gamma_{qv_1v_2} + 0.1\gamma_\sigma) + \gamma_{qv_1v_2} < 3\gamma_\sigma. \tag{1}$$

Thus, invariant (i) applies to $a$ and any vertex of $qv_1v_2$. We conclude that $qv_1v_2$ has edge lengths at least $\frac{1}{4}\lambda^2 R_X(a) \geq \frac{1}{4}\lambda^2\gamma_\sigma > \frac{1}{6}\lambda^2\gamma_{qv_1v_2}$. The angles in $qv_1v_2$ are then at least $\psi = \arcsin(\lambda^2/12)$. Similarly, all angles in $v_1v_2v_3$ are at least $\psi$.

Since we decide to insert $p$, $n_T(p) \geq \lambda R_X(u)$. As argued in proving (i), this leads to $n_T(p) \geq \frac{1}{4}\lambda^2 R_X(a)$, so $n_T(p) > \frac{1}{6}\lambda^2\gamma_{qv_1v_2}$. This implies that $d(p, v_1)$ and $d(p, v_2)$ are at least $\frac{1}{6}\lambda^2\gamma_{qv_1v_2}$. Since $\tilde{p}$ is the point on the current mesh nearest to $p$, Lemma 8(ii) implies that $d(p, \tilde{p}) \leq (40\kappa_{\mathrm{rad}}\varepsilon_X)\gamma_{v_1v_2v_3} \leq \frac{40\kappa_{\mathrm{rad}}\varepsilon_X}{\sin\psi}\gamma_{qv_1v_2} \leq \frac{40\kappa_{\mathrm{rad}}\varepsilon_1}{\sin\psi}\gamma_{qv_1v_2}$. So for a small enough $\varepsilon_1$, $d(\tilde{p}, v_1)$ and $d(\tilde{p}, v_2)$ are at least $\frac{1}{7}\lambda^2\gamma_{qv_1v_2}$.

Suppose that $\gamma_{qv_1v_2} \geq \gamma_{\tilde{p}v_1v_2}\sin(\psi/2)$. So $\sin\angle\tilde{p}v_1v_2 = \frac{d(\tilde{p},v_2)}{2\gamma_{\tilde{p}v_1v_2}} \geq \frac{\frac{1}{7}\lambda^2\gamma_{qv_1v_2}}{2\gamma_{qv_1v_2}/\sin(\psi/2)} \geq \frac{\lambda^2}{14}\sin(\psi/2)$. It follows that $\angle\tilde{p}v_2v_1 \geq \arcsin\left(\frac{\lambda^2}{14}\sin(\psi/2)\right)$. Similarly, $\angle\tilde{p}v_2v_1 \geq \arcsin\left(\frac{\lambda^2}{14}\sin(\psi/2)\right)$. Also, we have $\angle v_1\tilde{p}v_2 \geq \angle v_1v_3v_2 \geq \psi$.

Suppose that $\gamma_{qv_1v_2} \leq \gamma_{\tilde{p}v_1v_2}\sin(\psi/2)$. We claim that $v_1v_2$ is flippable. Since $\angle v_1\tilde{p}v_2 \geq \angle v_1v_3v_2 \geq \psi$ and $\sin\angle v_1\tilde{p}v_2 = \frac{d(v_1,v_2)}{2\gamma_{\tilde{p}v_1v_2}} \leq \frac{2\gamma_{qv_1v_2}}{2\gamma_{\tilde{p}v_1v_2}} \leq \sin(\psi/2)$, the angle $\angle v_1\tilde{p}v_2$ must be obtuse and greater than $\pi - \psi/2$. Imagine that we rotate $v_1\tilde{p}v_2$ while fixing $v_1v_2$ to make the dihedral angle between $v_1\tilde{p}v_2$ and $v_1qv_2$ larger. Since $\angle v_1\tilde{p}v_2 > \pi/2$, $\tilde{p}$ moves closer to the boundary of the diametric ball of $v_1qv_2$ as we rotate $v_1\tilde{p}v_2$. Let $p'$ be the point in the plane of $v_1v_2q$ such that $\angle v_1p'v_2 = \angle v_1\tilde{p}v_2$ and it is on the different side of $v_1v_2$ from $q$. So $\tilde{p}$ is in the diametric ball of $v_1qv_2$, if $p'$ is in the diametric ball of $v_1qv_2$, which is true because $\angle v_1p'v_2 + \angle v_1qv_2 < \pi$. We show below that $q$ also lies inside the diametric ball of $v_1v_2\tilde{p}$, and hence $v_1v_2$ is flippable. The plane of $v_1v_2q$ intersects $B_{v_1v_2\tilde{p}}$ in a disk $D$ with radius $\gamma \geq (1 - O(\varepsilon))\gamma_{v_1v_2\tilde{p}}$, as the dihedral angle between $v_1v_2q$ and $v_1v_2\tilde{p}$ is $\pi - O(\varepsilon)$. If $q$ is outside $B_{v_1v_2\tilde{p}}$, then $q$ is outside $D$, and $\psi \leq \angle v_1qv_2 \leq \arcsin\left(\frac{d(v_1,v_2)}{2\gamma}\right)$. Since $d(v_1, v_2) = 2\gamma_{v_1v_2\tilde{p}}\sin\angle v_1\tilde{p}v_2 \leq 2\gamma_{v_1\tilde{p}v_2}\sin(\psi/2)$, we get $\sin\psi \leq (1 + O(\varepsilon))\sin(\psi/2) \Leftrightarrow 2\cos(\psi/2) \leq 1 + O(\varepsilon)$, which is impossible for $\psi < \pi/3$. So $q$ lies inside the diametric ball of $v_1v_2\tilde{p}$.

Flipping $v_1v_2$ produces $v_1\tilde{p}q$ and $v_2\tilde{p}q$. First, $\angle\tilde{p}v_1q \geq v_2v_1q \geq \psi$, as the dihedral angle between $qv_1v_2$ and $\tilde{p}v_1v_2$ is obtuse. Next, if $\angle v_1\tilde{p}q$ is non-obtuse, since $v_1v_2$ is flippable, one can show that $\angle v_1\tilde{p}q \geq \angle v_1v_2q \geq \psi$ [8]. If $\angle v_1\tilde{p}q$ is obtuse, it is less than $\pi - \angle\tilde{p}v_1q \leq \pi - \psi$. This implies that $\sin\angle v_1\tilde{p}q \geq \sin\psi$. Finally, we can bound $\angle v_1q\tilde{p}$: $\sin\angle v_1q\tilde{p} = \frac{d(\tilde{p},v_1)}{d(q,v_1)}\cdot\sin\angle v_1\tilde{p}q \geq \frac{\frac{1}{7}\lambda^2\gamma_{qv_1v_2}}{2\gamma_{qv_1v_2}}\cdot\sin\psi = (\lambda^2/14)\sin\psi$. A similar argument bounds the angles in $v_2q\tilde{p}$ from below.

So all new angles are at least $\min\{\psi, \psi', \psi''\}$ before lifting the triangles incident to $\tilde{p}$ to $p$, where $\psi' = \arcsin\left(\frac{\lambda^2}{14}\sin(\psi/2)\right)$, and $\psi'' = \arcsin\left(\frac{\lambda^2}{14}\sin\psi\right)$. Since $p$

is very close to $\tilde{p}$, the lifting decreases the angles by only a constant factor. So an angle lower bound $\theta$ can be preserved by an appropriate setting of $\theta$. □

The lemma below summarizes the properties of the final mesh obtained.

**Lemma 11** *Let $X$ be an $(\varepsilon_X, \alpha_0)$-mesh satisfying* $\hat{\text{C}}1$–$\hat{\text{C}}3$. REFINE$(X)$ *runs in time linear in the number of sample points, and produces a mesh $Y$ such that:*

(i) *For every triangle $\tau$ in $Y$, there is no vertex in $B(c_\tau, 0.8\gamma_\tau)$, and if a vertex $v$ lies in $B\big(c_\tau, (\ell/6-4)\gamma_\tau\big)$, then $n_Y(v) \geq \frac{1}{6}\lambda^2\gamma_\tau$.*
(ii) *$Y$ is an $(\varepsilon_Y, \theta)$-mesh, where $\theta \in (0, \pi/3]$ and $\varepsilon_Y = \max\{0.3\varepsilon_0, 4\mu\lambda\varepsilon_X\}$.*

*Proof* Consider the running time of REFINE$(X)$. We first show that updating the nearest vertex for each sample point takes $O(1)$ time. For every non-vertex sample point $p$, it is initially stored in the list of some vertex $u$ such that $d(p,u) = O(R_X(u))$, as $X$ satisfies $\hat{\text{C}}3$. Set $\ell$ large enough so that $2d(p,u) \leq (\ell/2)R_X(u)$. The distance between $p$ and its nearest vertex $w$ is at most $d(p,u)$, so $d(w,u) \leq 2d(p,u)$. All triangles intersecting $B(u, 2d(p,u))$ are connected, and they project injectively in the tangent plane at $u$ by Lemma 7. $\hat{\text{C}}2$ ensures that all edges of those triangles have lengths at least $\Omega(R_X(u))$. Since all angles in $X$ are at least some constant, a standard packing argument shows that there are only $O(1)$ triangles intersecting $B(u, 2d(p,u))$. Therefore, we can search for the nearest vertex of $p$ by traversing the triangles intersecting $B(u, 2d(p,u))$ in $O(1)$ time.

Inside the main loop, for each sample point $p$, we need to find its nearest vertex again. The point $p$ is stored in the point list of the vertex $u$ in $X$ nearest to $p$. So $d(p,u) \leq 2R_X(u)$. Lemma 10(ii) ensures that all angles in an intermediate mesh are at least some constant. Lemma 10(i) implies that all edges intersecting $B(u, 4R_X(u))$ have lengths $\Omega(R_X(u))$. Then the nearest vertex of $p$ can be found in $O(1)$ time by the same reasoning. Consider edge flips before the search for the nearest triangle of $p$. There are constant number of vertices whose incident triangles intersect $B(u, 4R_X(u))$. So by Theorem 1(iii), it takes $O(1)$ time to flip edges so that their incident triangles have almost empty diametric balls. Finding the nearest triangle of $p$ can also be done in $O(1)$ time by traversing the triangles intersecting $B(u, 4R_X(u))$. The remaining operations in the main loop clearly take $O(1)$ time.

The post-processing, including the edge flips and point migrations, can be done in linear time. Therefore, REFINE runs in linear time.

Consider (i). Since edges are flipped in post-processing until none is flippable, Theorem 1(i) any triangle $\tau$ in $Y$, $B(c_\tau, 0.8\gamma_\tau)$ does not contain any vertex. Let $v$ be a vertex such that $d(v, c_\tau) \leq (\ell/6-4)\gamma_\tau$. By Lemma 9, we can find a triangle $\sigma$ in $X$ such that $d(c_\tau, \sigma) < 0.1(\gamma_\tau + \gamma_\sigma)$ and $\gamma_\sigma \geq 0.7\gamma_\tau$. Let $a$ be a vertex of $\sigma$. So we have $d(a,v) \leq d(a,c_\tau) + d(v,c_\tau) \leq 2\gamma_\sigma + 0.1(\gamma_\tau+\gamma_\sigma) + (\ell/6-4)\gamma_\tau \leq (\ell/4-3)\gamma_\sigma$. Lemma 10(i) applies and implies that $n_Y(v) \geq \frac{1}{4}\lambda^2 R_X(a) \geq \frac{1}{4}\lambda^2\gamma_\sigma > \frac{1}{6}\lambda^2\gamma_\tau$.

Consider (ii). The angle lower bound $\theta$ follows from Lemma 10(ii). It remains to show that the vertices of $Y$ form an $\varepsilon_Y$-sample of $\Sigma_{t+1}$. Take any point $x$ in $\Sigma_{t+1}$. Since $P_{t+1}$ remains an $\varepsilon$-sample of $\Sigma_{t+1}$ in our motion model, there exists $p \in P_{t+1}$ such that $d(p,x) \leq \varepsilon f_{t+1}(x)$. If $p$ is a vertex of $Y$, we are done. Suppose that $p$ is not a vertex of $Y$. Let $u$ be the vertex of $X$ where $p \in \text{points}(u)$. Recall that, before the main loop, we move every sample point to the point list of its nearest vertex in $X$, so $d(p,u) \le 2R_X(u)$. Let $w'$ be the nearest vertex to $p$ in the intermediate mesh when we decided not to insert $p$. Let $w$ be the vertex of $Y$ nearest to $p$. The point $p$ was not inserted because $d(p,w') \le 20(\sin\alpha_0)^{-1}\Delta_{t+1}(p)$ or $d(p,w') \le \lambda R_X(u)$. Since nearest vertex distances can only decrease in the insertion phase, we have $d(p,w) \le d(p,w') \le 20(\sin\alpha_0)^{-1}\Delta_{t+1}(p)$ or $d(p,w) \le d(p,w') \le \lambda R_X(u)$.

In the case that $d(p,w) \le 20(\sin\alpha_0)^{-1}\Delta_{t+1}(p)$, we have $d(p,w) \le 0.1\varepsilon_0 f_{t+1}(p)$ as $\Delta_{t+1}(p) \le (0.005\varepsilon_0 \sin\alpha_0) f_{t+1}(p)$ by the smoothness of the deformation. Then, $d(w,x) \le d(p,w) + d(p,x) \le 0.1\varepsilon_0 f_{t+1}(p) + \varepsilon f_{t+1}(x)$. The Lipschitzness of LFS implies that $d(w,x) \le 0.1\varepsilon_0(1+\varepsilon) f_{t+1}(x) + \varepsilon f_{t+1}(x) < 0.3\varepsilon_0 f_{t+1}(x)$ because we assume that $\varepsilon_0 \ge 10\varepsilon$. Therefore, $d(w,x) \le \varepsilon_Y f_{t+1}(x)$.

Thus, the requirement of an $\varepsilon_Y$-sample is fulfilled by the vertex $w$ of $Y$.

Consider the case that $d(p,w) \le \lambda R_X(u)$. Since $X$ is an $(\varepsilon_X,\alpha_0)$-mesh, $d(p,u) \le 2R_X(u) \le 2\mu\varepsilon_X f_{t+1}(u)$. So $d(u,x) \le d(p,u) + d(p,x) \le 2\mu\varepsilon_X f_{t+1}(u) + \varepsilon f_{t+1}(x)$. The Lipschitzness of LFS implies that $f_{t+1}(u) \le (1 + 2\varepsilon + 4\mu\varepsilon_X) f_{t+1}(x) < 2 f_{t+1}(x)$. So $d(p,w) \le \lambda R_X(u) \le \mu\lambda\varepsilon_X f_{t+1}(u) = 2\mu\lambda\varepsilon_X f_{t+1}(x)$, and $d(w,x) \le d(p,w) + d(p,x) \le (2\mu\lambda\varepsilon_X + \varepsilon) f_{t+1}(x) \le \max\{4\mu\lambda\varepsilon_X, 2\varepsilon\} \cdot f_{t+1}(x) < \varepsilon_Y f_{t+1}(x)$. □

## 5 Decimation

Some vertices must be deleted in order to increase the inter-vertex distances. This will restore conditions C1 and C2 and the angle lower bound $\alpha_0 = \Omega(\lambda)$. Let $Y$ be the output of the refinement phase. Our idea is to define a *decimation radius* $\delta_v$ for each vertex $v$ of $Y$ sensitive to the triangle circumradii nearby, so that if we keep $v$, then the vertices at distance less than radius $\delta_v$ from $v$ should be deleted. We want the decimation radius to satisfy the following properties.

P1: For any two vertices $v$ and $u$, $\delta_v - \delta_u \le (\lambda/\ell) d(v,u)$.
P2: There exists a constant $\kappa_{\text{dec}}$ such that for each vertex $v$, $\delta_v \ge 4\lambda R_Y(v)$ and there exists a triangle $\tau$ in $Y$ such that $d(v,c_\tau) \le \kappa_{\text{dec}}\gamma_\tau$ and $\delta_v \le \frac{1}{15}\gamma_\tau$.

P1 ensures that the function is smooth, so after the decimation, the nearest vertex distances of close vertices are similar. P2 provides lower and upper bounds on the decimation radius.

Define the decimation radius $\delta_v = \max\{4\lambda\gamma_\tau - (\lambda/\ell)d(v,c_\tau) : \text{triangle } \tau \text{ in } Y\}$. It is easy to verify that it satisfies P1 and P2 with $\kappa_{\text{dec}} = 4\ell$ for $4\lambda \le \frac{1}{15}$. To evaluate the decimation radii for all vertices, we perform a breadth-first search from each triangle $\tau$, update the decimation radii at the vertices visited, and stop at edges $e$ where $4\lambda\gamma_\tau \le (\lambda/\ell)d(e,c_\tau)$.

Vertex deletions cannot be performed in an arbitrary order because this may produce tiny angles in an intermediate mesh that makes it impossible to apply Theorem 1(iii) in subsequent vertex deletions. To avoid this problem, we decimate vertex neighborhoods gradually in rounds as explained below.

**Algorithm 1** DELETE(mesh $T$, vertex $w$)

```
1: Flip edges so that all triangles incident to $w$ have almost empty diametric balls.
2: $H \leftarrow$ the plane through a triangles incident to $u$.
3: Remove all the triangles incident to $w$, and let $P$ be the polygonal hole.
4: $Q \leftarrow$ the projection of $P$ in $H$.
5: Compute the constrained Delaunay triangulation of $Q$.
6: Fill the hole $P$ by lifting these triangles.
```

Define $n_{\min}$ to be the smallest distance between two vertices in $Y$. Define $\delta_{\max} = \max\big\{\max\{\delta_v, 20(\sin\alpha_0)^{-1}\Delta_{t+1}(v)\} : \text{vertex } v \text{ in } Y.\big\}$. Place the vertices of $Y$ into lists $S_0, \ldots, S_m$ such that $S_i$ contains $v$ iff $2^i n_{\min} \le n_Y(v) < 2^{i+1} n_{\min}$, where $m = \max\{0, \lfloor\log_2(\delta_{\max}/n_{\min})\rfloor\}$. This can be done in $O(m+n)$ time as follows without assuming constant-time integral logarithm operations. We find $m$ by computing $\delta_{\max}$ in $O(n)$ time and computing the integral logarithm in $O(m)$ time in a brute force manner. Then initialize $m+1$ empty lists, and insert the vertex $v_{\min}$ with the smallest nearest vertex distance in $S_m$. Next, start a breadth-first search to traverse the mesh from $v_{\max}$. Notice that for any two adjacent vertices, their nearest vertex distances differ by a constant factor. So for each vertex encountered during the traversal, knowing at which list any of its adjacent vertices is placed allows us to locate its list in $O(1)$ time.

Then, we initialize a mesh $T$ to be $Y$ and decimate its vertices in $m+1$ rounds. In round $i$, we decimate the neighborhood of each vertex $v \in S_i$ as follows.

1. Remove $v$ from $S_i$. If $2^i n_{\min} \le \max\{\delta_v, 20(\sin\alpha_0)^{-1}\Delta_{t+1}(v)\}$, delete the vertices in $B(v, 2^{i+1}n_{\min})$ other than $v$ from $T$. Algorithm 1 explains how to delete a vertex from $T$. Deleted vertices are also removed from the lists $S_0, \ldots, S_m$.
2. Compute the nearest vertex distance $n_T(v)$ of $v$ by finding its nearest vertex in the updated mesh $T$. The nearest vertex of $v$ can be found by traversing the triangles intersecting $B(v, d(u,v))$ for any edge $uv$ incident to $v$ in $T$.
3. Compute the index $j$ such that $2^j n_{\min} \le n_T(v) < 2^{j+1} n_{\min}$. We will show in Lemma 16 that $j = i + O(1)$, so $j$ can be computed in $O(1)$ time. If $j > m$, then $n_T(v) \ge \delta_{\max}$, so we can drop $v$. If $j \le m$, we put $v$ in $S_j$ for future processing.

In other words, vertices within distances $2n_{\min}, 4n_{\min}, 8n_{\min}, \ldots$ from $v$ are deleted in rounds. After processing all vertices in $S_{m+1}$, we need to migrate the points in points($w$) for each vertex $w$ deleted.

In the end, the algorithm flips edges until none is flippable, and places every point that is not a vertex to the point list of its nearest vertex.

In the following, we show that the algorithm correctly decimates the neighborhood of each vertex (Lemma 12); and it does not over-decimate in the sense that each deleted vertex is at distance $O\big(\max\{\delta_v, 20(\sin\alpha_0)^{-1}\Delta_{t+1}(v)\}\big)$ from some vertex $v$ in $Z$ (Lemma 13).

**Lemma 12** *Let $Z$ be the output mesh of* DECIMATE($Y$). *The distance between any vertex $v$ to its nearest vertex in $Z$ is at least* $\max\{\delta_v, 20(\sin\alpha_0)^{-1}\Delta_{t+1}(v)\}$.

*Proof* Throughout the decimation procedure, we maintain the invariant that each vertex $v$ in $S_i$ has nearest vertex distance $n_Z(v) \geq 2^i n_{\min}$, and we drop it only when $2^i n_{\min} > \max\{\delta_v, 20(\sin\alpha_0)^{-1}\Delta_{t+1}(v)\}$. So the lemma follows. □

Consider a chronological sequence of vertices $(w_0, w_1, \ldots, w_g)$ that satisfies the following properties: (1) for $k \in [0, g]$, $w_k$ is a vertex in the initial mesh $Y$; (2) $w_g$ is a vertex of an intermediate mesh $T$ in DECIMATE($Y$); (3) for $k \in [1, g]$, $w_{k-1} \in B(w_k, 2^{i_k+1} n_{\min})$ and $w_{k-1}$ was deleted when we processed $w_k$ in some list $S_{i_k}$. We call the chronological sequence $(w_0, w_1, \ldots, w_g)$ a *deletion chain*. It has the following property.

**Lemma 13** *Let $(w_0, w_1, \ldots, w_g)$ be a deletion chain. Suppose that $w_g \in S_j$ when $w_{g-1}$ was deleted. If $2^j n_{\min} \leq \delta_{w_g}$, then $d(w_0, w_g) \leq 2^{j+2} n_{\min} \leq 4\delta_{w_g}$; otherwise, $d(w_0, w_g) \leq 80(\sin\alpha_0)^{-1}\Delta_{t+1}(w_g) \leq 0.4\varepsilon_0 f(w_g)$.*

*Proof* For $k \in [0, g-1]$, after deleting the vertices in $B(w_k, 2^{i_k+1} n_{\min}) \setminus \{w_k\}$ for some $i_k$, we cannot delete $w_k$ when examining another vertex in the same list $S_{i_k}$. Thus, $w_{k+1} \in S_{i_{k+1}}$ for some $i_{k+1} > i_k$ when $w_k$ was deleted. We have $d(w_0, w_g) \leq \sum_{k=1}^{g} d(w_{k-1}, w_k) \leq \sum_{i=0}^{j} 2^{i+1} n_{\min} \leq 2^{j+2} n_{\min}$. Since $w_g \in S_j$ when $w_{g-1}$ was deleted, we have $2^j n_{\min} \leq \max\{\delta_{w_g}, 20(\sin\alpha_0)^{-1}\Delta_{t+1}(w_g)\}$. If $2^j n_{\min} \leq \delta_{w_g}$, we obtain $d(w_0, w_g) \leq 4\delta_{w_g}$. Suppose that $2^j n_{\min} \leq 20(\sin\alpha_0)^{-1}\Delta_{t+1}(w_g)$. Since we assume that $\Delta_{t+1}(w_g) \leq (0.005\varepsilon_0 \sin\alpha_0) f_{t+1}(w_g)$, we get $d(w_0, w_g) \leq 80(\sin\alpha_0)^{-1}\Delta_{t+1}(w_g) \leq 0.4\varepsilon_0 f_{t+1}(w_g)$. □

In DELETE($T$, $w$), the first step is to flip edges so that the triangles incident to $w$ have almost empty diametric balls. The next technical result shows the property ensured by this step.

**Lemma 14** *Consider the intermediate mesh $T$ during* DECIMATE($Y$). *Suppose that $\ell$ is sufficiently large and $\lambda$ is sufficiently small. (Their settings depend on the hidden constant in assumption* (i) *in our motion model.) Suppose that all the angles are at least some constant before calling* DELETE($T$, $w$). *Then after the execution of line 1 of* DELETE($T$, $w$), *for any vertex $v$ of $T$, $n_T(v) \geq \frac{1}{14}\lambda^2 R_T(v)$.*

*Proof* Let $\tau$ be the triangle incident to $v$ such that $\gamma_\tau = R_T(v)$. Recall that $Y$ stands for the output mesh of the insertion phase. Let $\sigma$ be the triangle in $Y$ closest to $\varphi_{\Sigma_{t+1}}(c_\tau)$. By Lemma 9, we have $d(c_\tau, \sigma) < 0.1(\gamma_\tau + \gamma_\sigma)$. Let $a$ be any vertex of $\sigma$. The distance $d(a, c_\tau)$ is at most $2.1\gamma_\sigma + 0.1\gamma_\tau$.

Suppose that the vertices of $\sigma$ lie outside $B(c_\tau, 0.5\gamma_\tau)$. Then, $\gamma_\sigma \geq 0.45\gamma_\tau$ by Lemma 9. We have $d(a, v) \leq d(a, c_\tau) + d(v, c_\tau) \leq 2.1\gamma_\sigma + 0.1\gamma_\tau + \gamma_\tau < 5\gamma_\sigma$. Then, for $\ell \geq 54$, Lemma 11(i) implies that $n_T(v) \geq n_Y(v) \geq \frac{1}{6}\lambda^2\gamma_\sigma \geq \frac{1}{14}\lambda^2\gamma_\tau$.

Suppose that some vertex $a$ of $\sigma$ lies in $B(c_\tau, 0.5\gamma_\tau)$. There is a deletion chain $(a, \ldots, u, w)$, where $w$ is a vertex in the current $T$. Since $B(c_\tau, 0.8\gamma_\tau)$ contains no vertex of $T$, $d(w, c_\tau) \geq 0.8\gamma_\tau$. So we have $d(a, w) \geq d(w, c_\tau) - d(a, c_\tau) \geq 0.3\gamma_\tau$

Hence, $d(v,w) \le d(v,c_\tau) + d(a,c_\tau) + d(a,w) < 1.5\gamma_\tau + d(a,w) \le 6d(a,w)$. Assume that $u$ was deleted when we processed $w$ in some list $S_i$.

Suppose that $2^i n_{\min} \le \delta_w$. Lemma 13 implies that $\delta_w \ge \frac{1}{4} d(a,w)$. Combining it with the previous inequality and the smoothness of the decimation radii, we have $\delta_v \ge \delta_w - (\lambda/\ell)d(v,w) \ge \delta_w - (24\lambda/\ell)\delta_w > \delta_w/2$. If we dropped $v$ before processing $w$ (i.e., did not place $v$ in any list for further processing), we must have $n_T(v) \ge \delta_v$, which is greater than $\frac{1}{2}\delta_w$. Then, Lemma 13 implies that $n_T(v) > \delta_w/2 \ge d(a,w)/8 \ge 0.3\gamma_\tau/8$, which is greater than $\lambda^2\gamma_\tau/14$. If we did not drop $v$ before processing $w$, we must have $n_T(v) \ge 2^i n_{\min}$. Then, Lemma 13 implies that $n_T(v) \ge 2^i n_{\min} \ge d(a,w)/4 \ge 0.3\gamma_\tau/4$, which is also greater than $\lambda^2\gamma_\tau/14$.

If $2^i n_{\min} > \delta_w$, then $d(a,w) \le 80(\sin\alpha_0)^{-1}\Delta_{t+1}(w)$ by Lemma 13. By assumption, $\Delta_{t+1}(w) \le c\Delta_t(w)$ for some constant $c$. Condition $\hat{\mathrm{C}}1$ and the working of REFINE ensure that $R_Y(w) \ge n_Y(w) \ge \frac{18\Delta_t(w)}{\sin\alpha_0} \ge \frac{18\Delta_{t+1}(w)}{c\sin\alpha_0} \ge \frac{9d(a,w)}{40c}$. Recall that $d(a,w) \ge 0.3\gamma_\tau$. By property P2 of decimation radii, we have $\delta_w \ge 4\lambda R_Y(w) > \frac{9}{10c}\lambda d(a,w) \ge \frac{27}{100c}\lambda\gamma_\tau$. Set $\ell$ to be a constant greater than $20c/3$. Then $\delta_v \ge \delta_w - \frac{\lambda}{\ell}\cdot d(w,v) \ge \delta_w - \frac{\lambda}{\ell}\cdot 6d(a,w) \ge \delta_w - \frac{\lambda}{\ell}\cdot\frac{10c}{9\lambda}\cdot 6\delta_w \ge \delta_w/2 \ge \frac{27}{200c}\lambda\gamma_\tau$. Then, we can invoke the same analysis in the previous paragraph and show that $n_T(v) \ge \lambda^2\gamma_\tau/14$ by setting $\lambda$ small enough. □

By Lemma 14, all triangles have angles at least $\arcsin\left(\frac{1}{28}\lambda^2\right)$ after the execution of line 1 in DELETE. This allows us to show that the deletion of a single vertex preserves an $\Omega(1)$ lower bound on angles. Then, inductively, any intermediate mesh during the decimation has constant lower bound on its smallest angle.

**Lemma 15** *At any time during* DECIMATE, DELETE$(T,w)$ *produces angles that are at least some constant.*

*Proof* Consider the deletion of a vertex $w$ from $T$. We first show that the distance between any two vertices adjacent to $w$ is at least $\Omega(n_T(w))$. Take any two vertices $a$ and $b$ incident to $w$. If $\angle awb \ge \pi/2$, then $d(a,b) \ge d(a,w) \ge n_T(w)$. Suppose $\angle awb < \pi/2$. Then $d(a,b) \ge d(a,w)\cdot\sin\angle awb \ge n_T(w)\cdot\sin\angle awb$. Since each angle at $w$ is at least $\arcsin\left(\frac{1}{28}\lambda^2\right)$ and triangles incident to $w$ are fairly flat, we have $\angle awb \ge \arcsin\left(\frac{1}{28}\lambda^2\right)$, and hence $d(a,b) \ge \frac{1}{28}\lambda^2 n_T(w)$.

Let $Q$ be the projection of the triangles incident to $w$ to the plane of one of these triangles. By Lemma 7, $Q$ is a simple polygon. The distance between any two vertices adjacent to $w$ is shortened by a constant factor by the projection. So the distance between any two vertices of $Q$ is $\Omega(\lambda^2 n_T(w))$. In the following, we prove that any triangle $\tau$ in the constrained Delaunay triangulation of $Q$ has circumradius at most $\frac{112}{\lambda^2}R_T(w)$. Then, the ratio of the shortest edge length of $\tau$ to $\gamma_\tau$ is $\Omega\left(\lambda^4\cdot\frac{n_T(w)}{R_T(w)}\right) = \Omega(1)$, which implies that the smallest angle in $\tau$ is $\Omega(1)$.

Assume to the contrary that $\gamma_\tau > \frac{112}{\lambda^2}R_T(w)$. Refer to Fig. 2. Since the distance between $w$ and any vertex incident to $w$ is at most $2R_T(w)$, each edge of $\tau$ is at most $4R_T(w)$. The large circumradius of $\tau$ implies that it has an obtuse angle. Moreover, $c_\tau$ lies outside $Q$ because $\gamma_\tau$ is larger than the largest distance between two vertices

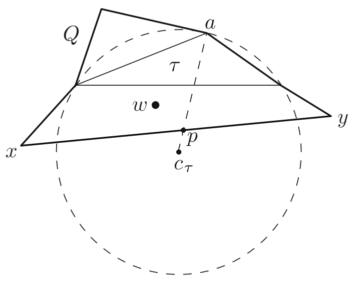


**Fig. 2** $\tau$ is a constrained Delaunay triangle of the projected polygon $Q$

of $Q$. Let $a$ be the vertex of $\tau$ with an obtuse angle. Since $Q$ contains $\tau$, the line segment $ac_\tau$ must intersect at least one edge of $Q$. Let $xy$ be the one closest to $a$. Let $p$ denote the intersection point $xy \cap ac_\tau$. Both $x$ and $y$ avoid the interior of the circumcircle of $\tau$ by the constrained Delaunay condition. The line segment $ap$ is inside $Q$ by our choice of $xy$. The polygon $Q$ is star-shaped with $w$ in the kernel, so the line segment $wp$ lies inside $Q$ as well. Thus, $w$ and $a$ are on the same side of $xy$. If $\angle wxy > \angle axy$ and $\angle wyx > ayx$, the triangle $wxy$ would contain $a$ in its interior, which is impossible. So $\angle wxy \le \angle axy$ or $\angle wyx \le ayx$. Assume the former is true.

Recall that all angles in the triangles incident to $w$ are at least $\arcsin\left(\frac{1}{28}\lambda^2\right)$. They can be affected by the projection only slightly. So $\angle axy \ge \angle wxy > \arcsin(\frac{1}{56}\lambda^2)$. Since $x$ and $y$ are outside the circumcircle of $\tau$ and $xy$ separates $a$ and $c_\tau$, we have $\gamma_\tau \le \gamma_{axy} \le \frac{d(a,y)}{2\sin\angle axy} \le \frac{28d(a,y)}{\lambda^2}$. Both $a$ and $y$ are at distance at most $2R_T(w)$ from $w$, so $d(a,y) \le 4R_T(w)$. It follows that $\gamma_\tau \le \frac{112}{\lambda^2}R_T(w)$, a contradiction. □

**Lemma 16** *Let $v$ be a vertex in list $S_i$. Lines 14–15 of* DECIMATE *place $v$ in list $S_j$ for some $j = i + O(1)$.*

*Proof* Let $T$ be the current mesh. Consider the mesh $T'$ right before the last vertex $u$ in $B(v, 2^i n_{\min}) \setminus \{v\}$ is deleted. Let $w \ne v$ be a vertex adjacent to $u$ in $T'$. By Lemma 15, $d(w,u) \le cn_{T'}(u) \le cd(u,v)$ for some constant $c$. So $d(v,w) \le d(v,u) + d(u,w) \le (c+1)d(v,u) \le (c+1)2^i n_{\min}$, and the nearest vertex distance of $v$ after deleting $u$ is at most $(c+1)2^i n_{\min}$.

If $u$ is deleted during the decimation of the neighborhood of $v$, then $w$ remains a vertex in $T$. Therefore, $n_T(n) \le d(v,w) \le (c+1)2^i n_{\min}$, and $j = i + O(\log c)$.

Suppose that $u$ is deleted during the decimation of the neighborhood of another vertex. Consider the deletion chain $(w_0, w_1, \ldots, w_g)$ that contains $u$, where $w_g$ is a vertex in $T$. The neighborhood of $w_g$ is decimated in round $i$ or earlier, so we have $d(u, w_g) \le \sum_{k=0}^{g-1} d(w_k, w_{k+1}) < 2d(w_{g-1}, w_g) \le 2^{i+1}n_{\min}$. So $n_T(v) \le d(v, w_g) \le d(v,u) + d(u, w_g) \le 3 \cdot 2^i n_{\min}$, meaning that $j \le i + 2$. □

Next, we show that the vertices of the output mesh $Z$ of DECIMATE is still a dense sample despite the vertex deletions.

**Lemma 17** *Let $Y$ be the output of the refinement phase, and $Z =$* REFINE$(Y)$. *The vertices of $Z$ form an $\varepsilon_Z$-sample, where $\varepsilon_Z = \max\{\varepsilon_0, 15\mu\lambda\varepsilon_X\}$.*

*Proof* Take any point $x \in \Sigma_{t+1}$. Let $w$ be the nearest vertex in $Y$ to $x$. So $d(w,x) \le \varepsilon_Y f_{t+1}(x)$ by Lemma 11. If $w$ is a vertex of $Z$, we are done. Suppose that $w$ was deleted. Then by Lemma 13, there exists a vertex $v$ in $Z$ such that $d(v,w) \le \max\{4\delta_v, 0.4\varepsilon_0 f_{t+1}(v)\}$.

Suppose $d(v,w) \le 0.4\varepsilon_0 f_{t+1}(w)$. Then, $d(x,v) \le d(x,w) + d(w,v) \le \varepsilon_Y f_{t+1}(x) + 0.4\varepsilon_0 f_{t+1}(v)$. The Lipschitzness of LFS implies that $d(x,v) \le \frac{0.4\varepsilon_0+\varepsilon_Y}{1-0.4\varepsilon_0} f_{t+1}(x) \le (0.7\varepsilon_0 + \varepsilon_Y)\cdot f_{t+1}(x)$. Since $\varepsilon_Y \le \max\{0.3\varepsilon_0, 4\mu\lambda\varepsilon_X\}$ by Lemma 11, it can be verified that $0.7\varepsilon_0 + \varepsilon_Y \le \max\{\varepsilon_0, 15\mu\lambda\varepsilon_X\}$. So $d(x,v) \le \varepsilon_Z f_{t+1}(x)$.

Consider the case that $d(v,w) \le 4\delta_v$. By property P2 of decimation radii, we can find a triangle $\tau$ in $Y$ such that $d(v,c_\tau) \le \kappa_{\text{dec}}\gamma_\tau$ and $\delta_v \le \frac{1}{15}\gamma_\tau$. Let $a$ be a vertex of $\tau$. By Theorem 1(ii), $\gamma_\tau \le 1.5\varepsilon_Y f_{t+1}(a)$. Therefore, we have $d(v,c_\tau) \le 1.5\kappa_{\text{dec}}\varepsilon_Y f_{t+1}(a)$ and $\delta_v \le \frac{1}{15}\gamma_\tau \le 0.1\varepsilon_Y f_{t+1}(a)$. The Lipschitzness of LFS implies that $\delta_v \le 0.1\varepsilon_Y\big(1 + 1.5(\kappa_{\text{dec}} + 1)\varepsilon_Y\big) f_{t+1}(v) \le 0.2\varepsilon_Y f_{t+1}(v)$. Thus, $d(x,v) \le 4\delta_v + \varepsilon_Y f_{t+1}(x) \le 0.8\varepsilon_Y f_{t+1}(v) + \varepsilon_Y f_{t+1}(x) < 3\varepsilon_Y f_{t+1}(x) \le \varepsilon_Z f_{t+1}(x)$. □

We are ready to show that DECIMATE restores the angle lower bound to $\alpha_0$, while ensuring a good sampling density.

**Lemma 18** *Let $X$ be an $(\varepsilon_X, \alpha_1)$-mesh satisfying* $\hat{\text{C}}1$–$\hat{\text{C}}3$ *for some constants $\varepsilon_X$ and $\alpha_1$. Let $Y$ be the output mesh of* REFINE$(X)$. *Assume that the decimation radii for the vertices in $Y$ are known. Then,* DECIMATE$(Y)$ *runs in linear time and produces an $(\varepsilon_Z, \alpha_0)$-mesh $Z$ that satisfies conditions* C1–C3, *where $\varepsilon_Z = \max\{\varepsilon_0, 15\mu\lambda\varepsilon_X\}$ and $\alpha_0 = \Omega(\lambda)$.*

*Proof* The angles in $Y$ are at least some constant by Lemma 11, so deleting the first vertex can be done in $O(1)$ time. Then by Lemma 15, all angles in the mesh after the deletion are still at least some constant. So Lemmas 14 and 15 are applicable inductively, and hence the deletion of each single vertex can be deleted in $O(1)$ time. Thus, we only need to bound the total number of migrations of vertices in the $m+1$ rounds of vertex deletions. Let $\tau$ be the triangle in $Y$ with the largest circumradius. Let $u$ be the vertex in $Y$ with the largest $\Delta_{t+1}(u)$. So, $m \le \log\left(\frac{1}{n_{\min}}\max\big\{\frac{1}{15}\gamma_\tau,\ 20(\sin\alpha_0)^{-1}\Delta_{t+1}(u)\big\}\right) = \log\frac{O(\gamma_\tau+\Delta_t(u))}{n_{\min}} = \log\frac{O(\gamma_\tau+R_Y(u))}{n_{\min}} = O\big(\log(\gamma_\tau/n_{\min})\big)$. Since each angle in $Y$ is at least some constant, the circumradii of two adjacent triangles differ by a constant factor, implying that the radio of the largest to the smallest circumradii is at most $2^{O(n)}$. Therefore, $m = O(n)$. Consider the processing of $v \in S_i$ in a round. In the case that $i < j \le m+1$, we deleted some vertex $p$ in a previous round where $d(p,v)$ equals the old $n_M(v)$. We charge to $p$ the migration of $v$ to $S_j$. We can show that $p$ is charged only $O(1)$ time as in bounding the degree of a planar minimum spanning tree by six. Therefore, the total number of migrations is $O(n)$, implying that the total running time is $O(n)$.

Condition C1 follows from Lemma 12.

Condition C2 requires that for any vertex $v$ and any triangle $\tau$ in $Z$, if $v \in B(c_\tau, \ell\gamma_\tau)$, then $n_M(v) > \lambda\gamma_\tau$. Let $\sigma$ be the triangle in $Y$ closest to $\varphi_{\Sigma_{t+1}}(c_\tau)$. Let $a$ be any vertex of $\sigma$. By Lemma 9, $d(a, c_\tau) < 2.1\gamma_\sigma + 0.1\gamma_\tau$. Suppose that the vertices of $\sigma$ lie outside $B(c_\tau, 0.7\gamma_\tau)$. Then, Lemma 9 implies that $\gamma_\sigma > 0.6\gamma_\tau$, and we have $d(a, v) \le d(a, c_\tau) + d(v, c_\tau) \le 2.1\gamma_\sigma + 0.1\gamma_\tau + \ell\gamma_\tau < 2\ell\gamma_\sigma$. By the properties of decimation radii, $\delta_a \ge 4\lambda\gamma_\sigma$ and $n_Z(v) \ge \delta_v \ge \delta_a - (\lambda/\ell)d(a, v) \ge 2\lambda\gamma_\sigma > \lambda\gamma_\tau$. If some vertex $a$ of $\sigma$ lies inside $B(c_\tau, 0.7\gamma_\tau)$, $d(a, v) \le \ell\gamma_\tau + 0.7\gamma_\tau < 2\ell\gamma_\tau$. We can invoke the same proof as that in Lemma 14: identify the deletion chain $(a, \ldots, w)$ where $w$ is a vertex of $Z$, and relate $\delta_v$ and $\delta_w$. We have $d(w, c_\tau) \ge 0.8\gamma_\tau$ and $0.1\gamma_\tau \le d(a, w) \le 4\delta_w$. Set $\lambda$ to be a small enough number. We have $\delta_v \ge \delta_w - (\lambda/\ell)d(v, w) \ge \delta_w - (\lambda/\ell)(d(v, a) + d(a, w)) \ge (1 - 4\lambda/\ell)\delta_w - (\lambda/\ell)(2\ell\gamma_\tau) > \frac{1}{80}\gamma_\tau > \lambda\gamma_\tau$.

Condition C3 is satisfied simply because we put every point $p \in P_{t+1}$ that is not a vertex of $M$ in points($w$) such that $w$ is the nearest vertex of $M$ to $p$.

Finally, by Lemma 17, the vertices of the output mesh $Z$ form an $\varepsilon_Z$-sample of $\Sigma_{t+1}$, and by C2, $\alpha_0$ can be set to $\arcsin(\lambda/2)$, so $Z$ is an $(\varepsilon_Z, \alpha_0)$-mesh. □

**Lemma 19** *Let $K_{t+1}$ be the deformed mesh at time $t + 1$, which is an $(\varepsilon_1, \alpha_1)$-mesh satisfying* $\hat{\text{C}}1 - \hat{\text{C}}3$. *We can iterate* Refine *and* Decimate $O(1)$ *times to return an $(\varepsilon_0, \alpha_0)$-mesh $M_{t+1}$ that satisfies* C1–C3.

*Proof* Our mesh update procedure iterates Refine and Decimate for $O(\log \frac{1}{\varepsilon_0}) = O(1)$ times. The angle bound $\alpha_0$ and the conditions *C1–C3* are enforced by Lemma 18. Consider the sampling condition. The vertex set of $K_{t+1}$ is an $\varepsilon_1$-sample, so after the first iteration, we obtain a mesh whose vertex set is an $\max\{\varepsilon_0, O(\lambda\varepsilon_1)\}$-sample by Lemma 18. Another iteration gives a $\max\{\varepsilon_0, O(\lambda^2\varepsilon_1)\}$-sample. Therefore, after $O(\log \frac{1}{\varepsilon_0})$ iterations, we can obtain an $(\varepsilon_0, \alpha_0)$-mesh. □

Finally, we obtain our deforming mesh maintenance result as stated in the theorem below. The $O(n)$ running time bound is obtained by using an octree to speed up the computation of the decimation radii.

**Theorem 2 ([7, 17])** *Consider the simulation of a deforming surface, specified by $n$ moving sample points, that progresses in unit time steps. Suppose that the sample points form an $\varepsilon$-sample for a sufficiently small $\varepsilon$ throughout the simulation. There exist constants $\varepsilon_0 \in (10\varepsilon, 1)$ and $\alpha_0 > 0$ such that an $(\varepsilon_0, \alpha_0)$-mesh can be built before the simulation begins and if the deformation is smooth, an $(\varepsilon_0, \alpha_0)$-mesh can be restored in $O(n)$ time at each subsequent time step.*

## 6 Experiments

We built a prototype of the mesh maintenance algorithm and experimented with two test cases on a machine with Intel Xeon E5450 3GHz CPU and 16G memory. Two iterations of the insertion and deletion phases were run on the two test cases, and we found that the mesh size and quality are nearly identical at the end of the first

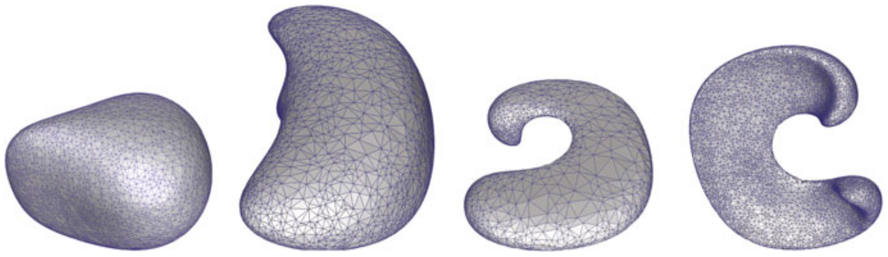


**Fig. 3** The *leftmost screen shots* are taken at time 0.2, 0.4 and 0.7 respectively. The *rightmost screen shot* is the back of the object at time 0.7

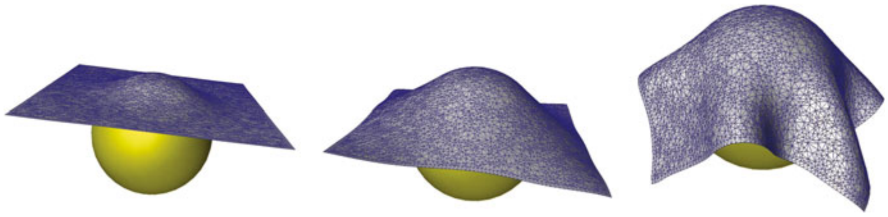


**Fig. 4** A piece of cloth falling on a ball

and second iterations. Since the iterations aim to increase the mesh density, it is reasonable to stop if the mesh size stays nearly the same.

The first test case is a deformation of a sphere [12, 20]. The velocity field is defined as: $v_x = 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)$, $v_y = -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)$, and $v_z = -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)$. Initially we have 10K random samples on a sphere centered at $(0.35, 0.35, 0.35)$ with radius 0.15. After the simulation starts, the points move with the velocity determined by its current position. Figure 3 shows how the object deforms. The rightmost two images in Fig. 3 show the front and back views of the surface mesh at time 0.7. As the object deforms, the front side is stretched and the sample points move to the back side of object. From the screen shots, one can see that our algorithm is indeed adaptive to the sampling: the triangles in the front are much larger than those at back. The average number of mesh vertices is around 6K, 90 % of angles are in the range $[30°, 120°]$, less than 0.02 % of them are less than 15°, and none is smaller than 11°. The average update time per time step is less than 0.36 s. Reduction in running time seems possible as we did not optimize the code. For comparison, we reconstructed the mesh from the sample points using Cocone [3]. Cocone took 3.7 s on average per time step.

We also ran our algorithm with different numbers of sample points and observed that the average update time increases linearly with the number of sample points.

Our second test case is the deformation of a piece cloth as it falls on a sphere [4]. 10K sample points are used. The cloth boundary is preserved by not allowing any boundary edge to be flipped. Figure 4 shows the results. The statistics is highly similar to the first test case.

**Acknowledgements** Research supported by the Research Grant Council, Hong Kong, China (project no. 612107).

## References

1. Adams, B., Pauly, M., Keiser, R., Guibas, L.J.: Adaptively sampled particle fluids. ACM Trans. Graph. **26**, 3–48 (2007)
2. Amenta, N., Bern, M.: Surface reconstruction by Voronoi filtering. Discrete Comput. Geom. **22**, 481–504 (1999)
3. Amenta, N., Choi, S., Dey, T.K., Leekha, N.: A simple algorithm for homeomorphic surface reconstruction. Int. J. Comput. Geom. Appl. **12**, 125–141 (2002)
4. Baraff, D., Witkin, A.: Large steps in cloth simulation. In: SIGGRAPH, pp. 43–54 (1998)
5. Beer, G., Smith, I., Duenser, C.: The Boundary Element Method with Programming. Springer, New York (2008)
6. Bredno, J., Lehmann, T.M., Spitzer, K.: A general discrete contour model in two, three, and four dimensions for topology-adaptive multichannel segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **25**, 550–563 (2003)
7. Cheng, S.-W., Jin, J.: Edge flips and deforming surface meshes. In: Proceedings of the 28th Annual Symposium on Computational Geometry, pp. 331–340 (2011)
8. Cheng, S.-W., Jin, J.: Edge flips in surface meshes. Manuscript. http://www.cse.ust.hk/faculty/scheng/pub/deform.pdf (2013)
9. Cheng, S.-W., Jin, J., Lau., M.-K.: A fast and simple surface reconstruction algorithm. In: Proceedings of the 28th Annual Symposium on Computational Geometry, pp. 69–78 (2012)
10. Delingette, H.: Towards realistic soft tissue modeling in medical simulation. In: Proceedings of the IEEE: Special Issue on Surgery Simulation, pp. 512–523 (1998)
11. Dey, T.K.: Curve and Surface Reconstruction: Algorithms with Mathematical Analysis. Cambridge University Press, New York (2006)
12. Enright, D., Fedkiw, R., Ferziger, J., Mitchell, I.: A hybrid particle level set method for improved interface capturing. J. Comput. Phys. **183**, 83–116 (2002)
13. Giesen, J., Wagner, U.: Shape dimension and intrinsic metric from samples of manifolds. Discrete Comput. Geom. **32**, 245–267 (2004)
14. Glimm, J., Grove, J.W., Li, X.L., Tan, D.C.: Robust computational algorithms for dynamic interface tracking in three dimensions. SIAM J. Sci. Comput. **21**, 2240–2256 (1999)
15. Hall, W.S.: The Boundary Element Method. Kluwer Academic Publishers, Dordrecht (1994)
16. Jiao, X.: Face offsetting: a unified approach for explicit moving interfaces. J. Comput. Phys. **220**, 612–625 (2007)
17. Jin, J.: Surface reconstruction and deformation. Doctoral Dissertation, The Hong Kong University of Science and Technology (2012)
18. Khayat, R.E.: Three-dimensional boundary element analysis of drop deformation in confined flow for Newtonian and viscoelastic systems. Int. J. Numer. Methods Fluids **34**, 241–275 (2000)
19. Koch, R.K., Gross, M.H., Carls, F.R., von Büren, D.F., Fankhauser, G., Parish, Y.I.H.: Simulating facial surgery using finite element methods. In: SIGGRAPH, pp. 421–428 (1996)
20. LeVeque, R.J.: High-resolution conservative algorithms for advection in incompressible flow. SIAM J. Numer. Anal. **33**, 627–665 (1996)
21. Liu, T., Shen, D., Davatzikos, C.: Deformable registration of cortical structures via hybrid volumetric and surface warping. NeuroImage **22**, 1790–1801 (2004)
22. Müller, M., Charypar, D., Gross, M.: Particle-based fluid simulation for interactive applications. In: SIGGRAPH, pp. 154–159 (2003)
23. Osher, S., Sethian, J.: Fronts propagating with curvature-dependent speed: algorithms based on Hamiltonian Jacobi formulations. J. Comput. Phys. **79**, 12–49 (1988)

24. Pauly, M., Keiser, R., Adams, B., Dutré, P., Gross, M., Guibas, L.J.: Meshless animation of fracturing solids. ACM Trans. Graph. **24**, 957–964 (2005)
25. Plantinga, S., Vegter, G.: Isotopic meshing of implicit surfaces. Vis. Comput. **23**, 45–58 (2007)
26. Pons, J., Boissonnat, J.D.: Delaunay deformable models: topology-adaptive meshes based on the restricted Delaunay triangulation. In: CVPR, 1–8 (2007)
27. Sethian, J.: Level Set Methods and Fast Marching Methods. Cambridge University Press, Cambridge (1999)
28. Tryggvason, G., Bunner, B., Esmaeeli, A., Juric, D., Al-Rawahi, N., Tauber, W., Han, J., Nas, S., Jan, Y.-J.: A front-tracking method for the computations of multiphase flow. J. Comput. Phys. **169**(2), 708–759 (2001)
29. Volino, P., Magnenat-Thalmann, N.: Comparing efficiency of integration methods for cloth simulation. In: Proceedings of the International Conference on Computer Graphics, pp. 265–272 (2001)
30. Wojtan, C., Thüey, N., Gross, M., Turk, G.: Deforming meshes that split and merge. ACM Trans. Graph. **28** (2009). Article 76

# An Optimization Based Method for the Construction of 2D Parameterizations for Isogeometric Analysis with T-Splines

**José Iván López, Marina Brovka, José María Escobar, José Manuel Cascón, and Rafael Montenegro**

**Abstract** We present a new strategy, based on the idea of the meccano method and a novel T-mesh optimization procedure, to construct a T-spline parameterization of 2D geometries for the application of isogeometric analysis. The proposed method only demands a boundary representation of the geometry as input data. The algorithm obtains, as a result, high quality parametric transformation between 2D objects and the parametric domain, the unit square. First, we define a parametric mapping between the input boundary of the object and the boundary of the parametric domain. Then, we build a T-mesh adapted to the geometric singularities of the domain in order to preserve the features of the object boundary with a desired tolerance. The key of the method lies in defining an isomorphic transformation between the parametric and physical T-mesh finding the optimal position of the interior nodes by applying a new T-mesh untangling and smoothing procedure. Bivariate T-spline representation is calculated by imposing the interpolation conditions on points sited both on the interior and on the boundary of the geometry. Proposed method also permits the modeling of objects with embedded geometries that can be used to solve problems with domains composed of several materials. Application of the isogeometric analysis to these type of domains are presented. The effectiveness of the proposed technique is shown in several examples.

J.I. López • M. Brovka • J.M. Escobar (✉) • R. Montenegro
University Institute for Intelligent Systems and Numerical Applications in Engineering, SIANI, University of Las Palmas de Gran Canaria, Las Palmas de Gran Canaria, Spain
e-mail: jilopez@siani.es; mbrovka@siani.es; jmescobar@siani.es; rmontenegro@siani.es

J.M. Cascón
Department of Economics and History of Economics, Faculty of Economics and Management, University of Salamanca, Salamanca, Spain
e-mail: casbar@usal.es

S. Perotto, L. Formaggia (eds.), *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, SEMA SIMAI Springer Series 5,
DOI 10.1007/978-3-319-06053-8_5

## 1 Introduction

An open problem in the context of isogeometric analysis [1] is how to obtain a spline parameterization of a complex computational domain from the CAD description of its boundary.

Parameterization is suitable for analysis if it does not have self-intersection, i.e., the Jacobian is strictly positive at any point of the computational domain. Moreover, in order to expect a high accuracy in numerical results it is necessary to obtain a good quality parameterization. Orthogonality and uniformity of isoparametric curves are desirable for the tensor-product structured parameterization. This task is not trivial and can be very time-consuming. For application of IGA it is essential to have an efficient method to construct T-spline parameterization. In the present work we investigate this problem for planar and embedded geometries.

There are only a few works addressing this problem. In [26], the parameterization is found by solving a constraint optimization problem for the control points of a planar B-spline surface. Constraints are defined by imposing injectivity sufficient conditions in terms of control points, and the optimization consists in the minimization of some energy functions in order to reach a good orthogonality and uniformity of the parametric mapping. Another similar technique was proposed by these authors in [27]. They use a harmonic mapping obtained by solving an optimization problem for the control points. Additional term is added to the objective function in order to improve the quality where needed. The use of harmonic mapping is a common characteristic of several works dealing with 2D and 3D parameterization methods [20–22].

In this paper, we propose a different approach where the parameterization is accomplished by transforming isomorphically a T-mesh from the parametric domain to the physical one. The construction of this transformation is mainly based on a simultaneous T-mesh untangling and smoothing procedure.

As far as we know, the only case of performing mesh untangling and smoothing procedure for a T-mesh, in order to construct T-spline representation of 3D domains, was described in [25, 28]. They remove tangled elements by maximizing the worst Jacobian. Smoothing is performed by moving each node towards the mass center of its neighboring elements. Another strategy for optimizing a grid with hanging nodes was described in [3]. Mesh optimization is carried out via a global grid smoothing functional based on shape quality and size control metrics for the elements of the mesh. Some constraints are imposed on the position of hanging nodes in order to place them in the middle point of their edge.

In our previous works [13, 14] we constructed the physical T-mesh of the solid using a volumetric parameterization obtained by deforming a tetrahedral mesh of the solid. In general, this approach does not provide an optimal T-mesh quality in the sense of its uniformity and orthogonality. Now, we propose a different approach where the optimization is applied directly to the T-mesh.

Our technique is simple and easy to implement. Satisfactory results are obtained with low computational effort for a variety of complex geometries.

This paper is organized as follows. In next section we describe the main steps of the proposed algorithm and explain the process of boundary parameterization and construction of a T-mesh adapted to the singularities of the object boundary. Section 3 describes the simultaneous T-mesh untangling and smoothing procedure that leads to the construction of a high quality T-mesh of the object. The modeling of the geometry by means of bivariate T-splines and a quality improvement strategy is developed in Sect. 4. Construction of embedded geometries is described in Sect. 5. In Sect. 6 we present applications of the proposed method to some 2D domains, including the resolution of the Poisson equation by means of the isogeometric analysis. Finally, in Sect. 7 we present the conclusions and set out some challenges.

## 2 General Scheme of the Method

In this section we summarize the main steps of our method in order to facilitate its understanding. Some ideas of the method are taken from our previous works on mesh untangling and smoothing and the meccano method [6–8, 10, 12, 23, 24], but they have been adapted to the requirements of the present work.

The algorithm includes the following stages:

1. *Boundary parameterization and construction of an adapted T-mesh:* A bijective correspondence between the input boundary of the object and the boundary of the parametric domain is defined. Then, an adapted T-mesh is generated by refining the initial mesh in order to approximate the geometry with a prescribed tolerance. During this process, the boundary nodes of the parametric domain are mapped to the boundary of the object.
2. *T-Mesh optimization:* We relocate the inner nodes of the T-mesh by applying a simultaneous mesh untangling and smoothing procedure. A previous relocation of the inner nodes is accomplished in order to facilitate this task.
3. *Construction of a T-spline representation of the geometry:* The T-spline parameterization is obtained by imposing interpolation conditions. As interpolation points, we take the vertices of the physical T-mesh obtained after the optimization process and other necessary additional points.
4. *Adaptive refinement to improve the mesh quality:* If the quality of the T-mesh is not satisfactory, we apply an adaptive refinement in order to increase the degree of freedom in the areas with high distortion. Then, we return to step 2 and repeat the process until reaching a good T-spline parameterization.

In the first stage, to define a parametric mapping between the input boundary of the object and the boundary of the parametric domain, the unit square, we have to select four points of the boundary that will correspond to the four corners of the square. These points divide the input boundary into four parts that are mapped via chord-length parameterization into its corresponding edge of the square, as shown in Fig. 1. Next, we construct an adapted T-mesh that approximates the input boundary with a prescribed tolerance $\epsilon$. To do that, an approximation error is calculated for

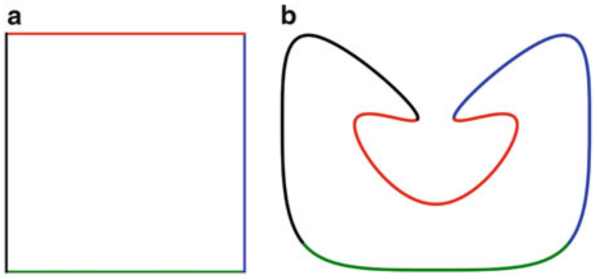


**Fig. 1** Corners selection and boundary parameterization. (**a**) Boundary of the parametric domain; (**b**) physical boundary where each part is mapped to a parametric edge. The *colors* represent the correspondence between boundaries

each boundary cell and the cell is refined if this error is greater than $\epsilon$. More details about the boundary approximation can be found in our previous work [4].

As result of the two first stages, the position of the boundary nodes in the physical domain are known and the position of the inner nodes will be defined by means of the T-mesh optimization procedure developed in next section.

Figure 2 illustrates the main stages of the algorithm applied to a *Test* geometry.

## 3 T-Mesh Optimization

The key of the proposed method lies in the optimization procedure that allows to obtain a high quality physical T-mesh used to construct the T-spline representation of the object.

It is preferable to perform a previous relocation of the inner nodes in order to reduce the computational effort during the optimization process. In the present work, we have used for this purpose Coons patch [9, 15] to define a surface that interpolates given boundary curves. This previous relocation procedure facilitates the untangling process, but in general does not obtain a satisfactory mesh quality and can produce self-intersections, as shown in Fig. 2d. Therefore, it is essential to apply an efficient optimization algorithm.

### *3.1 Objective Function*

The mesh optimization process is carried out by iterative relocation of each inner node of the mesh in such a way that the new position of the node improves the quality of the local submesh corresponding to this node. A local submesh is the set of all the elements connected with the movable or *free node*. The local objective function for a free node is based on algebraic shape quality metrics proposed by Knupp in [18, 19] for triangular and quadrilateral elements. Shape quality metric for a given triangle is defined in terms of the Jacobian matrix of the affine mapping from ideal triangle to the given one. This shape quality metric represents the deviation

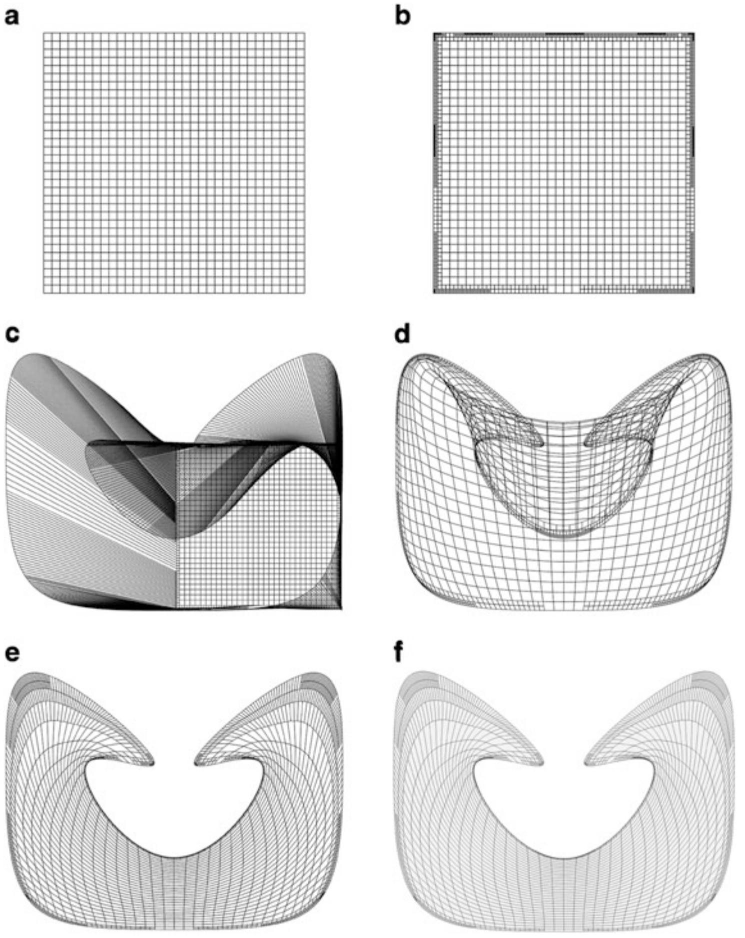


**Fig. 2** Stages of the method to parameterize a *Test* geometry. (**a**) Initial regular refinement of the parametric space; (**b**) parametric T-mesh adapted to the boundary of the geometry; (**c**) tangled physical mesh obtained after projecting the parametric boundary to the physical boundary. Note that the image of the parametric space is scaled respect to the previous ones; (**d**) resulting tangled mesh after previous relocation of the inner nodes by using Coons patch. This step is optional; (**e**) optimized physical T-mesh; (**f**) T-spline representation

of the physical triangle from the ideal one. It attains its maximum value, 1, if the triangle is similar to the ideal one, and it equals 0 if the triangle is degenerated. The distortion metric of an element is defined as the inverse of its quality metric. In order to asses the quality of the local submesh for a given free node of a T-mesh, we have to decompose each neighboring cell into triangles and asses the quality of each triangle. For a T-mesh, this decomposition depends on the type of the free node. There are two types of free node: a regular node and a hanging node. The optimal position of each free node is determined by minimizing a local objective function.

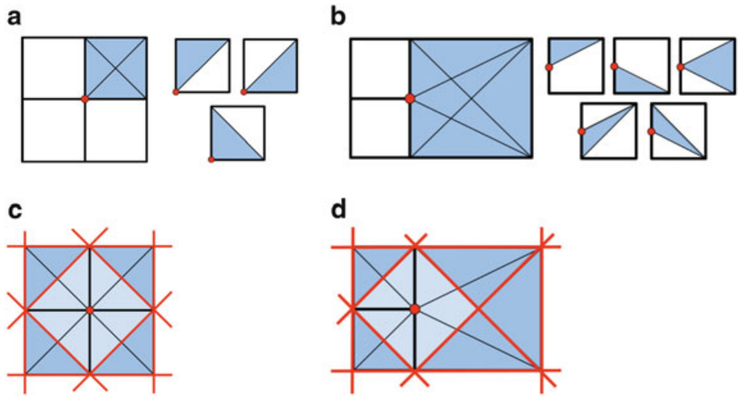


**Fig. 3** Triangular decomposition of the free node local submesh, where the *red point* is the free node. (**a**) Regular node case, where each cell is decomposed in three triangles; (**b**) hanging node case, where five triangles are formed in the cell where the node generates a T-junction; (**c**) barriers (*red lines*) and feasible region (*light blue*) induced by the 12 triangles in the objective function for a regular node; (**d**) barriers and feasible region induced by the 11 triangles in the objective function for a hanging node

We define the objective function as a sum of shape distortion metrics of the triangles of the local submesh. For each triangle of the physical mesh, the corresponding triangle of the parametric mesh is used as its ideal element. Therefore, each cell of the physical mesh tends to have the same shape as its counterpart cell of the parametric mesh. Thus, repeating this procedure for all the inner nodes of the mesh, we achieve the physical mesh of the object is as similar to the parametric one as possible.

A regular node is surrounded by four cells with equal or different sizes. In order to perform the mesh improvement, the local submesh is decomposed in twelve triangles, three triangles per cell whose qualities depends on the position of the free node. Figure 3a illustrates this decomposition and Fig. 3c shows the feasible region of the objective function.

In a hanging node case, the free node is surrounded by three cells and the local submesh is decomposed in eleven triangles. The cell in which the node forms a T-junction is decomposed in five triangles whose qualities depend on the position of the free node. Each one of the other two cells is decomposed in three triangles, as was described in the case of a regular node. Figure 3b shows the decomposition of a local submesh for hanging node case and the feasible region of the objective function. Note that, for the ideal case shown in Fig. 3d, the feasible region induced by these eleven triangles is the same as the one obtained after a refinement of the T-junction cell, see Fig. 3c.

In order to define the objective function we introduce the following concepts.

Let $T$ be a triangle whose vertices are given by $\mathbf{x}_k = (x_k, y_k)^T \in \mathbb{R}^2$, $k = 0, 1, 2$ and $T_R$ be the reference triangle with vertices $\mathbf{u}_0 = (0, 0)^T$, $\mathbf{u}_1 = (1, 0)^T$ and

$\mathbf{u}_2 = (0,1)^T$ . If we choose $\mathbf{x}_0$ as the translation vector, the affine map that takes $T_R$ to $T$ is $\mathbf{x} = A\mathbf{u} + \mathbf{x}_0$, where $A$ is the Jacobian matrix of the affine map referenced to node $\mathbf{x}_0$, and expressed as $A = (\mathbf{x}_1 - \mathbf{x}_0, \mathbf{x}_2 - \mathbf{x}_0)$.

Let consider that $T_I$ is our ideal or target triangle whose vertices are $\mathbf{v}_0$, $\mathbf{v}_1$ and $\mathbf{v}_2$. If we take $\mathbf{v}_0 = (0,0)^T$, the linear map that takes $T_R$ to $T_I$ is $\mathbf{v} = W\mathbf{u}$, where $W = (\mathbf{v}_1, \mathbf{v}_2)$ is its Jacobian matrix. As the parametric and real meshes are topologically identical, each triangle in the physical space has its counterpart in the parametric space.

Affine map that takes $T_I$ to $T$ is given by $\mathbf{x} = AW^{-1}\mathbf{v} + \mathbf{x}_0$, and its Jacobian matrix is $S = AW^{-1}$. Note that this weighted matrix $S$ depends on the node chosen as reference, so this node must be the same for $T$ and $T_I$. We can use matrix norms, determinant or trace of $S$ to construct algebraic quality metrics of $T$. For example, the *mean ratio*, $q = \frac{2\sigma}{\|S\|^2}$, is an easily computable algebraic quality metric of $T$, where $\sigma = \det(S)$ and $\|S\|$ is the Frobenius norm of $S$. The maximum value of $q$ is the unity, and it is reached when $A = \mu R W$, where $\mu$ is a scalar and $R$ is a rotation matrix. In other words, $q$ is maximum if and only if $T$ and $T_I$ are similar. Besides, any flat triangle has quality measure zero. We can derive an optimization function from this quality metric. Thus, let $\mathbf{x} = (x,y)^T$ be the position of the free node, and let $S_m$ be the weighted Jacobian matrix of the $m$-th triangle connected to this free node. We define the objective function of $\mathbf{x}$, associated to an $m$-th triangle as

$$\eta_m = \frac{\|S_m\|^2}{2\sigma_m}. \tag{1}$$

The local objective function used for mesh quality improvement is defined by means of the inverse of mean ratio quality metric of each triangle of the local submesh. The function to be minimized is given by

$$K(\mathbf{x}) = \left(\sum_{m=1}^{M} \eta_m^p\right)^{\frac{1}{p}} \qquad p \geqslant 1, \tag{2}$$

where $M$ is the number of triangles in the local submesh and $S_m$ is the Jacobian matrix associated to the affine mapping from the *ideal* triangle to the physical one. The value of the power $p$ may affect the result of the optimization. For quadrilateral meshes, we have obtained the best results with $p = 2$ in most cases.

The objective function defined by Eq. (2) is appropriate to improve the quality of a valid mesh, but it does not work properly when there are inverted elements [16, 17]. In the previous work [10] we introduced a modified objective function $K^*$, where the untangling and smoothing are carried out in the same stage. This modified objective function $K^*$ does not have singularities, it works as the original function

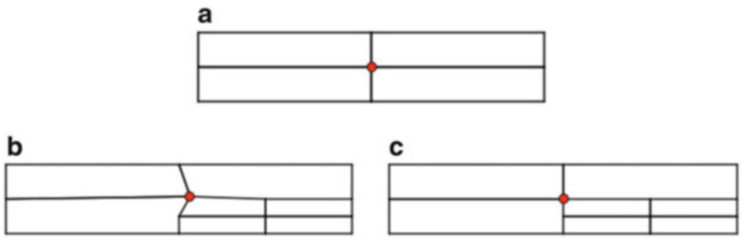


**Fig. 4** Resulting meshes after optimization with different objective functions. (**a**) Orthogonal mesh using $K^*$ or $K_\tau^*$; (**b**) no orthogonality result for a non-conformal local submesh using $K^*$; (**c**) orthogonal mesh using weighted objective function $K_\tau^*$

$K$ for the valid elements and tends to untangle the inverted and degenerated ones. This objective function is defined as

$$K^*(\mathbf{x}) = \left(\sum_{m=1}^{M} {\eta_m^*}^p\right)^{\frac{1}{p}} \tag{3}$$

being

$$\eta_m^* = \frac{\|S_m\|^2}{2h(\sigma_m)} \tag{4}$$

where $h(\sigma) = \frac{1}{2}(\sigma + \sqrt{\sigma^2 + 4\delta^2})$. Objective function $K^*$ is smooth in $\mathbb{R}^2$, so the unconstrained optimization problem can be easily solved with any standard method.

In case of a conformal local submesh, the result obtained by minimizing the objective function $K^*$ is, when possible, an orthogonal submesh, as shown in Fig. 4a. However, not the same result is obtained for a non-conformal submesh. In this case, two special situations can appear: a regular node surrounded by cells of different scales and a hanging node. In these situations, a variation in the position of the free node does not affect in the same way to the quality of the triangles of the local submesh. The objective function tends to form triangles as similar as possible to the reference ones, but the influence of the smaller cells are greater than the bigger ones. For example, in Fig. 4b it can be seen how the free node is moved toward the small cell and, therefore, the resulting mesh is not orthogonal.

In cases when it is desirable to have orthogonal mesh, this problem can be solved by a modification of the objective function $K^*$, namely multiplying the terms of the objective function by appropriate weights.

### 3.2 Weighted Objective Function

The terms of the objective function $K^*$ can be grouped according to the belonging to each cell of the local submesh. Each group is multiplied by an appropriated weight in order to avoid the mentioned problems in Sect. 3.1.

For a regular node, the weighted objective function is

$$K_\tau^*(\mathbf{x}) = \left( \tau_1 \sum_{m=1}^{3} \eta_m^{*\,p} + \tau_2 \sum_{m=4}^{6} \eta_m^{*\,p} + \tau_3 \sum_{m=7}^{9} \eta_m^{*\,p} + \tau_4 \sum_{m=10}^{12} \eta_m^{*\,p} \right)^{\frac{1}{p}} \tag{5}$$

where each summation is the group associated to each cell and $\tau_i$ is the applied weight. We can prove that this weight is equal to the scale factor of the cell in the parametric space, but it has not been included due to space limitation of this paper. We assume that the smallest cells in the local submesh have scale factor $\tau = 1$ and the other cells can have scale factor $\tau = 2$ or $\tau = 4$, as illustrated in Fig. 5a. Figure 4c shows the resulting orthogonal mesh when these weights are applied.

A hanging node is surrounded by three cells as it was mentioned above. In this case, the weighted objective functions is

$$K_\tau^*(\mathbf{x}) = \left( \tau_1 \sum_{m=1}^{3} \eta_m^{*\,p} + \tau_2 \sum_{m=4}^{6} \eta_m^{*\,p} + \tau_3 \sum_{m=7}^{11} \eta_m^{*\,p} \right)^{\frac{1}{p}}. \tag{6}$$

Hanging node is a more particular case because its local submesh is decomposed in different types of triangles. To guarantee the orthogonality in the local submesh after optimization, we have determined that the weights are $\tau_3 = \frac{8}{5}$ for the cell where the node forms a T-junction and $\tau_1 = \tau_2 = 1$ for the other two cells, as shown in Fig. 5b. More details about weighted objective function and the selection of the weights can be found in our previous work [4].

Two different options, weighted and no weighted objective functions, are available for T-mesh optimization, that should be chosen by the user according to

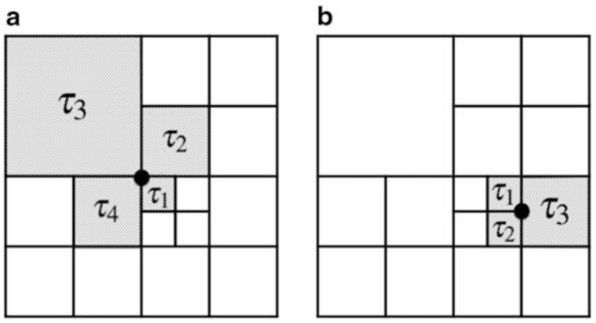


**Fig. 5** Weights for the objective functions $K_\tau^*$. (**a**) Regular node where $\tau_1 = 1$, $\tau_2 = \tau_4 = 2$ and $\tau_3 = 4$; (**b**) hanging node where $\tau_1 = \tau_2 = 1$ and $\tau_3 = \frac{8}{5}$

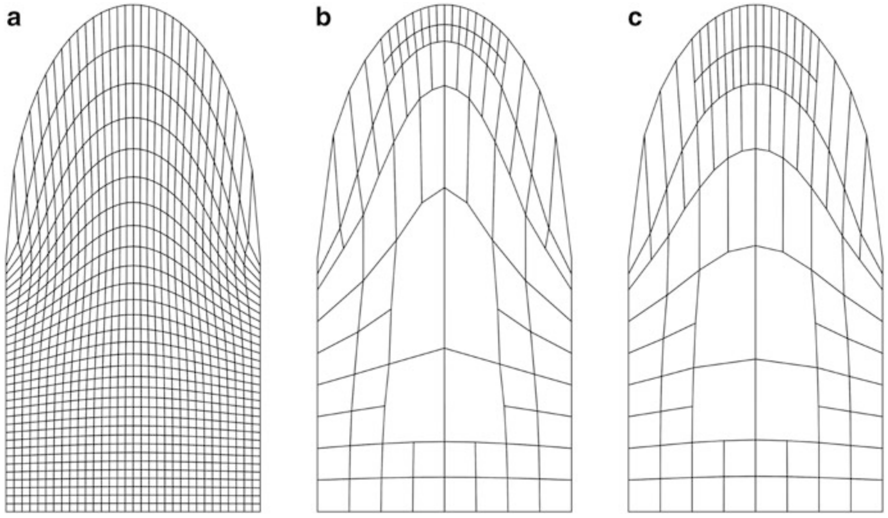


**Fig. 6** Comparison of optimization procedure with weighted and no weighted objective functions. (**a**) Regular mesh; (**b**) T-mesh optimized without weights; (**c**) the same T-mesh optimized with weights. Note that the positions of the nodes in case (**c**) are closer to the node positions of the regular mesh than in case (**b**)

necessities of the problem to solve. Optimization process with weighted objective function obtains a T-mesh as orthogonal and similar to a regular mesh as possible. However, no weighted objective function does not produce T-meshes so orthogonal, but it provides more flexibility in order to facilitates the untangling process in very complicated geometries. On the other hand, the quality of the mesh tends to be more uniform when it is optimized with the weighted objective function. Figure 6 shows the effects of optimizing a T-mesh without weights (b) and with weights (c) in comparison with an optimized regular mesh (a).

## 4 Construction of a T-Spline Geometry and Adaptive Refinement to Improve the Parametrization Quality

### *4.1 Geometry Construction via Interpolation*

In order to define T-spline basis functions of degree 3 over a given T-mesh in 2D, a local knot vector for both parametric directions should be assigned to each basis function $R_\alpha$: $\Xi_\alpha = \left(\xi_{\alpha_1}, \xi_{\alpha_2}, \xi_{\alpha_3}, \xi_{\alpha_4}, \xi_{\alpha_5}\right)$, $H_\alpha = \left(\eta_{\alpha_1}, \eta_{\alpha_2}, \eta_{\alpha_3}, \eta_{\alpha_4}, \eta_{\alpha_5}\right)$. These knot vectors are inferred by traversing T-mesh edges. This basis function is associated to the central knot $(\xi_{\alpha_3}, \eta_{\alpha_3})$ that is called *anchor*. As we are using open knot vector structure, there are some blending functions that have the same anchor.

The T-spline blending functions that we use in this work are rational B-spline functions defined as

$$R_\alpha(\boldsymbol{\xi}) = \frac{N_\alpha(\boldsymbol{\xi})}{\sum\limits_{\beta\in A} N_\beta(\boldsymbol{\xi})} \tag{7}$$

being $N_\alpha(\boldsymbol{\xi}) = N_\alpha^1(\xi)\, N_\alpha^2(\eta)$ the bivariate B-spline function defined over its local knot vectors $\boldsymbol{\Xi}_\alpha = \{\Xi_\alpha, H_\alpha\}$, and $A$ is the index set of the basis spanned by T-mesh.

A detailed report about T-splines and their relationship with isogeometric analysis can be found in [2].

We build bivariate T-spline surface representation of our physical domain as lineal combination of T-spline blending functions

$$\mathbf{S}(\boldsymbol{\xi}) = \sum_{\alpha\in A} \mathbf{P}_\alpha\, R_\alpha(\boldsymbol{\xi}) \tag{8}$$

where $\mathbf{P}_\alpha \in \mathbb{R}^2$ is the control point corresponding to the $\alpha$-th blending function.

Control points $\mathbf{P}_\alpha$ are found by imposing interpolation conditions. Assuming that the set of blending functions are linearly independent, we need as many interpolation points as blending functions. Nevertheless, we have not a theoretical proof of their linear independence and, therefore, of the solubility of the liner system. Buffa et al. [5] prove the linear independence of the T-splines by supposing that a sequence of nested T-meshes produces a sequence of nested spaces. Unfortunately, this supposition is not satisfied in our case, that is, our spaces are not nested although we have nested T-meshes. However, we have verify in numerous tests that the T-splines defined over a balanced quadtree mesh are linear independent.

As interpolation points, we use the vertices of the mesh. For each vertex, $\boldsymbol{\xi}_\alpha^v$, its position in the physical space, $\mathbf{x}_\alpha^v$, was determined by the mesh optimization process.

### *4.2 Mean Ratio and Adaptive Refinement Strategy to Improve Parameterization Quality*

Our objective is to get high-quality geometry parameterization suitable for isogeometric analysis. The parametric T-spline mapping of Eq. (8) is suitable for analysis if it has positive Jacobian in all the domain. High distortion of the geometry can produce a large variation of the Jacobian that can lead to a poor accuracy in the numerical results. Therefore, a good uniformity and orthogonality of the isoparametric curves are desired for the parametric mapping $\mathbf{S}$.

A high quality of the optimized T-mesh is a necessary, but not sufficient, condition for a high quality of the T-spline mapping. It can happen that the Jacobian

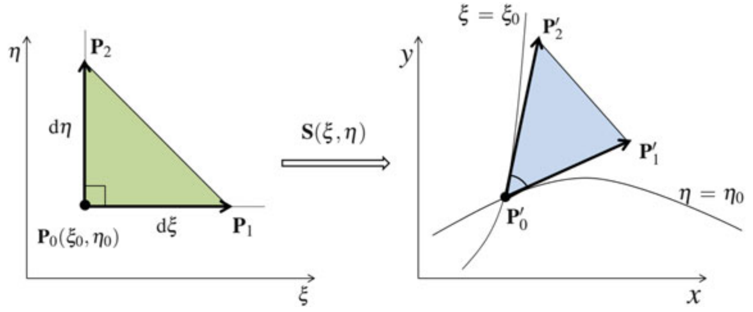


**Fig. 7** Mean ratio Jacobian. A quality metric of the parametric mapping $\mathbf{S}$ at any point $\mathbf{P}_0$ in terms of the mean ratio of the triangle $\mathbf{P}_0'\mathbf{P}_1'\mathbf{P}_2'$

of the spline parameterization takes negative values even if all the cells of the T-mesh are valid. In order to assess the quality of the constructed parametric transformation we analyze the mean ratio Jacobian, given by

$$J_r(\boldsymbol{\xi}) = \frac{2 \det (J)}{\|J\|^2}, \tag{9}$$

where $J$ is the Jacobian matrix of the mapping $\mathbf{S}$ at the point $\boldsymbol{\xi} = (\xi, \eta)$ and $\|J\|$ is its Frobenius norm.

The value of the mean ratio Jacobian at any point $\mathbf{P}_0$ of the parametric domain is a shape quality metric for the infinitesimal triangle formed by two isoparametric curves of the physical domain passing through the point $\mathbf{P}_0' = \mathbf{S}(\mathbf{P}_0)$, as illustrated in Fig. 7. In contrast to the scaled Jacobian, that represents a quality of the mapping $\mathbf{S}$ in the sense of the orthogonality of its isoparametric curves, the mean ratio Jacobian represents both: a quality of the mapping in the sense of the orthogonality and uniformity of its isoparametric curves. Scaled Jacobian attains its maximum value 1 at the given point if the mapping conserves orthogonality of the isoparametric curves. Mean ratio Jacobian is equal 1 at the point $\mathbf{P}_0$ if the mapping conserves orthogonality and produces the same length distortion in both parametric directions, i.e., the mapping is conformal at this point.

It is easy to see that $\forall \boldsymbol{\xi} : \quad 0 \leq |J_r(\boldsymbol{\xi})| \leq |J_s(\boldsymbol{\xi})| \leq 1$, where $J_s = \frac{\det(J)}{\|\mathbf{S}_\xi\| \|\mathbf{S}_\eta\|}$ is scaled Jacobian.

Parameterization of complex geometries entails a severe distortion that can lead to appearance of low quality cells, even cells with negative Jacobian. This can be explained by the lack of degrees of freedom provided by the inner nodes. In order to improve the mesh quality in this case, we propose an adaptive strategy that refines all the cells with low quality. A similar idea was implemented for tetrahedral meshes in [11].

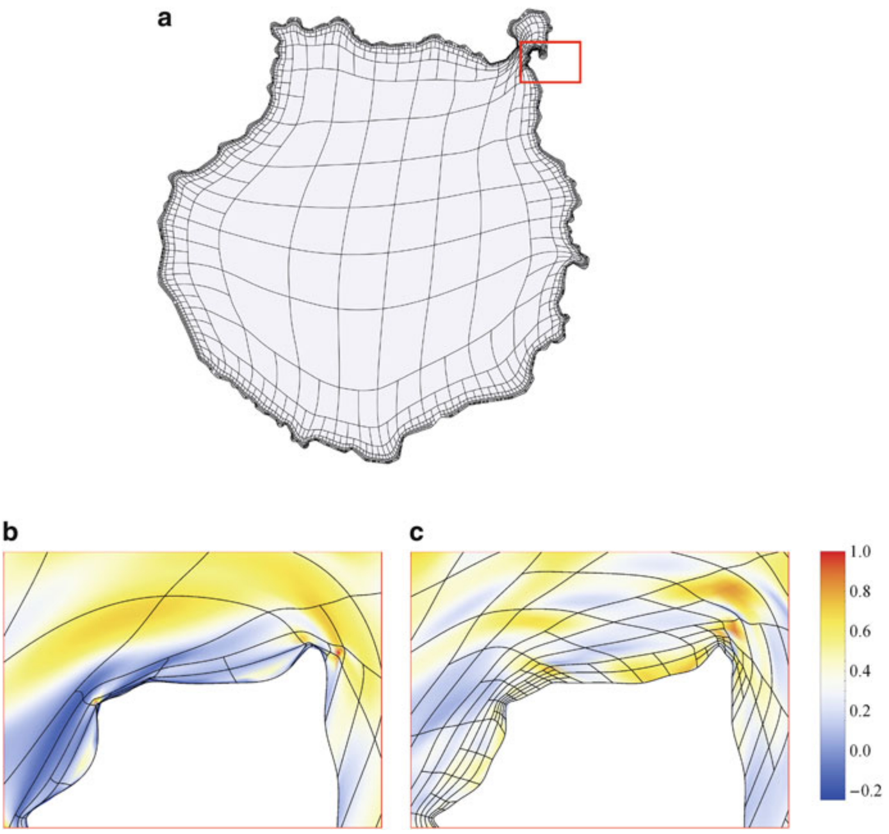


**Fig. 8** Adaptive refinement strategy to improve the parametric transformation quality in *Gran Canaria island* domain. (**a**) T-spline representation of the domain; (**b**) initial T-spline parametrization and color map of its mean ratio Jacobian with negative values; (**c**) resulting T-spline parameterization after applying adaptive refinement and color map of its mean ratio Jacobian with no negative values

We proceed as follows. For each cell of the mesh, the mean ratio Jacobian is calculated at Gauss quadrature points. We use $16 = 4 \times 4$ quadrature points per cell. A cell $\hat{\Omega}_e$ is marked to refine if, at least, one of its quadrature points has mean ratio Jacobian less than a certain threshold $\delta$. The refined T-mesh is optimized again and the process is repeated until a satisfactory mesh quality is obtained. Figure 8 illustrates the efficacy of the proposed strategy. Additional refinements were applied to *Gran Canaria Island* domain with $\delta = 0.2$. The initial mesh with 3,439 cells produces a T-spline parametric mapping with low quality in some areas and negative Jacobian in the North East part of the island. After adaptive refinement we have a mesh with 3,577 cells and positive Jacobian in all the domain. Moreover, the minimum value of mean ratio Jacobian at the quadrature points is 0.21.

In most cases, adaptive refinements only affect to a localized area of the geometry. A good strategy to optimize the mesh after adaptive refinement is to apply the optimization algorithm only to refined areas. Initially it is marked to optimize a list of nodes composed by the nodes of the new cells created in the refinement and the nodes that conform the local submesh of each of them. After optimization, if the position of the node does not change significantly, the node is extracted of the list. Otherwise, the node remains in the list and its local submesh is added. The process finalizes when the list of nodes is empty. This strategy avoids unnecessary iterations in zones where the optimization process does not produce relevant changes in the mesh. This accelerates the optimization after each adaptive refinement.

## 5 Parameterization of Embedded Geometries

The proposed method can be easily extended to parameterize embedded geometries. Embedded geometries are composed by the insertion of individuals figures into another one. Each of these figures can be parameterize individually with the method seen until now, obtaining the parametric and physical T-meshes of each one. The key to parameterize the composed geometry is to build a global parametric space based on the individual ones. In our method, the parametric space has a quadtree structure. A cell of a quadtree could be seen as the root of a new quadtree. This allows to insert the quadtree of an inner geometry in a cell or set of cells belonging to an outer quadtree.

After the insertion, we obtain a new full quadtree for the global geometry. This new quadtree is not balanced, so it is necessary to apply the 2:1 balance algorithm after the quadtree merging. The untangling and smoothing procedure is carried out with this new balanced quadtree, starting at the node positions obtained in the individual construction of each figure. In this untangling step, the nodes of the inner boundaries are fixed in the same manner that the outer boundary. Figure 9 describes the process to build a T-mesh with embedded geometries. The rest of the process to construct the T-spline representation of the geometry and the adaptive refinement is exactly the same that for a basic geometry.

With this strategy it is possible to insert any number of geometries into another one. Moreover, we can build geometries with holes, simply deleting the cells corresponding to an inner geometry.

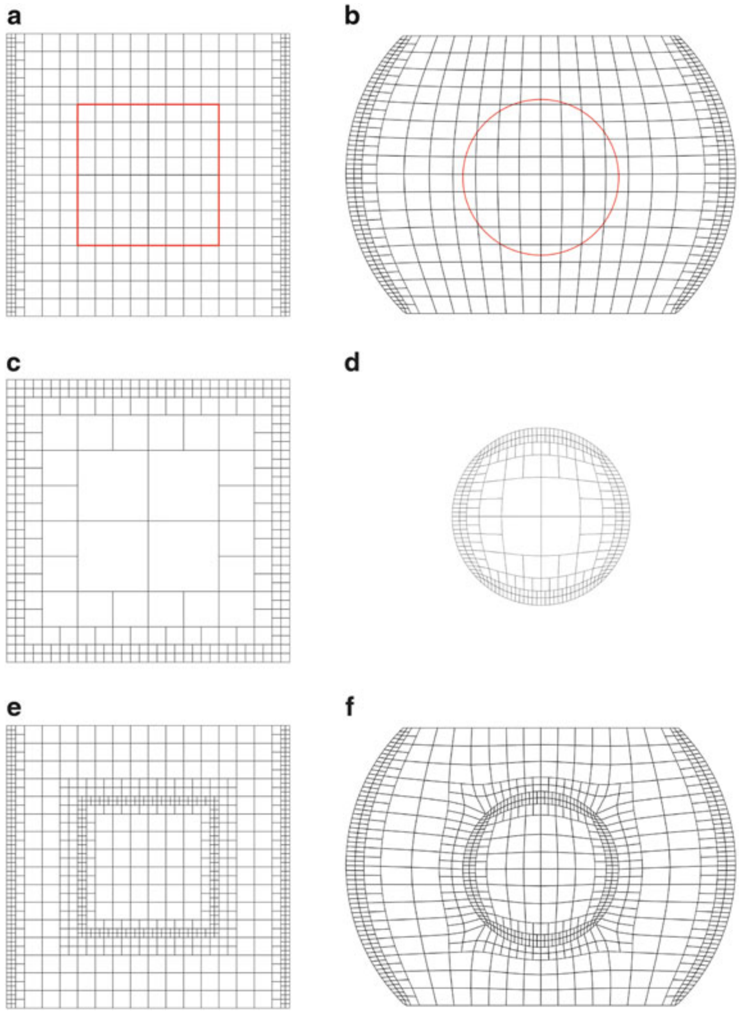


**Fig. 9** T-mesh construction of an embedded geometry. (**a**) Parametric space of the outer geometry. *Red lines* define the area where the inner parametric space will be inserted; (**b**) T-mesh of the outer geometry. The position of the inner boundary is marked in *red*; (**c**) parametric space of the inner geometry; (**d**) T-mesh of the inner geometry; (**e**) resulting parametric space after inserting the inner one into a region of the outer one, with a subsequent 2:1 balance; (**f**) T-mesh of the global geometry after optimization with $K_\tau^*$ and $p = 2$

## 6 Results and Applications

### 6.1 *Geometric Modeling Results*

The proposed algorithm was tested on several 2D domains. For all of them, we have obtained parametric mappings of high quality suitable for isogeometric analysis. In this section we present some results of the application of the method.

Figure 10 shows the parameterization quality of the *Test* geometry exposed in the description of the method, where the T-mesh has been optimized with $K_\tau^*$ and $p = 2$. The color map represents the mean ratio Jacobian of the parameterization. As it was described in Sect. 4.2, we have analyzed the quality of the parametric mapping by evaluating the mean ratio Jacobian at the quadrature points of each cell. In this case, no adaptive refinement was necessary.

Figure 11 shows the refined parametric space, the physical T-mesh and the color map of the mean ratio Jacobian in the *Puzzle* geometry. We have constructed the adapted T-mesh starting from an initial uniform $32 \times 32$ mesh in order to increase the degree of freedom in the inner of the geometry. It has been optimized with $K_\tau^*$ and $p = 2$. Only one adaptive refinement were necessary to improve the mesh quality and assure positive Jacobian in all quadrature points.

Figure 12 shows the parameterization quality of the *Embedded Test* geometry exposed in the previous section, optimized with $K_\tau^*$ and $p = 2$. In this case no adaptive refinement was necessary and minimum value of the mean ratio Jacobian at quadrature points is 0.26.

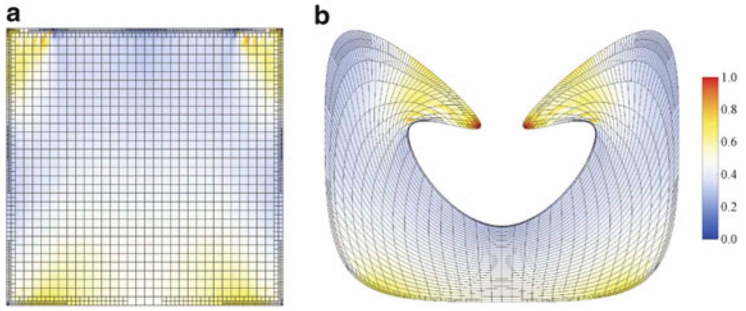


**Fig. 10** *Test* geometry with 2,287 cells and 3,363 control points, optimized with $K_\tau^*$ and $p = 2$. (**a**) Color map of the mean ratio Jacobian of the parametric transformation represented in the parametric domain; (**b**) color map of the mean ratio Jacobian in the physical domain. Minimal value of mean ratio Jacobian at quadrature points is 0.138

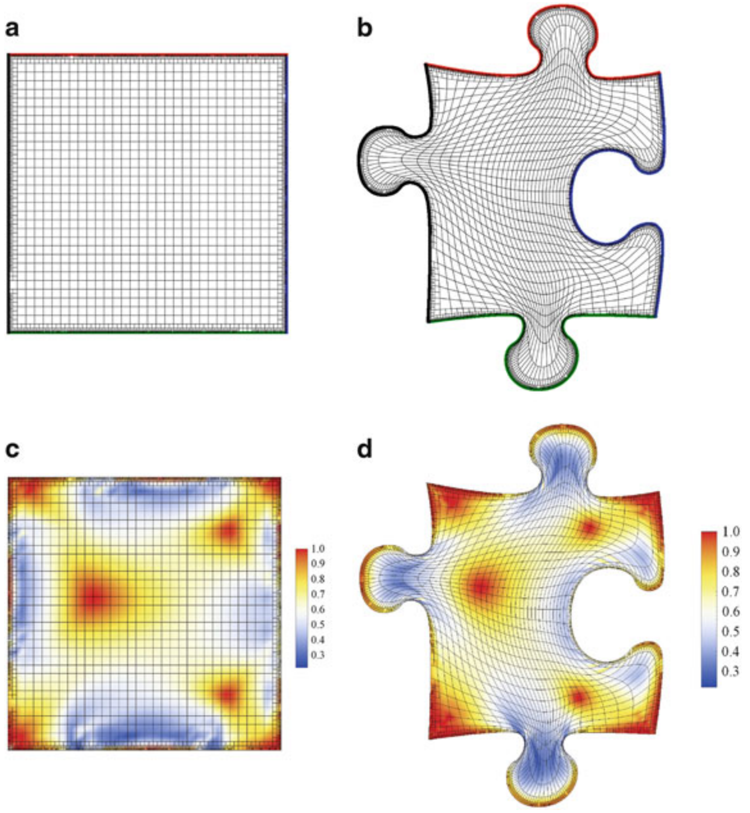


**Fig. 11** *Puzzle* geometry with 3,748 cells and 5,946 control points. (**a**) Parametric T-mesh; (**b**) physical T-mesh, optimized with $K_{\tau}^{*}$ and $p = 2$; (**c**) color map of the mean ratio Jacobian in the parametric domain; (**d**) T-spline representation and color map of the mean ratio Jacobian in the physical domain, where minimal value of mean ratio Jacobian at quadrature points is 0.19

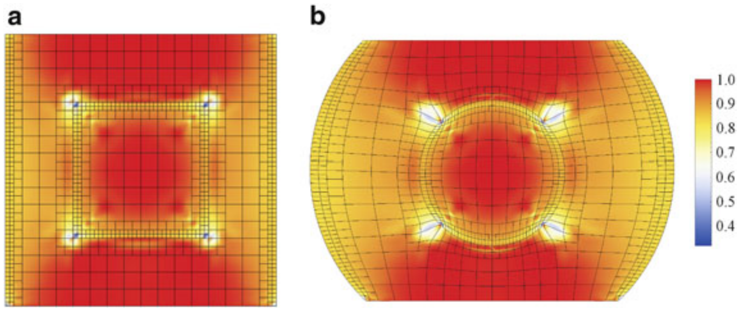


**Fig. 12** *Embedded Test* geometry with 916 cells and 1,317 control points, optimized with $K_{\tau}^{*}$ and $p = 2$. (**a**) Color map of mean ratio Jacobian in parametric domain; (**b**) color map of mean ratio Jacobian in physical domain, minimal value of mean ratio Jacobian at quadrature points is 0.26

## 6.2 Isogeometric Analysis Applications

In this section we present two examples of the resolution of the Poisson equation for domains composed of two materials using isogeometric analysis with T-splines. Let us consider the next problem

$$\begin{aligned} -\nabla(k(\mathbf{x})\nabla u) &= f \qquad \text{in } \Omega, \\ u &= g \qquad \text{on } \partial\Omega. \end{aligned} \tag{10}$$

### 6.2.1 Dielectric Cylinder in a Uniform Electric Field

The first problem is an infinity dielectric cylinder of radio $b$ immersed in uniform electric field $E_0\mathbf{i}$. Actually, it is a bi-dimensional problem as there is no dependence with $z$ coordinate. The domain $\Omega$ is a rectangle with a embedded circle of radius $b$ in its center where the dielectric constants are

$$k(\mathbf{x}) = \begin{cases} \epsilon_0\epsilon_r & \text{if } \rho < b \\ \epsilon_0 & \text{if } \rho \geq b. \end{cases}$$

The analytic solution is given in cylindrical coordinates by

$$\begin{aligned} u_{\rho<b} &= \frac{-2E_0\rho\cos\varphi}{(\epsilon_r+1)} \\ u_{\rho\geq b} &= E_0\cos\varphi\left(-\rho + \frac{b^2(\epsilon_r-1)}{\rho(\epsilon_r+1)}\right). \end{aligned}$$

By calculating the electric field $\mathbf{E} = -\nabla u$ it can be seen that the electric field tends to initial electric field $E_0\mathbf{i}$ as $\rho$ tends to infinity and it is constant inside the cylinder.

In order to obtain a numerical solution as accurate as possible, we impose the boundary Dirichlet conditions taking into account the exact solution.

The T-spline geometry of the embedded domains is shown in Fig. 13a and the result of the numerical simulation with $b = 0.5$ and $\epsilon_r = 20$ is presented in Fig. 13b. Note that the stream lines of the electric field are constant in the cylinder. Comparing with the analytic solution, we have measured a maximum error in the potential of 0.88 %.

### 6.2.2 Heat Conduction Problem

The other example is a heat conduction problem in a more complex geometry. We consider an interior "spot-shape" domain with conductivity $k_1 = 1$ immersed in

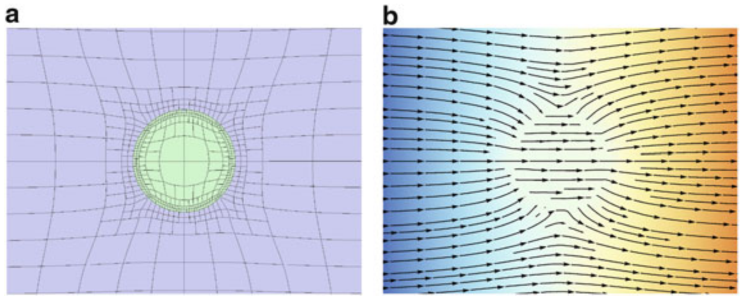


**Fig. 13** (**a**) Zoom of the T-spline centred in the immersed cylinder; (**b**) the stream lines of the electric field around of the cylinder

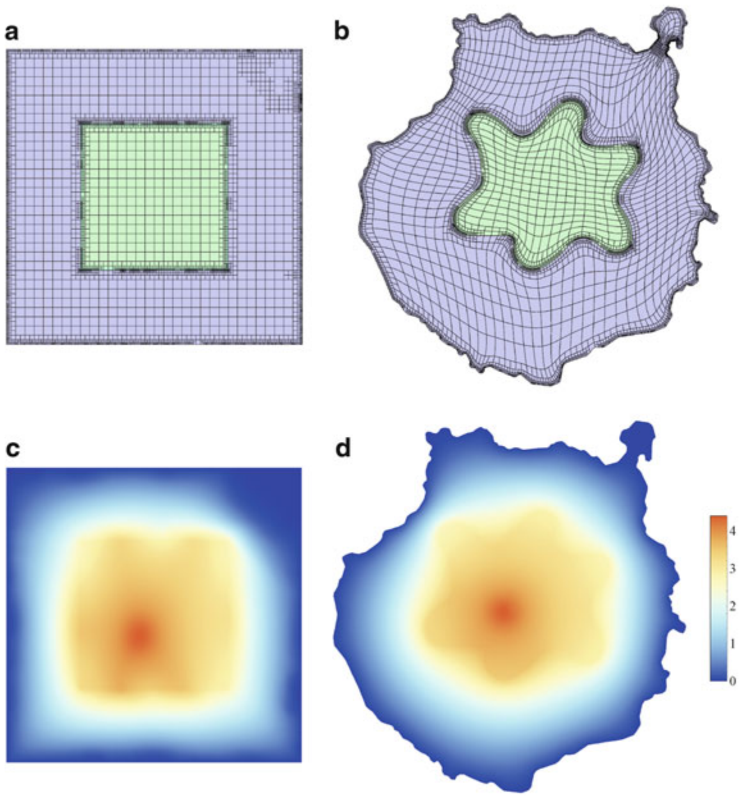


**Fig. 14** *Gran Canaria Island* geometry with two materials, 6,466 cells and 9,439 control points. T-mesh optimized with $K^*$ and $p = 2$. (**a**) Parametric domain; (**b**) T-spline representation of physical domain; (**c**) numerical solution for heat conduction problem in parametric domain; (**d**) numerical solution in physical domain

*Gran Canaria island* domain with conductivity $k_2 = 0.1$. The source term $f$ is a Gaussian function centred in the interior domain and the Dirichlet boundary condition $g = 0$ in $\partial\Omega$.

The T-spline parameterization and the solution of the heat conduction problem for *Gran Canaria Island* domain with interior subdomain of high conductivity is shown in Fig. 14.

## 7 Conclusions

We have proposed a new effective technique for obtaining a T-spline parameterization of 2D geometries for the application of isogeometric analysis. A new T-mesh untangling and smoothing procedure have been applied in order to define an isomorphic transformation between parametric and physical T-meshes. Presented technique is simple and easy to implement. The algorithm have been tested in several 2D geometries and, for all of them, we have obtained a high quality parametric transformation between the object and the parametric domain. To asses the quality of the parametric mapping, we evaluate its mean ratio Jacobian. Thereby, we detect the areas with low quality and perform an adaptive refinement in order to increase the degree of freedom in the areas with high distortion. This strategy allows to obtain a parameterization suitable for analysis with no negative Jacobian, even for complex geometries. Moreover, we have proposed an extension of the method to parameterize embedded geometries.

All the geometries presented in this work have been parameterized with the unit square. As a next step, we plan on overcome this limitation and to develop an algorithm to parameterize a 2D object with more complex polygon-type parametric domain that fits better the geometry. Also, in future research we expect to extend the presented method to 3D.

**Acknowledgements** This work has been supported by the Spanish Government, "Secretaría de Estado de Universidades e Investigación," "Ministerio de Economía y Competitividad,", "Programa de FPU del Ministerio de Educación, Cultura y Deporte", "Programa de FPI propio de la ULPGC" and FEDER, grant contracts: CGL2011-29396-C03-01 and CGL2011-29396-C03-03; "Junta Castilla León," grant contract: SA266A12-2. It has been also supported by CONACYT-SENER ("Fondo Sectorial CONACYT SENER HIDROCARBUROS," grant contract: 163723).

## References

1. Bazilevs, Y., Calo, V.M., Cottrell, J.A., Evans, J.A., Hughes, T.J.R., Lipton, S., Scott, M.A., Sederberg, T.W.: Isogeometric analysis: toward unification of computer aided design and finite element analysis. In: Papadrakakis, M., Topping, B.H.V. (eds.) Trends in Engineering Computational Technology, pp. 1–16. Saxe-Coburg Publications, Stirling (2008)

2. Bazilevs, Y., Calo, V.M., Cottrell, J.A., Evans, J.A., Hughes, T.J.R., Lipton, S., Scott, M.A., Sederberg, T.W.: Isogeometric analysis using T-splines. Comput. Methods Appl. Mech. Eng. **199**, 229–263 (2010)
3. Branets, L., Graham, F.C.: Smoothing and adaptive redistribution for grids with irregular valence and hanging nodes. In: Proceedings of the 13th International Meshing Roundtable, vol. 19–22, pp. 333–344. Springer, Berlin (2004)
4. Brovka, M., López, J.I., Escobar, J.M., Cascón, J.M., Montenegro, R.: A new method for T-spline parameterization of complex 2D geometries. Eng. Comput. **30**, 457–473 (2013). doi:10.1007/s00366-013-0336-8
5. Buffa, A., Cho, D., Sangalli, G.: Linear independence of the T-spline blending functions associated with some particular T-meshes. Comput. Methods Appl. Mech. Eng. **199**(23–24), 1437–1445 (2010)
6. Cascón, J.M., Montenegro, R., Escobar, J.M., Rodríguez, E., Montero, G.: A new *meccano* technique for adaptive 3-D triangulation. In: Proceedings of the 16th International Meshing Roundtable, pp. 103–120. Springer, Berlin (2007)
7. Cascón, J.M., Montenegro, R., Escobar, J.M., Rodríguez, E., Montero, G.: The meccano method for automatic tetrahedral mesh generation of complex genus-zero solids. In: Proceedings of the 18th International Meshing Roundtable, pp. 463–480. Springer, Berlin (2009)
8. Cascón, J.M., Rodríguez, E., Escobar, J.M., Montenegro, R.: Comparison of the meccano method with standard mesh generation techniques. Eng. Comput. (2013). doi:10.1007/s00366-013-0338-6
9. Coons, S.A.: Surfaces for Computer Aided Design, Springfield (1964)
10. Escobar, J.M., Rodríguez, E., Montenegro, R., Montero, G., González-Yuste, J.M.: Simultaneous untangling and smoothing of tetrahedral meshes. Comput. Methods Appl. Mech. Eng. **192**, 2775–2787 (2003)
11. Escobar, J.M., Montenegro, R., Montero, G., Rodríguez, E., González-Yuste, J.M.: Smoothing and local refinement techniques for improving tetrahedral mesh quality. Comput. Struct. **83**, 2423–2430 (2005)
12. Escobar, J.M., Rodríguez, E., Montenegro, R., Montero, G., González-Yuste, J.M.: SUS code: simultaneous mesh untangling and smoothing code. http://www.dca.iusiani.ulpgc.es/proyecto2012-2014/html/Software.html (2010)
13. Escobar, J.M., Cascón, J.M., Rodríguez, E., Montenegro, R.: A new approach to solid modeling with trivariate T-splines based on mesh optimization. Comput. Methods Appl. Mech. Eng. **200**, 3210–3222 (2011)
14. Escobar, J.M., Montenegro, R., Rodríguez, E., Cascón, J.M.: The meccano method for isogeometric solid modeling and applications. Eng. Comput., 1–13 (2012). doi:10.1007/s00366-012-0300-z
15. Farin, G., Hansford, D.: Discrete Coons patches. Comput. Aided Geom. Des. **16**, 691–700 (1999)
16. Freitag, L.A., Knupp, P.M.: Tetrahedral mesh improvement via optimization of the element condition number. Int. J. Numer. Methods Eng. **53**, 1377–1391 (2002)
17. Freitag, L.A., Plassmann, P.: Local optimization-based simplicial mesh untangling and improvement. Int. J. Numer. Methods Eng. **49**, 109–125 (2000)
18. Knupp, P.M.: Algebraic mesh quality metrics. SIAM J. Sci. Comput. **23**, 193–218 (2001)
19. Knupp, P.M.: A method for hexahedral mesh shape optimization. Int. J. Numer. Methods Eng. **58**(2), 319–332 (2003)
20. Li, X., Guo, X., Wang, H., He, Y., Gu, X., Qin, H.: Harmonic volumetric mapping for solid modeling applications. In: Proceedings of ACM Solid and Physical Modeling Symposium, pp. 109–120. Association for Computing Machinery, New York (2007)
21. Li, B., Li, X., Wang, K.: Generalized polycube trivariate splines. In: SMI 2010, International Conference of Shape Modeling and Applications, pp. 261–265 (2010)
22. Martin, T., Cohen, E., Kirby, R.: Volumetric parameterization and trivariate B-spline fitting using harmonic functions. Comput. Aided Geom. Des. **26**, 648–664 (2009)

23. Montenegro, R., Cascón, J.M., Escobar, J.M., Rodríguez, E., Montero, G.: An automatic strategy for adaptive tetrahedral mesh generation. Appl. Numer. Math. **59**, 2203–2217 (2009)
24. Montenegro, R., Cascón, J.M., Rodríguez, E., Escobar, J.M., Montero, G.: The meccano method for automatic three-dimensional triangulation and volume parametrization of complex solids. In: Topping, B.H.V., Adam, J.M., Pallarés, F.J., Bru, R., Romero, M.L. (eds.) Developments and Applications in Engineering Computational Technology, pp. 19–48. Saxe-Coburg Publications, Stirling (2010)
25. Wang, W., Zhang, Y., Liu, L., Hughes, T.J.R.: Trivariate solid T-spline construction from boundary triangulations with arbitrary genus topology. Comput. Aided Des. **45**, 351–360 (2013)
26. Xu, G., Mourrain, B., Duvigneau, R., Galligo, A.: Parametrization of computational domain in isogeometric analysis: methods and comparison. Comput. Methods Appl. Mech. Eng. **200**, 2021–2031 (2011)
27. Xu, G., Mourrain, B., Duvigneau, R., Galligo, A.: Variational harmonic method for parameterization of computational domain in 2D isogeometric analysis. In: 12th International Conference on Computer-Aided Design and Computer Graphics, pp. 223–228. IEEE, Jinan (2011)
28. Zhang, Y., Wang, W., Hughes, T.J.R.: Solid T-spline construction from boundary representations for genus-zero geometry. Comput. Methods Appl. Mech. Eng. **249–252**, 185–197 (2012)

# Thread-Parallel Anisotropic Mesh Adaptation

**Gerard J. Gorman, Georgios Rokos, James Southern, and Paul H.J. Kelly**

**Abstract** Mesh adaptation is a powerful way to minimise the computational cost of mesh based computation. It is particularly successful for multi-scale problems where the required mesh resolution can vary by orders of magnitude across the domain. The end result is local control over solution accuracy and reduced time to solution.

In the case of large scale simulations, where the time to solution is unacceptable or the memory requirements exceeds available RAM, mesh based computation is typically parallelised using domain decomposition methods using the Message Passing Interface (MPI). This allows a simulation to run in parallel on a distributed memory computer. While this has been a high successful strategy up until now, the drive towards low power multi- and many-core architectures means that an even higher degree of parallelism is required and the memory hierarchy exploited to maximise memory bandwidth.

For this reason application codes are increasingly adopting a hybrid parallel approach whereby decomposition methods, implemented using the Message Passing Interface (MPI), are applied for inter-node parallelisation, while a threaded programming model is used for intra-node parallelisation. Mesh adaptivity has been successfully parallelised using MPI by a number of groups, and can be implemented efficiently with few modifications to the serial code. However, thread-level parallelism is significantly more challenging because each thread modifies the mesh data and therefore must be carefully marshalled to avoid data races while still ensuring enough parallelism is exposed to achieve good parallel efficiency.

G.J. Gorman (✉)
Department of Earth Science and Engineering, Imperial College London, London SW8 2AZ, UK
e-mail: g.gorman@imperial.ac.uk

G. Rokos • P.H.J. Kelly
Department of Computing, Imperial College London, London SW7 2AZ, UK
e-mail: georgios.rokos09@imperial.ac.uk; p.kelly@imperial.ac.uk

J. Southern
Fujitsu Laboratories of Europe, Hayes Park Central, Hayes End Road, Hayes, Middlesex UB4 8FE, UK
e-mail: James.Southern@uk.fujitsu.com

S. Perotto, L. Formaggia (eds.), *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, SEMA SIMAI Springer Series 5,
DOI 10.1007/978-3-319-06053-8_6

Here we describe a new thread-parallel algorithm for anisotropic mesh adaptation algorithms. For each mesh optimisation phase (refinement, coarsening, swapping and smoothing) we describe how independent sets of tasks are defined. We show how a deferred updates strategy can be used to update the mesh data structures in parallel and without data contention. We show that despite the complex nature of mesh adaptation and inherent load imbalances in the mesh adaptivity, good parallel efficiency can be achieved.

## 1 Introduction

Anisotropic mesh adaptation methods provide an important means to minimise superfluous computation associated with over-resolving the solution while still achieving the required accuracy, [1, 5, 17, 18]. In order to use mesh adaptation within a simulation, the application code requires a method to estimate the local solution error. Given an error estimate it is then possible to calculate the required local mesh resolution in order to achieve a specific solution accuracy.

There are a number of examples where adaptive mesh methods have been parallelised in the context of distributed memory parallel computers. The main challenge in performing mesh adaptation in parallel is maintaining a consistent mesh across domain boundaries. One approach is to first lock the regions of the mesh which are shared between processes and for each process to apply the serial mesh adaptation operation to the rest of the local domain. The domain boundaries are then perturbed away from the locked region and the lock-adapt iteration is repeated [7]. Freitag et al. [12, 13] considers a fine grained approach whereby a global task graph is defined which captures the data dependencies for a particular mesh adaptation kernel. This graph is coloured in order to identify independent sets of operations. The parallel algorithm then progresses by selecting an independent set (vertices of the same colour) and applying mesh adaptation kernels to each element of the set. Once a sweep through a set has been completed, data is synchronised between processes, and a new independent set is selected for processing. In [3] each process applies the serial adaptive algorithm, however rather than locking the halo region, operations to be performed on the halo are first stashed in buffers and then communicated so that the same operations will be performed by all processes that share mesh information. For example, when coarsening is applied all the vertices to be removed are computed. All operations which are local are then performed while pending operations in the shared region are exchanged. Finally, the pending operations in the shared region are applied.

However, over the past decade there has been a trend towards multi- and many-core compute nodes. Indeed, it is assumed that the compute nodes of a future exascale supercomputer will each contain thousands or even tens of thousands of cores [9]. On multi-core architectures, a popular parallel programming paradigm is to use thread-based parallelism to exploit shared memory within a shared memory node and a message passing using MPI for interprocess communication. When the computational intensity is sufficiently high, a third level of parallelisation may be implemented via SIMD instructions at the core level. There are some opportunities to improve performance and scalability by reducing communication needs, memory consumption, resource sharing as well as improved load balancing [19]. However, the algorithms themselves must also have a high degree of thread parallelism if they are to have a future on multi-core architectures; whether it be CPU or coprocessor based.

Rokos et al. [20] and Gorman et al. [16] develop thread parallel algorithms for many-core and multi-core processors based on the independent set approach described in [12]. However, this approach does not easy carry over for a threaded implementation of the other mesh adaptation operations. Therefore, in this paper we take a fresh look at the anisotropic adaptive mesh methods in 2D to develop new scalable thread-parallel algorithms suitable for modern multi-core architectures. We show that despite the irregular data access patterns, irregular workload and need to rewrite the mesh data structures, good parallel efficiency can be achieved.

The algorithms described in this paper have been implemented in the open source code PRAgMaTIc (Parallel anisotRopic Adaptive Mesh ToolkIt).[1] The remainder of the paper is laid out as follows: Sect. 2 gives an overview of the anisotropic adaptive mesh procedure; Sect. 3 describes the thread-parallel algorithm; and Sect. 4 illustrates how well the algorithm scales for a benchmark problem. We conclude with a discussion on future work and possible implications of this work.

## 2 Overview

In this section we will give an overview of anisotropic mesh adaptation. In particular, we focus on the element quality as defined by an error metric and the anisotropic adaptation kernels which iteratively improve the local mesh quality as measured by the worst local element.

### *2.1 Error Control*

Solution discretisation errors are closely related to the size and the shape of the elements. However, in general meshes are generated using *a priori* information about the problem under consideration when the solution error estimates are not yet available. This may be problematic because:

- Solution errors may be unacceptably high.
- Parts of the solution may be over-resolved, thereby incurring unnecessary computational expense.

[1] https://code.launchpad.net/~pragmatic-core/pragmatic/pragmatic2d-2.0

A solution to this is to compute appropriate local error estimates, and use this information to compute a field on the mesh which specifies the local mesh resolution requirement. In the most general case this is a metric tensor field so that the resolution requirements can be specified anisotropically; for a review of the procedure see [14]. Size gradation control can be applied to this metric tensor field to ensure that there are not abrupt changes in element size [2].

## 2.2 Element Quality

As discretisation errors are dependent upon element shape as well as size, a number of different measures of element quality have been proposed, e.g. [1, 5, 18, 21, 22]. Here we use the element quality measure for triangles proposed by [22]:

$$q^M(\triangle) = \underbrace{12\sqrt{3}\frac{|\triangle|_M}{|\partial\triangle|_M^2}}_{I}\,\underbrace{F\left(\frac{|\partial\triangle|_M}{3}\right)}_{II}, \tag{1}$$

where $|\triangle|_M$ is the area of element $\triangle$ and $|\partial\triangle|_M$ is its perimeter as measured in a Riemannian space locally defined by the metric tensor $M$ as evaluated at the element's centre. The first factor $(I)$ is used to control the shape of element $\triangle$. For an equilateral triangle with sides of length $l$ in metric space, $|\triangle| = l^2\sqrt{3}/4$ and $|\partial\triangle| = 3l$; and so $I = 1$. For non-equilateral triangles, $I < 1$. The second factor ($II$) controls the size of element $\triangle$. The function $F$ is smooth and defined as:

$$F(x) = (\min(x, 1/x)(2 - \min(x, 1/x)))^3\,, \tag{2}$$

which has a single maximum of unity with $x = 1$ and decreases smoothly away from this with $F(0) = F(\infty) = 0$. Therefore, $II = 1$ when the sum of the lengths of the edges of $\triangle$ is equal to 3, i.e. an equilateral triangle with sides of unit length, and $II < 1$ otherwise. Hence, taken together, the two factors in (1) yield a maximum value of unity for an equilateral triangle with edges of unit length, and decreases smoothly to zero as the element becomes less ideal.

## 2.3 Overall Adaptation Procedure

The mesh is adapted through a series of local operations: edge collapse (Sect. 2.4.1); edge refinement (Sect. 2.4.2); element-edge swaps (Sect. 2.4.3); and vertex smoothing (Sect. 2.4.4). While the first two of these operations control the local resolution, the latter two operations are used to improve the element quality.

**Algorithm 1** Mesh optimisation procedure

```
Inputs: $\mathscr{M}, \mathscr{S}$.
$(\mathscr{M}^*, \mathscr{S}^*) \leftarrow coarsen(\mathscr{M}, \mathscr{S})$
**repeat**
    $(\mathscr{M}^*, \mathscr{S}^*) \leftarrow refine(\mathscr{M}^*, \mathscr{S}^*)$
    $(\mathscr{M}^*, \mathscr{S}^*) \leftarrow coarsen(\mathscr{M}^*, \mathscr{S}^*)$
    $(\mathscr{M}^*, \mathscr{S}^*) \leftarrow swap(\mathscr{M}^*, \mathscr{S}^*)$
**until** (maximum number of iterations reached) **or** (algorithm convergence)
$(\mathscr{M}^*, \mathscr{S}^*) \leftarrow smooth(\mathscr{M}^*, \mathscr{S}^*)$
**return** $\mathscr{M}^*$
```

Algorithm 1 gives a high level view of the anisotropic mesh adaptation procedure as described by Li et al. [17]. The inputs are $\mathscr{M}$, the piecewise linear mesh from the modelling software, and $\mathscr{S}$, the node-wise metric tensor field which specifies anisotropically the local mesh resolution requirements. Coarsening is initially applied to reduce the subsequent computational and communication overheads. The second stage involves the iterative application of refinement, coarsening and swapping to optimise the resolution and the quality of the mesh. The algorithm terminates once the mesh optimisation algorithm converges or after a maximum number of iterations has been reached. Finally, mesh quality is fine-tuned using some vertex smoothing algorithm (e.g. quality-constrained Laplacian smoothing [11], optimisation-based smoothing [12]), which aims primarily at improving worst-element quality.

## 2.4 Adaptation Kernels

### 2.4.1 Coarsening

Coarsening is the process of lowering mesh resolution locally by removing mesh elements, leading to a reduction in the computational cost. Here this is done by collapsing an edge to a single vertex, thereby removing all elements that contain this edge. An example of this operation is shown in Fig. 1.

### 2.4.2 Refinement

Refinement is the process of increasing mesh resolution locally. It encompasses two operations: splitting of edges; and subsequent division of elements. When an edge is longer than desired, it is bisected. An element can be split in three different ways, depending on how many of its edges are bisected:

1. When only one edge is marked for refinement, the element can be split across the line connecting the mid-point of the marked edge and the opposite vertex. This

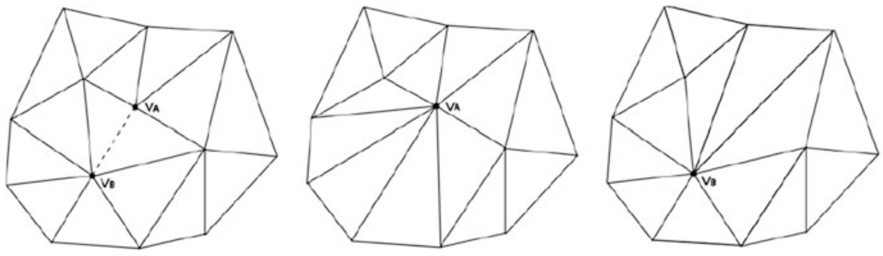


**Fig. 1** Edge collapse: the dashed edge in the *left figure* is reduced into a single vertex by bringing vertex $V_B$ on top of vertex $V_A$, as can be seen in the *middle figure*. The two elements that used to share the dashed edge are deleted. Edge collapse is an oriented operation, i.e. eliminating the edge by moving $V_B$ onto $V_A$ results in a different local patch than moving $V_A$ onto $V_B$, which can be seen in the *right figure*

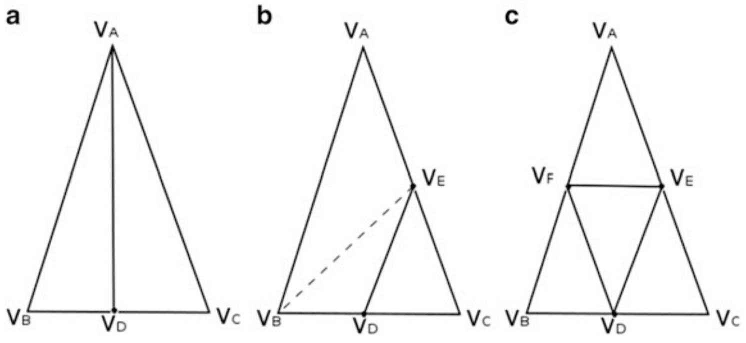


**Fig. 2** Mesh resolution can be increased either by bisecting an element across one of its edges (1:2 split, **a**), by performing a 1:3 split (**b**) or by performing regular refinement to that element (1:4 split, **c**)

operation is called bisection and an example of it can be seen on the left side of Fig. 2 (left shape).

2. When two edges are marked for refinement, the element is divided into three new elements. This case is shown in Fig. 2 (middle shape). The parent element is split by creating a new edge connecting the mid-points of the two marked edges. This leads to a newly created triangle and a non-conforming quadrilateral. The quadrilateral can be split into two conforming triangles by dividing it across one of its diagonals, whichever is shorter.
3. When all three edges are marked for refinement, the element is divided into four new elements by connecting the mid-points of its edges with each other. This operation is called regular refinement and an example of it can be seen in Fig. 2 (right shape).

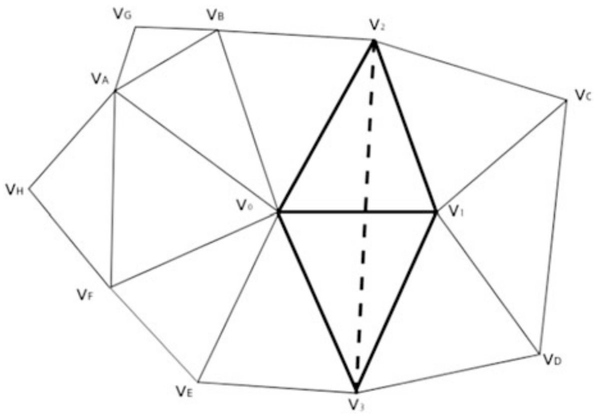


**Fig. 3** Flipping the common edge $\overline{V_0V_1}$ results in the removal of triangles $\widehat{V_0V_1V_2}$ and $\widehat{V_0V_1V_3}$ and their replacement with new triangles $\widehat{V_0V_2V_3}$ and $\widehat{V_1V_2V_3}$

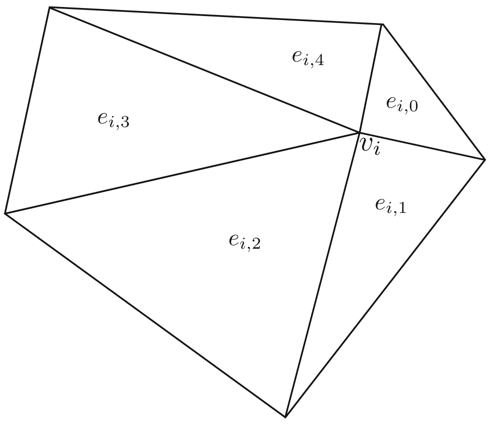


**Fig. 4** Local mesh patch: $\mathbf{v}_i$ is the vertex being relocated; $\{e_{i,0}, \ldots, e_{i,m}\}$ is the set of elements connected to $\mathbf{v}_i$

### 2.4.3 Swapping

In 2D, swapping is done in the form of edge flipping, i.e. flipping an edge shared by two elements, see Fig. 3. The operation considers the quality of the swapped elements—if the minimum element quality has improved then the original mesh triangles are replaced with the edge swapped elements.

### 2.4.4 Quality Constrained Laplacian Smooth

The kernel of the vertex smoothing algorithm should relocate the central vertex such that the local mesh quality is increased (see Fig. 4). Probably the best known heuristic for mesh smoothing is Laplacian smoothing, first proposed by Field [10]. This method operates by moving a vertex to the barycentre of the set of vertices connected by a mesh edge to the vertex being repositioned. The same approach can be implemented for non-Euclidean spaces; in that case all measurements of

**Algorithm 2** Smart smoothing kernel: a Laplacian smooth operation is accepted only if it does not reduce the infinity norm of local element quality

```
$\mathbf{v}_i^0 \leftarrow \mathbf{v}_i$
$quality^0 \leftarrow Q(\mathbf{v}_i)$
$n \leftarrow 1$
$\mathbf{v}_i^n \leftarrow \mathbf{v}_i^{\mathscr{L}}$    ▷ Initialise vertex location using Laplacian smooth
$M_i^n \leftarrow metric\_interpolation(\mathbf{v}_i^n)$
$quality^n = Q(\mathbf{v}_i^n)$    ▷ Calculate the new local quality for this relocation.
while $(n \leq max\_iteration)$ and $(quality_i^n - quality_i^0 < \sigma_q)$ do
    $\mathbf{v}_i^{n+1} \leftarrow (\mathbf{v}_i^n + \mathbf{v}_i^0)/2$
    $M_i^{n+1} \leftarrow metric\_interpolation(\mathbf{v}_i^{n+1})$
    $quality^{n+1} \leftarrow Q(\mathbf{v}_i^{n+1})$
    $n = n + 1$
if $quality_i^n - quality_i^0 > \sigma_q$ then    ▷ Accept if local quality is improved
    $\mathbf{v}_i \leftarrow \mathbf{v}_i^n$
    $\mathbf{M_i} \leftarrow \mathbf{M_i^n}$
```

length and angle are performed with respect to a metric tensor field that describes the desired size and orientation of mesh elements [18]. Therefore, in general, the proposed new position of a vertex $\mathbf{v}_i^{\mathscr{L}}$ is given by

$$\mathbf{v}_i^{\mathscr{L}} = \frac{\sum_{j=1}^{J} ||\mathbf{v}_i - \mathbf{v}_j||_M \mathbf{v}_j}{\sum_{j=1}^{J} ||\mathbf{v}_i - \mathbf{v}_j||_M}, \tag{3}$$

where $\mathbf{v}_j$, $j = 1, \ldots, J$, are the vertices connected to $\mathbf{v}_i$ by an edge of the mesh, and $|| \cdot ||_M$ is the norm defined by the edge-centred metric tensor $M_{ij}$. In Euclidean space, $M_{ij}$ is the identity matrix.

As noted by Field [10], the application of pure Laplacian smoothing can actually decrease local element quality; at times, elements can even become inverted. Therefore, repositioning is generally constrained in some way to prevent local decreases in mesh quality. One variant of this, termed *smart Laplacian smoothing* by Freitag and Ollivier-Gooch [11] (while they only consider the Euclidean geometry it is straightforward to extend to Riemannian geometry), is summarised in Algorithm 2. This method accepts the new position defined by a Laplacian smooth only if it increases the infinity norm of local element quality, $Q_i$ (i.e. the quality of the worst local element):

$$Q(\mathbf{v}_i) \equiv \|\boldsymbol{q}\|_\infty, \tag{4}$$

where $i$ is the index of the vertex under consideration and $\boldsymbol{q}$ is the vector of the element qualities from the local patch.

## 3 Thread-Level Parallelism in Mesh Optimisation

To allow fine grained parallelisation of anisotropic mesh adaptation we make extensive use of maximal independent sets. This approach was first suggested in a parallel framework proposed by Freitag et al. [13]. However, while this approach avoids updates being applied concurrently to the same neighbourhood, data writes will still incur significant lock and synchronisation overheads. For this reason we incorporate a deferred updates strategy, described below, to minimise synchronisations and allow parallel writes.

In the same paper [13] the authors describe the need for propagation of operations. Adaptive operations need to be propagated to adjacent vertices/edges because topological changes or changes in element quality might give rise to new configurations of better quality.

### *3.1 Design Choices*

Before presenting the adaptive algorithms, it is necessary to give a brief description of the data structures used to store mesh-related information. Following that, we present a set of techniques which help us avoid hazards and data races and guarantee fast and safe concurrent read/write access to mesh data.

#### 3.1.1 Mesh Data Structures

The minimal information necessary to represent a mesh is an element-node list (we refer to it in this article as `ENList`), which is implemented in PRAgMaTIc as a C++ Standard template library (STL) vector container class storing vertex IDs (`std::vector<int>`), and an array of vertex coordinates (referred to as `coords`), which is an STL vector of coordinates (`std::vector<double>`). Element `eid` is comprised of vertices `ENList[3*eid]`, `ENList[3*eid+1]` and `ENList[3*eid+2]`, whereas the $x$- and $y$-coordinates of vertex `vid` are stored in `coords[2*vid]` and `coords[2*vid+1]` respectively. The metric tensor field is similarly stored in the STL vector `metric`.

All necessary structural information about the mesh can be extracted from `ENList`. However, it is convenient to create and maintain two additional adjacency-related structures, the node-node adjacency list (referred to as `NNList`) and the node-element adjacency list (referred to as `NEList`). As `NNList` is a ragged array it is implemented as `std::vector< std::vector<int> >` where the vector `NNList[vid]` contains the IDs of all vertices adjacent to vertex `vid`. Similarly, `NEList` is implemented as an STL vector of STL sets of element IDs (`std::vector< std::set<int> >`) and `NEList[vid]` contains the IDs of all elements which vertex `vid` is part of.

It should be noted that, contrary to common approaches in other adaptive frameworks, we do not use other adjacency-related structures such as element-element or edge-edge lists. Manipulating these lists and maintaining their consistency throughout the adaptation process is quite complex and constitutes an additional parallel overhead. Instead, we opted for the approach of finding all necessary adjacency information on the fly using `ENList`, `NNList` and `NEList`.

### 3.1.2 Colouring

There are two types of hazards when running mesh optimisation algorithms in parallel: structural hazards and data races. The term *structural hazards* refers to the situation where an adaptive operation results in invalid or non-conforming edges and elements. For example, on the local patch in Fig. 3, if two threads flip edges $\overline{V_0V_1}$ and $\overline{V_0V_2}$ at the same time, the result will be two new edges $\overline{V_2V_3}$ and $\overline{V_1V_B}$ crossing each other. Structural hazards for all adaptive algorithms are avoided by colouring a graph whose nodes are defined by the mesh vertices and edges are defined by the mesh edges. Maximal independent sets are readily selected by calculating the intersection between the set of vertices of each colour and the set of active vertices.

The fact that topological changes are made to the mesh means that after an independent set has been processed the graph colouring has to be recalculated. Therefore, a fast scalable graph colouring algorithm is vital to the overall performance. In this work we use a parallel colouring algorithm described by [15]. This algorithm can be described as having three stages: (a) initial pseudo-colouring where vertices are coloured in parallel and invalid colourings are possible; (b) loop over the graph to detect invalid colours arising from the first stage; (c) the detected invalid colours are resolved in serial. Between adaptive sweeps through independent sets it is only necessary to execute stages (b) and (c) to resolve the colour conflicts introduced by changes to the mesh topology.

### 3.1.3 Deferred Operations Mechanism

Data race conditions can appear when two or more threads try to update the same adjacency list. An example can be seen in Fig. 5. Having coloured the mesh, two threads are allowed to process vertices $V_B$ and $V_C$ at the same time without structural hazards. However, `NNList[` $V_A$ `]` and `NEList[` $V_A$ `]` must be updated. If both threads try to update them at the same time there will be a data race which could lead to data corruption. One solution could be a distance-2 colouring of the mesh (a distance-$k$ colouring of $\mathscr{G}$ is a colouring in which no two vertices share the same colour if these vertices are up to $k$ edges away from each other or, in other words, if there is a path of length $\leq k$ from one vertex to the other). Although this solution guarantees a race-free execution, a distance-2 colouring would increase the chromatic number, thereby reducing the size of the independent sets and therefore the available parallelism. Therefore, an alternative solution is sought.

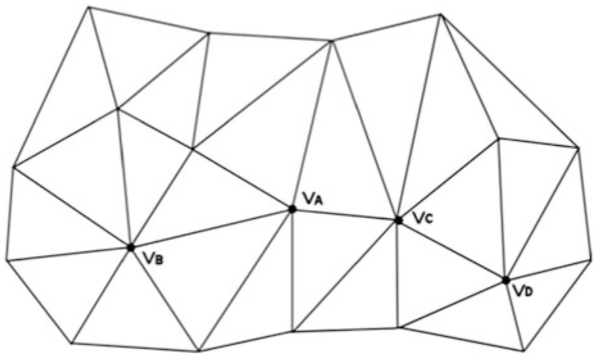


**Fig. 5** Example of hazards when running edge collapse in parallel. $V_B$ is about to collapse onto $V_A$. The operation is executed by thread $T_1$. Clearly, $V_A$ cannot collapse at the same time. Additionally, $V_C$ cannot collapse either, because it affects $V_A$'s adjacency list. If a thread $T_2$ executes the collapse operation collapse on $V_C$, then both $T_1$ and $T_2$ will attempt to modify $V_A$'s adjacency list concurrently, which can lead to data corruption. This race can be eliminated using the deferred-updates mechanism

In a shared-memory environment with `nthreads` OpenMP threads, every thread has a private collection of `nthreads` lists, one list for each OpenMP thread. When `NNList[i]` or `NEList[i]` have to be updated, the thread does not commit the update immediately; instead, it pushes the update back into the list corresponding to thread with ID $tid = i \% nthreads$. At the end of the adaptive algorithm, every thread `tid` visits the private collections of all OpenMP threads (including its own), locates the list that was reserved for `tid` and commits the operations which are stored there. This way, it is guaranteed that for any vertex $V_i$, `NNList[` $V_i$ `]` and `NEList[` $V_i$ `]` will be updated by only one thread. Because updates are not committed immediately but are deferred until the end of the iteration of an adaptive algorithm, we call this technique the *deferred updates*. A typical usage scenario is demonstrated in Algorithm 3.

### 3.1.4 Worklists and Atomic Operations

There are many cases where it is necessary to create a worklist of items which need to be processed. An example of such a case is the creation of the active sub-mesh in coarsening and swapping, as will be described in Sect. 3.3. Every thread keeps a local list of vertices it has marked as active and all local worklists have to be accumulated into a global worklist, which essentially is the set of all vertices comprising the active sub-mesh.

One approach is to wait for every thread to exit the parallel loop and then perform a prefix sum [4] on the number of vertices in its private list. After that, every thread knows its index in the global worklist at which it has to copy the private list. This method has the disadvantage that every thread must wait for all other threads to

**Algorithm 3** Typical example of using the deferred updates mechanism

```
typedef std::vector<Updates> DeferredOperationsList;
int nthreads = omp_get_max_threads();

// Create nthreads collections of deferred operations lists
std::vector< std::vector<DeferredOperationsList> > defOp;
defOp.resize(nthreads);

#pragma omp parallel
{
  // Every OMP thread executes
  int tid = omp_get_thread_num();
  defOp[tid].resize(nthreads);
  // By now, every OMP thread has allocated one list per thread

  // Execute one iteration of an adaptive algorithm in parallel
  // Defer any updates until the end of the iteration
  #pragma omp for
  for(int i=0; i<nVertices; ++i){
    execute kernel(i);
    // Update will be committed by thread i%nthreads
    // where the modulo avoids racing.
    defOp[tid][i%nthreads].push_back(some_update_operation);
  }

  // Traverse all lists which were allocated for thread tid
  // and commit any updates found
  for(int i=0; i<nthreads; ++i){
    commit_all_updates(defOp[i][tid]);
  }
}
```

exit the parallel loop, synchronise with them to perform the prefix sum and finally copy its private data into the global worklist. Profiling data indicates that this way of manipulating worklists is a significant limiting factor towards achieving good scalability.

Experimental evaluation showed that, at least on the Intel Xeon, a better method is based on atomic operations on a global integer variable which stores the size of the worklist needed so far. Every thread which exits the parallel loop increments this integer atomically while caching the old value. This way, the thread knows immediately at which index it must copy its private data and increments the integer by the size of this data, so that the next thread which will access this integer knows in turn its index at which its private data must be copied. Caching the old value before the atomic increment is known in OpenMP terminology as *atomic capture*. Support for atomic capture operations was introduced in OpenMP 3.1. This functionality has also been supported by GNU extensions (intrinsics) since GCC 4.1.2, known under the name *fetch-and-add*. An example of using this technique is shown in Algorithm 4.

**Algorithm 4** Example of creating a worklist using OpenMP's atomic capture operations

```
int worklistSize = 0; // Points to end of the global worklist
std::vector<Item> globalWorklist;

// Pre-allocate enough space
globalWorklist.resize(some_appropriate_size);

#pragma omp parallel
{
  std::vector<Item> private_list;

  // Private list - no need to synchronise at end of loop.
  #pragma omp for nowait
  for(all items which need to be processed){
    do_some_work();
    private_list.push_back(item);
  }

  // Private variable - the index in the global worklist
  // at which the thread will copy the data in private_list.
  int idx;

  #pragma omp atomic capture
  {
    idx = worklistSize;
    worklistSize += private_list.size();
  }

  memcpy(&globalWorklist[idx], &private_list[0],
             private_list.size() * sizeof(Item));

}
```

Note the `nowait` clause at the end of the `#pragma omp for` directive. A thread which exits the loop does not have to wait for the other threads to exit. It can proceed directly to the atomic operation. It has been observed that dynamic scheduling for OpenMP for-loops is what works best for most of the adaptive loops in mesh optimisation because of the irregular load balance across the mesh. Depending on the nature of the loop and the chunk size, threads will exit the loop at significantly different times. Instead of having some threads waiting for others to finish the parallel loop, with this approach they do not waste time and proceed to the atomic increment. The profiling data suggests that the overhead or spinlock associated with atomic-capture operations is insignificant.

### 3.1.5 Reflection on Alternatives

Our initial approach to dealing with structural hazards, data races and propagation of adaptivity was based on a thread-partitioning scheme in which the mesh was split into as many sub-meshes as there were threads available. Each thread was then free to process items inside its own partition without worrying about hazards and races. Items on the halo of each thread-partition were locked (analogous to the MPI parallel strategy); those items would be processed later by a single thread. However, this approach did not result in good scalability for a number of reasons. Partitioning the mesh was a significant serial overhead, which was incurred repeatedly as the adaptive algorithms changed mesh topology and invalidated the existing partitioning. In addition, the single-threaded phase of processing halo items was another hotspot of this thread-partition approach. In line with Amdahl's law, these effects only become more pronounced as the number of threads is increased. For these reasons this thread-level domain decomposition approach was not pursued further.

## *3.2 Refinement*

Every edge can be processed and refined without being affected by what happens to adjacent edges. Being free from structural hazards, the only issue we are concerned with is thread safety when updating mesh data structures. Refining an edge involves the addition of a new vertex to the mesh. This means that new coordinates and metric tensor values have to be appended to `coords` and `metric` and adjacency information in `NNList` has to be updated. The subsequent element split leads to the removal of parent elements from `ENList` and the addition of new ones, which, in turn, means that `NEList` has to be updated as well. Appending new coordinates to `coords`, metric tensors to `metric` and elements to `ENList` is done using the thread worklist strategy described in Sect. 3.1.4, while updates to `NNList` and `NEList` can be handled efficiently using the deferred operations mechanism.

The two stages, namely edge refinement and element refinement, of our threaded implementation are described in Algorithms 5 and 6, respectively. The procedure begins with the traversal of all mesh edges. Edges are accessed using `NNList`, i.e. for each mesh vertex $V_i$ the algorithm visits $V_i$'s neighbours. This means that edge refinement is a directed operation, as edge $\overline{V_i V_j}$ is considered to be different from edge $\overline{V_j V_i}$. Processing the same physical edge twice is avoided by imposing the restriction that we only consider edges for which $V_i$'s ID is less than $V_j$'s ID. If an edge is found to be longer than desired, then it is split in the middle (in metric space) and a new vertex $V_n$ is created. $V_n$ is associated with a pair of coordinates and a metric tensor. It also needs an ID. At this stage, $V_n$'s ID cannot be determined. Once an OpenMP thread exits the edge refinement phase, it can proceed (without synchronisation with the other threads) to fix vertex IDs and append the new data it created to the mesh. The thread captures the number of mesh vertices *index* =

**Algorithm 5** Edge-refinement

```
Global worklist of split edges $\mathscr{W}$, refined_edges_per_element[NElements]
#pragma omp parallel
    private : $split\_cnt \leftarrow 0, newCoords, newMetric, newVertices$
    #pragma omp for schedule(dynamic)
    for all vertices $V_i$ do
        for all vertices $V_j$ adjacent to $V_i$, $ID(V_i) < ID(V_j)$ do
            if length of edge $\overline{V_i V_j} > L_{max}$ then
                $V_n \leftarrow$ new vertex of split edge $\overline{V_i V_n V_j}$; Append new
                coordinates, interpolated metric, split edge to newCoords,
                newMetric, newVertices; $split\_cnt \leftarrow split\_cnt + 1$
    #pragma omp atomic capture
        $index \leftarrow NNodes; NNodes \leftarrow NNodes + splint\_cnt$
    Copy newCoords into coords, newMetric into metric
    for all edges $e_i \in newVertices$ do
        $e_i = \overline{V_i V_n V_j}$; increment ID of $V_n$ by index
    Copy newVertices into $\mathscr{W}$
    #pragma omp barrier
    #pragma omp parallel for schedule(dynamic)
    for all Edges $e_i \in \mathscr{W}$ do
        Replace $V_j$ with $V_n$ in NNList[$V_i$]; replace $V_i$ with $V_n$ in NNList[$V_j$]
        Add $V_i$ and $V_j$ to NNList[$V_n$]
        for all elements $E_i \in \{NEList[V_i] \cap NEList[V_j]\}$ do
            Mark edge $e_i$ as refined in refined_edges_per_element[$E_i$].
```

**Algorithm 6** Element refinement phase

```
#pragma omp parallel
    private : $newElements$
    #pragma omp for schedule(dynamic)
    for all elements $E_i$ do
        REFINE_ELEMENT($E_i$)
        Append additional elements to newElements.
    Resize ENList.
    Parallel copy of newElements into ENList.
```

*NNodes* and increments it atomically by the number of new vertices it created. After capturing the index, the thread can assign IDs to the vertices it created and also copy the new coordinates and metric tensors into `coords` and `metric`, respectively.

Before proceeding to element refinement, all split edges are accumulated into a global worklist. For each split edge $\overline{V_i V_j}$, the original vertices $V_i$ and $V_j$ have to be connected to the newly created vertex $V_n$. Updating `NNList` for these vertices cannot be deferred. Most edges are shared between two elements, so if the update was deferred until the corresponding element were processed, we would run the risk of committing these updates twice, once for each element sharing the edge. Updates can be committed immediately, as there are no race conditions when accessing `NNList` at this point. Besides, for each split edge we find the (usually two) elements sharing it. For each element, we record that this edge has been split. Doing so makes element refinement much easier, because as soon as we visit an element we will know immediately how many and which of its edges have been split. An array of length `NElements` stores this type of information.

During mesh refinement, elements are visited in parallel and refined independently. It should be noted that all updates to `NNList` and `NEList` are deferred operations. After finishing the loop, each thread uses the worklist method to append the new elements it created to `ENList`. Once again, no thread synchronisation is needed.

Compared to Freitag's task graph approach, this parallel refinement algorithm has the advantage of not requiring any mesh colouring and having low synchronisation overhead as. Additionally, the element refinement phase is based on the results of the edge refinement phase, so we completely avoid having non-conformities and the subsequent need to propagate operations in order to eliminate them.

## 3.3 Coarsening

Because any decision on whether to collapse an edge is strongly dependent upon what other edges are collapsing in the immediate neighbourhood of elements, an operation task graph for coarsening has to be constructed. Edge collapse is based on the removal of vertices, i.e. the elemental operation for edge collapse is the removal of a vertex. Therefore, the operation task graph $\mathcal{G}$ is the mesh itself.

Figure 5 demonstrates what needs to be taken into account in order to perform parallel coarsening safely. It is clear that adjacent vertices cannot collapse concurrently, so a distance-1 colouring of the mesh is sufficient in order to avoid structural hazards. This colouring also enforces processing of vertices topologically at least *every other one* which prevents skewed elements forming during significant coarsening [8, 17].

An additional consideration is that vertices which are two edges away from each other share some common vertex $V_{common}$. Removing both vertices at once means that $V_{common}$'s adjacency list will have to be modified concurrently by two different threads, leading to data races. These races can be avoided using the deferred operations mechanism.

Algorithm 7 illustrates a thread parallel version of mesh edge collapse. Coarsening is divided into two phases: the first sweep through the mesh identifies what edges are to be removed, see Algorithm 8; and the second phase actually applies the coarsening operation, see Algorithm 9. Function *coarsen_identify(*$V_i$*)* takes as argument the ID of a vertex $V_i$, decides whether any of the adjacent edges can collapse and returns the ID of the target vertex $V_t$ onto which $V_i$ should collapse (or a negative value if no adjacent edge can be removed). *coarsen_kernel(*$V_i$*)* performs the actual collapse, i.e. removes $V_i$ from the mesh, updates vertex adjacency information and removes the two deleted elements from the element list.

**Algorithm 7** Edge collapse

```
Allocate $dynamic\_vertex, worklist$.
#pragma omp parallel
    #pragma omp for schedule(static)
    for all vertices $V_i$ do $dynamic\_vertex[V_i] \leftarrow -2$
    Colour mesh
    repeat
        #pragma omp for schedule(dynamic)
        for all vertices $V_i$ do
            if $dynamic\_vertex[V_i] == -2$ then
                $dynamic\_vertex[V_i] \leftarrow$COARSEN_IDENTIFY($V_i$)
        if dynamic vertex count $== 0$ then break
        $\mathscr{I}_m \leftarrow$ maximal independent set of dynamic vertices
        #pragma omp for schedule(dynamic)
        for all $V_i \in \mathscr{I}_m$ do
                                                        ▷ mark all neighbours for re-evaluation
            for all vertices $V_j \in$ NNList[$V_i$] do
                $dynamic\_vertex[V_j] \leftarrow -2$
            $dynamic\_vertex[V_i] \leftarrow -1$
            COARSEN_KERNEL($V_i$)
        Commit deferred operations.
        Repair colouring
    until true
```

**Algorithm 8** coarsen_identify

```
procedure COARSEN_IDENTIFY($V_i$)
    $S_i \leftarrow$ the set of all edges connected to $V_i$
    $S^0 \leftarrow S_i$
    repeat
        $E_j \leftarrow$ shortest edge in $S^j$
        if length of $E_j > L_{min}$ then             ▷ if shortest edge is of acceptable
            return -1                                 ▷ length, no edge can be removed
        $V_t \leftarrow$ the other vertex that bounds $E_j$
        evaluate collapse of $E_j$ with the collapse of $V_i$ onto $V_t$
        if ($\forall$ edges $\in S_i \leq L_{max}$) and ($\nexists$ inverted elements) then
            return $V_t$
        else
            remove $E_j$ from $S^j$                   ▷ $E_j$ is not a candidate for collapse
    until $S_i = \emptyset$
```

**Algorithm 9** Coarsen_kernel with deferred operations

```
procedure COARSEN_KERNEL($V_i$)
    $V_t \leftarrow dynamic\_vertex[V_i]$
    $removed\_elements \leftarrow$ NEList[$V_i$] $\cap$ NEList[$V_t$]
    $common\_patch \leftarrow$ NNList[$V_i$] $\cap$ NNList[$V_t$]
    for all $E_i \in removed\_elements$ do
        $V_o \leftarrow$ the other vertex of $E_i = \widehat{V_i V_t V_o}$
        NEList[$V_o$].erase($E_i$)                          ▷ deferred operation
        NEList[$V_t$].erase($E_i$)                          ▷ deferred operation
        NEList[$V_i$].erase($E_i$)
        ENList[3*$E_i$] $\leftarrow -1$          ▷ erase element by resetting its first vertex
    for all $E_i \in$ NEList[$V_i$] do
        replace $V_i$ with $V_t$ in ENList[3*$E_i$+{0,1,2}]
        NEList[$V_t$].add($E_i$)                            ▷ deferred operation
    remove $V_i$ from NNList[$V_t$]                         ▷ deferred operation
    for all $V_c \in common\_patch$ do
        remove $V_i$ from NNList[$V_c$]                     ▷ deferred operation
    for all $V_n \notin common\_patch$ do
        replace $V_i$ with $V_t$ in NNList[$V_n$]
        add $V_n$ to NNList[$V_t$]                          ▷ deferred operation
    NNList[$V_i$].clear()
    NEList[$V_i$].clear()
```

Parallel coarsening begins with the initialisation of array *dynamic_vertex* which is defined as:

$$dynamic\_vertex[V_i] = \begin{cases} -1 & V_i \text{ cannot collapsed,} \\ -2 & V_i \text{ must be re-evaluated,} \\ V_t & V_i \text{ is about to collapse onto } V_t. \end{cases}$$

At the beginning, the whole array is initialised to -2, so that all mesh vertices will be considered for collapse.

In each iteration of the outer coarsening loop, *coarsen_identify_kernel* is called for all vertices which have been marked for (re-)evaluation. Every vertex for which $dynamic\_vertex[V_i] \geq 0$ is said to be *dynamic* or *active*. At this point, a reduction in the total number of active vertices is necessary to determine whether there is anything left for coarsening or the algorithm should exit the loop.

Next up, we find the maximal independent set of active vertices $\mathscr{I}_m$. Working with independent sets not only ensures safe parallel execution, but also enforces the *every other vertex* rule. For every active vertex $V_r \in \mathscr{I}_m$ which is about to collapse, the local neighbourhood of all vertices $V_a$ formerly adjacent to $V_r$ changed and target vertices $dynamic\_vertex[V_a]$ may not be suitable choices any more. Therefore, when $V_r$ is erased, all its neighbours are marked for re-evaluation. This is how propagation of coarsening is implemented.

Algorithm 9 describes how the actual coarsening takes place in terms of modifications to mesh data structures. Updates which can lead to race conditions have been pointed out. These updates are deferred until the end of processing of the independent set. Before moving to the next iteration, all deferred operations are committed and colouring is repaired because edge collapse may have introduced inconsistencies.

## *3.4 Swapping*

The data dependencies in edge swapping are virtually identical to those of edge coarsening. Therefore, it is possible to reuse the same thread parallel algorithm as for coarsening in the previous section with slight modifications

In order to avoid maintaining edge-related data structures (e.g. edge-node list, edge-edge adjacency lists etc.), an edge can be expressed in terms of a pair of vertices. Just like in refinement, we define an edge $E_{ij}$ as a pair of vertices $(V_i, V_j)$, with $ID(V_i) < ID(V_j)$. We say that $E_{ij}$ is outbound from $V_i$ and inbound to $V_j$. Consequently, the edge $E_{ij}$ can be marked for swapping by adding $V_j$ to *marked_edges*$[V_i]$. Obviously, a vertex $V_i$ can have more than one outbound edge, so unlike *dynamic_vertex* in coarsening, *marked_edges* in swapping needs to be a vector of sets.

The algorithm begins by marking all edges. It then enters a loop which is terminated when no marked edges remain. The maximal independent set $\mathscr{I}_m$ of active vertices is calculated. A vertex is considered active if at least one of its outbound edges is marked. Following that, threads process all active vertices of $\mathscr{I}_m$ in parallel. The thread processing vertex $V_i$ visits all edges in *marked_edges*$[V_i]$ one after the other and examines whether they can be swapped, i.e. whether the operation will improve the quality of the two elements sharing that edge. It is easy to see that swapping two edges in parallel which are outbound from two independent vertices involves no structural hazards.

Propagation of swapping is similar to that of coarsening. Consider the local patch in Fig. 3 and assume that a thread is processing vertex $V_0$. If edge $\overline{V_0V_1}$ is flipped, the two elements sharing that edge change in shape and quality, so all four edges surrounding those elements (forming the rhombus in bold) have to be marked for processing. This is how propagation is implemented in swapping.

One last difference between swapping and coarsening is that $\mathscr{I}_m$ needs to be traversed more than once before proceeding to the next one. In the same example as above, assume that all edges adjacent to $V_0$ are outbound and marked. If edge $\overline{V_0V_1}$ is flipped, adjacency information for $V_1$, $V_2$ and $V_3$ has to be updated. These updates have to be deferred because another thread might try to update the same lists at the same time (e.g. the thread processing edge $\overline{V_CV_1}$). However, not committing the changes immediately means that the thread processing $V_0$ has a stale view of the local patch. More precisely, `NEList`$[V_2]$ and `NEList`$[V_3]$ are invalid and cannot be used to find what elements edges $\overline{V_0V_2}$ and $\overline{V_0V_3}$ are part of. Therefore, these two edges cannot be processed until the deferred operations have been committed. On the other hand, the rest of $V_0$'s outbound edges are free to be processed. Once all

**Algorithm 10** Thread-parallel mesh smoothing

```
repeat
    $relocate\_count \leftarrow 0$
    for $colour = 1 \rightarrow k$ do
        #pragma omp for schedule(static)
        for all $i \in \mathscr{V}^c$ do
                                        ▷ $move\_success$ is true if vertex was relocated,
            $move\_success \leftarrow smooth\_kernel(i)$          ▷ false otherwise.
            if $move\_success$ then
                $relocate\_count \leftarrow relocate\_count + 1$
until $(n \geq max\_iteration)$ or $(relocate\_count = 0)$
```

threads have processed whichever edges they can for all vertices of the independent set, deferred operations are committed and threads traverse the independent set again (up to two more times in 2D) to process what had been skipped before.

## 3.5 Smoothing

Algorithm 10 illustrates the colouring based algorithm for mesh smoothing and is described in greater detail in [16]. In this algorithm the graph $\mathscr{G}(\mathscr{V}, \mathscr{E})$ consists of sets of vertices $\mathscr{V}$ and edges $\mathscr{E}$ that are defined by the vertices and edges of the computational mesh. By computing a vertex colouring of $\mathscr{G}$ we can define independent sets of vertices, $\mathscr{V}^c$, where $c$ is a computed colour. Thus, all vertices in $\mathscr{V}^c$, for any $c$, can be updated concurrently without any race conditions on dependent data. This is clear from the definition of the smoothing kernel in Sect. 2.4.4. Hence, within a node, thread-safety is ensured by assigning a different independent set $\mathscr{V}^c$ to each thread.

## 4 Results

In order to evaluate the parallel performance, an isotropic mesh was generated on the unit square with using approximately $200 \times 200$ vertices. A synthetic solution $\psi$ is defined to vary in time and space:

$$\psi(x, y, t) = 0.1 \sin(50x + 2\pi t/T) + \arctan(-0.1/(2x - \sin(5y + 2\pi t/T))), \tag{5}$$

where $T$ is the period. An example of the field at $t = 0$ is shown in Fig. 6. This is a good choice as a benchmark as it contains multi-scale features and a shock front. These are the typical solution characteristics where anisotropic adaptive mesh methods excel.

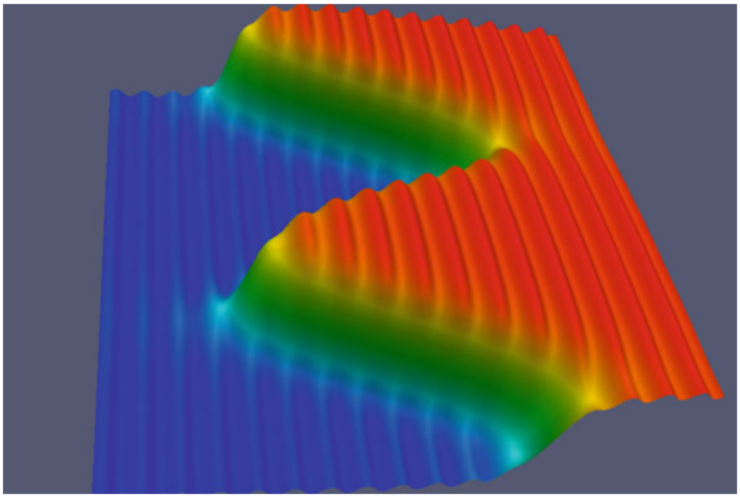


**Fig. 6** Benchmark solution field

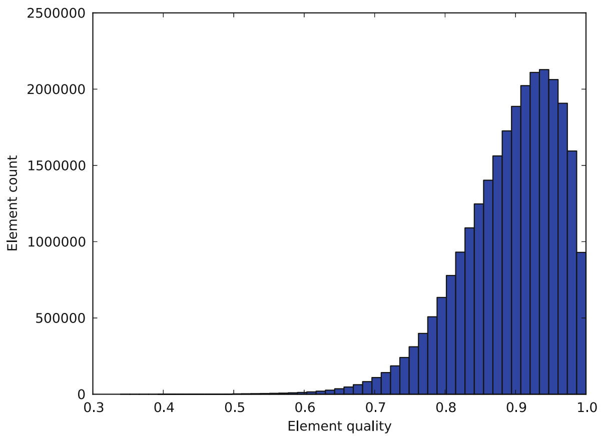


**Fig. 7** Histogram of element qualities aggregated over all iterations

Because mesh adaptation has a very irregular workload we simulate a time varying scenario where $t$ varies from 0 to 51 in increments of unity and we use the mean and standard deviations when reporting performance results. To calculate the metric we used the $L^p$-norm as described by [6], where $p \equiv 2$. The number of mesh vertices and elements maintains an average of approximately $250k$ and $500k$ respectively. As the field evolves all of the adaptive operations are heavily used, thereby giving an overall profile of the execution time.

In order to demonstrate the correctness of the adaptive algorithm we plot a histogram (Fig. 7) showing the quality of all element aggregated over all time steps.

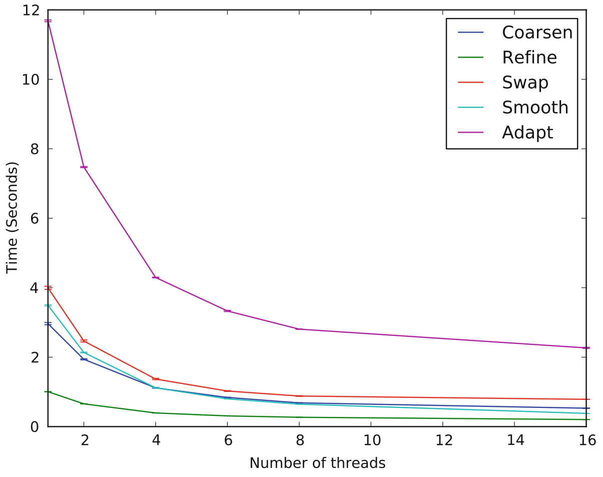


**Fig. 8** Wall time for each mesh adaptation phase

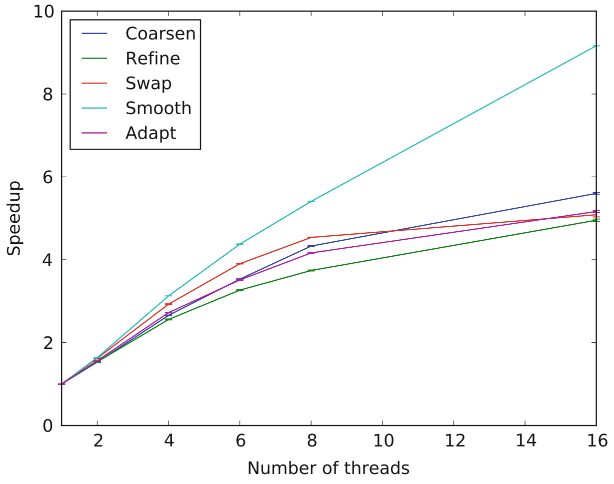


**Fig. 9** Speedup for each mesh adaptation phase

We can see that the vast majority of the elements are of very quality. The lowest quality element had a quality of 0.34, and in total only ten elements out of 26 million have a quality less than 0.4.

The benchmarks were run on a Intel(R) Xeon(R) E5-2650 CPU. The code was compiled using the Intel compiler suite, version 14.0.1 and with the compiler flags `-Ofast`. In all cases we used Intel's thread-core affinity support - specifically `scatter` which distributes the threads as evenly as possible across the entire system.

Figures 8, 9 and 10 show the wall time, speedup and efficiency of each phase of mesh adaptation. Simulations using between 1 and 8 cores are run on a single

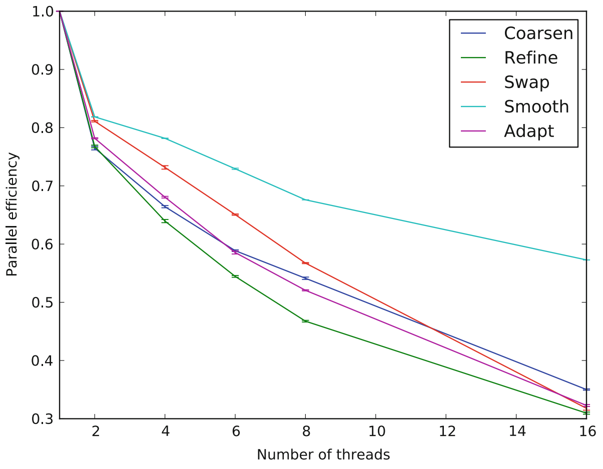


**Fig. 10** Parallel efficiency for each mesh adaptation phase

socket while the 16 core simulation runs across two CPU sockets and thereby incurring NUMA overheads. From the results we can see that all operations achieve good scaling, including for the 16 core NUMA case. The dominant factors limiting scaling are the number of synchronisations and load-imbalances. Even in the case of mesh smoothing, which involves the least data-writes, the relatively expensive optimisation kernel is only executed for patches of elements whose quality falls below a minimum quality tolerance. Indeed, the fact that mesh refinement, coarsening and refinement are comparable is very encouraging as it indicates that despite the invasive nature of the operations on these relatively complex data structures it is possible to get good intra-node scaling.

## 5 Conclusions

This paper is the first to examine the scalability of anisotropic mesh adaptivity using a thread-parallel programming model and to explore new parallel algorithmic approaches to support this model. Despite the complex data dependencies and inherent load imbalances we have shown it is possible to achieve practical levels of scaling. To achieve this two key ingredients were required. The first was to use colouring to identify maximal independent sets of tasks that would be performed concurrently. In principle this facilitates scaling up to the point that the number of elements of the independent set is equal to the number of available threads. The second important factor contributing to the scalability was the use of worklists and deferred whereby updates to the mesh are added to worklists and applied in parallel at a later phase of an adaptive sweep. This avoids the majority of serial overheads otherwise incurred with updating mesh data structures.

While the algorithms presented are for 2D anisotropic mesh adaptivity, we believe many of the algorithmic details carry over to the 3D case as the challenges associated with exposing a sufficient degree of parallelism are very similar.

**Acknowledgements** The authors would like to thank Fujitsu Laboratories of Europe Ltd. and EPSRC grants EP/I00677X/1 and EP/L000407/1 for supporting this work.

## References

1. Agouzal, A., Lipnikov, K., Vassilevski, Y.: Adaptive generation of quasi-optimal tetrahedral meshes. East West J. Numer. Math. **7**(4), 223–244 (1999)
2. Alauzet, F.: Size gradation control of anisotropic meshes. Finite Elem. Anal. Des. **46**(1), 181–202 (2010)
3. Alauzet, F., Li, X., Seol, E.S., Shephard, M.S.: Parallel anisotropic 3D mesh adaptation by mesh modification. Eng. Comput. **21**(3), 247–258 (2006)
4. Blelloch, G.E.: Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University (1990)
5. Buscaglia, G.C., Dari, E.A.: Anisotropic mesh optimization and its application in adaptivity. Int. J. Numer. Methods Eng. **40**(22), 4119–4136 (1997)
6. Chen, L., Sun, P., Xu, J.: Optimal anisotropic meshes for minimizing interpolation errors in $L^p$-norm. Math. Comput. **76**(257), 179–204 (2007)
7. Coupez, T., Digonnet, H., Ducloux, R.: Parallel meshing and remeshing. Appl. Math. Model. **25**(2), 153–175 (2000)
8. De Cougny, H., Shephard, M.S.: Parallel refinement and coarsening of tetrahedral meshes. Int. J. Numer. Methods Eng. **46**(7), 1101–1125 (1999)
9. Dongarra, J.: What you can expect from exascale computing. In: International Supercomputing Conference (ISC'11), Hamburg (2011)
10. Field, D.A.: Laplacian smoothing and Delaunay triangulations. Commun. Appl. Numer. Methods **4**, 709—712 (1988)
11. Freitag, L., Ollivier-Gooch, C.: A comparison of tetrahedral mesh improvement techniques. In: Fifth International Meshing Roundtable (1996)
12. Freitag, L., Jones, M., Plassmann, P.: An efficient parallel algorithm for mesh smoothing. In: Proceedings of the 4th International Meshing Roundtable, Sandia National Laboratories, Citeseer, pp. 47–58 (1995)
13. Freitag, L.F., Jones, M.T., Plassmann, P.E.: The scalability of mesh improvement algorithms. In: Improvement Algorithms. IMA Volumes in Mathematics and Its Applications, pp. 185–212. Springer, New York (1998)
14. Frey, P.J., Alauzet, F.: Anisotropic mesh adaptation for cfd computations. Comput. Methods Appl. Mech. Eng. **194**(48), 5068–5082 (2005)
15. Gebremedhin, A.H., Manne, F.: Scalable parallel graph coloring algorithms. Concurrency Pract. Experience **12**(12),1131–1146 (2000)
16. Gorman, G., Southern, J., Farrell, P., Piggott, M., Rokos, G., Kelly, P.: Hybrid OpenMP/MPI anisotropic mesh smoothing. In: Proceedings of the International Conference on Computational Science. Procedia Computer Science, vol. 9, pp. 1513–1522 (2012)
17. Li, X., Shephard, M., Beall, M.: 3D anisotropic mesh adaptation by mesh modification. Comput. Methods Appl. Mech. Eng. **194**(48-49), 4915–4950 (2005)
18. Pain, C.C., Umpleby, A.P., de Oliveira, C.R.E., Goddard, A.J.H.: Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations. Comput. Methods Appl. Mech. Eng. **190**(29-30), 3771–3796 (2001)

19. Rabenseifner, R., Hager G, Jost, G.: Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes. In: 17th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, 2009, pp. 427–436. IEEE (2009)
20. Rokos, G., Gorman, G., Kelly, P.H.J.: Accelerating anisotropic mesh adaptivity on nVIDIA's CUDA using texture interpolation. In: Proceedings of the 17th International Conference on Parallel Processing - Volume Part II, Euro-Par'11, pp. 387–398. Springer, Berlin (2011). http://dl.acm.org/citation.cfm?id=2033408.2033453
21. Tam, A., Ait-Ali-Yahia, D., Robichaud, M., Moore, M., Kozel, V., Habashi, W.: Anisotropic mesh adaptation for 3D flows on structured and unstructured grids. Comput. Methods Appl. Mech. Eng. **189**(4), 1205–1230 (2000)
22. Vasilevskii, Y., Lipnikov, K.: An adaptive algorithm for quasioptimal mesh generation. Comput. Math. Math. Phys. **39**(9), 1468–1486 (1999)

# Immersed NURBS for CFD Applications

**Jeremy Veysset, Ghina Jannoun, Thierry Coupez, and Elie Hachem**

**Abstract** We present a new immersed method for Computational Fluid Dynamics applications. It is based on the use of Non Uniform Rational B-Splines (NURBS). The distance function to an immersed solid is computed directly from its Computer Aided Design (CAD) description. This allows to bypass the generation of surface meshes and to obtain accurate levelset functions for complex geometries. Combined with a metric based anisotropic mesh adaptation and stabilized Finite Elements Method (FEM), it allows a novel, efficient and flexible approach to deal with a wide range of fluid structure interaction problems. The metric field is computed directly at the node of the mesh using the length distribution tensor and an edge based error analysis. Several 2D and 3D numerical examples will demonstrate the applicability of the proposed method.

## 1 Introduction

Immersed methods for Fluid Structure Interaction (FSI) are gaining popularity in many scientific and engineering applications. Different approaches can be found such as the embedded boundary method [1], the immersed boundary method [2], the fictitious domain [3], the immersed volume method [4–7] and the cartesian method [8]. All these methods are attractive because they simplify a number of issues in Fluid-Structure applications such as meshing the fluid domain, using a fully Eulerian algorithm, problems involving large structural motion and deformation [9] or topological changes [10].

However they use non-body fitted grids which require special interface treatments. Indeed recent developments are focusing on issues related to the immersion of a surface mesh for complex 3D geometries [7], the detection and the intersection algorithms for the interface and finally the transmission of boundary conditions

J. Veysset • G. Jannoun • T. Coupez • E. Hachem (✉)
MINES ParisTech, Center for Materials Forming (CEMEF), UMR CNRS 7635, BP 207, 06904 Sophia-Antipolis, France
e-mail: jeremy.veysset@mines-paristech.fr; ghina.jannoun@mines-paristech.fr; thierry.coupez@mines-paristech.fr; elie.hachem@mines-paristech.fr

S. Perotto, L. Formaggia (eds.), *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, SEMA SIMAI Springer Series 5, DOI 10.1007/978-3-319-06053-8_7

between the solid and the fluid regions. In particular these methods appear to be limited by the quality and the accuracy of the surface mesh description of a given immersed solid.

In this work, we present a new immersion technique that simplifies and bypasses the generation of a surface mesh. It is based on the use of Non Uniform Rational B-Splines (NURBS) curves or surfaces, representing simple or complex geometries. We compute the distance function from any point in the fluid mesh to these NURBS, thus representing the immersed solid by the zero iso-value of this function. The computation of the distance mainly relies on patching the NURBS functions [11] and using a Newton method [12]. Although, many methods and techniques have been already developed to compute the distance to NURBS functions, none of them has been used to compute level-set functions for immersed objects needed to solve FSI problems. Therefore instead of relying on the resolution of the surface mesh, the proposed method uses directly the Computer Aided Design (CAD) definition which keeps the quality of its analytical description. In practice, it eliminates the cost of the surface mesh generation step and reduces the complexity to set up a Fluid-Structure application.

Combined with anisotropic mesh adaptation, it provides an attractive immersed framework. Therefore, for the mesh adaptation, we retain the use of a metric constructed directly at the nodes of the mesh without any direct information from the elements, neither considering any underlying interpolation [13–15]. It is performed by introducing a statistical concept: the length distribution function. First, we use a second order tensor to approximate the distribution of lengths defined by gathering the edges at the node. Then we compute the error along and in the direction of each edge. Finally we extend the approach to deal with multicomponent fields (tensors, vectors, scalars). It uses a single metric to account for different fields such as the levelset function of the immersed solid and all components of the velocity field. Note also that the proposed algorithm is implemented in the context of adaptive meshing under the constraint of a fixed number of nodes. With such an advantage, we can provide a very useful tool for practical FSI problems and avoid a drastic increase in the number of nodes. The paper is structured as follows. Section 2 presents the details of the new immersion technique. Section 3 describes the used error estimator for anisotropic mesh adaptation. In Sect. 4 several numerical examples are used to highlight the capability of the approach. Finally conclusions and perspectives are given in Section 5.

## 2 NURBS Immersion

Let us first recall some notations and definitions of the NURBS functions. All the steps that constitute the computation of the distance function to an immersed solid defined by NURBS functions will be outlined. First, we highlight the relation between the computation of the distance function and the geometrical problem. Based on the principle of an elimination criterion, we obtain the needed initial guess for the Newton resolution. Finally a simple algorithm is used to sign the obtained distance function, positive in the solid domain and negative outside.

## 2.1 Definition of NURBS Functions

NURBS or Non-Uniform Rational B-Spline functions are piecewise-polynomial parametric functions. They were introduced in the 1950s [16, 17] in the industrial engineering field to represent complicated geometries like ship hulls and aircraft exterior surfaces. They are now widely implicated in the CAD field and used in many designing softwares (CATIA, Pro Engineer, SolidWorks…). With such mathematical functions, it is possible to represent any geometry with different levels of complexity. Their main advantage is that they can be locally modified by just moving control points without affecting the rest of the geometry. Figure 1 shows an example of a NURBS curve with the corresponding control points and knots. The definition of a NURBS curve c is as follows:

$$c(u) = \frac{\sum_{i=1}^{n} N_{i,p}(u)\omega_i P_i}{\sum_{i=1}^{n} N_{i,p}(u)\omega_i} \tag{1}$$

where $p$ is the degree of the curve, $N_{i,p}$ the basis functions, $P_i$ the control points, $n$ the number of control points, $\omega_i$ the weights and $u$ the parameter taking its values in the knot vector $U$. The knot vector $U$ has $n + p + 1$ knots. The first and last knots have multiplicity $p + 1$ ($U = \{\underbrace{u_0, \ldots, u_0}_{p+1}, u_1, \ldots, u_{n-1}, \underbrace{u_n, \ldots, u_n}_{p+1}\}$). The basis functions are defined by the Cox-De Boor recursion formula [18, 19]:

$$N_{i,0}(u) = \begin{cases} 1 \; if \; u_i \leq u < u_{i+1} \\ 0 \; otherwise \end{cases} \tag{2}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u), \; with \; p \in \mathbb{N}^*. \tag{3}$$

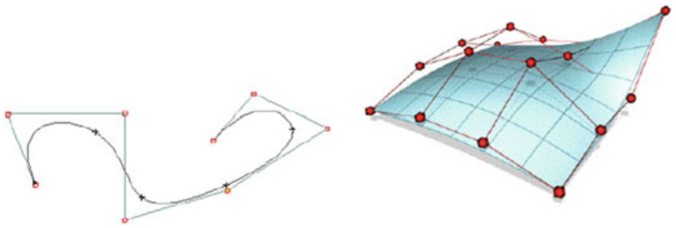


**Fig. 1** Example of a NURBS curve and a NURBS surface, their control points and knots

Following the definition given by (1), a NURBS surface is defined as follows:

$$s(u,v) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} N_{i,p}(u)N_{j,q}(v)\omega_{ij}P_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n} N_{i,p}(u)N_{j,q}(v)\omega_{ij}} \tag{4}$$

where $p$ and $q$ are the polynomial degrees in the $u$ and $v$ directions, $N_{i,p}$ and $N_{j,q}$ the basis functions in the $u$ and $v$ directions, $P_{ij}$ the control points, $\omega_{ij}$ the weights and $u$ and $v$ the parameters taking their values in the $U$ and $V$ knot vectors. The latters are constructed in the same way as mentioned previously in the NURBS curve definition.

## 2.2 *The Closest Point Problem*

The objective is to compute the level-set of the immersed objects involved in the simulations directly from their CAD definition, i.e. their CAD files. Indeed, in these files, each object is commonly characterized by NURBS curves or surfaces. Let $\Omega$, $\Omega_f$, $\Omega_s$ and $\Gamma$ represent respectively the whole domain, the fluid domain, the solid domain and the interface verifying:

$$\begin{cases} \Omega_f \bigcup \Omega_s = \Omega \\ \Omega_f \bigcap \Omega_s = \Gamma \end{cases} . \tag{5}$$

Then for each node $X$ of the computational domain $\Omega$, the level-set function $\alpha$ which is the signed distance from the interface reads:

$$\alpha(X) \begin{cases} > 0 \text{ if } X \in \Omega_s \\ = 0 \text{ if } X \in \Gamma \\ < 0 \text{ if } X \in \Omega_f \end{cases} . \tag{6}$$

The immersed solid is implicitly defined by the zero iso-value of this function $\alpha$. In what follows, we describe the algorithm to compute the minimum distance between the nodes of the computational mesh and the surface of the immersed object. This can be achieved by solving the closest point problem, which can be seen as a root finding problem [20]. In fact, if we consider a point $P$ and a NURBS curve $c$, the projection of the point $P$ on the curve $c$ is mathematically equivalent to finding the parameter $u^*$ such that:

$$(P - c(u^*)).c'(u^*) = 0. \tag{7}$$

This kind of problem can be solved by using a Newton method. It requires a good starting value in order to obtain fast and accurate results. Different approaches are proposed in the literature. In [12], the authors make a sampling of points on the curve and take as an initial value the closest one. However, this method has been described as time consuming. We will adopt here a more efficient way to find a good initial value [21]. It consists first in selecting the part of the curve containing the root. Then the initial value is taken on this part of the curve and the Newton method is performed only on this part. Therefore, we subdivide the NURBS curve into rational Bezier segments as a preparation phase. We recall that a rational Bezier curve $c$ of degree $p$ is defined by:

$$c(u) = \frac{\sum_{i=0}^{p} B_{i,p}(u)\omega_i P_i}{\sum_{i=0}^{p} B_{i,p}(u)\omega_i} \tag{8}$$

where $P_i$ are the control points, $\omega_i$ the weights and $B_{i,p}$ the Bernstein polynomials defined by the following formula:

$$B_{i,p}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}. \tag{9}$$

Then we eliminate the Bezier segments that do not satisfy a certain criterion (this criterion is detailed thereafter). Finally we use a Newton-Raphson method to solve the point projection problem (7) on the remaining rational Bezier segments. Analogously, a NURBS surface can be decomposed into a set of rational Bezier surfaces. Then the same scheme is performed in order to find the minimum distance relatively to the NURBS surface.

$$s(u,v) = \frac{\sum_{i=0}^{p} \sum_{j=0}^{q} B_{i,p}(u) B_{j,q}(v)\omega_{ij} P_{ij}}{\sum_{i=0}^{p} \sum_{j=0}^{q} B_{i,p}(u) B_{j,q}(v)\omega_{ij}} \tag{10}$$

where $s$ is the rational Bezier surface, $p$ and $q$ the degrees of $s$, $P_{ij}$ the control points and $\omega_{ij}$ the weights.

Different alternatives are proposed in the literature. For instance, in [22] and [23], the Bezier segments are subdivided until the created control polygons become simple and convex or until a flatness condition is reached. In our case, the Bezier segments are not subdivided. In [24], the authors prefer to use a geometric criterion for the elimination of the rational Bezier curves based on computing the tangent cone of every rational Bezier curve. In [25], they introduce an algebraic function instead of subdividing the NURBS geometry. Consequently, Eq. (7) is transformed into a polynomial equation and the roots of this new equation are extracted using a Sturm method.

**Algorithm 1** Closest Extremity

```
if ∀ i ∈ [1,n] $P_1P_i.PP_1 \geq 0$ then
    $P_1$ is the closest point
else if ∀ i ∈ [1,n] $P_nP_i.PP_n \geq 0$ then
    $P_n$ is the closest point
else
    Go to Algorithm 2
end if
```

**Algorithm 2** Segment Elimination

```
if ∀ i ∈ [1,$p+1$] $P_{k,1}P_{k,i}.PP_{k,1} \geq 0$ then
    $P_{k,1}$ is the closest point and the Bezier patch $B_k$ is eliminated
else if ∀ i ∈ [1,$p+1$] $P_{k,n}P_{k,i}.PP_{k,n} \geq 0$ then
    $P_{k,n}$ is the closest point and the Bezier patch $B_k$ is eliminated
else
    Apply Newton method
end if
```

## 2.3 *Outline of the Algorithm*

Inspired by all the above described works, we present a new modified algorithm, adapted mainly from [21]. We first subdivide the curve into a set of rational Bezier segments $B_k$ as a preparation phase. Then we check if one of the extremities of the NURBS curve is the closest point (Algorithm 1).

If the closest point is not an endpoint, we eliminate all the subcurves $B_k$ whose closest point from point $P$ is an extremity of $B_k$ (Algorithm 2).

$P_{k,i}$ being the control points of the Bezier segment $B_k$. If all the subcurves $B_k$ have been suppressed, then the curve has got at least a cust and the closest point is one of these custs (point of multiplicity equal to $p$, $p$ being the degree of the curve). Thus we compute the distance for all the singular points and check which one is the closest. Otherwise we look for the closest point with a Newton method on the remaining sub segments.

Analogously, we transpose the algorithm to compute the closest distance between a point and a NURBS surface. We first subdivide the surface into a set of rational Bezier patches (i.e. surfaces) $B_k$. Then, as for NURBS curves, we check if one of the corners of the NURBS surface is the closest point (Algorithm 3). If none of the corners of the NURBS surface is selected as the closest point of the query point $P$, we eliminate the rational Bezier patches $B_k$ whose closest point from point $P$ is a corner of $B_k$ (Algorithm 4). If all the sub patches $B_k$ have been suppressed, then the surface has got at least a cust and the closest point is one of these custs (point of multiplicity equal to $p$ and $q$, $p$ and $q$ being the degrees of the curve respectively in the $u$ and $v$ directions). Thus we compute the distance for all the singular points and check which one is the closest. Otherwise we look for the closest point using a Newton method on the remaining sub-patches.

**Algorithm 3** Closest Corner

```
if ∀ i ∈ [1,m] and ∀ j ∈ [1,n] $P_{11}P_{ij}.PP_{11} \geq 0$ then
    $P_{11}$ is the closest point.
else if ∀ i ∈ [1,m] and ∀ j ∈ [1,n] $P_{m1}P_{ij}.PP_{m1} \geq 0$ then
    $P_{11}$ is the closest point.
else if ∀ i ∈ [1,m] and ∀ j ∈ [1,n] $P_{1n}P_{ij}.PP_{1n} \geq 0$ then
    $P_{11}$ is the closest point.
else if ∀ i ∈ [1,m] and ∀ j ∈ [1,n] $P_{mn}P_{ij}.PP_{mn} \geq 0$ then
    $P_{11}$ is the closest point.
else
    Go to Algorithm 4
end if
```

**Algorithm 4** Patch Elimination

```
if ∀ i ∈ [1,m] and ∀ j ∈ [1,n] $P_{k,11}P_{k,ij}.PP_{k,11} \geq 0$ then
    $P_{k,11}$ is the closest point and the Bezier patch $B_k$ is eliminated
else if ∀ i ∈ [1,m] and ∀ j ∈ [1,n] $P_{k,m1}P_{k,ij}.PP_{k,m1} \geq 0$ then
    $P_{k,m1}$ is the closest point and the Bezier patch $B_k$ is eliminated
else if ∀ i ∈ [1,m] and ∀ j ∈ [1,n] $P_{k,1n}P_{k,ij}.PP_{k,1n} \geq 0$ then
    $P_{k,1n}$ is the closest point and the Bezier patch $B_k$ is eliminated
else if ∀ i ∈ [1,m] and ∀ j ∈ [1,n] $P_{k,mn}P_{k,ij}.PP_{k,mn} \geq 0$ then
    $P_{k,mn}$ is the closest point and the Bezier patch $B_k$ is eliminated
else
    Apply Newton method
end if
```

## 2.4 *The Newton Method Resolution*

The last part of the algorithm consists in solving, using the Newton method, the point inversion problem (7) on the selected segments or patches of the NURBS function. Since the corners of the NURBS function have been treated in the previous subsection, the distance between a point and a NURBS function can now simply be expressed as an orthogonal point projection problem [c.f. Eq. (7)]. The segment constituted by the query point and the closest point on the curve is orthogonal to the derivative of the curve at this closest point. From the Taylor expansion of Eq. (7), we can state that the parameter of the curve in the Newton algorithm is computed as follows:

$$u_{i+1} = u_i - \frac{(c(u_i) - P).c'(u_i)}{(c(u_i - P).c''(u_i) + \|c'(u_i)\|^2}. \tag{11}$$

The algorithm is performed as far as the parameter value does not change significantly or until Eq. (7) is satisfied under a given precision. Analogously, the problem statement for finding the distance between a point and a NURBS surface is the following, find the parameters $u$ and $v$ such that:

$$\begin{cases} a(u, v) = (s(u, v) - P).s_u(u, v) = 0 \\ b(u, v) = (s(u, v) - P).s_v(u, v) = 0 \end{cases} \tag{12}$$

where $s_u$ and $s_v$ are the partial derivatives respectively in the $u$ and $v$ directions of the NURBS surface s. The problem is transformed by solving iteratively the following system:

$$\begin{bmatrix} a_u(u_i, v_i) a_v(u_i, v_i) \\ b_u(u_i, v_i) b_v(u_i, v_i) \end{bmatrix} \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = \begin{bmatrix} -a(u_i, v_i) \\ -b(u_i, v_i) \end{bmatrix} \tag{13}$$

where $a_u$, $a_v$, $b_u$ and $b_v$ are the partial derivatives respectively in the $u$ and $v$ directions of $a$ and $b$. Replacing (12) in (13) gives:

$$J_i \cdot \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = \begin{bmatrix} -(s(u_i, v_i) - P).s_u(u_i, v_i) \\ -(s(u_i, v_i) - P).s_v(u_i, v_i) \end{bmatrix} \tag{14}$$

$$\text{with } J_i = \begin{bmatrix} \|s_u(u_i, v_i)\|^2 + (s(u_i, v_i) - P).s_{uu}(u_i, v_i) s_u(u_i, v_i).s_v(u_i, v_i) \\ \quad +(s(u_i, v_i) - P).s_{uv}(u_i, v_i) \\ s_u(u_i, v_i).s_v(u_i, v_i) + (s(u_i, v_i) - P).s_{vu}(u_i, v_i) \|s_v(u_i, v_i)\|^2 \\ \quad +(s(u_i, v_i) - P).s_{vv}(u_i, v_i) \end{bmatrix}. \tag{15}$$

Finally the parameters are computed by the following equation:

$$\begin{bmatrix} u_{i+1} \\ v_{i+1} \end{bmatrix} = \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} + \begin{bmatrix} u_i \\ v_i \end{bmatrix}. \tag{16}$$

The method is performed iteratively until the $u$ and $v$ parameters do not change significantly or both equations in (12) are satisfied under a given precision.

## 2.5 Computing the Sign of the Distance

Now that the distance has been obtained with the detailed algorithm, we need to sign it in order to check whether the point lies inside or outside the object. If the point is outside the object, then the distance will take a negative sign and vice versa. We propose two methods for signing the distance. The first one consists in defining a point $O$ lying inside the object and computing the scalar product $\overrightarrow{P_pP}.\overrightarrow{P_pO}$, $P$ being the query point and $P_p$ the closest point of $P$ on the object boundary (Fig. 2).

If the sign of the obtained scalar product is negative, then it means that the point $P$ is outside of the object and the distance takes a negative sign. This method is efficient and easy to implement but its main drawback lies in the fact that it works only for convex objects. The second method is more generic and works for any type of objects. It consists in computing the number of intersections between the edge constituted by the query point $P$ and the inside point $O$ and the object's boundary

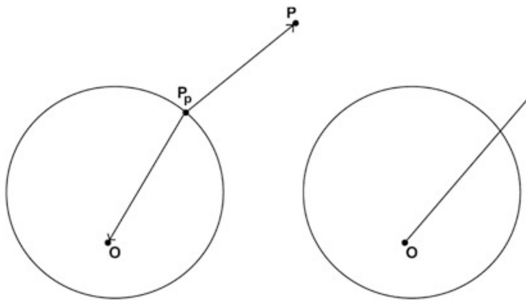


**Fig. 2** Scalar product signing method (*left*) and intersection signing method (*right*)

(Fig. 2). If the number of intersections is odd, then the distance takes a negative sign. The outline of the new implemented algorithm takes finally the following form:

1. The NURBS curve (respectively surface) is subdivided into rational Bezier segments (respectively patches).
2. Then we check if one of the corner of the NURBS function is the closest point.
3. If it is the case, go to step 6.
4. Eliminate the rational Bezier segments (respectively patches) that do not contain the closest point.
5. Compute the closest point with a Newton method on the remaining segment (respectively patch).
6. Sign the distance.

## 3 Construction of an Anisotropic Mesh

In this section, we recall important features of the anisotropic meshing approach relying on the length distribution tensor approach and the associated edge based error analysis as developed in [13].

### 3.1 Edge Based Error Estimation

We consider $u \in \mathscr{C}^2(\Omega) = \mathscr{V}$ and $\mathscr{V}_h$ a simple $P^1$ finite element approximation space:

$$\mathscr{V}_h = \left\{ w_h \in \mathscr{C}^0(\Omega), w_h|_K \in P^1(K), K \in \mathscr{K} \right\}$$

where $\Omega = \bigcup_{K \in \mathscr{K}} K$ and $K$ is a simplex (segment, triangle, tetrahedron, ...).

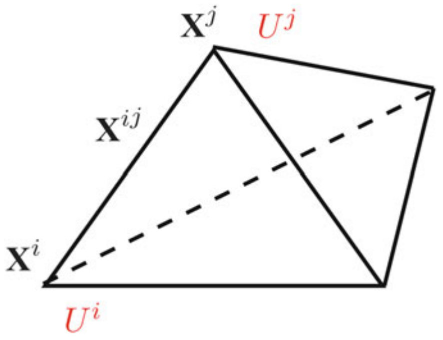


**Fig. 3** Length $\mathbf{X}^{\mathbf{ij}}$ of the edge joining nodes $i$ and $j$

We define $\mathbf{X} = \left\{\mathbf{X}^i \in \mathbb{R}^d, \, i = 1, \cdots, N\right\}$ as the set of nodes of the mesh and we denote by $U^i$ the nodal value of $u$ at $\mathbf{X}^i$ and we let $\Pi_h$ be the Lagrange interpolation operator from $\mathscr{V}$ to $\mathscr{V}_h$ such that:

$$\Pi_h u(\mathbf{X}^i) = u(\mathbf{X}^i) = U^i \, , \, \forall i = 1, \cdots, N.$$

As shown in Fig. 3, we define the set of nodes connected to node $i$ by $\Gamma(i) = \left\{j \, , \exists^i K \in \mathscr{K} \, , \mathbf{X}^i, \mathbf{X}^j \text{ are nodes of } K\right\}$.

By introducing the following notation: $\mathbf{X}^{ij} = \mathbf{X}^j - \mathbf{X}^i$ and using the analysis carried in [13], we can set the following results:

$$\nabla u_h \cdot \mathbf{X}^{ij} = U^{ij}, \tag{17}$$

$$|| \underbrace{\nabla u_h \cdot \mathbf{X}^{ij}}_{U^{ij}} - \nabla u(X^i) \cdot \mathbf{X}^{ij}|| \leq \max_{Y \in [X^i, X^j]} |\mathbb{H}(u)(Y)\mathbf{X}^{ij} \cdot \mathbf{X}^{ij}| \, , \tag{18}$$

where $\mathbb{H}(u) = \nabla^{(2)} u$ is the associated Hessian of $u$. Recall that taking $u \in \mathscr{C}^2(\Omega)$ we obtain $\nabla u \in \mathscr{C}^1(\Omega)$.

Applying the interpolation operator on $\nabla u$ and using (17) we obtain a definition of the projected second derivative of $u$ in terms of only the values of the gradient at the extremities of the edge:

$$\nabla g_h \mathbf{X}^{\mathbf{ij}} \cdot \mathbf{X}^{\mathbf{ij}} = g^{ij} \cdot \mathbf{X}^{\mathbf{ij}} \tag{19}$$

where $\nabla g_h = \Pi_h \nabla u$, $g^i = \nabla u(\mathbf{X}^\mathbf{i})$ and $g^{ij} = g^j - g^i$.

Using a mean value argument, we set that:

$$\exists y \in [x^i, x^j] | g^{ij} \cdot \mathbf{X}^{\mathbf{ij}} = \mathbb{H}(u)(Y)\mathbf{X}^{ij} \cdot \mathbf{X}^{ij} \, .$$

We use this projection as an expression of the error along the edge:

$$e_{ij} = g^{ij} \cdot \mathbf{X}^{\mathbf{ij}}. \tag{20}$$

However this equation cannot be evaluated exactly as it requires that the gradient of $u$ be known and continuous at the nodes of the mesh. For that reason, we resort to a gradient recovery procedure.

### 3.2 Gradient Recovery

Based on an optimization analysis, the author in [13] proposes a recovery gradient operator defined by:

$$G^i = (\mathbb{X}^i)^{-1} \sum_{j \in \Gamma(i)} U^{ij} \mathbf{X}^{ij} \tag{21}$$

where $\mathbb{X}^i = \frac{d}{|\Gamma(i)|} \sum_{j \in \Gamma(i)} \mathbf{X}^{ij} \otimes \mathbf{X}^{ij}$ is what we call the length distribution tensor at node $\mathbf{X}^i$. Note that this construction preserves the second order:

$$\left|\left(G^i - g^i\right) \cdot \mathbf{X}^{\mathbf{ij}}\right| \sim \left(\mathbb{H}(u)\mathbf{X}^{ij} \cdot \mathbf{X}^{ij}\right)$$

where $G^i$ is the recovery gradient at node $i$ [given by (21)] and $g^i$ being the exact value of the gradient at node $i$.

The error is evaluated by substituting $G$ by $g$ in (20):

$$e_{ij} = G^{ij} \cdot \mathbf{X}^{\mathbf{ij}}.$$

### 3.3 Metric Construction from the Edge Distribution Tensor

Taking into account this error analysis, we construct the metric for the unit mesh as follows:

$$\mathbb{M}^i = \left( \frac{d}{|\Gamma(i)|} \sum_{j \in \Gamma(i)} \mathbf{X}^{ij} \otimes \mathbf{X}^{ij} \right)^{-1}.$$

For a complete justification of this result, the reader is referred to [13].

### 3.4 Error Behavior due to Varying the Edge Length

In this section, we introduce a new way to enforce the number of nodes $N$ and we propose a novel approach to compute the stretching factor without using the dimensional parameter $p$ as was proposed in [13]. First, we start by examining how the error behaves when we change the length of the edges by stretching coefficients

$$\mathscr{S} = \left\{ s_{ij} \in \mathbb{R}^{+} \, , \, i = 1, \cdots, N \, , \, j = 1, \cdots, N \, , \, \Gamma(i) \cap \Gamma(j) \neq \phi \right\}.$$

In order to obtain a new metric depending on the error analysis, one has to calculate first a new length for each edge and then to use it for rebuilding the length distribution tensor. An interesting way of linking the error variations to the changes in edge lengths is by introducing a stretching factor $s \in \mathbb{R}$ such that

$$\begin{cases} \widetilde{\mathbf{X}_{ij}} = s\mathbf{X}_{ij} \\ ||\widetilde{e_{ij}}|| = s^2 ||e_{ij}|| = s^2 ||G^{ij} \cdot \mathbf{X}_{ij}|| \end{cases} \tag{22}$$

where $\widetilde{e_{ij}}$ and $\widetilde{\mathbf{X}_{ij}}$ are the target error at edge $ij$ and its associated edge length.

Following the lines of [13] we can simply define the metric associated with $\mathscr{S}$ by:

$$\widetilde{\mathbb{M}^i} = \frac{1}{d} \left( \widetilde{\mathbb{X}^i} \right)^{-1} \tag{23}$$

where

$$\widetilde{\mathbb{X}^i} = \frac{1}{\Gamma(i)} \sum_{j \in \Gamma(i)} s_{ij}^2 \mathbf{X}^{ij} \otimes \mathbf{X}^{ij}$$

is the length distribution tensor. Let $n_{ij}$ be the number of created nodes in relation with the stretching factor $s_{ij}$ and along the edge $ij$. When scaling the edges by a factor $s_{ij}$, the error changes quadratically so that the number of created nodes (number of sub-edges as shown in Fig. 4) along the edge $ij$ is given by:

$$n_{ij} = \left( \frac{\widetilde{e_{ij}}}{e_{ij}} \right)^{\frac{1}{2}} = s_{ij}^{-1}.$$

Here $\widetilde{e_{ij}}$ denotes the induced error for edge $\widetilde{\mathbf{X^{ij}}}$.

Giving the number of nodes (or sub-edges) created along the current edge, it is possible now to build a tensor of distribution of nodes in all directions by solving

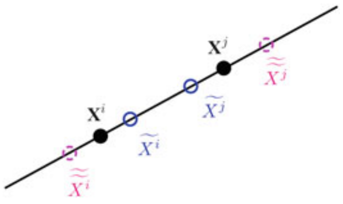


**Fig. 4** Varying the edge in its own direction

the following optimization problem:

$$\min_i \sum_{j\in\Gamma(i)} |N^i \cdot \mathbf{X}^{ij} - n_{ij}\mathbf{X}^{ij} \otimes \mathbf{X}^{ij}|^2$$

where

$$N^i = \det(N^i) = \det\left((\mathbb{X}^i)^{-1} \sum_{j\in\Gamma(i)} n_{ij}\mathbf{X}^{ij} \otimes \mathbf{X}^{ij}\right).$$

By considering the averaging process of the number of nodes distribution function, the total number of nodes in the adapted mesh is given by

$$N = \sum_i N^i.$$

Assuming a uniform totally balanced error along the edge, $\widetilde{e_{ij}} = e$ is constant, we get a direct relation between $N$ and $e$ as follows:

$$N^{ij}(e) = s_{ij}^{-1}(e) = \left(\frac{\widetilde{e_{ij}}}{e_{ij}}\right)^{+\frac{1}{2}}.$$

For a node $i$ we have

$$N^i(e) = \det\left(\left(\frac{1}{d}\right)(X^i)^{-1} \sum_{j\in\Gamma(i)} N_{ij}(e)\mathbf{X}^{ij} \otimes \mathbf{X}^{ij}\right)$$

with

$$N^i(e) = e^{\frac{2}{d}} N^i(1)$$

so that

$$N = e^{\frac{2}{d}} \sum_i N^i(1).$$

Hence, the global induced error for a given total number of nodes $N$ can be determined by:

$$e(N) = \left(\frac{N}{\sum_i N^i(1)}\right)^{-\frac{4}{d}}.$$

Therefore the corresponding stretching factors under the constraint of a fixed number of nodes $N$ are given by:

$$s_{ij} = \left(\frac{\widetilde{e_{ij}}}{e(N)}\right)^{-\frac{1}{2}}.$$

### 3.4.1 Extension to Multi-Component Field

Here we propose to construct a unique metric directly from a multi-component vector field containing, for instance, all the components of the velocity field and/or different levelset functions of the immersed solids. Consequently, we do not need to intersect several metrics but construct it using the following error vector: $\mathbf{e}_{ij} = \left\{e_{ij}^1, e_{ij}^2, \cdots, e_{ij}^n\right\}$.

Let us introduce $u = \{u_1, u_2, \cdots, u_n\}$,

$$\mathbb{Z} = \mathscr{V} \times \mathscr{V} \times \cdots \times \mathscr{V}$$

and

$$\mathbb{Z}_h = \mathscr{V}_h \times \mathscr{V}_h \times \cdots \times \mathscr{V}_h.$$

In the view of constructing a unique metric, we choose to apply the above theory for each component of $u$. It comes out immediately that the error is now a vector given by the following expression:

$$\overrightarrow{e_{ij}} = \left\{e_{ij}^1, e_{ij}^2, \cdots, e_{ij}^n\right\}$$

and then

$$s_{ij} = \left(\frac{||\widetilde{e_{ij}}||}{||\overrightarrow{e_{ij}}||}\right)^{-\frac{1}{2}}.$$

Here, the norm can be $L_2$, $L_1$ or $L_\infty$. In the following numerical experiments, we used the $L^2$ case to compute the error.

## 3.5 *Application to the Velocity Field and to the Levelset Function*

Let $v_h(X^i) = V^i \in \mathbb{R}^d$, $d = 2, 3$ the finite element solution of the Navier-Stokes equations. Introduce the vector field $\mathscr{Y} = \left(\frac{v}{|v|}, |v|, \alpha\right)$ made of $d + 1$ components vector fields. Recall that $\alpha$ is the level set function used to localize an immersed body. We obtain then for every node $i$,

$$\Pi_h \mathscr{Y}(X^i) = \left\{ \frac{V^i}{|V^i|}, |V^i|, \alpha \right\} = \mathscr{Y}^i.$$

Obviously the case $|v| = 0$ must be accounted for by using $\frac{V^i}{\max(|v^i|, \varepsilon)}$ with $\varepsilon \approx 10^{-6}$ chosen as a small value so that $\mathscr{Y}_k^i = 0$ when $|v^i| = 0$.

Using the vector $\mathscr{Y}^i$, the adaptivity will now take into account, using one unique metric, the variations in the velocity directions, the velocity norm and the levelset functions. Indeed, the adaptivity will focus mainly on the change of direction rather than the intensity of the velocity. Consequently, and as presented by the numerical results in the following section, even the small vortices developed by the solution will be very well captured. What is even more interesting is the capability of the method to automatically detect the boundary layers at the fluid-solid interfaces due to the anisotropically adapted mesh exhibiting highly stretched elements. Finally, we recall that we use a mesh technique (MTC) based on the local modification and the conformity control through the theorem for minimal volume preserving. This was introduced in [26] and extended to anisotropic mesh adaptation in [13, 27].

## 4 Applications

The performance of the new NURBS immersed method will be assessed using several 2D and 3D examples. First we show that combining the new immersed method with anisotropic mesh adaptation can lead to a novel, efficient and flexible immersed framework able to handle simple and very complex geometries. Then, we combine it with flow solvers based on a stabilized three-fields velocity-pressure-stress finite element formulation, designed for the computation of rigid bodies in an incompressible Navier-Stokes flow at high Reynolds number. Indeed, this formulation consists of considering the whole domain as a single one, meshed by a single grid, and solved with an Eulerian framework. Continuity at the fluid-solid interface is then obtained naturally and there is no need to enforce it. Then, it imposes the use of an appropriate constitutive equation describing both the fluid and the solid domain. For instance, the presence of the solid is taken into account as an extra stress in the Navier-Stokes equation [6, 28]. The results show that the method is very efficient and robust in particular at high Reynolds numbers using anisotropic meshes with highly stretched elements.

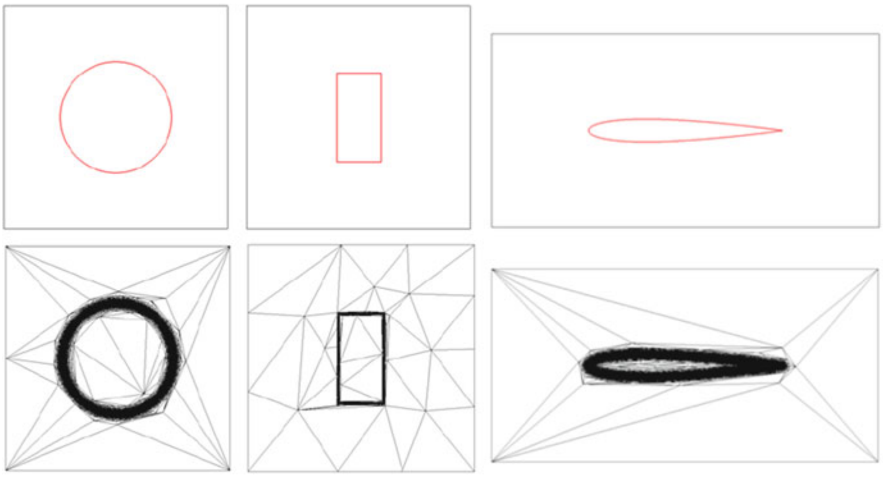


**Fig. 5** 2D applications of the immersed NURBS method: level-set zero iso-value (*top*); adapted meshes (*bottom*)

## 4.1 Immersed 2D and 3D Simple Geometries

First we test the method by immersing simple objects. Indeed, the distance function for the circle and the rectangle can be obtained easily using analytical functions. Therefore, they will be used first to test the implemented algorithm, in particular in the presence of curvatures, sharp angles and singularity. We immerse the CAD descriptions of a circle, a rectangle and a NACA profile in 2D, a sphere and cube in 3D. We use the computed levelset functions as the mesh criterion.

Figure 5 presents the zero isovalues of the immersed objects inside the computational domain. As expected, it reflects the sharp capture of the geometries and the right orientation and deformation of the mesh elements (longest edges parallel to the boundary). This yields a great reduction of the number of triangles and consequently a reduction in the computational costs. These first results show that the method works properly and that the obtained results are accurate and respect well the geometry of the objects.

The extension of the method to deal with 3D objects described this time by NURBS surfaces is tested on a sphere and a cube immersed inside a larger domain. Figure 6 shows the zero-isovalues of the computed levelset functions and several cut in the planes highlighting the obtained meshes at the interfaces. Once again the results prove that the implemented method works well and shows that combining the new immersed method with anisotropic mesh adaptation lead to a very practical tool for immersed methods.

Taking a closer look at the mesh near the interfaces, we can detect the good orientation of the elements with the stretching in the right direction. This demonstrates

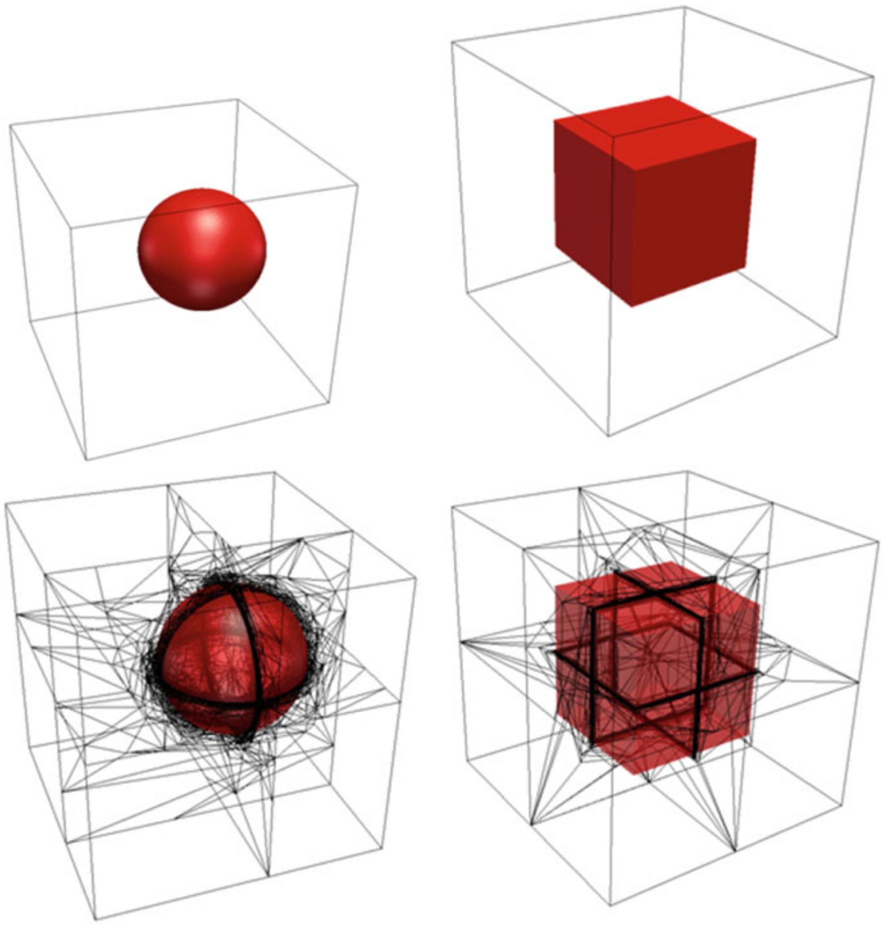


**Fig. 6** 3D applications of the immersed NURBS method: level-set zero iso-value (*top*); adapted meshes (*bottom*)

the ability of the algorithm to work under the constraint of a fixed number of nodes and to effectively control the elements sizes, orientations and locations.

## *4.2 Immersed 3D Complex Geometries*

In this section, we test the immersed method on complex geometries: a ship hull and a large airship. Two difficulties must be underlined. The first is clearly the edge of the ship hull while the second is the presence of the hole all along the airship. Note also that both geometries are described this time by several NURBS surfaces.

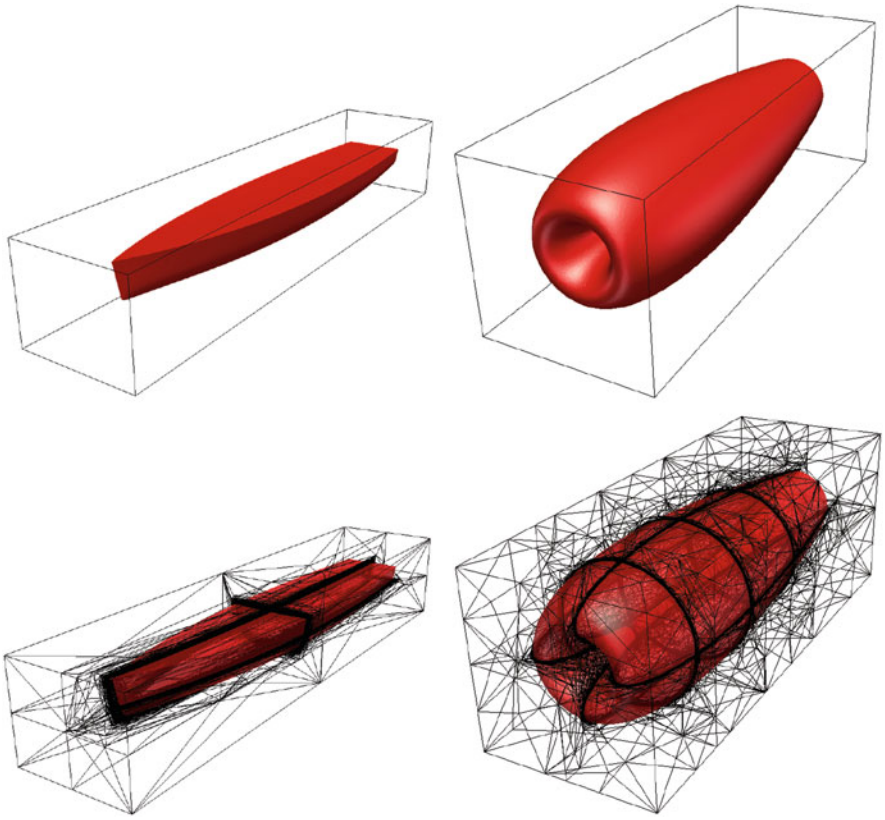


**Fig. 7** 3D applications of the immersed NURBS method: level-set zero iso-value (*top*); adapted meshes (*bottom*)

The same algorithm is applied iteratively on both geometries: (1) distance function computation using NURBS, (2) sign determination and (3) anisotropic mesh adaptation. The obtained results are shown in Fig. 7. As expected, the algorithm progressively detects and refines the mesh at the interfaces leading to a well respected shape in terms of curvature, angles, etc. All the small details in the given geometries are captured accurately. These observations reflect the ability of the anisotropic mesh adaptation algorithm to automatically adjust the shape and orientation of the elements while optimizing their numbers. For instance, the singularity of these edges could not be recovered without an accurate distance computation and anisotropic refined mesh adaptation.

It is worth mentioning that both the use of NURBS and anisotropic mesh adaptation are complementary. As mentioned previously, immersed objects are usually surface meshes. Therefore the anisotropic mesh adaptation can be limited by the facetization of the object, i.e. the accuracy of the surface mesh file. By immersing

**Table 1** Computational time in seconds of the distance calculation of the ship hull immersed with an IGES file, a STL file and the transportation method

| n cores | NURBS | Surface mesh | NURBS + transport |
|---|---|---|---|
| 1 | 138.10 | 13.37 | 2.72 |
| 2 | 70.92 | 6.99 | 2.23 |
| 4 | 43.14 | 3.53 | 2.12 |
| 8 | 22.30 | 2.03 | 0.70 |

NURBS objects we overcome this issue as the object geometry is kept analytical. Thus the anisotropic mesh adaptation reaches its full potential.

We present in Table 1 the computational time taken to compute the distance function of the ship hull. We compare several techniques and we use different number of cores (1, 2, 4 and 8) also to test the implementation in a parallel environment. First, we notice that the algorithm works well in parallel and shows a good scalability. Note that we did not extend further this study since it is not in the scope of this paper. Secondly, we compare the present method to the computation of the distance function obtained by immersing a surface mesh (i.e. STL file). Even though the comparison is not fair since the execution time to obtain the surface mesh is not counted and the quality of the surface mesh remains unclear, the purpose of this comparison still gives us an idea on the potential of the method and the possibilities for improvement. However, to make the comparisons fair, we immersed first the ship hull inside a smaller domain using the NURBS, and then we transport the obtained distance function on this refined mesh to the larger computational domain. In the latter case, the cost of this method referred as NURBS + Transport becomes negligible and interesting for practical CFD applications.

## 4.3 CFD Applications

The objective of this test case is to show the utility of the immersed NURBS method. Indeed, combined with flow solvers it allows to easily and accurately deal with complex fluid structure interaction problems. Therefore, we consider a turbulent flow past an immersed large scale airship. This 3D computations have been obtained using 64 2:4 GHz Opteron cores. The air movement around the airship is quite complex and interesting; i.e. it allows the study of the influence of different airfoils and their positions to optimize the aerodynamic design. A number of vortices between the objects and the surroundings can be observed due to the turbulence dissipation. All these observations are highlighted by the streamlines in Fig. 8. Moreover, we can clearly see on the vertical planes cutting through the airships that the solid region satisfies the zero velocity and, hence, the no-slip condition on the extremely refined interface is also verified. The airship slows down the air circulation on the surface and influences the main air circulation along the hole.

Note also in Fig. 9 the concentration of the resolution not only along all the boundary layers but also at the detachment and in the wake regions. This reflects

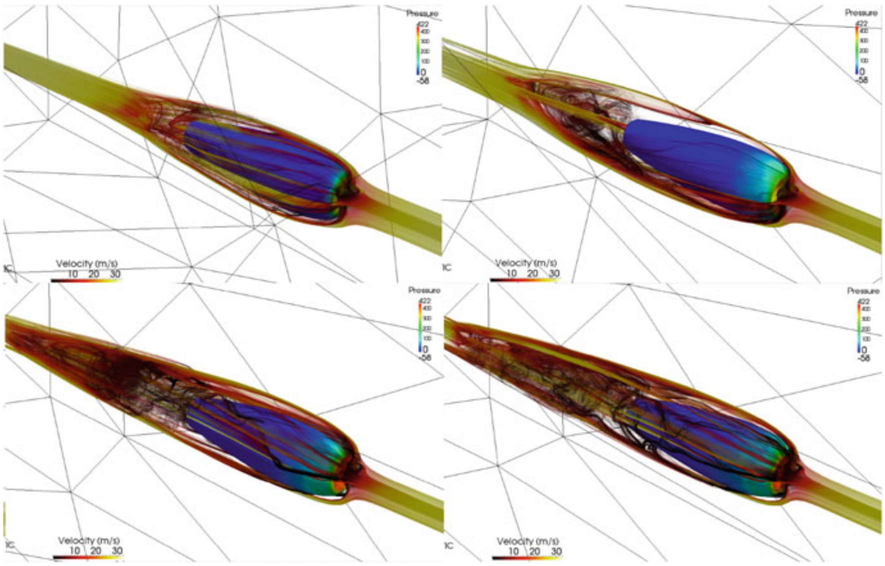


**Fig. 8** Snapshots of the streamlines around an airship described by NURBS surfaces

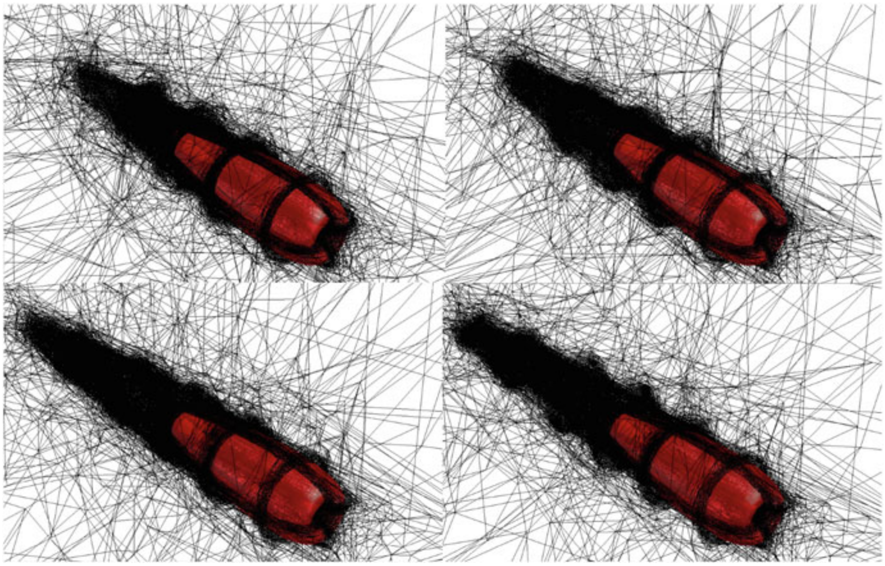


**Fig. 9** Snapshots of the adapted mesh around an airship described by NURBS surfaces

well the anisotropy of the solution caused by the discontinuity of the boundary conditions and the nature of the flow. The elements far from the immersed solid are mostly isotropic and increase in size as the velocity gradient decreases. Again, this reflects and explains why, for a controlled number of nodes, the mesh is naturally and automatically coarsened in that region with the goal of reducing the mesh size around the boundaries and in the wake regions.

## 5 Conclusions

We present a new NURBS immersed method for Computational Fluid Dynamics applications. This method is an extension of the standard Immersed Volume method and more accurate. The immersion of an object described by surface meshes is replaced by the direct use of the CAD definition keeping the quality of its analytical description. The distance computation is performed using a modified algorithm based on the decomposition of the NURBS functions in sub-curves or surfaces and a selection criterion. The Newton method is presented and used to solve the distance problem. The numerical examples show that combined with anisotropic mesh adaptation and flow solver, it leads to a novel, accurate and efficient method to deal with complex fluid structure interaction problems. The natural extension of this work is to optimize and accelerate the implemented algorithm.

**Acknowledgements** The authors gratefully acknowledge the help of the Agence Nationale de la Recherche Scientifique (ANR), France, under the project ANR-10-REALISTIC-0065.

## References

1. Kreiss, H., Petersson, A.: A second order accurate embedded boundary method for the wave equation with dirichlet data. SIAM J. Sci. Comput. **27**, 1141–1167 (2006)
2. Peskin, C.: Flow patterns around heart valves: a numerical method. J. Comput. Phys. **10**, 252–271 (1972)
3. Glowinski, R., Pan, T., Kearsley, A., Periaux, J.: Numerical simulation and optimal shape for viscous flow by a fictitious domain method. Int. J. Numer. Methods Fluids **20**, 695–711 (2005)
4. Hachem, E., Digonnet, H., Massoni, E., Coupez, T.: Immersed volume method for solving natural convection, conduction and radiation of a hat-shaped disk inside a 3d enclosure. Int. J. Numer. Methods Heat Fluid Flow **22**, 718–741 (2012)
5. Hachem, E., Kloczko, T., Digonnet, H., Coupez, T.: Stabilized finite element solution to handle complex heat and fluid flows in industrial furnace using the immersed volume method. Int. J. Numer. Methods Fluids **68**, 99–121 (2012)
6. Hachem, E., Feghali, S., Codina, R., Coupez, T.: Anisotropic adaptive meshing and monolithic variational multiscale method for fluid-structure interaction. Comput. Struct. **122**, 88–100 (2013)
7. Hachem, E., Feghali, S., Codina, R., Coupez, T.: Immersed stress method for fluid structure interaction. Int. J. Numer. Methods Eng. **94**, 805–825 (2013)

8. Johansen, H., Colella, P.: A cartesian grid embedded boundary method for Poisson's equation on irregular domains. J. Comput. Phys. **147**, 60–85 (1998)
9. Farhat, C., Rallu, A., Wang, K., Belytschko, T.: Robust and provably second-order explicit-explicit and implicit-explicit staggered time-integratorsfor highly nonlinear fluid-structure interaction problems. Int. J. Numer. Methods Eng. **84**(1), 73–107 (2010)
10. Farhat, C., Maute, K., Argrow, B., Nikbay, M.: Shape optimization methodology for reducing the sonic boom initial pressure rise. AIAA J. Aircraft **45**, 1007–1018 (2007)
11. Piegl, L., Rajab, K., Smarodzinava, V., Valavanis, K.: Point-distance computations: a knowledge-guided approach. Comput. Aid. Des. Appl. **5**(6), 855–866 (2008)
12. Piegl, L., Tiller, W.: The NURBS Book, 2nd edn. Springer, Berlin Heidelberg (1996)
13. Coupez, T.: Metric construction by length distribution tensor and edge based error for anisotropic adaptive mesing. J. Comput. Phys. **230**, 2391–2405 (2011)
14. Coupez, T., Hachem, E.: Solution of high-reynolds incompressible flow with stabilized finite element and adaptive anisotropic meshing. Comput. Methods Appl. Mech. Eng. **267**, 65–85 (2013)
15. Coupez, T., Jannoun, G., Nassif, N., Nguyen, H., Digonnet, H., Hachem, E.: Adaptive time-step with anisotropic meshing for incompressible flows. J. Comput. Phys. **241**, 195–211 (2013)
16. De Casteljau, P.: Outillages methodes calcul. Tech. rep., A. Citro n, Paris (1959)
17. Bezier, P.: Definition numerique des courbes et surfaces I. Automatisme **XI**, 625–632 (1966)
18. Cox, M.: The numerical evaluation of B-splines. Tech. rep., National Physics Laboratory DNAC4 (1971)
19. De Boor, C.: On calculating with B-splines. J. Approx. Theory **6**, 50–62 (1972)
20. Schneider, P., Eberly, D.: Geometric Tools for Computer Graphics. Morgan Kaufmann, San Francisco (2003)
21. Selimovic, I.: Improved algorithms for the projection of points on NURBS curves and surfaces. Comput. Aid. Geom. Des. **23**, 439–445 (2006)
22. Ma, Y., Hewitt, W.: Point inversion and projection for NURBS curve and surface: control polygon approach. Comput. Aid. Geom. Des. **20**, 79–99 (2003)
23. Dyllong, E., Luther, W.: Distance calculation between a point and a NURBS surface. In: Curve and Surface Design. Saint-Malo, pp. 55–62 (1999)
24. Cohen, E., Johnson, D.: Distance extrema for spline models using tangent cones. In: Proceedings of the Graphics Interface 2005 Conference, Victoria, pp. 169–175, 9–11 May 2005
25. Chen, X.: Improved algebraic algorithm on point projection for Bezier curves. In: Second International Multisymposium on Computer and Computational Sciences, pp. 158–169 (2007)
26. Coupez, T.: A mesh improvement method for 3D automatic remeshing. In: Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, pp. 615–626. Pineridge Press, Swansea (1994)
27. Gruau, C., Coupez, T.: 3D tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric. Comput. Methods Appl. Mech. Eng. **194**, 4951–4976 (2005)
28. Hachem, E., Rivaux, B., Kloczko, T., Digonnet, H., Coupez, T.: Stabilized finite element method for incompressible flows with high reynolds number. J. Comput. Phys. **229**, 8643–8665 (2010)

# Strategies for Generating Well Centered Tetrahedral Meshes on Industrial Geometries

**Sean Walton, Oubay Hassan, and Kenneth Morgan**

**Abstract** This paper outlines some recent developments in the process of generating well centered tetrahedral meshes. A well centered tetrahedron contains its circumcentre, which is a basic property required for a valid co-volume discretisation. Although most work in this area has focussed on improving meshes generated using classical techniques, in this paper we consider modification of the generation procedure itself. A simple lattice point insertion technique is introduced and the potential of the technique for generating well centered meshes is demonstrated. This is accomplished by comparing, for some complex geometries, the meshes generated with the meshes created by a standard Delaunay mesh refinement technique. Despite the simplicity of the lattice point insertion method, the comparison is found to be favourable and the method is shown to produce good well centered elements in the vicinity of the geometry.

## 1 Introduction

Co-volume algorithms, such as the marker and cell (MAC) algorithm for the solution of the Navier Stokes equations [1] or the Yee scheme for the solution of Maxwell's equations [2], exhibit a high degree of computational efficiency in terms of their low operation count and their low storage requirements. These properties, together with the simplicity of the algorithm, have made the Yee scheme a favoured computational solution technique for industrial electromagnetic simulations. However, these algorithms are highly sensitive to the quality of the mesh employed. Specifically, they require well centered meshes, i.e. meshes in which each element contains its circumcentre.

The primary goal of this work is to develop a technique which automatically generates a well centered unstructured mesh around an arbitrary geometry in three dimensions. It has been shown that it is possible for uniform meshes, and with enough optimisation, for non-uniform meshes, to obtain well centered meshes for

S. Walton • O. Hassan (✉) • K. Morgan
College of Engineering, Swansea University, Swansea SA2 8PP, Wales, UK
e-mail: s.p.walton@swansea.ac.uk

S. Perotto, L. Formaggia (eds.), *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, SEMA SIMAI Springer Series 5, DOI 10.1007/978-3-319-06053-8_8

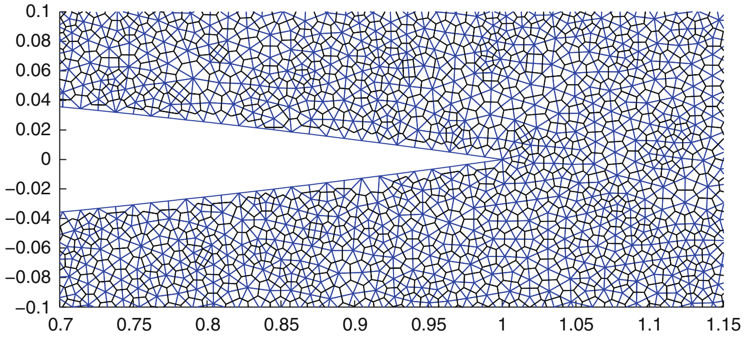


**Fig. 1** A detail of an example well centered mesh in two dimensions

general two dimensional geometries [3–5]. An example of such a mesh is shown in Fig. 1. This mesh was generated using a Delaunay refinement technique and was optimised using a modified cuckoo search [4–6]. Our work is currently focussed on attempting to obtain the equivalent level of mesh quality in three dimensions.

This paper details some recent developments aimed at addressing the problem of generating well centered meshes for industrial geometries. At present, the problem remains unsolved, but the results presented here demonstrate the recent progress that has been achieved. The paper is structured in the following way. The mesh quality required for successful implementation of co-volume algorithms is discussed. Volume mesh generation that is accomplished by extending the stitching methods that was successfully implemented in two dimensions [3] is described. Near field meshes generated by the Delaunay insertion of ideal lattice is highlighted. Previously developed mesh optimisation techniques are then briefly outlined and applied to the meshes initially generated using the lattice insertion technique. The quality of the meshes generated using the proposed technique for three representative geometries, consisting of an ONERA M6 wing, a full B60 geometry and a full F16 geometry, is illustrated. The paper concludes with a discussion and suggestions for further work.

## 1.1 *Co-volume Algorithms and Their Mesh Requirements*

In its most basic form, a classical co-volume method is staggered in time and is implemented on a pair of orthogonal staggered Cartesian meshes in space. The mutually orthogonal pair of meshes are termed the primal and the dual mesh. The resulting discretisation is second order accurate, in both space and time, on uniform meshes and has the additional advantages of preserving the energy, and maintaining the amplitude, of plane waves. This enables the approximation of electromagnetic fields near sharp edges, vertices and wire structures, without a need for excessive local mesh refinement [7].

The success of a co-volume solver is, however, particularly sensitive to the quality of the meshes employed [4]. On general meshes, second order accuracy of a co-volume scheme can only be ensured if [8]:

- The primal and dual meshes are mutually orthogonal, i.e. the edges of the primal mesh pass orthogonally through the faces of the dual mesh.
- A dual mesh node must lie inside its corresponding primal mesh element, otherwise integrals over primal mesh elements will be approximated in terms of solution values located outside the elements.
- The nodes of the dual mesh lie at the centroid of corresponding primal mesh and the dual mesh edges pass through the centroids of the corresponding primal mesh faces.

The most obvious choice, for meshes with these properties, is to employ a primal Delaunay mesh and its Voronoi dual [8]. This is advantageous because the first condition above is then met automatically. The second and third conditions are, however, not met for a general set of nodes. When all the conditions are met, the set of nodes is said to form a centroidal Voronoi tesselation (CVT). The aim of the current work is to develop new mesh generation and optimisation techniques, which will make the use of co-volume techniques feasible for complex three dimensional geometries.

In isolation, a tetrahedral element whose faces are equilateral triangles has a Voronoi node which sits at its centroid. Note, however, that such tetrahedra do not fill space. To date, only one type of tetrahedron has been found which fills space and satisfies all of the above conditions [9–11]. Each face of this tetrahedron is an isosceles triangle, with one side of length 1 and the other two, shorter, sides of length $\sqrt{3}/2$. The tiling of space which is created by these tetrahedra is illustrated in Fig. 2, where each internal node is connected to 14 neighbouring nodes in a regular lattice. Clearly, a tiling of this form will be unable to mesh the volume surrounding a general surface. In addition, in the analysis of practical problems, it is normally desirable that the local mesh size matches a user-defined target mesh spacing at each point in space. It has proved possible to obtain well centered volume meshes, for regions lying inside simple shapes, by using optimisation techniques [4, 12, 13]. However, it is yet to be shown that this can also be achieved for general geometries with specific mesh spacing functions.

Since the use of the Delaunay Voronoi dual ensures mutual orthogonality, our current work is focused on producing well centered meshes. Much of the current literature, in this field, employs traditional meshing techniques and attempts to improve the mesh through the use of a variety of mesh cosmetics procedures. Cosmetic procedures are when either the positions, connectivity of the mesh nodes are adjusted after the initial generation, in an attempt to improve quality. Although it is not technically the same as the problem of interest here, the problem of calculating a CVT of a set of nodes is particularly relevant. CVT techniques aim to minimise an energy function [14], which is related to the deviation of the primal Delaunay nodes from the mass centroid of the Voronoi cell in which they lie. Such methods, which are iterative by nature, require that the CVT energy is calculated several times and

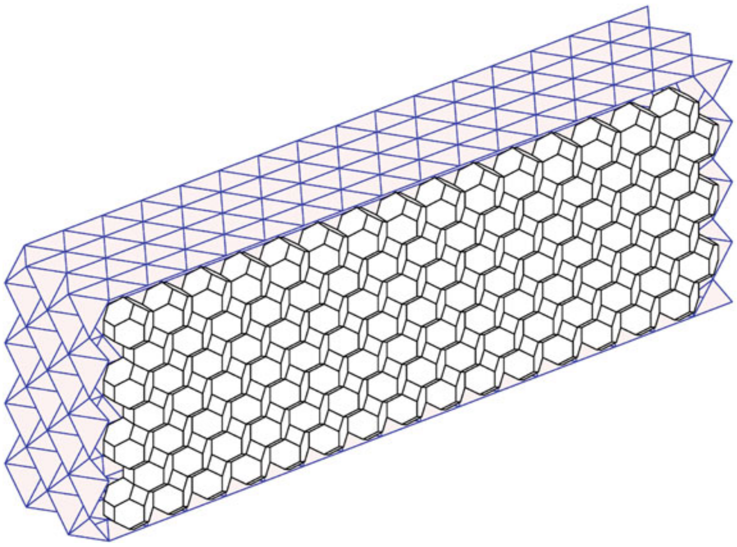


**Fig. 2** Detail of a mesh of ideal tetrahedral elements, showing the surface Delaunay faces and the internal Voronoi cells

this can be an expensive task, when it is applied to large meshes of practical interest. It is apparent that it is possible to find a CVT for a range of examples, but even state of the art techniques, using quasi-Newton based optimisation methods, require that the CVT energy be calculated hundreds of times for simple geometries [15].

During our own efforts at directly optimising meshes for this type of application we introduced the concept of reduced order mesh optimisation [4, 5]. By applying reduced order modelling techniques we were able to represent full meshes, with a large number of nodes, with much fewer degrees of freedom. The reduced number of degrees of freedom allowed the application of gradient free optimisation techniques. To date this approach has only been explored once so more work in this area is needed, to ascertain its strengths and weaknesses. However, our work still showed that a large number of iterations is required to eliminate all non well centered elements [4, 5]. For this reason, it is important to develop mesh generation techniques that aim, at the outset, to create a high quality mesh. If this can be achieved, it will, hopefully, ensure a reduction in the time spent on cosmetics.

## 2 Volume Mesh Generation Techniques

Two dimensional surface meshing is now well established. For example, using advancing front techniques, close to ideal surface meshes can be obtained for a wide range of geometries, provided, of course, that the geometry definitions are water-tight. Therefore, it will be assumed that high quality surface meshes can be obtained for the geometries of interest. However, it might be necessary, at some later stage, to consider introducing additional specific surface mesh generation techniques, if we find that existing surface meshes prevent us from obtaining the volume meshes of the desired quality.

The challenge of generating well centered tetrahedral meshes for complex geometries has previously been approached using a stitching method [3]. This method starts by employing an advancing front technique [16] to generate a local high quality triangulation close to the boundaries. This triangulation is then stitched to an ideal mesh which covers the remainder of the computational domain. The advancing front technique places nodes at ideal locations and generates the elements in layers, while advancing away from the boundary surfaces. Although this approach is effective and relatively simple in two dimensions, its extension to three dimensions is clearly dependent on the ability to efficiently create high quality triangulation in the vicinity of the geometry.

Here we consider an alternative mesh generation approach that combine the classical Delaunay refinement method with an ideal mesh lattice for the creation of high quality near field mesh.

## 2.1 Delaunay Refinement

The Delaunay refinement method [17] requires a surface mesh defining the boundary and a user-defined mesh spacing function. Initially, the surface mesh nodes are inserted into a convex hull which contains the entire computational domain. Once this is completed, the boundary triangulation of the surface is recovered, resulting in a Delaunay triangulation of these nodes. New nodes are then inserted sequentially at the centroids of existing elements that do not comply with the required mesh spacing, with a new Delaunay mesh of the nodes being produced at each stage. Since we are looping over the elements in the mesh, we immediately know the element into which the node should be inserted. Furthermore, since the insertion of new nodes takes place inside existing elements, there is no need to check for self intersection of the primal meshes faces and edges. This results in a very fast algorithm. This process of centroid insertion continues until the node density is within a user defined-factor of the local value specified by the mesh spacing function [18].

## 2.2 Lattice Point Insertion

To ensure the creation of well centered meshes, we start by considering the ideal mesh shown in Fig. 2. Here, each node in this lattice is connected to 14 neighbouring nodes, as shown in Fig. 3. Each tetrahedron is well centered when these nodes are triangulated in a Delaunay fashion. The method that is proposed for generating

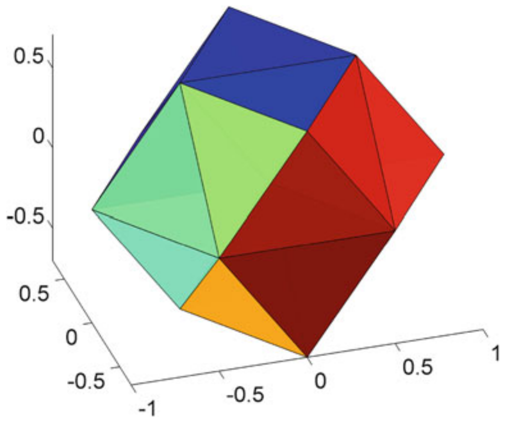


**Fig. 3** A single point from the lattice of Fig. 2, with its 14 neighbours lying either on the sphere of radius $\sqrt{3}/2$ or on the sphere of radius $1/\sqrt{2}$

volume meshes is then to use these 14 nodes as a template for locating nodes in space. This approach, which requires a surface mesh and a defined spacing function, may then be simply described by the following steps:

1. For each node on the surface:
   (a) Calculate the value $\delta$ of mesh spacing function at the node.
   (b) Generate the co-ordinates of the 14 lattice nodes described above, such that the long edges of the isosceles triangular faces are of length $\delta$.
   (c) Check each of the 14 nodes in turn; store the nodes lying inside the computational domain in a list, $\mathbf{x}_{temp}$.
2. When $\mathbf{x}_{temp}$ is generated, loop over each node currently in the volume mesh; delete any point in $\mathbf{x}_{temp}$ that lies within $0.5\delta$ of an existing point.
3. Now loop over the remaining nodes in $\mathbf{x}_{temp}$ and, for each point:
   (a) Calculate the spacing function, $\delta^i$, at the point $\mathbf{x}^i_{temp}$.
   (b) Perform a search for nodes either already in the mesh or in the list which lie within a radius $\delta^i$ of $\mathbf{x}^i_{temp}$.
   (c) Calculate the spacing, $\delta^j$, at each of the nodes found by this search.
   (d) If any point lies within $0.5max(\delta^i, \delta^j)$ of $\mathbf{x}^i_{temp}$ then delete $\mathbf{x}^i_{temp}$.
4. Insert the remaining nodes in $\mathbf{x}_{temp}$ into the mesh in a Delaunay fashion. During this insertion, a check is made to ensure that the minimum distance between a new point and the cavity into which it is inserted is greater than the local spacing. If this is not the case, the point is rejected. This step is the main difference between the lattice technique considered here and others [19].

This process is then repeated for the desired number of layers using the last set of nodes $\mathbf{x}_{temp}$ as the starting surface or until all the nodes are rejected in the final step. Connecting the nodes in a Delaunay fashion rather than specifying a connectivity is justified since it is known that, if a well centered triangulation of a set of nodes exists it is unique and it is the Delaunay triangulation [20].

At first glance, it may appear that this algorithm will be expensive, due to the nearest neighbour searching. However, many data structures have been developed for this operation and, in this implementation, a KD-tree library is employed. This enables searches for neighbours within a given range to be performed in $O(\log N)$ operations, where $N$ denotes the total number of nodes [21]. Other costs associated with point insertion and background space calculation can be reduced by employing careful data structure and storage. For instance when building the mesh we store the elements connected to each node, this allows a good first guess for which element the new lattice nodes originating from that node lie inside.

## 2.3 Examples

It is recognised that meshing techniques can be highly sensitive to the nature of the geometry under consideration. Several three dimensional examples, are considered to illustrate the potential of the proposed methodology. The examples that have been selected are of particular interest to the aerospace community. Meshes are generated using both lattice insertion and the original Delaunay refinement. No mesh cosmetic steps are performed on the meshes presented in this section and the target mesh spacing function is the same for both generation techniques. Since the extension of the stitching method requires the generation of near field elements that exhibit the required qualities for co-volume methods, at this stage, only elements that are attached to boundary faces and boundary nodes are considered in the analysis. In the following, the term bad element is meant to refer to an element whose circumcentre lies outside the element.

The first example is the problem of generating a volume mesh, for the ONERA M6 wing. The surface mesh, shown in Fig. 4 consists of 16,178 triangular elements and 8,091 nodes. Table 1 details the mesh statistics for the two meshes generated for

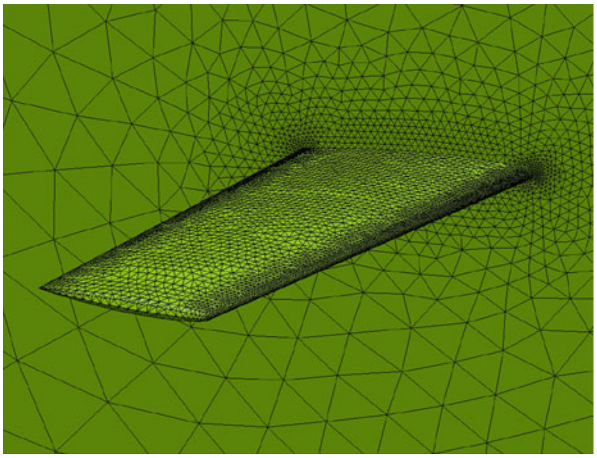


**Fig. 4** ONERA M6 wing: detail of the surface mesh

**Table 1** Comparison of mesh quality measures for the ONERA M6 wing

| | Delaunay refinement | Lattice point insertion |
|---|---|---|
| No. nodes | 93,066 | 79,591 |
| No. elements | 572,711 | 485,324 |
| No. bad elements touching a surface | 24,637 | 15,157 |
| No. (%) surface elements connected to a bad element | 2,309 (14 %) | 318 (2 %) |

this geometry. It is clear that a considerable reduction in the number of bad elements has been achieved. The percentage of surface elements which are connected to bad elements is very low, at 2 % when using lattice point insertion and at 14 % when using Delaunay refinement. This is due to the fact that the proposed technique was able to better represent the spacing function. Figure 5a shows a view of the volume mesh generated using the Delaunay refinement technique. The elements shown in black are the bad elements. There also appears to be a high concentration of bad elements at the leading edge of the wing, where the curvature is the highest. Figure 5b shows a view, at the same location, of the volume mesh generated using the lattice insertion technique. The bad elements are again shaded black, while a number of elements on the surface of the wing are well structured and are almost ideal. This mesh for the ONERA M6 geometry appears to be more structured and the effect of the lattice insertion is quite clear.

The second example is that of a B60 full aircraft configuration. the surface mesh consists of 144,240 triangular elements and 72,122 nodes. A view of the surface mesh is shown in Fig. 6. The corresponding mesh statistics for the B60 geometry are presented in Table 2. For this configuration, significantly fewer nodes are inserted using the lattice insertion method. As with the previous ONERA M6 wing example, the total percentage of bad elements is similar for both generation techniques, but the percentage of bad elements connected to the surface is much less when lattice insertion is employed. Figure 7a, b show similar views of the meshes created using Delaunay refinement and lattice point insertion respectively. As reflected by the statistics, in Fig. 7b it is apparent that there are areas of the mesh, attached to the boundaries, which almost match the ideal mesh. This can be explained by the fact that, the Delaunay insertion of the lattice nodes results in the ideal that satisfy the mesh quality requirements. There is the obvious question of how the constant orientation meets with the surfaces. It may be beneficial to adjust the orientation of the inserted lattice nodes at the surface. Alternatively, it may be possible to move the nodes on the surface, so as to produce a better match with the orientation of the lattice. These are possible modifications that we are currently considering.

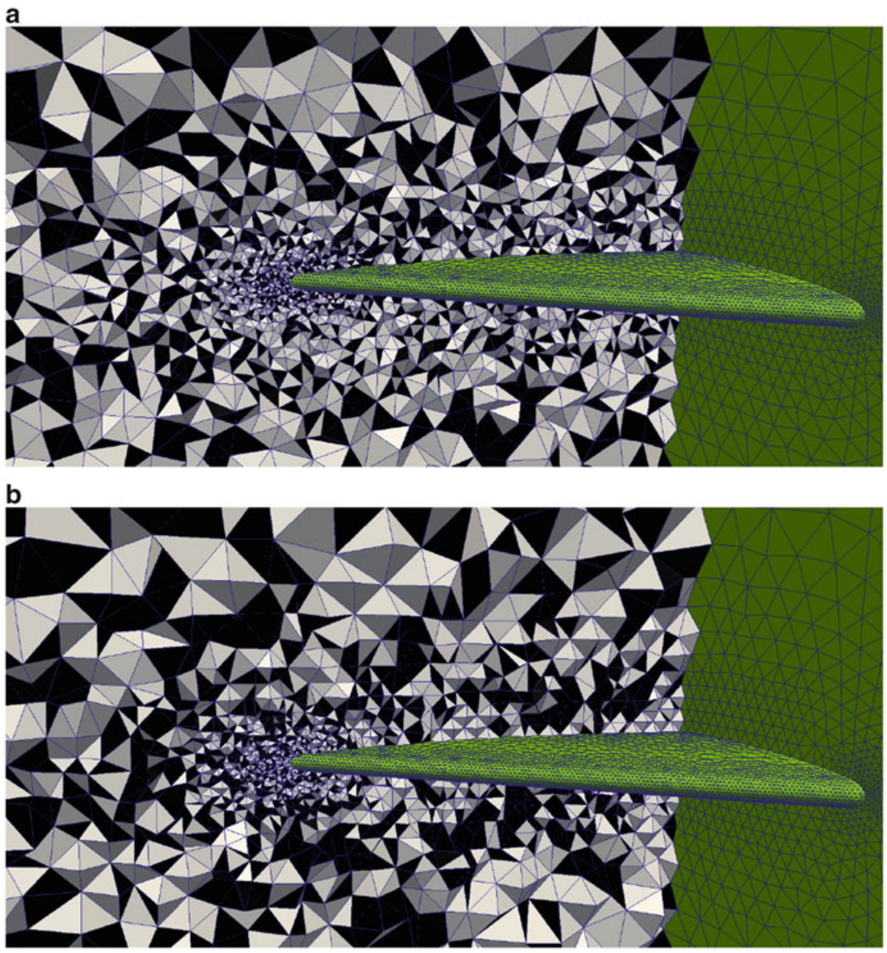


**Fig. 5** ONERA M6 wing: detail of the volume mesh generated using (**a**) Delaunay refinement; (**b**) lattice point insertion

The final example is a complete generic F16 configuration that provides a geometry that is challenging to mesh. The surface mesh, shown in Fig. 8 consists of 291,378 triangular element and 145,699 nodes.The mesh statistics that are obtained for this example are given in Table 3. Despite the increased geometric complexity, the conclusions reached are similar to those for the other two examples. The percentage of bad elements in the complete mesh is about the same for both techniques, but the percentage of surface triangles which connect to bad elements is

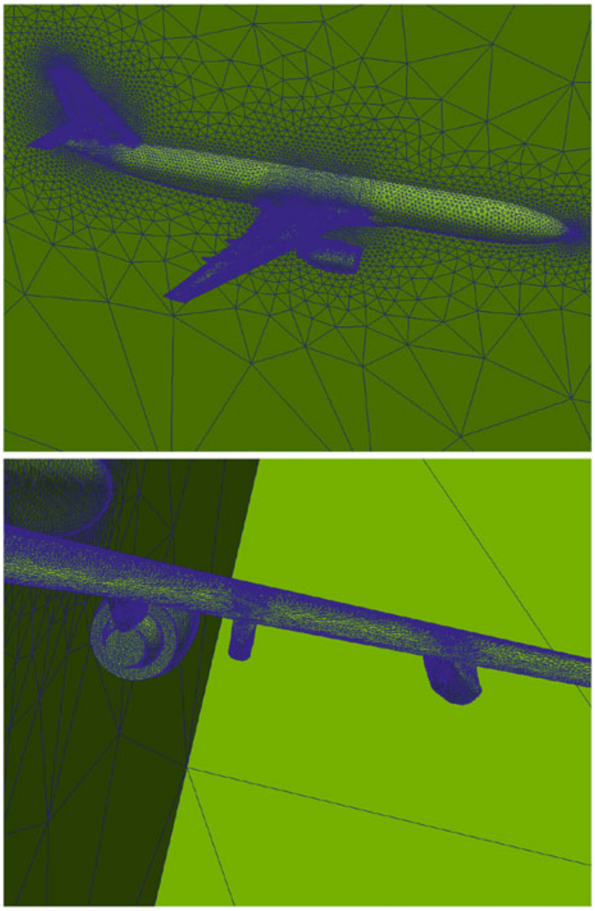


**Fig. 6** B60: details of the surface mesh, indicating the complicated nature of the geometry on the underside of the wing

**Table 2** Comparison of mesh quality measures for the B60

| | Delaunay refinement | Lattice point insertion |
|---|---|---|
| No. nodes | 1,358,561 | 1,221,477 |
| No. elements | 8,511,731 | 7,665,517 |
| No. bad elements touching a surface | 204,293 | 142,817 |
| No. (%) surface elements connected to a bad element | 18,417 (13 %) | 4,356 (3 %) |

significantly less when using lattice point insertion. Figure 9a, b show similar views of the meshes created by Delaunay refinement and lattice insertion respectively. In these figures, the surface elements are not plotted. It is apparent that there are areas of the mesh generated using lattice insertion which closely resemble the structured

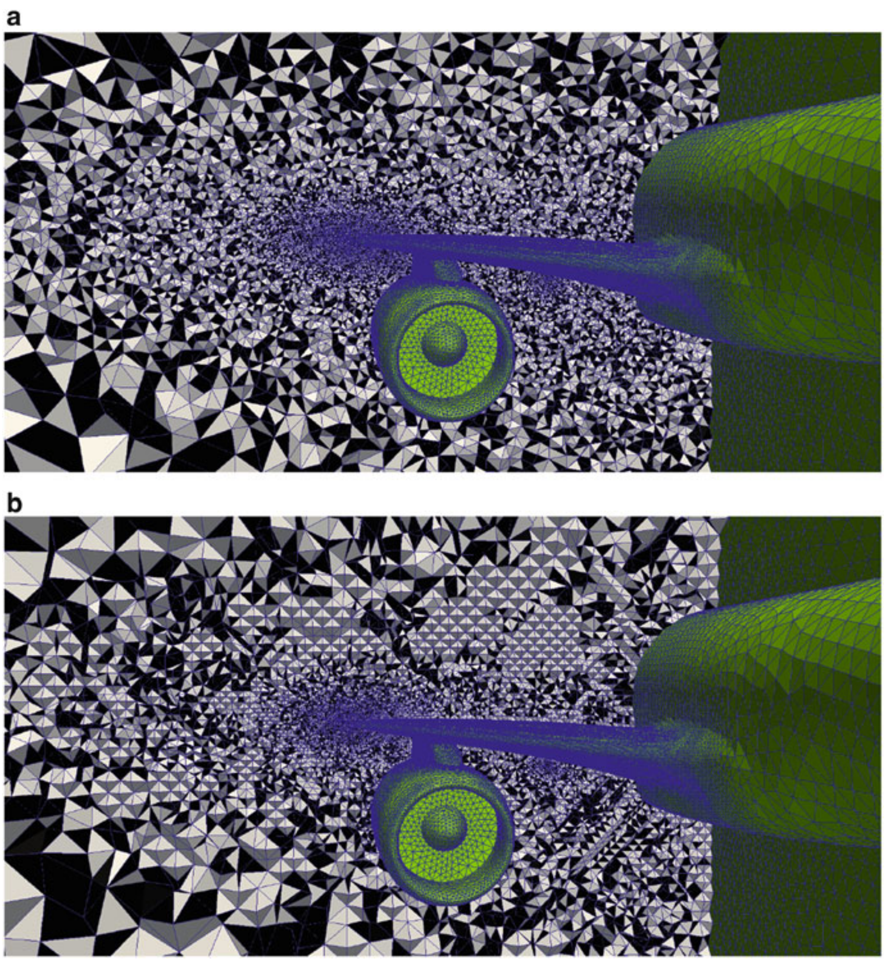


**Fig. 7** B60: detail of the volume mesh generated using (**a**) Delaunay refinement; (**b**) lattice point insertion

ideal mesh. This effect is even clearer when comparing Fig. 10a, b, which show wider views of the meshes generated using Delaunay refinement and lattice insertion respectively. Bad elements are not coloured in these figures. Pairs of elements in Fig. 10b which lie on the same circumsphere may be potential candidates for merging [7] before the mesh is employed with a co-volume scheme.

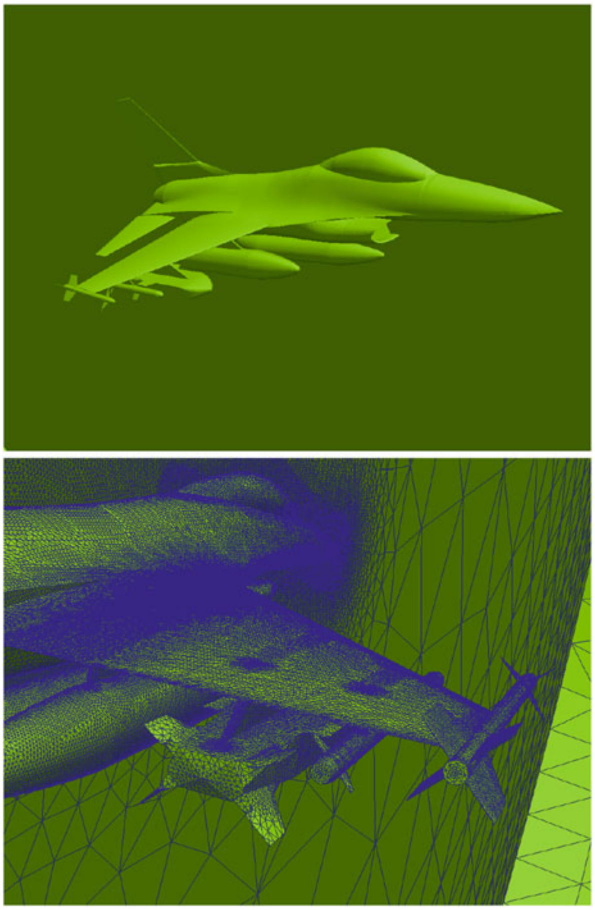


**Fig. 8** F16 configuration, details of the surface and the surface mesh, indicating the challenges posed by the presence of the attached stores

**Table 3** Comparison of mesh quality measures for the generic F16

| | Delaunay refinement | Lattice point insertion |
|---|---|---|
| No. nodes | 2,308,738 | 1,848,771 |
| No. elements | 14,389,566 | 11,367,474 |
| No. bad elements touching a surface | 424,796 | 309,181 |
| No. (%) surface elements connected to a bad element | 42,292 (14 %) | 8,150 (3 %) |

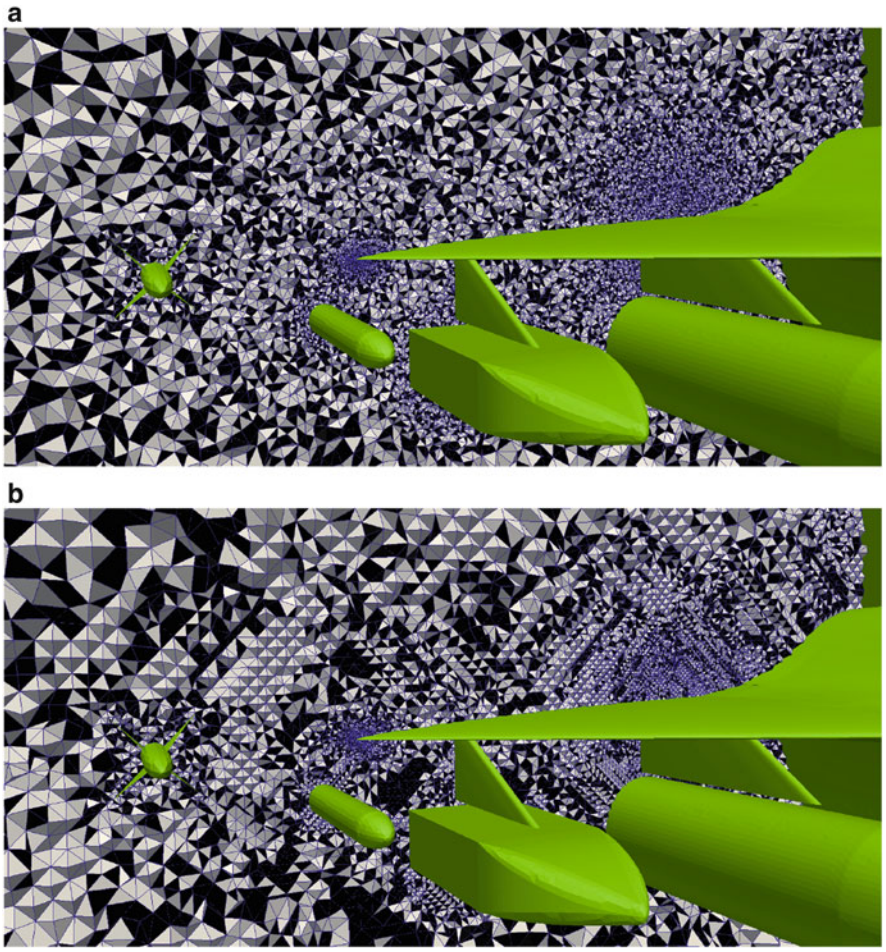


**Fig. 9** Generic F16: detail of the volume mesh generated using (**a**) Delaunay refinement; (**b**) lattice point insertion

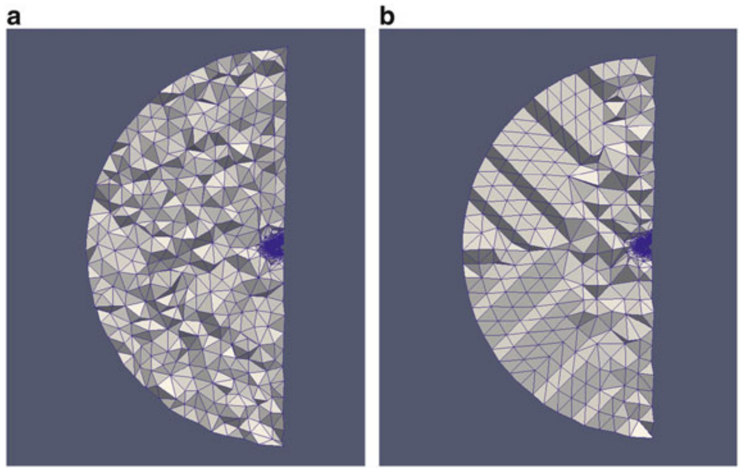


**Fig. 10** Generic F16: wider view of the volume mesh generated using (**a**) Delaunay refinement; (**b**) lattice point insertion

## 3 Mesh Optimisation

Mesh optimisation techniques attempt to minimise an objective function, which has been constructed to represent any mesh quality measure of interest. Classical gradient based optimisation techniques prove ineffective at finding global optima, particularly when considering non-smooth objective functions [22], which are typical of mesh quality measures [23]. This motivates the use of gradient free techniques, which employ large populations of potential solutions and which move around the search space evaluating the objective function. This means any optimum that is found is likely to be global and, since no objective function gradient information is required, the smoothness of the objective function has no effect on the process. Examples illustrating the application of gradient free techniques to mesh optimisation can be found in the literature [4, 23, 24]. Here, we consider the application of the gradient free optimisation technique, modified cuckoo search, to the meshes that have been generated using the lattice point insertion technique.

### *3.1 Modified Cuckoo Search (MCS)*

MCS [6] is a metaheuristic search algorithm that is inspired by the reproduction strategy of cuckoos. Cuckoo eggs have evolved to mimic the eggs of local birds by the process of laying eggs in the nests of host birds. These birds will discover eggs that do not resemble their own and destroy them. Thus, only cuckoo eggs which resemble the host bird eggs will pass genetic information to the next generation of cuckoos. MCS uses this simple mechanism as a basis for an optimisation algorithm. The algorithm turns out to be very effective, for high-dimensional objective functions. It has also been shown on benchmark functions that MCS has a high convergence rate to the global minimum.

### 3.2 Local Coordinate Optimisation

As noted above, the number of nodes in industrial meshes makes it impossible to apply a global gradient optimisation technique without some modification. Instead, a local approach is adopted, in which the position of each node of the Delaunay primal mesh is optimised in turn. Since MCS has a very high convergence rate, this is computationally feasible [4]. The objective function

$$F(k) = \sum_{i=1}^{E} W_i \frac{\|\mathbf{C}_i - \mathbf{V}_i\|}{\delta_i} \tag{1}$$

is minimised at each node, where $k$ is the node index, $E$ is the number of elements which include node $k$, the index $i$ nodes to the elements including node $k$, $\mathbf{C}_i$ is the position vector of the centroid of element $i$, $\mathbf{V}_i$ is the position vector of the Voronoi vertex of $i$, and $\delta_i$ is the mean edge length of element $i$. The weight $W_i$ is equal to zero if $V_i$ is inside element $i$ and equal to one otherwise. For each node, MCS is applied for five generations and the position of the node is moved towards the optimally located position [4]. After the volume mesh is generated, this procedure is applied recursively until no improvement in global mesh quality is obtained.

### 3.3 Local Weight Optimisation

Using this technique, it is not always possible to eliminate all bad elements from a mesh. A second approach is adopted, in which the requirement that a dual edge be a bisector of the corresponding Delaunay edge is relaxed, but the corresponding Voronoi vertex is moved in such a way to retain orthogonality [7]. To illustrate this process consider the two dimensional element shown in Fig. 11. If circles of equal radius are positioned such that their centres lie on the nodes of a Delaunay element, then the corresponding Voronoi vertex is located at the point of intersection of the common chords of these circles, as shown in Fig. 11a. By reducing the radius of the circle at $B$, the Voronoi vertex is pulled inside the element as shown in Fig. 11b. Changing this radius will also affect the position of Voronoi vertices in surrounding elements such that orthogonality is obtained. By allocating a weight to each Delaunay node, which corresponds to the change in the radius of the circle

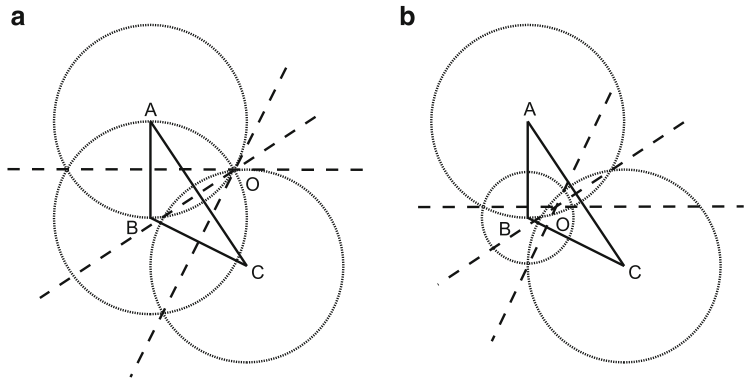


**Fig. 11** (**a**) A bad element ABC and the corresponding Voronoi vertex O; (**b**) moving the Voronoi vertex O to repair the bad element ABC

**Table 4** Quality measures for the ONERA M6 wing mesh generated using lattice point insertion and modified cuckoo search optimisation

| | |
|---|---|
| No. nodes | 78,999 |
| No. elements | 485,254 |
| No. of bad elements touching a surface | 4,154 |
| No. (%) surface elements connected to a bad element | 99 (0.6 %) |

(or the sphere in three dimensions), the coordinates of each Voronoi vertex can be readily obtained [4].

After this coordinate optimisation procedure is completed, MCS is used to optimise the weight of each node, in turn, using the objective function of Eq. (1). This procedure is repeated until no further improvement is obtained.

## 3.4 Examples

The optimisation procedures described above are applied to the meshes generated previously using the lattice point insertion technique. The results are presented in the same fashion as previously. All optimisation processes were run until no further improvement was produced. To give an idea of the runtime requirements for this approach, the M6 example was completed in around half an hour, while the other examples were left to run overnight for around 8–12 h.

The quality measures for the ONERA M6 volume mesh, generated and optimised as described above, are given in Table 4. A view of a cut through the mesh is shown in Fig. 12. Bad elements are coloured black in this figure. The optimisation process reduces the total percentage of faces which are connected to a bad element has reduced from 2 to 0.6 %.

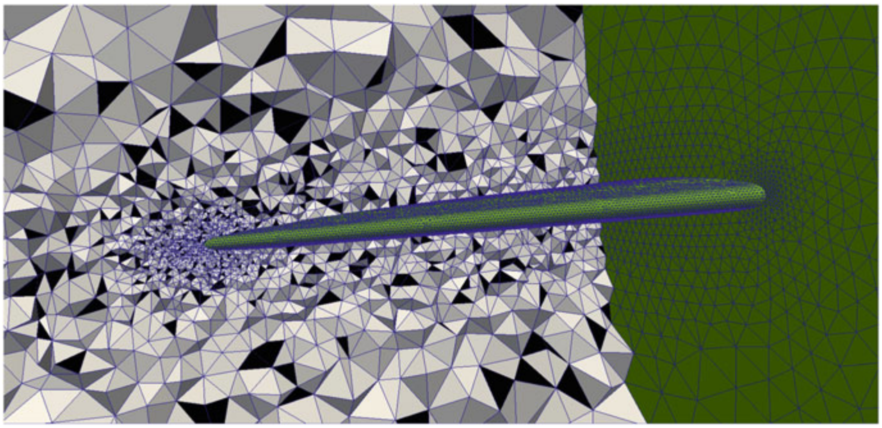


**Fig. 12** ONERA M6 wing: detail of the volume mesh generated using lattice point insertion and modified cuckoo search optimisation

**Table 5** Quality measures for the B60 mesh generated using lattice insertion and modified cuckoo search optimisation

| | |
|---|---|
| No. nodes | 1,120,620 |
| No. elements | 7,086,250 |
| No. bad elements touching a surface | 47,305 |
| No. (%) surface elements connected to a bad element | 1,550 (1 %) |

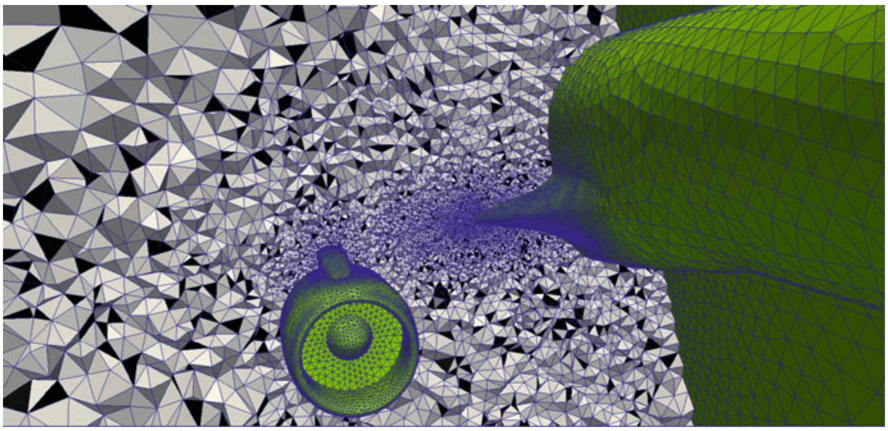


**Fig. 13** B60: detail of the volume mesh generated using lattice point insertion and modified cuckoo search optimisation

The quality measures for the mesh produced for the B60, after optimisation, are presented in Table 5. A view of a cut through this mesh is shown in Fig. 13 and the bad elements are again coloured black. The percentage of surface elements connected to bad elements has also dropped from 3 to 1 %.

**Table 6** Quality measures for the generic F16 mesh generated using lattice insertion and modified cuckoo search optimisation

| | |
|---|---|
| No. nodes | 1,657,926 |
| No. elements | 10,280,144 |
| No. bad elements touching a surface | 106,631 |
| No. (%) surface elements connected to a bad element | 3,432 (2 %) |

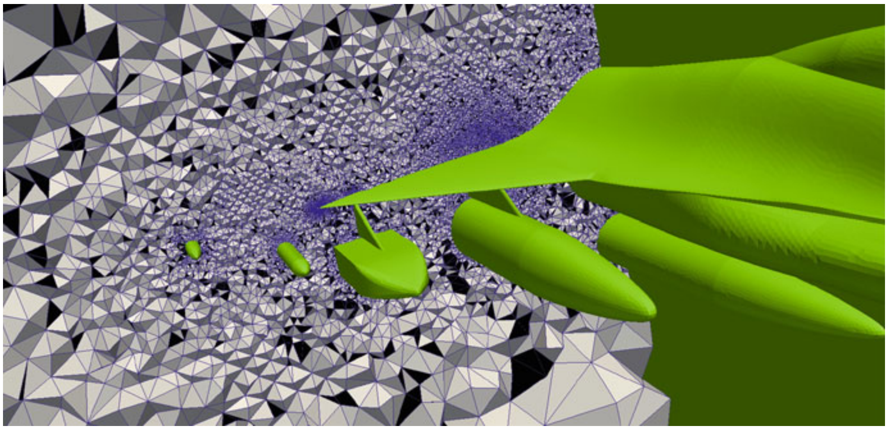


**Fig. 14** Generic F16: detail of the volume mesh generated using lattice point insertion and modified cuckoo search optimisation

The results for the mesh created for the generic F16 show the same trend as the other examples. The mesh quality measures are given in Table 6 and a detail of a cut through the mesh is shown in Fig. 14. In this figure, black elements indicate bad elements. The optimisation process reduces the percentage of faces which are connected to bad elements from 3 to 2 %.

## 4 Conclusions

In this paper, we have detailed our current line of investigation into the problem of generating well centred tetrahedral meshes for general geometries. The lattice insertion method appears to be a promising approach, which offers certain advantages over competing methods to produce the quality required for a co-volume schemes. The steps involved in the algorithm are based on a few simple rules, which gives confidence that the method will be robust when it is applied to problems involving complex geometries. This claim is supported by the quality of the initial results obtained for the examples presented here. Future investigations will concentrate on the possibility of controlling the orientation of the lattice, of merging elements as part of the mesh generation process and of preparing the surface mesh by moving nodes into positions better suited for building the lattice. We have also shown that, by additionally applying previously developed optimisation techniques, we can significantly improve the quality of these meshes. The generation of this high quality near field meshes will then enable the extension of the stitching methods into three dimensions.

**Acknowledgements** Sean Walton acknowledges the financial support for this work provided by the UK Engineering and Physical Sciences Research Council (EPSRC) under Research Grant EP/K000705.

## References

1. Harlow, F.H., Welch, J.E.: Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. Phys. Fluids **8**, 2182–2189 (1965)
2. Yee, K.: Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. IEEE Trans. Antennas Propag. **14**, 302–307 (1966)
3. Sazonov, I., Wang, D., Hassan, O., Morgan, K., Weatherill, N.: A stitching method for the generation of unstructured meshes for use with co-volume solution techniques. Comput. Methods Appl. Mech. Eng. **195**, 1826–1845 (2006)
4. Walton, S., Hassan, O., Morgan, K.: Reduced order mesh optimisation using proper orthogonal decomposition and a modified cuckoo search. Int. J. Numer. Methods Eng. **93**, 527–550 (2013)
5. Walton, S.: Gradient free optimisation in selected engineering applications. PhD Thesis, Swansea University (2013)
6. Walton, S., Hassan, O., Morgan, K.: Modified cuckoo search: A new gradient free optimisation algorithm. Chaos Solitons Fractals **44**, 710–718 (2011)
7. Xie, Z.Q., Hassan, O., Morgan, K.: Tailoring unstructured meshes for use with a 3D time domain co-volume algorithm for computational electromagnetics. Int. J. Numer. Methods Eng. **87**, 48–65 (2011)
8. Sazonov, I., Hassan, O., Morgan, K., Weatherill, N.P.: Smooth Delaunay–Voronoi dual meshes for co-volume integration schemes. In: Pebay, P.P. (ed.) Proceedings of the 15th International Meshing Roundtable, pp. 529–541. Springer, Berlin (2006)
9. Naylor, D.J.: Filling space with tetrahedra. Int. J. Numer. Methods Eng. **44**, 1383–1395 (1999)
10. Eppstein, D., Sullivan, J.M., Üngör, A.: Tiling space and slabs with acute tetrahedra. Comput. Geom. Theory Appl. **27**, 237–255 (2004)
11. Sazonov, I., Hassan, O., Morgan, K., Weatherill, N.P.: Generating the Voronoï–Delaunay dual diagram for co-volume integration schemes. In: Gold, C.M. (ed.) Proceedings of the 4th International Symposium on Voronoï Diagrams in Science and Engineering, ISPRS, IEEE CPS, pp. 199–204 (2007)
12. van der Zee, E., Hirani, A., Guoy, D.: Triangulation of simple 3D Shapes with well-centered tetrahedra. In: Garimella, R. (ed.) Proceedings of the 17th International Meshing Roundtable. Springer, Berlin (2008)
13. van der Zee, E., Hirani, A., Guoy, D., Ramos, E.: Well-centered triangulation. SIAM J. Sci. Comput. **31**, 4497–4523 (2010)
14. Yan, D.-M., Wang, W., Lévy, B., Liu, Y.: Efficient computation of clipped Voronoi diagram for mesh generation. Comput. Aid. Des. **45**, 843–852 (2013)
15. Liu, Y., Wang, W., Lévy, B., Sun, F., Yan, D.-M., Lu, L., Yang, C.: On centroidal Voronoi tessellation – Energy smoothness and fast computation. ACM Trans. Graph. **28**, 4 (2009)
16. Peraire, J., Vahdati, M., Morgan, K., Zienkiewicz, O.C.: Adaptive remeshing for compressible flow computations. J. Comput. Phys. **72**, 449–466 (1987)

17. Weatherill, N.P., Hassan, O.: Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. Int. J. Numer. Methods Eng. **37**, 2005–2040 (1994)
18. Morgan, K., Peraire, J., Peiro, J.: Unstructured grid methods for compressible flows. Report 787: Special Course on Unstructured Grid Methods for Advection Dominated Flows, AGARD, Paris, pp. 5.1–5.39 (1992)
19. Radovitzky, R., Ortiz, M.: Tetrahedral mesh generation based on node insertion in crystal lattice arrangements and advancing-front-Delaunay triangulation. Comput. Methods Appl. Mech. Eng. **187**, 543–569 (2000)
20. Rajan, V.T.: Optimality of the Delaunay triangulation in Rd. Discrete Comput. Geom. **12**, 189–202 (1994)
21. Kennel, M.: KDTREE 2: Fortran 95 and C++ software to efficiently search for near neighbors in a multi-dimensional Euclidean space. ArXiv Physics e-prints arXiv:physics/0408067v2 (2004)
22. Praveen, C., Duvigneau, R.: Low cost PSO using metamodels and inexact pre-evaluation: application to aerodynamic shape design. Comput. Methods Appl. Mech. Eng. **198**, 1087–1096 (2009)
23. Park, J., Shontz, S.M.: Two derivative-free optimization algorithms for mesh quality improvement. Proc. Comput. Sci. **1**, 387–396 (2010)
24. Yilmaz, A., Kuzuoglu, M.: A particle swarm optimization approach for hexahedral mesh smoothing. Int. J. Numer. Methods Fluids **60**, 55–78 (2009)

# Enhanced Viscous Mesh Generation with Metric-Based Blending

**David Marcum and Frédéric Alauzet**

**Abstract** In this work we describe a unified approach that blends the best characteristics of both a near body pseudo-structured boundary-layer (BL) and generalized anisotropic metric approaches. Specifically, near-body physics with anisotropy are resolved using an a priori pseudo-structured process and off-body or field features are resolved using an adaptive generalized approach. In particular the metric field of the adaptive approach is derived from the BL region of the pseudo-structured approach. The derived metric is based on local aspect ratio and geometry. This metric is then blended from the BL region into the overall field to allow for a smooth transition to the generalized field. The result is a flexible and optimal overall mesh generation process that can be used with or without adaptation. Metric-based formulations for quality functions and other geometric quantities require for mesh generation are presented. Results are presented that demonstrate the overall approach in the context of blending between the near body pseudo-structured region and the outer tetrahedral field region. These results point out that the metric-based transition can be used to improve mesh quality and density for configurations with anisotropic surface meshes and BL regions that do not reach outer region length scale.

## 1 Introduction

Large-scale computational field simulation applications with unstructured meshes are widely used to help solve real world problems found in industry and government. In many of these cases the physics involved includes widely varying gradients. In particular, computational fluid dynamic (CFD) applications often involve significant

D. Marcum (✉)
Center for Advanced Vehicular Systems (CAVS) and Mechanical Engineering Department, Mississippi State University, Mississippi, MS 39762, USA
e-mail: marcum@cavs.msstate.edu

F. Alauzet
GAMMA3 Team, INRIA Paris-Rocquencourt, 78153 Le Chesnay, France
e-mail: Frederic.Alauzet@inria.fr

S. Perotto, L. Formaggia (eds.), *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, SEMA SIMAI Springer Series 5,
DOI 10.1007/978-3-319-06053-8_9

flow field regions where viscous effects are dominant. These regions have gradients that vary by orders of magnitude in different directions. While viscous regions are typically prominent at the vehicle surface in the boundary-layer (BL) region, they can also be significant in the field with shear layers and vortical flows. In addition, high-speed flow fields can also include discontinuous phenomena, with shock waves and contact surfaces. These features may also have significant interaction with viscous dominated regions. High-resolution simulation of such cases with ideal isotropic discretization is simply not possible. Discretization of the field using anisotropic elements with length scales that match the gradients of the physics is the only known way to numerically solve such problems with a high level of resolution.

There are two primary approaches to generating unstructured meshes with anisotropic elements. The most general involves anisotropic triangulation using generalized metric terms [2, 6, 7, 10, 12, 13, 17, 23]. If the metrics are based upon the flow field as it evolves in the simulation [2, 3] then the result is a mesh adapted to the physics. Such a mesh typically has length scales that are directionally optimal for the given flow field. However, in BL regions the characteristics of a mesh generated using a generalized approach are not always ideal. Viscous BL regions near a surface often have very stringent numerical requirements as they involve high-gradient and non-linear physics that usually includes turbulence. These regions are known a priori and ideally suited to a pseudo-structured approach that generates elements with an advancing layer/normal type process [11, 14, 18, 19, 21, 24]. The resulting mesh is highly aligned, precisely spaced and very structured in at least the normal direction. Often such regions are generated with pentahedral and hexahedral elements for optimal flow solver efficiency. The characteristics of a pseudo-structured type mesh are ideal for the BL regions. Combining an advancing layers approach with a high-quality tetrahedral mesh generator for the field region results in an ideal approach for many applications. However, if there are significant field features then this approach requires an aligning surface for off-body features that may not be know a priori and it may be difficult to smoothly blend between attached and detached regions.

In this work we describe an approach that blends the two very different processes for a more elegant and optimal solution to overall problem. A good example application is an aircraft in a landing or takeoff condition as shown in Fig. 1. In such cases there are numerous regions of viscous dominated flow with fully extended flaps and slats that interact with the main wing in widely varying ways and produce substantial detached viscous shear layers and vortices. Also, small features can be significant contributors to overall performance and high-resolution discretization of all features is essential to accurate simulation. In addition, anisotropy is required in multiple directions, not just normal to the surface. A blended approach that smoothly blends the near body pseudo-structured region into a generalized anisotropic field region to capture attached and detached shear layers considerably improves the knowledge of and ability to predict such flow fields [22].

The unified approach taken here utilizes the best characteristics of both near body BL and generalized approaches. Specifically, near-body physics with anisotropy are resolved using an a priori pseudo-structured process [19, 21] and off-body or field features are resolved using an adaptive generalized approach [8, 9, 15, 17]. In

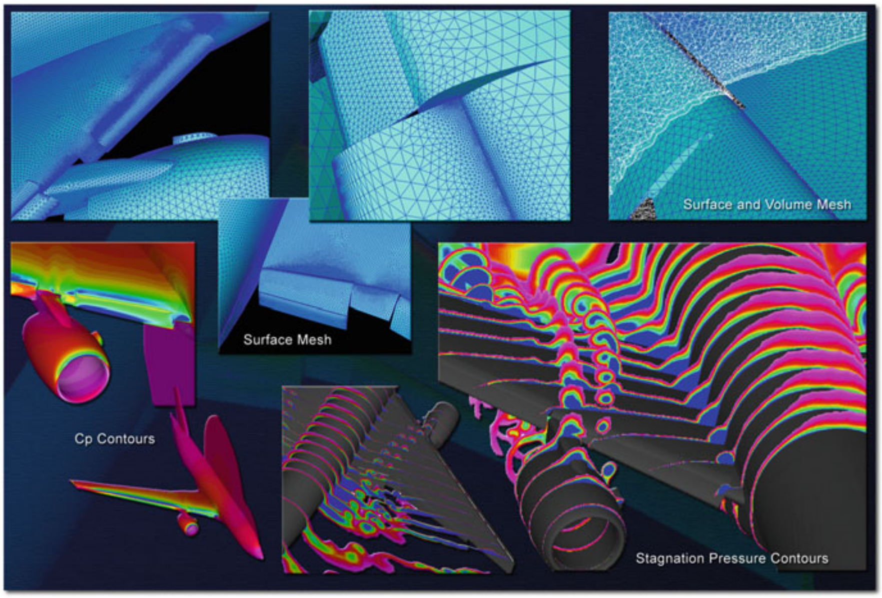


**Fig. 1** Typical unstructured mesh and solution for a high-lift aircraft configuration

particular the metric field of the adaptive approach is derived from the BL region of the pseudo-structured approach. The derived metric is based on local aspect ratio and geometry. This metric is then blended from the BL region into the overall field to allow for a smooth transition to the generalized field. The result is a flexible and optimal overall mesh generation process that can be used with or without adaptation. Results are presented here without adaptation that demonstrate the overall approach in the context of blending between the near body pseudo-structured region and the outer tetrahedral field region.

## 2 Mesh Generation Algorithms

### *2.1 Advancing-Front and Local Reconnection (AFLR) Algorithm*

The volume meshing process used in the present work is based on the Advancing-Front and Local Reconnection (AFLR) algorithm [19, 21]. It has proven to be capable of generating high-quality unstructured volume meshes suitable for large-scale high-resolution CFD applications and is widely used in aerospace and other engineering disciplines.

The overall grid generation procedure used in the present work is a combination of automatic point creation, advancing type ideal point placement, and connectivity optimization schemes. A valid grid is maintained throughout the grid generation process. This provides a framework for implementing efficient local search operations using a simple data structure. It also provides a means for smoothly distributing the desired point spacing in the field using a point distribution function. This function is propagated through the field by interpolation from the boundary point spacing or by specified growth normal to the boundaries. Points are generated using either advancing-front type point placement for isotropic elements, advancing-point type point placement for isotropic right angle elements, or advancing-normal type point placement for high-aspect-ratio elements. The connectivity for new points is initially obtained by direct subdivision of the elements that contain them. Connectivity is then optimized by local-reconnection with a combined Delaunay/min-max type (minimize the maximum angle, maximize the edge weight, etc.) type criterion. The overall procedure is applied repetitively until a complete field grid is obtained.

Complete details of the algorithm and results are presented in [19, 21]. For the present work the algorithm is modified to operate in metric space. The overall algorithm above utilizes an isotropic length scale defined by the distribution function. For generalized anisotropic elements (not pseudo-structured boundary-layer (BL) elements) an anisotropic length scale definition is required. For the present work a generalized metric approach is used to generate anisotropic elements outside of the BL region. Modifications to the original algorithm for anisotropy are accomplished by using a functional approach for calculation of all geometric properties such as dot products, cross products, reconnection criterion, etc. Processes that require modification include; ideal point placement for creation of new points, proximity checking, and local-reconnection criteria. All of these processes are modified to operate in metric space rather than physical space. These modifications are described in Sect. 3.

## *2.2 Boundary Layer Meshing Algorithm*

Within the BL region a pseudo-structured mesh aligned with the boundary surface is optimal for accuracy and performance of typical CFD solvers. This is the approach taken in the present work. While the standard unstructured meshing procedure previously described can be utilized to generate pseudo-structured elements in the BL region, an open (extrusion) or closed (displacement) approach is far more efficient. A modified procedure using an advancing-normal/layers approach [19, 20] is used for volume mesh generation. In this approach, the element connectivity is generated along with new points in high-aspect-ratio regions. Local-reconnection is not used to determine the connectivity in these regions. Instead, the connectivity is directly determined as each new point is generated. This produces a very structured connectivity and allows the tetrahedral elements to be easily combined into structured type elements. Typically, the majority of the tetrahedral elements within the high-aspect-ratio region can be combined into prismatic elements. If the surface mesh contains quad faces then hexahedral elements can be formed. The outer layer of this region may require some pyramid elements to match the outer tetrahedral element region. In all cases, the pentahedral elements have strict node, edge, and face matching to each other and to neighboring tetrahedral elements. Hexahedral elements may have either full matching with an attached pyramid element or split-face matching without.

For the present work the advancing-normal/layers algorithm does not require a generalized anisotropic metric as the a priori assumption of BL gradients is part of the process. In an adaptive process the tangential and normal spacing could be adapted using a generalized metric approach [8, 9, 15, 17] by treating each separately. In a typical complex aerospace configuration the elements at the edge of the BL region often have considerable anisotropy from the normal spacing not having reached outer region length scales. Further, optimal surface meshes for high-resolution of leading edge type or high-curvature regions require a structured or pseudo-structured mesh aligned with the curvature. Example surface meshes are shown for a wing-body in Fig. 5 and for a nacelle in Fig. 9. As with BL regions, aligned pseudo-structured quad-faces or right-angle-tria-faces are optimal. Surface aspect-ratios for realistic configurations often involve anisotropy similar in scale to BL regions. Unfortunately this results in issues with transition to isotropic elements that are more ideal in the outer field region. This creates practical limits on anisotropy of the surface faces to order ten. Consequently the combined pseudo-structured and generalized anisotropic meshing approach is advocated in this work to eliminate these limits. In the overall implementation a metric is required to accomplish the blending from BL to outer region. This metric is derived from the normal and tangential spacing of BL region elements in the outer layer. A discussion of this is provided in Sect. 3.3.

## 3 Metric-Based Transition for Viscous Flowfields

This section deals with the blending or the transition of the mesh between the pseudo-structured boundary-layer (BL) region mesh and the generalized outer mesh region. We propose to use metric-based anisotropic mesh operators to generate a smooth anisotropic transition between the boundary layer and the rest of the flow field. These operators have been widely used successfully for anisotropic mesh adaptation [2, 6, 7, 10, 12, 13, 17, 23].

The blending requires to compute a metric field associated with the last layer of the boundary layer, to extrapolate this metric field into the flow field and to write the operators of Sect. 2.1 into the metric space, in other words, all geometric quantities thus quality function are computed in the given metric space. Moreover, the addition of metric-based operators provides the necessary framework and capability for adaptation with generalized anisotropic elements.

## *3.1 Basics of Metric and Notion of Unit Mesh*

### 3.1.1 Euclidean Metric Space

For the sake of clarity, we recall the differential geometry notions that are used in the sequel. We use the following notations: bold face symbols, as $\mathbf{a}, \mathbf{b}, \mathbf{u}, \mathbf{v}, \mathbf{x}, \ldots$, denote vectors or points of $\mathbb{R}^3$. The natural dot and cross product between two vectors $\mathbf{u}$ and $\mathbf{v}$ of $\mathbb{R}^3$ are denoted by: $\mathbf{u} \cdot \mathbf{v} = \langle \mathbf{u},\ \mathbf{v} \rangle$ and $\mathbf{u} \times \mathbf{v}$.

An **Euclidean metric space** $(\mathbb{R}^3, \mathcal{M})$ is a vector space of finite dimension where the dot product is defined by means of a symmetric definite positive tensor $\mathcal{M}$ :

$$\mathbf{u} \cdot_{\mathcal{M}} \mathbf{v} = \langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{M}} = \langle \mathbf{u}, \mathcal{M}\mathbf{v} \rangle = {}^t\mathbf{u}\, \mathcal{M}\, \mathbf{v}, \quad \text{for } (\mathbf{u}, \mathbf{v}) \in \mathbb{R}^3 \times \mathbb{R}^3 . \tag{1}$$

In the following, the matrix $\mathcal{M}$ is simply called **a metric tensor** or **a metric**. In such space, the **length** $\ell_{\mathcal{M}}$ of a segment $\mathbf{ab}$ is given by the distance between its extremities:

$$\ell_{\mathcal{M}}(\mathbf{ab}) = d_{\mathcal{M}}(\mathbf{a}, \mathbf{b}) = \|\mathbf{ab}\|_{\mathcal{M}} = \sqrt{\mathbf{ab} \cdot_{\mathcal{M}} \mathbf{ab}}. \tag{2}$$

As metric tensor $\mathcal{M}$ is symmetric, it is diagonalizable in an orthonormal basis:

$$\mathcal{M} = {}^t\mathcal{R}\, \Lambda\, \mathcal{R},$$

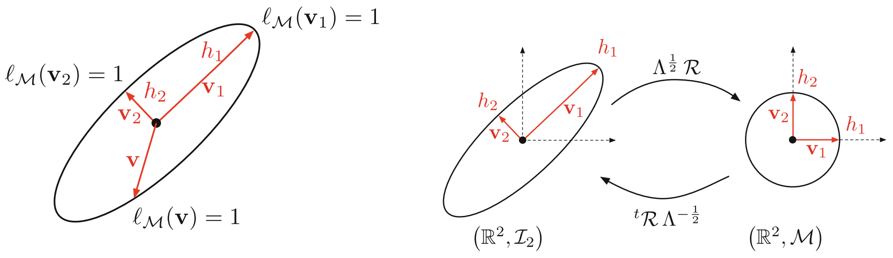
where $\mathcal{R}$ is an orthonormal matrix the lines of which are composed of the **eigenvectors** $(\mathbf{v}_i)_{i=1,3}$ of $\mathcal{M}$ verifying ${}^t\mathcal{R}\mathcal{R} = \mathcal{R}^t\mathcal{R} = \mathcal{I}_3$. $\Lambda = \mathrm{diag}(\lambda_i)$ is a diagonal matrix composed of the **eigenvalues** of $\mathcal{M}$, denoted $(\lambda_i)_{i=1,3}$, which are strictly positive. From the previous definition, we deduce that application $\Lambda^{\frac{1}{2}}\mathcal{R}$ where $\Lambda^{\frac{1}{2}} = \mathrm{diag}(\lambda_i^{\frac{1}{2}})$ defines the **mapping** from the physical space $(\mathbb{R}^3, \mathcal{I}_3)$, where $\mathcal{I}_3$ is the identity matrix, to the Euclidean metric space $(\mathbb{R}^3, \mathcal{M})$.

We are now able to deduce the following expression of the 3D **cross product** with respect to $\mathcal{M}$:

$$\mathbf{u} \times_{\mathcal{M}} \mathbf{v} = \big(\Lambda^{\frac{1}{2}}\mathcal{R}\big)\mathbf{u} \times \big(\Lambda^{\frac{1}{2}}\mathcal{R}\big)\mathbf{v} = \sqrt{\det \mathcal{M}}\, \big({}^t\mathcal{R}\, \Lambda^{-\frac{1}{2}}\big)\, (\mathbf{u} \times \mathbf{v}) . \tag{3}$$

In an Euclidean metric space, volumes and angles are still well defined. These features are of main interest when dealing with meshing. For instance using Relations (1) and (3), given a parallelepiped $K$ of $\mathbb{R}^3$, the **volume** of $K$ computed with respect to metric tensor $\mathcal{M}$ is:

$$|K|_{\mathcal{M}} = \mathbf{u} \cdot_{\mathcal{M}} (\mathbf{v} \times_{\mathcal{M}} \mathbf{w}) = \sqrt{\det \mathcal{M}}\, |K|_{\mathcal{I}_3}, \tag{4}$$



$\ell_{\mathcal{M}}(\mathbf{v}_1) = 1$
$\ell_{\mathcal{M}}(\mathbf{v}_2) = 1$
$h_1$
$h_2$
$\mathbf{v}_1$
$\mathbf{v}_2$
$\mathbf{v}$
$\ell_{\mathcal{M}}(\mathbf{v}) = 1$
$\Lambda^{\frac{1}{2}}\mathcal{R}$
${}^t\mathcal{R}\,\Lambda^{-\frac{1}{2}}$
$(\mathbb{R}^2, \mathcal{I}_2)$
$(\mathbb{R}^2, \mathcal{M})$

**Fig. 2** *Left*: geometric interpretation of the unit ball $\mathscr{B}_{\mathscr{M}}$ where $\mathbf{v}_i$ are the eigenvectors of $\mathscr{M}$ and $\lambda_i = h_i^{-2}$ are the eigenvalues of $\mathscr{M}$. *Right*: mappings between physical space $(\mathbb{R}^3, \mathscr{I}_3)$ and Euclidean metric space $(\mathbb{R}^3, \mathscr{M})$

where $|K|_{\mathscr{I}_3}$ is the Euclidean volume of $K$. The **angle** between two non-zero vectors $\mathbf{u}$ and $\mathbf{v}$ is defined by the unique real-value $\theta_{\mathscr{M}} \in [0, \pi]$ verifying:

$$\cos(\theta_{\mathscr{M}}) = \frac{\langle \mathbf{u}, \mathbf{v}\rangle_{\mathscr{M}}}{\|\mathbf{u}\|_{\mathscr{M}} \|\mathbf{v}\|_{\mathscr{M}}} . \tag{5}$$

In three dimensions, **dihedral angles**[1] of a tetrahedron can be computed in a given Euclidean space from this definition as it is the angle between the faces normal. These relations are very useful in 3D for computing the area of faces with respect to $\mathscr{M}$, let $F$ be a triangle in $\mathbb{R}^3$:

$$2\,|F|_{\mathscr{M}} = \|\big(\Lambda^{\frac{1}{2}}\mathscr{R}\big)\mathbf{u} \times \big(\Lambda^{\frac{1}{2}}\mathscr{R}\big)\mathbf{v}\| = \|\mathbf{u}\|_{\mathscr{M}}\, \|\mathbf{v}\|_{\mathscr{M}}\, \sin(\theta_{\mathscr{M}}) .$$

We will often refer to the geometric interpretation of a metric tensor. In the vicinity $\mathscr{V}(\mathbf{a})$ of point $\mathbf{a}$, the set of points that are at distance 1 of $\mathbf{a}$ is called the **unit ball** of $\mathscr{M}$ and is given by:

$$\mathscr{B}_{\mathscr{M}} = \big\{\mathbf{x} \in \mathscr{V}(\mathbf{a}) \mid {}^t(\mathbf{x} - \mathbf{a})\, \mathscr{M}\, (\mathbf{x} - \mathbf{a}) = 1\big\} .$$

The above relation defines an ellipsoid centered at $\mathbf{a}$ with its axes aligned with the eigen directions of $\mathscr{M}$. Sizes along these directions are given by $h_i = \lambda_i^{-\frac{1}{2}}$. This ellipsoid depicted in Fig. 2 (left).

### 3.1.2 Riemannian Metric Space

In differential geometry, a Riemannian manifold or Riemannian space $(M, \mathscr{M})$ is a smooth manifold $M$ in which each tangent space is equipped with a dot product

[1]The dihedral angle is the angle between two triangular faces of a tetrahedron.

defined by a metric tensor $\mathcal{M}$, a Riemannian metric, in a manner which varies smoothly from point to point. Even if no global definition of the scalar product exists, various geometric notions can be defined on a Riemannian manifold. In the context of mesh adaptation, we do not know any manifold. For our concern, we work with a simpler object called a **Riemannian metric space** defined by $\mathbf{M} = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$. In that specific case, we only know $\mathbf{M}$ a Riemannian metric and $\Omega \subset \mathbb{R}^n$ a common space of parametrization which is our computational domain. There is still no global notion of scalar product.

In the context of meshing, notions of length and volume defined in Sect. 3.1.1 can be easily extend to Riemannian metric spaces because we want to evaluate then for a given parametrization. Indeed, as regards edge length computation, we want to compute the length of the path between these two points defined by the straight line parameterization. To take into account the variation of the metric along the edge, the edge length is evaluated with an integral formula. Formally speaking, in Riemannian metric space $\mathbf{M} = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$, the **length** of edge **ab** is computed using the straight line parameterization $\gamma(t) = \mathbf{a} + t\,\mathbf{ab}$, where $t \in [0, 1]$:

$$\ell_{\mathcal{M}}(\mathbf{ab}) = \int_0^1 \|\gamma'(t)\|_{\mathcal{M}}\,\mathrm{d}t = \int_0^1 \sqrt{{}^t\mathbf{ab}\;\mathcal{M}(\mathbf{a} + t\,\mathbf{ab})\;\mathbf{ab}}\,\mathrm{d}t\,, \tag{6}$$

and, given a bounded subset $K$ of $\Omega$, the **volume** of $K$ computed with respect to $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ is:

$$|K|_{\mathcal{M}} = \int_K \sqrt{\det \mathcal{M}(\mathbf{x})}\,\mathrm{d}\mathbf{x}\,. \tag{7}$$

### 3.1.3 Metric-Based Mesh Generation

To generate anisotropic meshes, one must be able to prescribe at each point of the domain privileged sizes and orientations for the elements. This information will be transmitted to the mesher which will try to best fit these demands. The use of Riemannian metric spaces is an elegant and efficient way to achieve this goal. The main idea of metric-based mesh adaptation, initially introduced in [9], is to generate **a unit mesh** in a prescribed Riemannian metric space.

*Unit Element* A tetrahedron $K$, defined by its list of edges $(\mathbf{e_i})_{i=1..6}$, is **unit** with respect to a metric tensor $\mathcal{M}$ if the length of all its edges is unit in metric $\mathcal{M}$:

$$\forall i = 1,\ldots,6,\;\; \ell_{\mathcal{M}}(\mathbf{e_i}) = 1 \text{ with } \ell_{\mathcal{M}}(\mathbf{e_i}) = \sqrt{{}^t\mathbf{e_i}\,\mathcal{M}\,\mathbf{e_i}}.$$

If $K$ is composed only of unit length edges, then its volume $|K|_{\mathcal{M}}$ in $\mathcal{M}$ is constant equal to:

$$|K|_{\mathcal{M}} = \frac{\sqrt{2}}{12} \quad \text{and} \quad |K| = \frac{\sqrt{2}}{12}(\det(\mathcal{M}))^{-\frac{1}{2}},$$

where $|K|$ is its Euclidean volume.

*Unit Mesh* The notion of unit mesh is far more complicated than the notion of unit element as the existence of a mesh composed only of unit regular simplices with respect to a given Riemannian metric space is not guaranteed. A discrete mesh $\mathcal{H}$ of a domain $\Omega \subset \mathbb{R}^n$ is a **unit mesh** with respect to Riemannian metric space $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ if all its elements are quasi-unit. The definition of unity for an element is relaxed by taking into account technical constraints imposed by mesh generators. To avoid cycling while analyzing edges lengths, a tetrahedron $K$ defined by its list of edges $(\mathbf{e_i})_{i=1...6}$ is said to be quasi-unit if, $\forall i,\ \ell_{\mathcal{M}}(\mathbf{e_i}) \in [\frac{1}{\sqrt{2}}, \sqrt{2}]$. The study in [16] shows that several non-regular space filling tetrahedra verify this constraint, which guarantees the existence for constant Riemannian metric space. Unfortunately, this weaker constraint on edges lengths can lead to the generation of quasi-unit elements with a null volume. Consequently, controlling only the edges length is not sufficient, the volume must also be controlled to relax the notion of unit element which is practically achieved by managing a quality function.

*Generating Adapted Anisotropic Meshes* Using the previous notions, the problem of mesh generation can be considerably simplified. Indeed, whatever the kind of desired mesh (uniform, adapted isotropic, adapted anisotropic), the mesh generator will always generate a unit mesh in the prescribed Riemannian metric space [9]. Consequently, the generated mesh is uniform and isotropic in the Riemannian metric space while it is adapted and anisotropic in the Euclidean space. This idea has turned out to be a huge breakthrough in the generation of anisotropic adapted meshes.

## 3.2 *Practical Use of Metrics*

The main advantage when working with metric spaces is the well-posedness of operations on metric tensors which enable us to manage directional sizes. These operations have a straightforward geometric interpretation when considering the ellipsoid associated with a metric. In this section, the practical uses of metrics inside the mesh generator is described.

### 3.2.1 Metric Interpolation

In practice, the Riemannian metric space or metric field is only known discretely at mesh vertices. The definition of an interpolation procedure on metrics is therefore

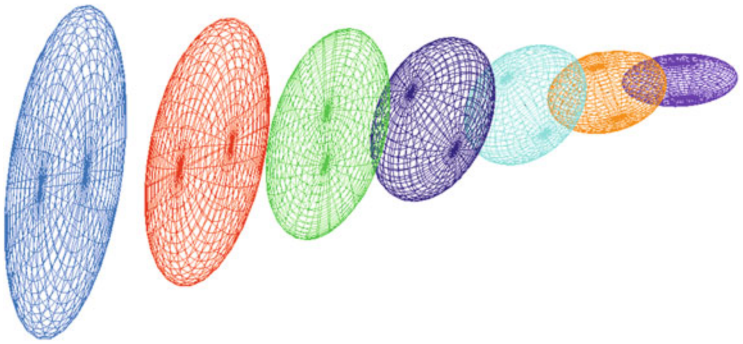


**Fig. 3** Metric interpolation along a segment where the endpoints metrics are the blue and violet ones

mandatory to be able to compute the metric at any point of the domain. Figure 3 illustrates metric interpolation along a segment, for which the initial data are the endpoints metrics. The proposed metric interpolation is based on the log-Euclidean framework introduced in [4].

*Log-Euclidean Framework* We first define the notion of metric logarithm and exponential:

$$\ln(\mathcal{M}) := {}^t\mathcal{R}\ln(\Lambda)\mathcal{R} \quad \text{and} \quad \exp(\mathcal{M}) := {}^t\mathcal{R}\exp(\Lambda)\mathcal{R}\,,$$

where $\ln(\Lambda) = \text{diag}(\ln(\lambda_i))$ and $\exp(\Lambda) = \text{diag}(\exp(\lambda_i))$. We can now define the *logarithmic addition* $\oplus$ and the *logarithmic scalar multiplication* $\odot$:

$$\mathcal{M}_1 \oplus \mathcal{M}_2 := \exp\left(\ln(\mathcal{M}_1) + \ln(\mathcal{M}_2)\right) \quad \text{and} \quad \alpha \odot \mathcal{M} := \exp\left(\alpha . \ln(\mathcal{M})\right) = \mathcal{M}^\alpha\,.$$

The logarithmic addition is commutative and coincides with matrix multiplication whenever the two tensors $\mathcal{M}_1$ and $\mathcal{M}_2$ commute in the matrix sense. The space of metric tensors, supplied with the logarithmic addition $\oplus$ and the logarithmic scalar multiplication $\odot$ is a vector space.

*Metric Interpolation in the Log-Euclidean Framework* Let $(\mathbf{x}_i)_{i=1\ldots k}$ be a set of vertices and $(\mathcal{M}(\mathbf{x}_i))_{i=1\ldots k}$ their associated metrics. Then, for a point $\mathbf{x}$ of the domain such that:

$$\mathbf{x} = \sum_{i=1}^{k} \alpha_i . \mathbf{x}_i \quad \text{with} \quad \sum_{i=1}^{k} \alpha_i = 1\,,$$

the interpolated metric is defined by:

$$\mathcal{M}(\mathbf{x}) = \bigoplus_{i=1}^{k} \alpha_i \odot \mathcal{M}(\mathbf{x}_i) = \exp\left(\sum_{i=1}^{k} \alpha_i \ \ln(\mathcal{M}(\mathbf{x}_i))\right). \tag{8}$$

This interpolation is commutative. Moreover, it has been demonstrated in [4] that this interpolation preserves the maximum principle, *i.e.,* for an edge $\mathbf{pq}$ with endpoints metrics $\mathcal{M}(\mathbf{p})$ and $\mathcal{M}(\mathbf{q})$ such that $\det(\mathcal{M}(\mathbf{p})) < \det(\mathcal{M}(\mathbf{q}))$ then we have $\det(\mathcal{M}(\mathbf{p})) < \det(\mathcal{M}(\mathbf{p} + t\,\mathbf{pq})) < \det(\mathcal{M}(\mathbf{q}))$ for all $t \in [0, 1]$.

*Remark 1* The interpolation formulation (8) reduces to $\mathcal{M}(\mathbf{x}) = \prod_{i=1}^{k} \mathcal{M}(\mathbf{x}_i)^{\alpha_i}$, if all the metrics commute. Therefore, an arithmetic mean in the log-Euclidean framework could be interpreted as a geometric mean in the space of metric tensors.

### 3.2.2 Computation of Geometric Quantities in Riemannian Metric Space

In this section, we describe how geometric quantities are computed numerically in Riemannian metric space as such quantities involve integrals. Let $\mathcal{M}$ be a discrete metric field defined at the vertices of a mesh $\mathcal{H}$ of a domain $\Omega_h$. Thanks to the interpolation operation, we have a continuous metric field in the whole domain, *i.e.,* a Riemannian metric space $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega_h}$. This representation of the metric field depends on $\mathcal{H}$ as the interpolation law is applied at the element level.

*Computation of Edge Length* Approximation can be used to evaluate edge length in Riemannian metric space given by Relation (6). However, an analytical expression can be obtained if we consider that the metric field conforms to a geometric variation law as described in Sect. 3.2.1.

Let $\mathbf{e} = \mathbf{p_1 p_2}$ be an edge of the mesh of Euclidean length $\|\mathbf{e}\|_2$, and $\mathcal{M}(\mathbf{p}_1)$ and $\mathcal{M}(\mathbf{p}_2)$ be the metrics at the edge extremities $\mathbf{p}_1$ and $\mathbf{p}_2$. We denote by $\ell_{\mathcal{M}_i}(\mathbf{e}) = \sqrt{{}^t\mathbf{e}\mathcal{M}(\mathbf{p}_i)\mathbf{e}}$ the length of the edge in metric $\mathcal{M}(\mathbf{p}_i)$. We assume $\ell_{\mathcal{M}_1}(\mathbf{e}) > \ell_{\mathcal{M}_2}(\mathbf{e})$ and we set $a = \dfrac{\ell_{\mathcal{M}_1}(\mathbf{e})}{\ell_{\mathcal{M}_2}(\mathbf{e})}$. The restriction of the (multi-dimensional) metric interpolation operator given by Relation (8) to an edge $\mathbf{e} = \mathbf{p_1 p_2}$ leads to a geometric interpolation law on eigen values $\lambda$ and size $h$:

$$\lambda(t) = \exp\left((1-t)\ln(\lambda_1) + t\ln(\lambda_2)\right) = \lambda_1^{(1-t)}\lambda_2^{t} \quad \Longleftrightarrow \quad h(t) = h_1^{(1-t)}\,h_2^{t}.$$

Under these assumption, we deduce [1]:

$$\ell_{\mathcal{M}}(\mathbf{e}) = \ell_{\mathcal{M}_1}(\mathbf{e})\frac{a-1}{a\ln(a)}. \tag{9}$$

*Computation of Element Volume* The evaluation of a tetrahedron volume in a Riemannian metric space consists in computing numerically Integral (7). Higher order approximation can be obtained by using Gaussian quadrature and metric interpolation based on the Log-Euclidean framework. For instance, if one considers a $k$-point Gaussian quadrature with weights $(\omega_j)_{j=1\ldots k}$ and barycentric coefficients $(\beta_j^1, \beta_j^2, \beta_j^3, \beta_j^4)_{j=1\ldots k}$, it yields:

$$|K|_{\mathcal{M}} \approx |K|_{\mathcal{I}_3} \sum_{j=1}^{k} \omega_j \sqrt{\det\Big(\exp\Big(\sum_{i=1}^{4} \beta_j^i \log(\mathcal{M}_i)\Big)\Big)}\,.$$

However, for faster volume evaluation, only a first order approximation is considered in this work:

$$|K|_{\mathcal{M}} \approx \sqrt{\det \mathcal{M}_K}\, |K|_{\mathcal{I}_3} = |K|_{\mathcal{M}_K} \quad \text{where } \mathcal{M}_K = \exp\Big(\frac{1}{4}\sum_{i=1}^{4} \log\big(\mathcal{M}_{\mathbf{p}_i}\big)\Big) \tag{10}$$

and $\mathbf{p}_i$ are the four vertices of tetrahedron $K$.

### 3.2.3 Metric-Based Quality Function

In this section, we propose the generalization of two quality functions to metric spaces.

*Minimum Edge Weight Function* This quality function corresponds to the weight [5] for an element edge shared by two faces:

$$Q_{\mathcal{M}}(K) = \min_{i=1..6} \frac{\|\mathbf{e}_i\|_{\mathcal{M}_K}}{\tan \theta^i_{\mathcal{M}_K}} = \max_{i=1..6} \frac{\mathbf{n}_{F_{i,1}} \cdot_{\mathcal{M}_K} \mathbf{n}_{F_{i,2}}}{6|K|_{\mathcal{M}_K}}\,,$$

where $\theta^i_{\mathcal{M}}$ is the dihedral angle between two faces $F_{i,1}$ and $F_{i,2}$ sharing edge $\mathbf{e}_i$. It represents the geometric contribution to the value of the diagonal terms in a solution matrix for an elliptic equation. Consequently, it represents a quantity that can degrade performance. This is directly analogous to minimizing the maximum angle in 2D and similar to that in 3D. This quality function can be rewritten:

$$Q_{\mathcal{M}}(K) = \max_{i=1..6} \frac{({}^t\mathbf{e}_i \cdot_{\mathcal{M}_K} \mathbf{e}_j)({}^t\mathbf{e}_i \cdot_{\mathcal{M}_K} \mathbf{e}_k) - \ell^2_{\mathcal{M}_K}(\mathbf{e}_i)({}^t\mathbf{e}_j \cdot_{\mathcal{M}_K} \mathbf{e}_k)}{6|K|_{\mathcal{M}_K}}\,.$$

*Tetrahedron Dihedral Angle Function* The maximum value corresponds to the function for the edge with the minimum dihedral angle between the two faces that share the edge. The minimum value corresponds to the function for the edge with the maximum dihedral angle between the two faces that share the edge:

$$Q_{\mathscr{M}}(\mathbf{e}_i) = \frac{\cos\theta^i_{\mathscr{M}}}{|\cos(\theta^i_{\mathscr{M}})|} = \frac{(\mathbf{n}_{F_{i,1}} \cdot_{\mathscr{M}} \mathbf{n}_{F_{i,2}})|\mathbf{n}_{F_{i,1}} \cdot_{\mathscr{M}} \mathbf{n}_{F_{i,2}}|}{16|F_{i,1}|^2_{\mathscr{M}}|F_{i,2}|^2_{\mathscr{M}}}.$$

The maximum element angle is useful as a basic measure of quality for a given element. Particularly at extremes. Large dihedral angles always correspond to low quality elements as defined by any quality criteria.

## 3.3 Metric-Based Anisotropic Mesh Transition

This section presents the new blended approach that smoothly blends the near-body pseudo-structured region into a generalized anisotropic field region. The unified approach taken here utilizes the best characteristics of both near body BL with pseudo-structured elements and field region unstructured mesh with metric-based anisotropic element.

First, the BL region mesh is generated using the advancing-normal BL mesh generation algorithm. Second, the unstructured anisotropic mesh of the outer field region is generated utilizing the advancing-front based/local-reconnection based approach. Here, the metric field does not come from an error estimate as in mesh adaptation but, instead, it is derived on the fly by interpolation when each new vertex is inserted into the volume mesh. The initial metric values are specified to be isotropic at all surface points except at the interface between the BL and outer tetrahedral region. That metric is determined from the normal and tangential spacing at the edge of the BL region as shown in Fig. 4.

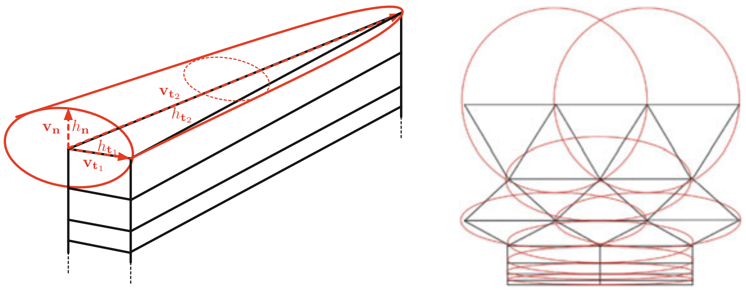

$\mathbf{v}_{\mathbf{n}}$
$h_{\mathbf{n}}$
$h_{\mathbf{t}_1}$
$\mathbf{v}_{\mathbf{t}_1}$
$\mathbf{v}_{\mathbf{t}_2}$
$h_{\mathbf{t}_2}$

**Fig. 4** *Left*: metric associated with boundary layer prismatic elements. *Right*: metric-based anisotropic mesh transition

Two different methods have been tested. For the first one, called metric-based transition based on metric interpolation, each time a new vertex $\mathbf{x}$ is inserted in tetrahedron $K$ its metric is simply interpolated from the metric of the vertices of $K$:

$$\mathcal{M}(\mathbf{x}) = \frac{\exp\left(\sum_{p_i \in K} \omega_i \log\left(\mathcal{M}(\mathbf{p}_i)\right)\right)}{\sum_{p_i \in K} \omega_i},$$

where the weight are chosen to be $\omega_i$ the barycentrics of point $\mathbf{x}$ with respect to $K$, *i.e.,* the classical FEM basis function coefficient.

For the second method, when a new vertex $\mathbf{x}$ is inserted in tetrahedron $K$, its metric is computed from growth metric issued from vertices of $K$. For each vertex $\mathbf{p}$ of tetrahedron $K$ supplied with a metric $\mathcal{M}(\mathbf{p})$, the growth metric issued from $\mathbf{p}$ at position $\mathbf{x}$, denoted $\mathcal{M}_{\mathbf{p}}(\mathbf{x})$, is computed. To this end, the metric growth law proposed in [1] which is homogeneous in the physical space is chosen. This means that the eigenvalues are growing separately and differently, the shape of the metric is not more preserved. This law gradually makes the metric more and more isotropic as it gradually propagates in the domain. The growth factor associated independently with each eigenvalue of $\mathcal{M}$ is given by:

$$\eta_i^2(\mathbf{px}) = \left(1 + \sqrt{\lambda_i}\,\|\mathbf{px}\|_2 \ln(\beta)\right)^{-2} \quad \text{for } i = 1, \ldots, 3,$$

where $\beta$ is the specified growth rate factor. The growth metric at $\mathbf{x}$ is given by:

$$\mathcal{M}_{\mathbf{p}}(\mathbf{x}) = {}^t\mathcal{R}\mathcal{N}(\mathbf{px})\Lambda\mathcal{R} \quad \text{where} \quad \mathcal{N}(\mathbf{px}) = \begin{pmatrix} \eta_1^2(\mathbf{px}) & 0 & 0 \\ 0 & \eta_2^2(\mathbf{px}) & 0 \\ 0 & 0 & \eta_3^2(\mathbf{px}) \end{pmatrix}. \tag{11}$$

Finally, the metric at new vertex $\mathbf{x}$ is obtained by a weighted interpolation of all the vertices growth metrics of tetrahedron $K$:

$$\mathcal{M}(\mathbf{x}) = \frac{\exp\left(\sum_{p_i \in K} \omega_i \log\left(\mathcal{M}_{\mathbf{p}_i}(\mathbf{x})\right)\right)}{\sum_{p_i \in K} \omega_i},$$

where $\omega_i$ are again the barycentrics of point $\mathbf{x}$ with respect to $K$. We will refer to this method as metric-based transition with growth.

## 4 Numerical Comparison

Numerical results are presented here to demonstrate the basic characteristics of approach described in this work. These cases where chosen to best illustrate the properties in a graphical context. However, their resolution is relatively coarse (for ease of graphical presentation) in comparison to what is more typical in realistic aerospace configurations. In each case, an anisotropic pseudo-structured surface mesh is used and three distinctly different volume meshes were generated using the same BL region mesh characteristics. The volume meshes were generated using pentahedral BL elements to facilitate the graphic process. With hexahedra elements the number of BL elements are reduced by up to a factor of two. Of the three different volume meshes, one is generated uses a pure isotropic approach with an immediate transition from the BL region to the outer tetrahedral region. The other two use metric-based anisotropic blending described in Sect. 3.3. Of those one uses standard metric interpolation and the other uses metric-based transition with growth.

## *4.1 Wing-Body*

The first test case is a wing-body profile with relatively high-aspect ratio quad faces on the wing surface. Length scale transition between anisotropic and isotropic faces has been selected with relatively high growth to amplify the effect of the blending. Geometry and the initial surface mesh are depicted in Fig. 5 for this configuration. The surface mesh is composed of 19,724 triangles and 1,170 quads.

The results for a purely isotropic approach are depicted in Fig. 6. As shown, mesh quality suffers in the transition region. For this case, this is largely driven by the anisotropic surface mesh and there is simply no way to gracefully transition immediately from a high-aspect ratio face to an isotropic tet element.

For the case with metric interpolation the results are depicted in Fig. 7. As shown the transition now is blended. And the mesh quality is improved from a length scale and element volume perspective. However, the transition is very slow and the mesh density maybe inappropriate for typical applications. Compared to the no blending case, the number of vertices has increased by a factor two and the number of tetrahedra in the outer region are nearly ten times more (Table 1). Finally, the results for the case using metric-based transition with growth are depicted in Fig. 8. As with metric interpolation, the transition is blended and the mesh quality is again

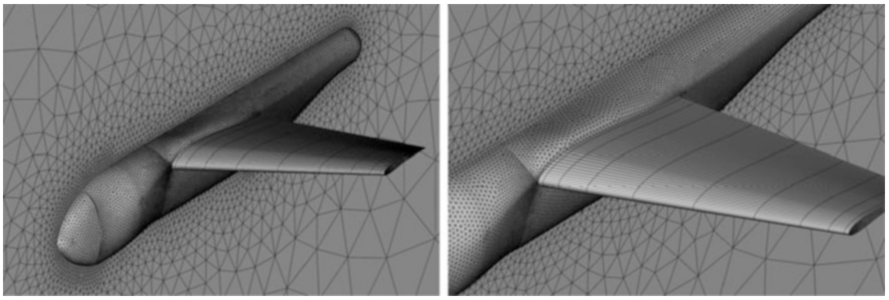


**Fig. 5** Wing-body hybrid surface mesh with high-aspect ratio quad faces on the wing

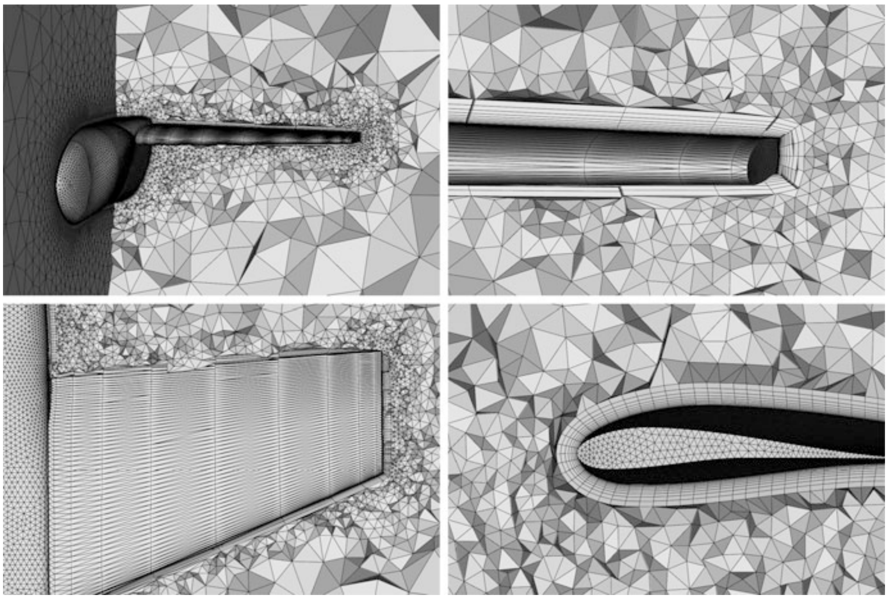


**Fig. 6** Wing-body meshes obtained with no anisotropic blending

**Table 1** Wing-body mesh size statistics

| Case | # Vertices | # Tets | # Prisms | # Pyramids |
|---|---|---|---|---|
| No blending | 377,563 | 313, 766 | 625,859 | 389 |
| Interpolation | 838,477 | 3, 049, 515 | 625,859 | 389 |
| Growth | 469,390 | 859, 133 | 625,859 | 389 |

improved. However, growth has significantly improved the rate of transition and provided a more reasonable mesh density for the given configuration. Compared to the no blending case, the number of vertices has increased by a factor 1.25 and the number of tetrahedra in the outer region are 2.75 times more. By varying the parameter in the growth function, density can be increased. Here, it has been set to 1.01. Increasing it to 1.05 reduces the number of tetrahedra by a factor 2.75. And, reducing it to 1.0025 increases the number of tetrahedra by a factor two.

## *4.2 Nacelle*

The second test case is a nacelle configuration with a predominantly structured quad faces on the surface. Geometry and the initial surface mesh are depicted in Fig. 9 for this configuration. The surface mesh is composed of 14,192 triangles and 24,640 quads.

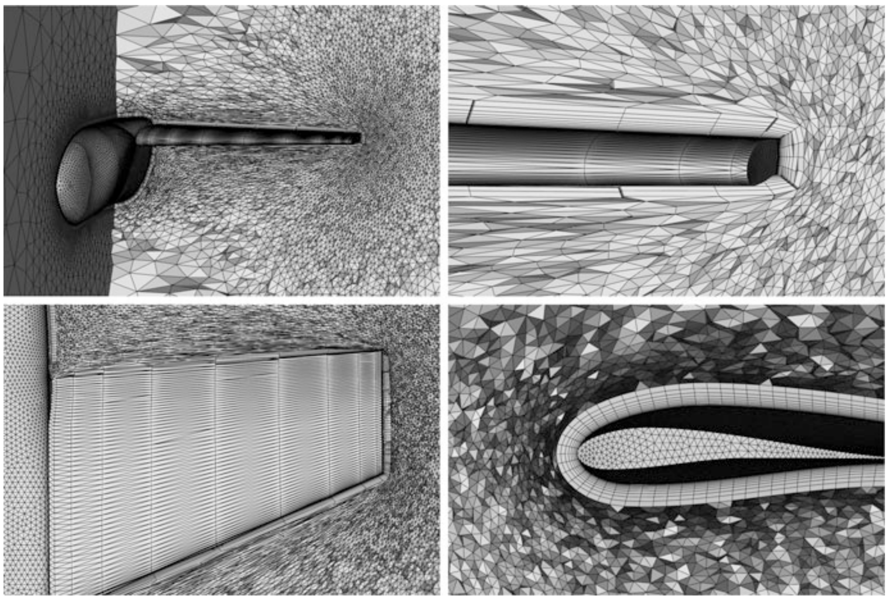


**Fig. 7** Wing-body meshes obtained with metric-based transition based on metric interpolation

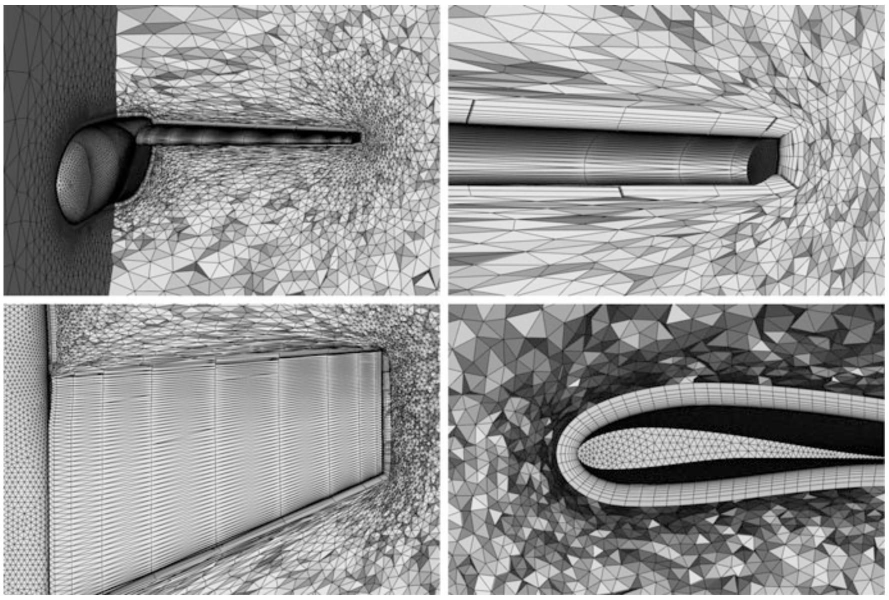


**Fig. 8** Wing-body meshes obtained with metric-based transition with growth

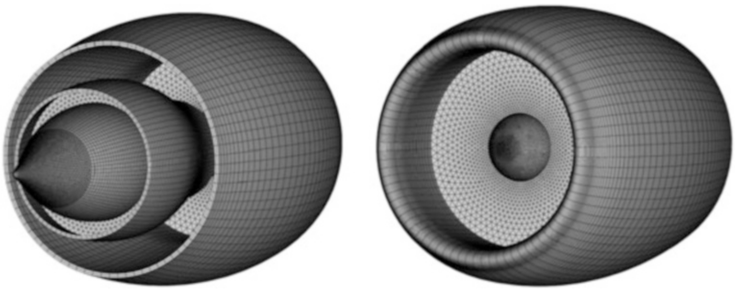


**Fig. 9** Nacelle hybrid surface mesh with high-aspect ratio quad element

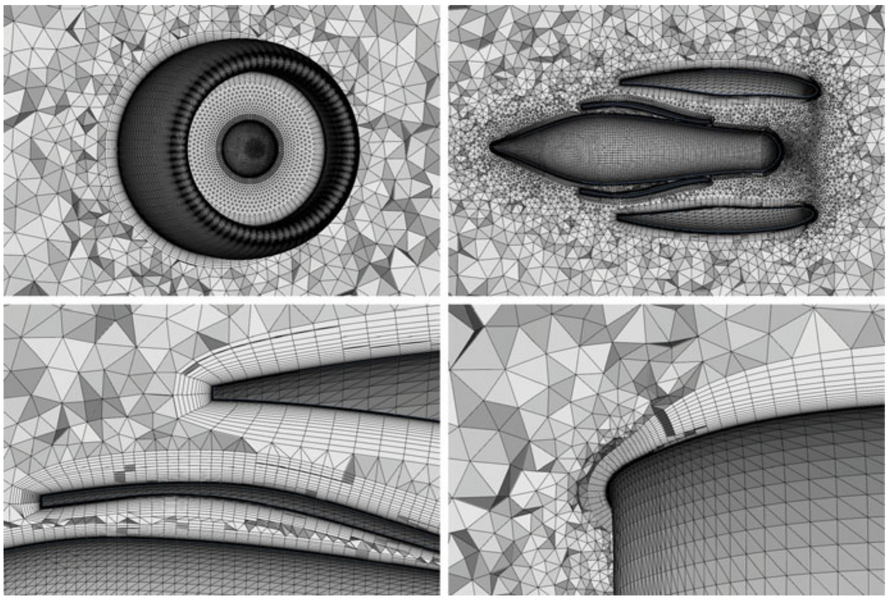


**Fig. 10** Nacelle meshes obtained with no anisotropic blending

The results for the pure isotropic approach with an immediate transition from the BL region to the outer tetrahedral region are presented in Fig. 10. As before, mesh quality suffers in the transition region. For this case, this is driven by both the anisotropic surface mesh at the leading edge and the normal spacing at the edge of the BL region. Graceful and immediate transition from BL elements with anisotropy in both normal and tangential directions to the isotropic outer elements is not possible.

For the case with metric interpolation the results are depicted in Fig. 11. As shown the transition now is blended. And the mesh quality is improved from a length scale and element volume perspective. However, the transition is again very slow and the mesh density maybe inappropriate for typical applications. Compared to

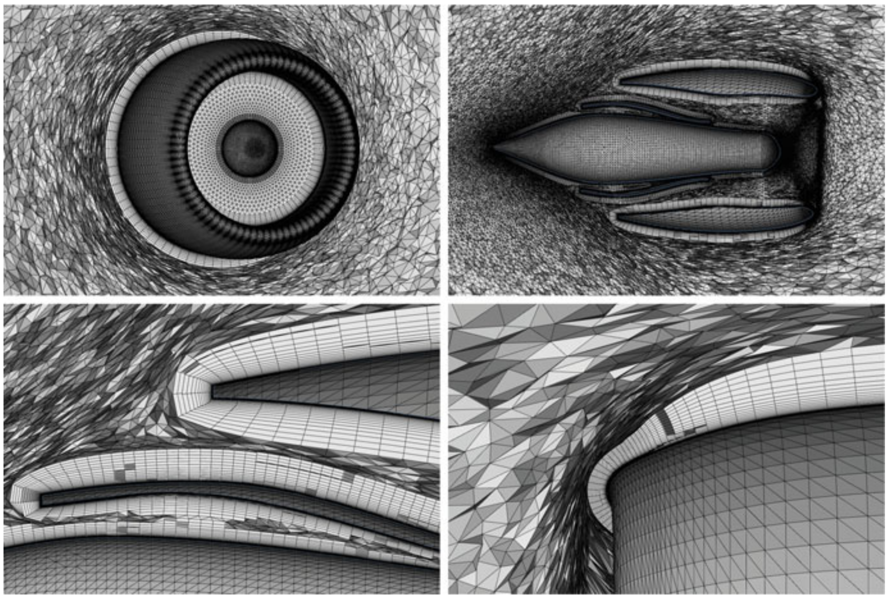


**Fig. 11** Nacelle meshes obtained with metric-based transition based on metric interpolation

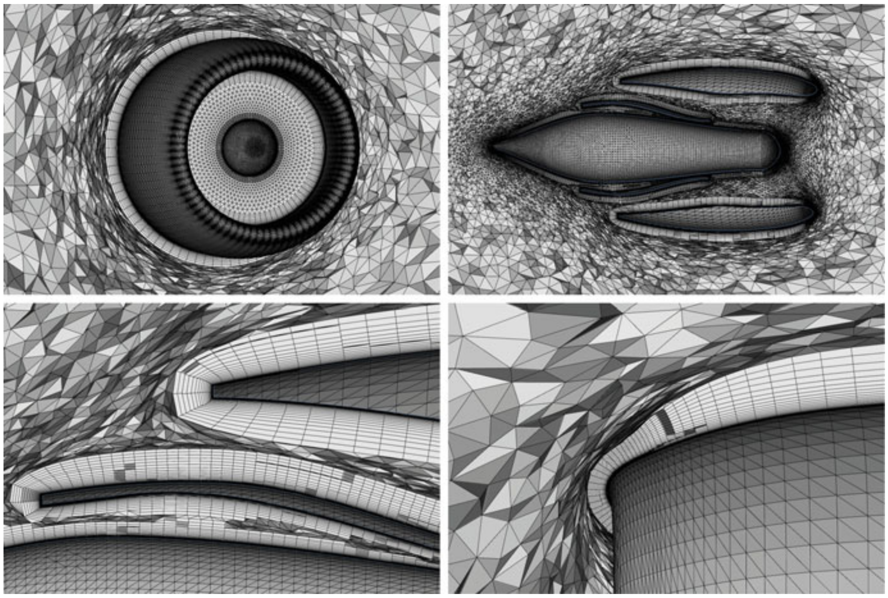


**Fig. 12** Nacelle meshes obtained with metric-based transition with growth

**Table 2** Nacelle mesh size statistics

| Case | # Vertices | # Tets | # Prisms | # Pyramids |
|---|---|---|---|---|
| No blending | 1,591,712 | 875, 740 | 2,855,180 | 4,356 |
| Interpolation | 2,569,867 | 6, 715, 561 | 2,855,180 | 4,356 |
| Growth | 1,729,584 | 1, 702, 926 | 2,855,180 | 4,356 |

the no blending case, the number of vertices has increased by a factor 1.6 and the number of tetrahedra in the outer region are nearly eight times more (Table 2).

The results for the case using metric-based transition with growth are depicted in Fig. 12. As with metric interpolation, the transition is blended and the mesh quality is again improved. Also, the metric-based transition is accounting for both the normal and tangential anisotropy. However, growth has once again significantly improved the rate of transition and provided a more reasonable mesh density for the given configuration. Compared to the no blending case, the number of vertices has increased by a nominal amount and the number of tetrahedra by a factor two (Table 2).

## 5 Conclusions

A metric-based approach for smooth transition between BL region and tetrahedral outer region has been proposed. This unified method utilizes the best characteristics of both near body BL and generalized metric-based approaches. Metric-based formulations for quality functions and other geometric quantities require for mesh generation have been presented. Results point out that the metric-based transition can be used to improve mesh quality and density for configurations with anisotropic surface meshes and BL regions that do not reach outer region length scale.

Further work is required to demonstrate this approach combined with an adaptive approach to resolve off-body or field features.

## References

1. Alauzet, F.: Size gradation control of anisotropic meshes. Finite Elem. Anal. Des. **46**, 181–202 (2010)
2. Alauzet, F., Loseille, A.: High order sonic boom modeling by adaptive methods. J. Comput. Phys. **229**, 561–593 (2010)
3. Alauzet, F., Frey, P.J., George, P.L., Mohammadi, B.: 3D transient fixed point mesh adaptation for time-dependent problems: Application to CFD simulations. J. Comput. Phys. **222**, 592–623 (2007)
4. Arsigny, V., Fillard, P., Pennec, X., Ayache, N.: Log-Euclidean metrics for fast and simple calculus on diffusion tensors. Magn. Reson. Med. **56**(2), 411–421 (2006)

5. Barth, T.J.: Steiner triangulation for isotropic and stretched elements. In: 33th AIAA Aerospace Sciences Meeting (January 1995)
6. Bottasso, C.L.: Anisotropic mesh adaption by metric-driven optimization. Int. J. Numer. Methods Eng. **60**, 597–639 (2004)
7. Dobrzynski, C., Frey, P.J.: Anisotropic Delaunay mesh adaptation for unsteady simulations. In: Proceedings of the 17th International Meshing Roundtable, pp. 177–194. Springer, Berlin Heidelberg (2008)
8. Frey, P.J., George, P.L.: Mesh Generation. Application to Finite Elements, 2nd edn. ISTE, London (2008)
9. George, P.L., Hecht, F., Vallet, M.G.: Creation of internal points in Voronoi's type method. Control and adaptation. Adv. Eng. Software **13**(5–6), 303–312 (1991)
10. Gruau, C., Coupez, T.: 3D tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric. Comput. Methods Appl. Mech. Eng. **194**(48–49), 4951–4976 (2005)
11. Ito, Y., Nakahashi, K.: Unstructured mesh generation for viscous flow computations. In: Proceedings of the 11th International Meshing Roundtable, pp. 367–377 (2002)
12. Jones, W.T., Nielsen, E.J., Park, M.A.: Validation of 3D adjoint based error estimation and mesh adaptation for sonic boom reduction. In: 44th AIAA Aerospace Sciences Meeting and Exhibit, AIAA-2006-1150, Reno (January 2006)
13. Li, X., Shephard, M.S., Beal, M.W.: 3D anisotropic mesh adaptation by mesh modification. Comput. Methods Appl. Mech. Eng. **194**(48-49), 4915–4950 (2005)
14. Löhner, R.: Generation of unstructured grids suitable for RANS calculations. In: 37th AIAA Aerospace Sciences Meeting (January 1999)
15. Loseille, A., Alauzet, F.: Optimal 3D highly anisotropic mesh adaptation based on the continuous mesh framework. In: Proceedings of the 18th International Meshing Roundtable, pp. 575–594. Springer, Berlin Heidelberg (2009)
16. Loseille, A., Alauzet, F.: Continuous mesh framework. Part I: Well-posed continuous interpolation error. SIAM J. Numer. Anal. **49**(1), 38–60 (2011)
17. Loseille, A., Löhner, R.: Adaptive anisotropic simulations in aerodynamics. In: 48th AIAA Aerospace Sciences Meeting, AIAA Paper 2010-169, Orlando (January 2010)
18. Loseille, A., Löhner, R.: Boundary layer mesh generation and adaptivity. In: 49th AIAA Aerospace Sciences Meeting, AIAA Paper 2011-894, Orlando (January 2011)
19. Marcum, D.L.: Unstructured grid generation using automatic point insertion and local reconnection. In: Thompson, J.F., Soni, B., Weatherill, N.P. (eds.) The Handbook of Grid Generation, Chap. 18, pp. 1–31. CRC, Boca Raton (1998)
20. Marcum, D.L., Gaither, J.A.: Mixed element type unstructured grid generation for viscous flow applications. In: 14th AIAA Computational Fluid Dynamics Conference (June 1999)
21. Marcum, D.L., Weatherill, N.P.: Unstructured grid generation using iterative point insertion and local reconnection. AIAA J. **33**(9), 1619–1625 (1995)
22. Michal, T.: The benefits of a smooth transition in viscous grids generation for Boeing CFD applications. Private Communication, The Boeing Company (November 2013)
23. Michal, T., Krakos, J.: Anisotropic mesh adaptation through edge primitive operations. In: 50th AIAA Aerospace Sciences Meeting (January 2012)
24. Pirzadeh, S.: Viscous unstructured three dimensional grids by the advancing-layers method. In: 32th AIAA Aerospace Sciences Meeting (January 1994)

# On the Generation of Curvilinear Meshes Through Subdivision of Isoparametric Elements

**David Moxey, Mashy D. Green, Spencer J. Sherwin, and Joaquim Peiró**

**Abstract** Recently, a new mesh generation technique based on the isoparametric representation of curvilinear elements has been developed in order to address the issue of generating high-order meshes with highly stretched elements. Given a valid coarse mesh comprising of a prismatic boundary layer, this technique uses the shape functions that define the geometries of the elements to produce a series of subdivided elements of arbitrary height. The purpose of this article is to investigate the range of conditions under which the resulting meshes are valid, and additionally to consider the application of this method to different element types. We consider the subdivision strategies that can be achieved with this technique and apply it to the generation of meshes suitable for boundary-layer fluid problems.

## 1 Introduction

In recent years, interest in high-order finite element methods has increased dramatically. Their attractive dispersion properties, exponential convergence of approximate solutions and computational performance when compared to traditional low-order methods make high-order methods an attractive prospect for researchers in both academia and industry across a wide range of application areas. However, one of the main issues to be overcome before these methods can be widely adopted is the development of methods for reliable generation of curvilinear meshes for complex three-dimensional domains.

A particularly important problem to be solved is the generation of meshes where highly stretched elements are desired, either as function of the geometry or for reasons of computational cost. In a typical high-order generation process one begins by generating a linear mesh, and then deforming the elements connected to curved surfaces by projecting points lying on the surface geometry into the interior of each face. When the linear elements are highly stretched so that their thickness

D. Moxey • M.D. Green • S. J. Sherwin • J. Peiró (✉)
Department of Aeronautics, Imperial College London, London SW7 2AZ, UK
e-mail: d.moxey@imperial.ac.uk; mdh10@imperial.ac.uk; s.sherwin@imperial.ac.uk; j.peiro@imperial.ac.uk

S. Perotto, L. Formaggia (eds.), *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, SEMA SIMAI Springer Series 5,
DOI 10.1007/978-3-319-06053-8_10

is small, then in regions where the curvature of the geometry is high, deforming only the elements connected to the surface can result in self-intersection and thus the mesh becomes unsuitable for computations [5]. Whilst techniques exist to deform linearly refined meshes using linear or non-linear elastic analogies [4, 7] or alternatively untangle and optimise meshes that have self-intersecting elements [6], they are relatively expensive and their success has been limited when applied to these problems.

One solution to this problem that has been recently proposed by the authors of this work is to consider an isoparametric approach to producing highly stretched meshes [3]. Given an existing valid high-order mesh of prismatic elements, in this technique one subdivides each prism by using the shape functions defining the geometry of the original prismatic element. This method is simple, cheap to implement and leads to the generation of meshes that are guaranteed to be valid so long as the original mapping is valid. Furthermore, this subdivision technique can be adapted to address other issues, such as the generation of meshes containing only high-order simplex elements for solvers which do not support hybrid meshes.

The purpose of this paper is to frame the subdivision technique in the context of a more general mathematical framework and demonstrate how it can be utilised to subdivide a broader range of elemental types in both two and three dimensions. We note that in general, the subdivision of elements in this manner often requires the enrichment of the polynomial space so that the subdivided elements capture all curvature of the original element. One of aims of this paper therefore is to establish the necessary conditions for the validity of the resulting subelements under various subdivision strategies, demonstrate through numerical examples the applicability of the method to mesh generation and that such conditions are indeed required.

The paper is structured as follows. Section 2 outlines the motivation for the subdivision technique and gives a brief overview of the process through which an element is subdivided as presented in [3]. The mathematical framework for a generalisation of the method to other element types is given in Sect. 3. We then demonstrate some applications of the method in Sect. 4 to problems in aeronautics and biomechanics, and the subdivision of elements to produce meshes containing only simplex elements. Finally we conclude with some remarks on further applications and improvements in Sect. 5.

## 2 Motivation

One of the main application areas of high-order methods is the simulation of fluid flow over aeronautical geometries where, near walls, flow gradients in the direction normal to the wall are several orders of magnitude larger than those tangential to the wall. Sufficient resolution in the near-wall boundary layer in order to resolve this shear is crucial since the vortices which lead to turbulent instabilities develop close to this region, and so under-resolution will usually lead to unphysical or biased results. In these simulations therefore, the size of elements in wall-normal directions must be small so that the steep gradient of the near-wall flow profile is adequately resolved. In the other directions however, such resolution is not usually required, leading to the generation of elements with a large stretching ratio. Introducing curvature into elements near the boundary layer will often lead to self-intersection in regions of high curvature.

The isoparametric subdivision technique proposed in [3] addresses this problem. Firstly, we assume that a coarse mesh, comprising of a prismatic boundary layer and tetrahedra elsewhere, has been generated. As part of the usual high-order mesh generation procedure, we construct a mapping $\chi$ which maps coordinates $\xi$ in a reference element into the Cartesian coordinates of $\Omega$. In order to produce a series of refined prismatic elements in the physical domain, we instead refine the standard elemental region, and utilise the mapping $\chi$ to map this back into physical space.

An overview of this process can be seen for a representative quadrilateral element in Fig. 1, where we assume the bottom edge of the element is attached to the wall, and therefore require extra resolution in the vertical direction. The top row shows how the standard quadrilateral element $\Omega_{\mathrm{st}}$ is deformed under the mapping $\chi$ to produce a curved element $\Omega$. To refine the element in physical space, we first split $\Omega_{\mathrm{st}}$ into a series of smaller elements as shown in bottom left of the figure. Applying the mapping $\chi$ to these subelements of the standard region leads to the production of curved subelements of the physical element as desired.

What remains to be presented, and indeed is the focus of the rest of this paper, is an examination of the conditions under which the subdivision process produces valid elements, not only for the refinement procedure outlined here, but for more generic transformations of the standard element. In the following section we

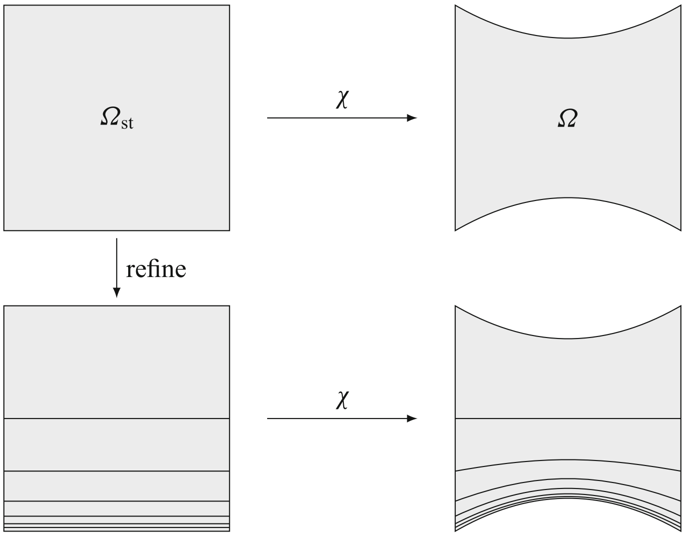


**Fig. 1** Overview of boundary layer refinement technique presented in [3]

describe the mathematical framework in which this problem is defined, and describe more precisely how a subdivision procedure impacts the polynomial spaces which define the elements.

## 3 Mathematical Framework

We begin by providing a brief mathematical framework for the method. Let $\Omega^e$ denote an element, which in general belongs to a mesh arising from the tessellation $\mathscr{T}(\Theta) = \{\Omega^1, \ldots, \Omega^{N_{\rm el}}\}$ of some domain $\Theta \subset \mathbb{R}^n$ with $n = 2, 3$ of $N_{\rm el}$ elements, so that

$$\Theta = \bigcup_{e=1}^{N_{\rm el}} \Omega^e, \qquad \Omega^e \cap \Omega^f = \emptyset \text{ if } e \neq f.$$

In two dimensions, we consider quadrilateral and triangular shaped elements, and in three dimensions tetrahedral, prismatic and hexahedral elements. In order to introduce curvature into an element $\Omega$ (where we drop the superscript $e$ for convenience), we assume there exists a mapping $\chi : \Omega_{\rm st} \to \Omega$ which projects a canonical standard element $\Omega_{\rm st}$ into the Cartesian coordinates defining an element. In this work we define reference elements to be

$$\begin{aligned}
\Omega_{\rm st}^{\rm quad} &= \{(\xi_1, \xi_2) \quad\ \mid -1 \le \xi_1, \xi_2 \le 1\},\\
\Omega_{\rm st}^{\rm tri} &= \{(\xi_1, \xi_2) \quad\ \mid -1 \le \xi_1 + \xi_2 \le 1\},\\
\Omega_{\rm st}^{\rm hex} &= \{(\xi_1, \xi_2, \xi_3) \mid -1 \le \xi_1, \xi_2, \xi_3 \le 1\},\\
\Omega_{\rm st}^{\rm pri} &= \{(\xi_1, \xi_2, \xi_3) \mid -1 \le \xi_1 + \xi_3 \le 1, -1 \le \xi_2 \le 1\},\\
\Omega_{\rm st}^{\rm tet} &= \{(\xi_1, \xi_2, \xi_3) \mid -1 \le \xi_1 + \xi_2 + \xi_3 \le 1\},
\end{aligned}$$

respectively. Inside the standard elements we define a polynomial space in terms of the reference coordinates $\xi = (\xi_1, \xi_2, \xi_3)$ from which an expansion basis is selected. Assuming that we select a polynomial order $P$, $Q$ and $R$ for each coordinate direction, the polynomial spaces take the form

$$\mathscr{P}(\Omega_{\rm st}) = \text{span}\{\xi_1^p \xi_2^q \xi_3^r \mid (pqr) \in \mathscr{I}\}$$

where $\mathscr{I}$ represents an indexing set, defined for each element as

$$\begin{aligned}
\mathscr{I}^{\rm quad} &= \{(pqr) \mid 0 \le p \le P,\ 0 \le q \le Q,\ r = 0\}\\
\mathscr{I}^{\rm tri} &= \{(pqr) \mid 0 \le p \le P,\ 0 \le p + q \le Q,\ r = 0,\ P \le Q\}
\end{aligned}$$

$$\begin{aligned}
\mathscr{I}^{\text{hex}} &= \{(pqr) \mid 0 \le p \le P,\ 0 \le q \le Q,\ 0 \le r \le R\} \\
\mathscr{I}^{\text{pri}} &= \{(pqr) \mid 0 \le p \le P,\ 0 \le q \le Q,\ 0 \le p+r \le P,\ P \le R\} \\
\mathscr{I}^{\text{tet}} &= \{(pqr) \mid 0 \le p \le P,\ 0 \le p+q \le Q,\ 0 \le p+q+r \le R,\ P \le Q \le R\}.
\end{aligned}$$

In order to preserve the positivity of discretised spatial operators, we insist that given the components of $\chi = (\chi_1, \ldots, \chi_n)$ the determinant of the Jacobian matrix

$$[J_\chi(\xi)]_{ij} = \frac{\partial \chi_i(\xi)}{\partial \xi_j}, \qquad i, j = 1, \ldots, n$$

is positive for all $\xi \in \Omega_{\text{st}}$, so that $\chi$ preserves orientation and is invertible. Furthermore we consider an isoparametric representation of $\chi$ in terms of a set of shape functions $\phi_{pqr}$, so that

$$\chi_i(\xi) = \sum_{(pqr)\in\mathscr{I}} (\hat{\chi}_i)_{pqr}\phi_{pqr}(\xi).$$

In the numerical demonstrations below we consider an expansion in terms of a tensor product of modified hierarchical modal functions which permits a boundary-interior decomposition [2]. We note however that in this setting the choice of shape function is relatively unimportant, so long as they span the polynomial space of the element. However, as we will demonstrate later, this choice of basis is useful for certain types of elemental subdivisions as it permits fewer restrictions on the resulting subelement polynomial spaces.

### *3.1 Subdivision into the Same Element Type*

In this section we demonstrate how the isoparametric mapping $\chi$, which we assume has positive Jacobian for all $\xi \in \Omega_{\text{st}}$, can be used to subdivide an element into smaller elements of the same type. The goal of the subdivision process is to obtain a mapping $\zeta : \Omega_{\text{st}} \to \tilde{\Omega}$ where $\tilde{\Omega} \subset \Omega$ and $\det J_\zeta(\xi) > 0$ for all $\xi \in \Omega_{\text{st}}$.

In the isoparametric approach we adopt here, instead of attempting to determine the exact subdomain $\tilde{\Omega}$ of the physical element $\Omega$, we select a subdomain of the standard region, $\widetilde{\Omega_{\text{st}}}$, and construct an invertible mapping $f : \Omega_{\text{st}} \to \widetilde{\Omega_{\text{st}}}$ with $\det J_f(\xi) > 0$. Initially, we also assume that the polynomial expansion in each direction is equal so that $P = Q = R$. Setting $\zeta$ as the composition $\chi \circ f$ we then obtain a subelement $\tilde{\Omega} = \zeta(\Omega_{\text{st}})$.

The justification for the validity of $\zeta$, and moreover the resulting element $\tilde{\Omega}$ under the restriction of equal polynomial order is as follows. Firstly, it is clear that the determinant of the Jacobian of $\zeta$ is positive for any $\xi \in \Omega_{\mathrm{st}}$, since through an application of the chain rule we have that

$$\det J_\zeta(\xi) = \det J_\chi(f(\xi)) \det J_f(\xi) > 0. \tag{1}$$

Let us assume that each component of $\chi$ lies in the polynomial space $\mathscr{P}(\Omega_{\mathrm{st}})$. In order for $\zeta$ to retain the isoparametric representation of the subelements, we note in turn that each of its components must be defined in a polynomial space $\mathscr{P}'(\Omega_{\mathrm{st}})$ where in the most general case, $\mathscr{P}(\Omega_{\mathrm{st}}) \subset \mathscr{P}'(\Omega_{\mathrm{st}})$. A consequence of subdivision therefore is that the subdivided elements may have a higher polynomial order than the parent element depending on the choice of $f$.

Figure 2 shows a simple application of this subdivision strategy for a quadrilateral element. Here we choose for example an affine mapping $f(\xi_1, \xi_2) = (\xi_1, c\xi_2)$ for some $c \in (0, 1)$ so that the standard element is scaled in the $\xi_2$ direction. Applying the original $\chi$ mapping we obtain a new element $\tilde{\Omega}$ which is appropriately scaled, and naturally introduces curvature into the resulting subelement. In this case, any polynomial term $\xi_1^p \xi_2^q$ is mapped under $f$ to the term $c^q \xi_1^p \xi_2^q$ which clearly lies in $\mathscr{P}(\Omega_{\mathrm{st}}^{\mathrm{quad}})$, and indeed it is clear that by Eq. (1) that $\det_\zeta(\xi)$ is simply a scalar multiple of $\det_\chi(\xi)$. We may therefore choose $\mathscr{P}'(\Omega_{\mathrm{st}}) = \mathscr{P}(\Omega_{\mathrm{st}})$ and the order of the subelements may be the same as the parent element.

Since the restriction of equal polynomial order is somewhat burdensome, we now consider the case where the polynomial order in each direction is not equal. Whilst a similar argument to the previous explanation can be used in this case, more

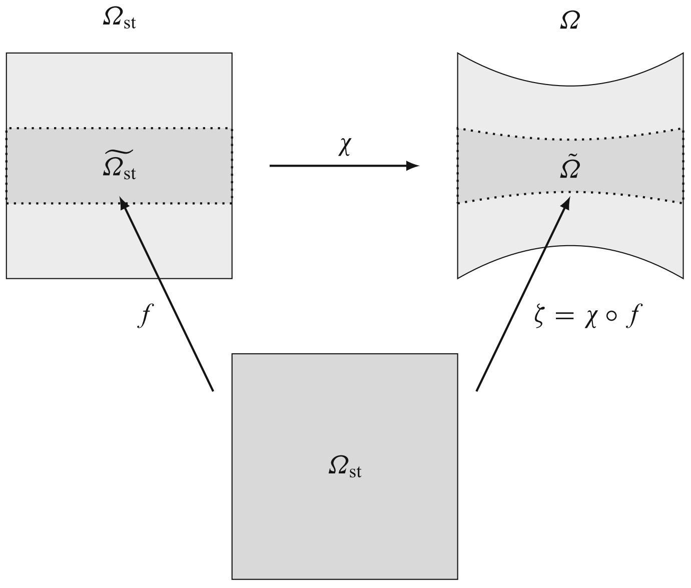


**Fig. 2** Construction of the mapping $\zeta$ for the subdivision of a quadrilateral element

care must be taken either in the choice of the mapping $f$ or in the order of the resulting subelements to ensure that the polynomial space is correctly spanned. For example, consider a quadrilateral element with expansion orders $P = 2$ and $Q = 1$ which has corresponding polynomial space $\mathscr{P}$, and suppose we choose to produce a trivially subdivided element by applying the transformation $f(\xi_1, \xi_2) = (-\xi_2, \xi_1)$. This map has positive Jacobian determinant and indeed is affine, as in the previous example. However, since $\xi_1$ and $\xi_2$ are permuted in the composition with $f$, the expansion has polynomial terms which lie outside of $\mathscr{P}$ leading to unpredictable element generation.

There are two solutions in this case. Firstly we may choose to obey the general condition $\mathscr{P}(\Omega_{\mathrm{st}}) \subset \mathscr{P}'(\Omega_{\mathrm{st}})$, and enrich the polynomial order of the subelement so that $P = Q = 2$. Alternatively however, we may permute the polynomial orders of the resulting subelements, so that $P = 1$ and $Q = 2$, to form a space $\mathscr{Q}$. We see in this instance that the resulting subelement is still valid as all of the terms of the original $\chi$ expansion are represented in $\zeta$, but the previous condition is not held since $\mathscr{P} \not\subset \mathscr{Q}$. We therefore note that $\mathscr{P}(\Omega_{\mathrm{st}}) \subset \mathscr{P}'(\Omega_{\mathrm{st}})$ represents a sufficient, but not necessary condition on the validity of subelements in this case.

A similar warning also applies to the other element types, and in particular triangles, prisms and tetrahedra since additional conditions are placed on the summation of mode indices which must be observed. In the next section, we discuss an enrichment strategy to permit the subdivision of elements into different element types.

## 3.2 Subdivision into Different Element Types

Another possible strategy one may adopt when subdividing elements is to consider their division into elements of a different type; for instance, we may subdivide a quadrilateral into triangles in two dimensions, or alternatively hexahedra into prisms or prisms into tetrahedra in three dimensions. Such techniques are well understood for linear finite elements [1] but for curvilinear elements self-intersection may occur if the interior deformation is not taken into account. In this section we demonstrate how the technique introduced in the previous section can be adapted to introduce curvature into the subelements in such a way as to prevent them becoming invalid.

We must adapt the previous argument above since now $f : \Omega'_{\mathrm{st}} \to \tilde{\Omega}$ where $\Omega'_{\mathrm{st}} \subsetneq \Omega_{\mathrm{st}}$ and so the polynomial spaces which span these standard elements obey the relation $\mathscr{P}(\Omega'_{\mathrm{st}}) \subsetneq \mathscr{P}(\Omega_{\mathrm{st}})$. In the same way that the technique needs an enrichment of the polynomial space if direction-dependent polynomial orders are used, if we naively apply the method then the polynomial expansion $\chi$ can contain terms which are not contained inside $\mathscr{P}(\Omega'_{\mathrm{st}})$, and so the resulting mapping $\zeta$ may not produce valid elements.

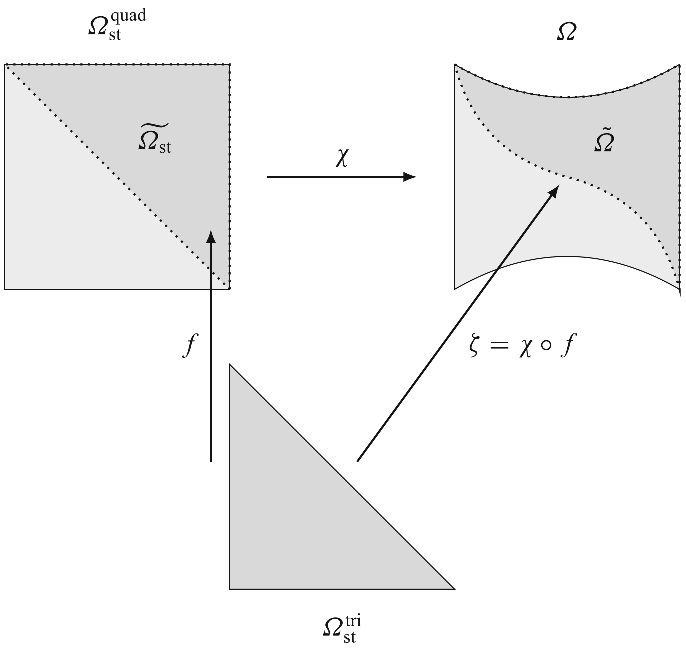
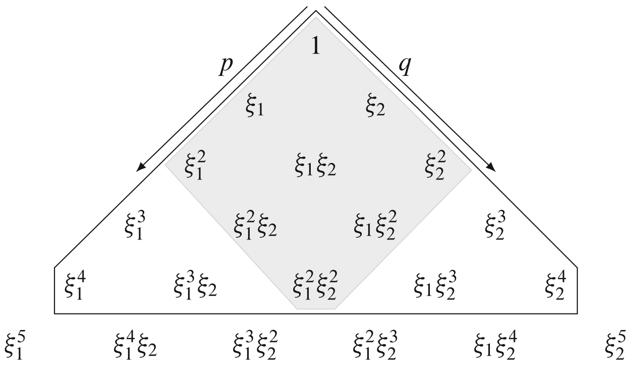

$\Omega_{\mathrm{st}}^{\mathrm{quad}}$
$\widetilde{\Omega_{\mathrm{st}}}$
$\chi$
$\Omega$
$\tilde{\Omega}$
$f$
$\zeta = \chi \circ f$
$\Omega_{\mathrm{st}}^{\mathrm{tri}}$

**Fig. 3** Construction of the map $\zeta$ in the case of a quadrilateral being split into two triangles

To demonstrate this point, we first examine the problem of Fig. 3, which depicts an example where a quadrilateral is split along a diagonal edge in order to obtain two triangles. We may again utilise an affine mapping $f(\xi) = -\xi$ in order to map $\Omega_{\mathrm{st}}^{\mathrm{tri}}$ onto a subdomain $\widetilde{\Omega_{\mathrm{st}}}$ of $\Omega_{\mathrm{st}}^{\mathrm{quad}}$. From our previous argument we see that each component of $\zeta = \chi \circ f$ has degree $2P$ in general if the original quadrilateral is of order $P$.

Since $\zeta \in [\mathscr{P}(\Omega_{\mathrm{st}}^{\mathrm{tri}})]^2$ we must select a sufficiently large polynomial order for the triangular space so that all terms of the expansion are represented in the resulting expansion. To guarantee this for a general quadrilateral-to-triangle split, given a quadrilateral of order $P$ we must generate triangles of order $2P$. Then the space $\mathscr{P}(\Omega_{\mathrm{st}}^{\mathrm{quad}}) \subset \mathscr{P}(\Omega_{\mathrm{st}}^{\mathrm{tri}})$ and thus $\zeta$ captures all curvature of the original mapping. For a visual illustration of this, we may represent the polynomial spaces of the triangular and quadrilateral elements in the form of a Pascal's triangle as shown in Fig. 4.

Figure 5 illustrates the problem of using triangular elements which are not sufficiently enriched. On the left, a second-order ($\mathbb{P}^2$) quadrilateral is split into two second-order triangles. Splitting the quadrilateral into two $\mathbb{P}^2$ triangles leads to the generation of degenerate elements. In this case, the symmetry of the deformed element coupled with the quadratic order of the triangles means that the diagonal edge which bisects the quadrilateral is forced to remain straight and thus causes a self-intersection. We note that in this example, the interior quadrilateral mode $\xi_1^2\xi_2^2$ is not energised since curvature is only introduced in one coordinate direction.



p
q
1
$\xi_1$ $\xi_2$
$\xi_1^2$ $\xi_1\xi_2$ $\xi_2^2$
$\xi_1^3$ $\xi_1^2\xi_2$ $\xi_1\xi_2^2$ $\xi_2^3$
$\xi_1^4$ $\xi_1^3\xi_2$ $\xi_1^2\xi_2^2$ $\xi_1\xi_2^3$ $\xi_2^4$
$\xi_1^5$ $\xi_1^4\xi_2$ $\xi_1^3\xi_2^2$ $\xi_1^2\xi_2^3$ $\xi_1\xi_2^4$ $\xi_2^5$

**Fig. 4** Pascal's triangle representing the polynomial spaces of $\mathbb{P}^2$ quadrilateral (*shaded grey*) and $\mathbb{P}^4$ triangular (*black outline*) elements. The triangle shows that in order to split a general $\mathbb{P}^2$ quadrilateral we require $\mathbb{P}^4$ triangles so that all terms can be represented in the resulting mapping

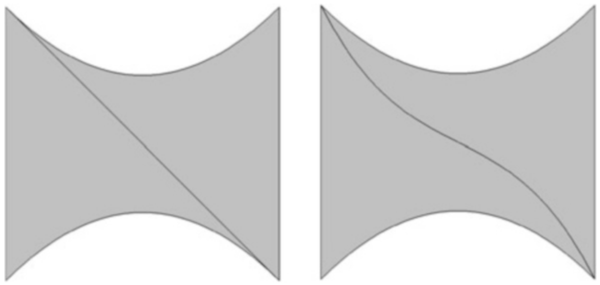


**Fig. 5** Qualitative example of the necessary condition for subdivision. A $\mathbb{P}^2$ quadrilateral is split into $\mathbb{P}^2$ (*left*) and $\mathbb{P}^3$ (*right*) triangles. Since a $\mathbb{P}^2$ triangular expansion does not capture some of the terms of the original mapping, an additional order is required to produce valid elements

We additionally note that this can be very intuitively achieved by the choice of a boundary-interior hierarchical expansion in which edge and vertex degrees of freedom are decoupled from the interior. Other basis types, such as a nodal Lagrange scheme, will not in general have this property, although the use of the classical Gordon-Hall blending does have this property.

Consulting the Pascal triangle of polynomial spaces we therefore see that only a $\mathbb{P}^3$ expansion is required for the triangular elements. Using this insight, from a qualitative perspective we can predict how the diagonal edge will be deformed under this mapping, since if we consider a parametrisation $r(t) = (t, -t)$ for $t \in [-1, 1]$ then the composition $\chi(r(t))$ should be a cubic polynomial in $t$. The resulting subdivision, shown on the right-hand side of Fig. 5, confirms this observation and consequently we obtain two valid triangular elements.

We note that the same logic can be used in the splitting of prismatic and hexahedra elements into tetrahedra. In general an order $P$ prismatic or hexahedral element also requires enrichment so that the resulting tetrahedra have order $2P$ and

$3P$ respectively. However by applying the logic above, if curvature is introduced only into the triangular faces of the prisms, then it is only necessary to produce order $P + 1$ tetrahedra. Since visualisation of the Pascal's triangle structure is more difficult in three dimensions, this can alternatively be seen from a brief analysis of the prismatic and tetrahedral spaces. If a linear expansion is used in the homogeneous direction of the prismatic element (i.e. $Q = 1$) and $P = R$ then the resulting polynomial space is

$$\mathscr{P}^{\mathrm{pri}}(\Omega_{\mathrm{st}}) = \{\xi_1^p \xi_2^q \xi_3^r \mid 0 \leq p + r \leq P,\ q = 0, 1\}.$$

A tetrahedron with equal polynomial order $P$ in each direction has the restriction on a triple $(pqr)$ that $0 \leq p + q + r \leq P$. If $q = 1$ then we obtain the restriction $0 \leq p + r \leq P - 1$, and so the tetrahedral space at order $P$ does not contain the prismatic space, leading to possible invalid elements. In order to guarantee validity of elements we therefore require tetrahedra of order $P + 1$.

In the following section, we give a demonstration of this prism-to-tetrahedral splitting and also highlight the application of the refinement method in boundary-layer problems.

## 4 Applications

This section demonstrates the usefulness of the subdivision method by showing how it can be used to generate three-dimensional meshes for challenging applications. Firstly we consider the subdivision of a coarse prismatic boundary-layer mesh into a series of progressively thinner elements as the distance to the wall decreases. We then show how the prismatic elements can be subdivided to obtain a boundary-layer mesh comprising only tetrahedra for use by solvers supporting only simplex elements.

### *4.1 Boundary Layer Mesh Generation*

Figure 6 shows how the subdivision technique can be used to generate a boundary layer mesh for an intercostal pair of a rabbit aorta. In these simulations one wishes to simulate the flow of blood through the aortic arch. From this, one may simulate an advection-diffusion equation in order to measure the concentration of particles which are transported by the flow of blood. Whilst in this case, the Reynolds number of the flow is not particularly large, the diffusion coefficient is inversely proportional to the Peclet number, which in turn is inversely proportional to the size of particle being advected. At high Peclet numbers, in a similar fashion to if the Reynolds number were high, one must use a thin boundary layer in order to resolve the steep gradient of the scalar variable representing concentration.

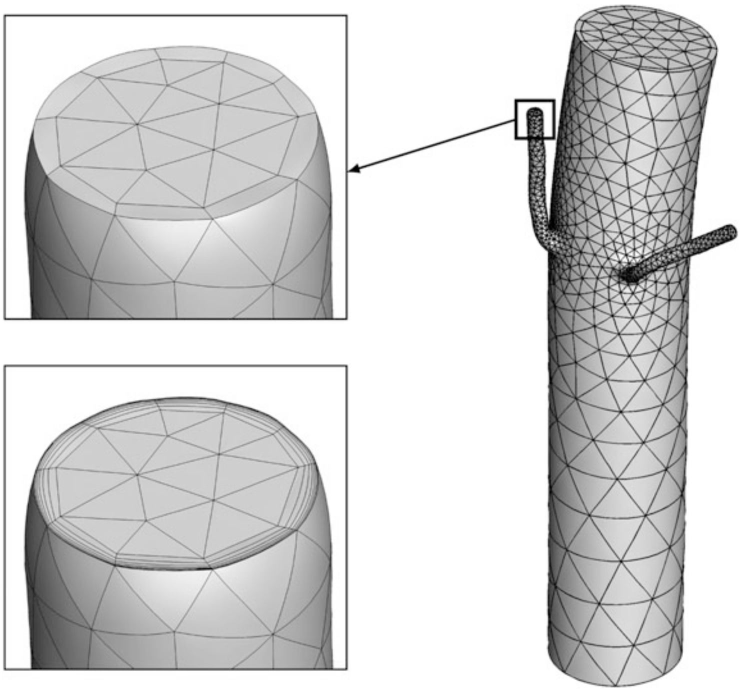


**Fig. 6** Boundary layer refinement for an intercostal pair of a rabbit aorta pictured right. The *upper left* image shows the high curvature of one smaller vessel, and *below* the resulting boundary layer mesh is visualised

In order to generate a sequence of $N$ subelements which gradually become more slender towards the surface of the domain, we define a spacing distribution $\Delta_k$ for $1 \leq k \leq N$ which defines the height of each prismatic subelement inside $\Omega_{\text{st}}^{\text{prism}}$. In this case, we choose $\Delta_k$ to be a geometric progression so that

$$\Delta_k = ar^k, \quad a = \frac{2(1-r)}{1-r^{N+1}}$$

where $r$ denotes a ratio dictating the relative height of each element. Under the framework of Sect. 3 then, we define a straightforward affine scaling function similar to that used in Fig. 2 which obeys the necessary conditions in order to generate valid subelements. We additionally note that as long as the same spacing distribution is used for all prismatic elements, the resulting mesh is conformal. One of the major advantages of this method for the generation of boundary layer meshes is that the resulting subelements are guaranteed to be valid, as shown in Sect. 3, and thus we are able to produce boundary layers of arbitrary thickness.

## 4.2 Generating Meshes of Simplex Elements

Certain solvers only have support for meshes which are composed only of simplex elements. For problems where boundary layers are required, this poses an additional problem for mesh generation software. In Fig. 7, we show how the same method can be used to split the prismatic elements of Fig. 6 into three tetrahedra. Firstly we note that in order for the resulting mesh to be conformal, we must employ a strategy so that the quadrilateral faces which connect prismatic elements are split in a consistent fashion, such as the one outlined by Dompierre et al. [1].

Once this strategy is applied, we may utilise the subdivision strategy to split the standard prismatic element into three tetrahedra by using an affine transformation similar to that used in Fig. 3. We note that in the specific case of Fig. 7, since the curvature of the original prisms is only imposed on the triangular surface, we may obtain valid tetrahedra by enriching the polynomial space by one order.

An important point to note is that whilst the validity of the resulting tetrahedra is guaranteed through our previous arguments, this method may lead to the production of tetrahedra which have suboptimal quality in terms of interior angles, depending on the curvature of the original prismatic elements. However, when tetrahedral boundary layers are required this is often unavoidable since the elements are required to possess a large stretching ratio. In the very worst cases, the use of these meshes as a starting point for a mesh deformation procedure may lead to better quality elements. We suggest that the validity of the meshes produced here may lead to improved convergence speeds in such methods.

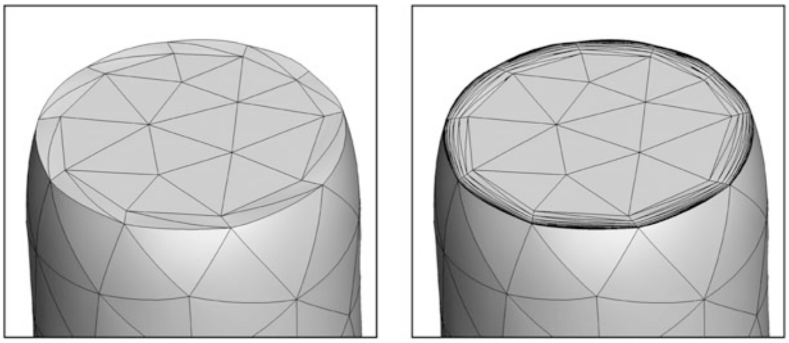


**Fig. 7** The result of a tetrahedralisation of the intercostal pair mesh of Fig. 6. The *left-hand figure* shows a prism to tetrahedron split of the original mesh; on the *right* we apply the splitting after the boundary layer refinement has been performed

## 5 Conclusions

In this paper we have derived the mathematical conditions that are necessary for the subdivision of high-order isoparametric elements, and show how this technique can be applied to tackle challenges in high-order mesh generation. We posit that the simplicity of the method outlined here will prove to be a valuable tool in improving both the efficiency and robustness of curvilinear mesh generation software, and particularly for the generation of meshes for high Reynolds number computational fluid dynamics problems or high Peclet number advection-diffusion problems.

**Acknowledgements** This work was partly supported by EU Grant No. 265780 as part of the EU FP7 project "IDIHOM: Industrialisation of High-Order Methods – A Top-Down Approach." Spencer J. Sherwin additionally acknowledges Royal Academy of Engineering support under their research chair scheme and support under the EPSRC Laminar Flow Control Centre (EP/H050507/1).

## References

1. Dompierre, J., Labbé, P., Vallet, M.G., Camarero, R.: How to subdivide pyramids, prisms and hexahedra into tetrahedra. In: 8th International Meshing Roundtable, Lake Tahoe, California (1999)
2. Karniadakis, G., Sherwin, S.: Spectral/*hp* Element Methods for Computational Fluid Dynamics, 2nd edn. Oxford University Press, Oxford (2005)
3. Moxey, D., Green, M.D., Sherwin, S.J., Peiró, J.: An isoparametric approach to high-order curvilinear boundary-layer meshing. Comp. Meth. App. Mech. Eng. **283**, 636–650 (2015). doi:10.1016/j.cma.2014.09.019
4. Persson, P.O., Peraire, J.: Curved mesh generation and mesh refinement using Lagrangian solid mechanics. In: 47th AIAA Aerospace Sciences Meeting and Exhibit, Orlando. AIAA paper 2009-949 (2009)
5. Sherwin, S., Peiró, J.: Mesh generation in curvilinear domains using high-order elements. Int. J. Numer. Methods Eng. **53**(1), 207–223 (2002)
6. Toulorge, T., Geuzaine, C., Remacle, J.F., Lambrechts, J.: Robust untangling of curvilinear meshes. J. Comput. Phys. **254**, 8–26 (2013)
7. Xie, Z., Sevilla, R., Hassan, O., Morgan, K.: The generation of arbitrary order curved meshes for 3D finite element analysis. Comput. Mech. **51**(3), 361–374 (2013)

# Anisotropic, Adaptive Finite Elements for a Thin 3D Plate

**Marco Picasso and Adrien Loseille**

**Abstract** An adaptive, anisotropic finite element algorithm is proposed to solve the 3D linear elasticity equations in a thin 3D plate. Numerical experiments show that adaptive computations can be performed in thin 3D domains having geometrical aspect ratio 1:1000.

## 1 The Linear Elasticity Model and the Numerical Method

Anisotropic adaptive algorithms are now widely used to solve complex systems based on partial differential equations, see for instance [5, 6, 11, 16]. Our goal is to experiment such techniques for the 3D linear elasticity system, the computational domain being a thin 3D plate.

Let $\Omega$ be the reference configuration of a bounded, polyhedral, elastic body of $\mathbb{R}^3$, $\partial\Omega = \Gamma_D \cup \Gamma_N$, $\Gamma_D$ not empty, $\Gamma_D \cap \Gamma_N = \emptyset$, $n$ the unit outer normal of $\partial\Omega$. Given $f \in L^2(\Omega)^3$, given the positive Lamé coefficients $\lambda, \mu$, we are looking for a displacement vector $u = (u_1, u_2, u_3)^T$ and a symmetric stress tensor $\sigma$ such that

$$\begin{aligned}
-\operatorname{div}\sigma &= f && \text{in } \Omega, && (1)\\
\sigma &= 2\mu D(u) + \lambda \operatorname{div} u\, I && \text{in } \Omega, && (2)\\
u &= 0 && \text{on } \Gamma_D, &&\\
\sigma n &= 0 && \text{on } \Gamma_N. &&
\end{aligned}$$

Hereabove, we have used the notation $D_{ij}(u) = \frac{1}{2}(\partial u_i/\partial x_j + \partial u_j/\partial x_i)$, and $I$ is the unit $3 \times 3$ tensor. Introducing $V = \{v \in H^1(\Omega)^3, v = 0 \text{ on } \Gamma_D\}$, the displacement

M. Picasso (✉)
EPFL, MATHICSE, Station 8, 1015 Lausanne, Switzerland
e-mail: marco.picasso@epfl.ch

A. Loseille
INRIA Paris-Rocquencourt, Gamma3, 78153 Le Chesnay, France
e-mail: adrien.loseille@inria.fr

© Springer International Publishing Switzerland 2015
S. Perotto, L. Formaggia (eds.), *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, SEMA SIMAI Springer Series 5,
DOI 10.1007/978-3-319-06053-8_11

weak form corresponding to this problem is to find $u \in V$ such that

$$\int_{\Omega} \Big( 2\mu D(u) : D(v) + \lambda \text{div}\, u \ \text{div}\, v \Big) dx = \int_{\Omega} f \cdot v dx \quad \forall v \in V, \tag{3}$$

where we have set $D(u) : D(v) = \sum_{i,j=1}^{3} D_{ij}(u) D_{ij}(v)$. Thanks to Korn's inequality

$$v \to \left( \sum_{i,j=1}^{3} \| D_{ij}(v) \|_{L^2(\Omega)}^2 \right)^{1/2}$$

is a norm on $V$ and the above problem has a unique solution. For any $h > 0$, let $\mathscr{T}_h$ be a conforming mesh of $\overline{\Omega}$ into tetrahedrons $K$ with diameter $h_K$ less than $h$. Assume that the mesh is such that $\Gamma_D$ is the union of triangles lying on $\partial\Omega$. Let $V_h$ be the usual finite element space of continuous displacements having components that are linear on the tetrahedrons of $\mathscr{T}_h$, zero valued on $\Gamma_D$. Then, the Galerkin formulation corresponding to (3) is to find $u_h = (u_{1,h}, u_{2,h}, u_{3,h})^T \in V_h$ such that

$$\int_{\Omega} \Big( 2\mu D(u_h) : D(v_h) + \lambda \text{div}\, u_h \ \text{div}\, v_h \Big) dx = \int_{\Omega} f \cdot v_h \, dx \quad \forall v_h \in V_h. \tag{4}$$

The matrix of the linear system corresponding to (4) is symmetric positive definite so that the Conjugate Gradient (CG) method can be used. From Korn, Poincaré and the inverse inequalities, it can be shown that the number of iterations required to solve the linear system with a Jacobi preconditioner is $O(1/h)$, thus doubles when the mesh size is divided by two. Thus, the complexity is $O(1/h^4)$; it can be reduced to $O(1/h^3)$—which is optimal—when using multigrid as a preconditioner, this will not be the case in this paper.

Our goal is to consider the case when the computational domain $\Omega$ is a thin 3D volume—for instance $\Omega = (0,1) \times (0,1) \times (0,\varepsilon)$ with $\varepsilon$ small—and when anisotropic finite elements are used—that is tetrahedrons with large aspect ratio.

## 2 An Anisotropic Error Indicator

We now use the notations of [2, 3] in order to describe the mesh anisotropy, similar results can be found in [8]. For any tetrahedron $K$ of the mesh, let $T_K : \hat{K} \to K$ be the affine transformation which maps the reference tetrahedron $\hat{K}$ into $K$. Let $M_K$ be the Jacobian of $T_K$ that is

$$\mathbf{x} = T_K(\hat{\mathbf{x}}) = M_K \hat{\mathbf{x}} + \mathbf{t}_K.$$

Since $M_K$ is invertible, it admits a singular value decomposition $M_K = R_K^T \Lambda_K P_K$, where $R_K$ and $P_K$ are orthogonal and where $\Lambda_K$ is diagonal with positive entries. In the following we set

$$\Lambda_K = \begin{pmatrix} \lambda_{1,K} & 0 & 0 \\ 0 & \lambda_{2,K} & 0 \\ 0 & 0 & \lambda_{3,K} \end{pmatrix} \qquad \text{and} \qquad R_K = \begin{pmatrix} r_{1,K}^T \\ r_{2,K}^T \\ r_{3,K}^T \end{pmatrix}, \tag{5}$$

with the choice $\lambda_{1,K} \geq \lambda_{2,K} \geq \lambda_{3,K}$. In the frame of anisotropic meshes, the classical minimum angle condition is not required. However, for each vertex, the number of neighbouring vertices should be bounded from above, uniformly with respect to the mesh size $h$. Also, for each tetrahedron $K$ of the mesh, there is a restriction related to the patch $\Delta_K$, the set of tetrahedrons having a vertex common with $K$. More precisely, the diameter of the reference patch $\Delta_{\hat{K}}$, that is $\Delta_{\hat{K}} = T_K^{-1}(\Delta_K)$, must be uniformly bounded independently of the mesh geometry. This assumption excludes some too distorted reference patches, see [15]. This assumption is needed in order to prove the interpolation estimates—Clément's interpolant involves quantities on the reference patch—and implies that the local geometric quantities $\lambda_{i,K}$, $r_{i,K}$, $i = 1, 2, 3$, vary smoothly on neighbouring tetrahedrons. Two examples of admissible and non-admissible patches are presented in Fig. 1. The anisotropic mesh generator used in this paper has always produced admissible patches.

Let us now introduce our anisotropic error indicator. It is similar to the one presented in [14, 15] for the Laplace equation. For all $K \in \mathscr{T}_h$, let $\ell_{i,K}$, $i = 1, 2, 3, 4$ be the four faces of tetrahedron $K$, with unit normal $n_{i,K}$ (in arbitrary direction), let $[\cdot]$ denote the jump of the bracketed quantity across $\ell_{i,K}$, with the convention $[\cdot] = 0$ for a face $\ell_{i,K}$ on the boundary $\Gamma_D$. Then, our error indicator on tetrahedron $K$ is defined by

$$\eta_K^2 = \rho_K(u_h)\omega_K(e), \tag{6}$$

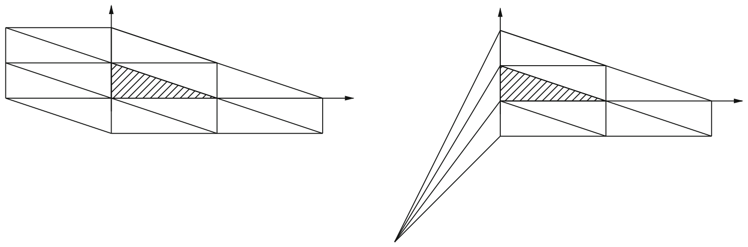


**Fig. 1** Admissible (*left*) and non-admissible (*right*) patches

where

$$\rho_K(u_h) = \frac{1}{2}\sum_{i=1}^{4}\left(\frac{|\ell_{i,K}|}{\lambda_{1,K}\lambda_{2,K}\lambda_{3,K}}\right)^{1/2} \left(\int_{\ell_{i,K}}\left([\nabla u_{1,h}\cdot n_{i,K}]^2 + [\nabla u_{2,h}\cdot n_{i,K}]^2 + [\nabla u_{3,h}\cdot n_{i,K}]^2\right)dx\right)^{1/2},$$

where $e = (e_1, e_2, e_3)^T = u - u_h$ is the true error, and $\omega_K(e)$ is defined by

$$(\omega_K(e))^2 = \lambda_{1,K}^2\left(r_{1,K}^T G_K(e) r_{1,K}\right)+\lambda_{2,K}^2\left(r_{2,K}^T G_K(e) r_{2,K}\right)+\lambda_{3,K}^2\left(r_{3,K}^T G_K(e) r_{3,K}\right). \tag{7}$$

Here $G_K(e)$ denotes the $3\times 3$ matrix defined by

$$G_K(e) = \sum_{j=1}^{3}\begin{pmatrix} \int_{\Delta_K}\left(\frac{\partial e_j}{\partial x_1}\right)^2 dx & \int_{\Delta_K}\frac{\partial e_j}{\partial x_1}\frac{\partial e_j}{\partial x_2}dx & \int_{\Delta_K}\frac{\partial e_j}{\partial x_1}\frac{\partial e_j}{\partial x_3}dx \\ \int_{\Delta_K}\frac{\partial e_j}{\partial x_1}\frac{\partial e_j}{\partial x_2}dx & \int_{\Delta_K}\left(\frac{\partial e_j}{\partial x_2}\right)^2 dx & \int_{\Delta_K}\frac{\partial e_j}{\partial x_2}\frac{\partial e_j}{\partial x_3}dx \\ \int_{\Delta_K}\frac{\partial e_j}{\partial x_1}\frac{\partial e_j}{\partial x_3}dx & \int_{\Delta_K}\frac{\partial e_j}{\partial x_2}\frac{\partial e_j}{\partial x_3}dx & \int_{\Delta_K}\left(\frac{\partial e_j}{\partial x_3}\right)^2 dx \end{pmatrix}. \tag{8}$$

The indicator (6) is not a usual error estimator since $e = u - u_h$ (and therefore $u$) is still involved. However, the error $e$ can be estimated using post-processing techniques, so that (6) can be used to derive a computable quantity. An efficient anisotropic error indicator has been previously obtained replacing the derivatives

$$\frac{\partial e_j}{\partial x_i} \text{ in (8) by } \frac{\partial u_{j,h}}{\partial x_i} - \Pi_h\frac{\partial u_{j,h}}{\partial x_i}, \text{ i,j=1,2,3} \tag{9}$$

where $\Pi_h$ is an approximate $L^2(\Omega)$ projection onto $V_h$. More precisely, from constant values of $\partial u_{j,h}/\partial x_i$ on triangles, we build values at vertices $P$ using the formula

$$\Pi_h\left(\frac{\partial u_{j,h}}{\partial x_i}\right)(P) = \frac{1}{\sum\limits_{\substack{K\in\mathscr{T}_h\\ P\in K}}|K|}\sum_{\substack{K\in\mathscr{T}_h\\ P\in K}}|K|\left(\frac{\partial u_{j,h}}{\partial x_i}\right)_{|K} \qquad i,j = 1,2,3.$$

Approximating $\partial e_j/\partial x_i$ by $(I - \Pi_h)\partial u_{j,h}/\partial x_i$ is at the base of the celebrated Zienkiewicz-Zhu error estimator and can be justified theoretically whenever superconvergence occurs, that is when $\nabla u_j - \Pi_h \nabla u_{j,h}$ convergences faster to zero than $\nabla u_j - \nabla u_{j,h}$. To our knowledge, the most recent and general result has been obtained for a second order elliptic problem and 2D mildly structured anisotropic meshes in [1]. In practice, on general 3D unstructured anisotropic meshes, superconvergence is not observed; however, $\nabla u_j - \Pi_h \nabla u_{j,h}$ is much smaller than $\nabla u_j - \nabla u_{j,h}$.

## 3 An Adaptive Algorithm

We have considered the anisotropic, adaptive algorithm presented in [14, 15], the goal being to build an anisotropic triangulation such that the estimated relative error is close to a preset tolerance *TOL*, namely

$$(1-\beta)TOL \leq \frac{\left(\sum_{K\in\mathscr{T}_h} \eta_K^2\right)^{1/2}}{\|\nabla u_h\|_{L^2(\Omega)}} \leq (1+\beta)TOL. \tag{10}$$

Here $\eta_K$ is defined by (6)–(8) and the post-processing (9) has been used in order to approximate the error gradient $G_K(e)$. Also, $0 < \beta < 1$ is the equidistribution parameter, $\beta = 0.25$ throughout the paper.

Our goal is to equidistribute locally the error in the three directions of stretching $r_{1,K}, r_{2,K}, r_{3,K}$, and to align the mesh along the eigenvectors of the matrix $G_K(e)$. In practice, all the meshes are generated using the `feflo` software [13] which requires a metric to be prescribed at the mesh vertices. The method used to build this metric is now described. For each vertex $P$ of the mesh, we compute

$$G_P(e) = \sum_{\substack{K\in\mathscr{T}_h \\ P\in K}} G_K(e),$$

where (9) has been used to estimate $G_K(e)$. We then compute an orthonormal basis $Q_P(e)$ of the eigenvectors of $G_P(e)$. Our goal is to align the tetrahedron around vertex $P$ with the eigenvectors of $G_P(e)$. The metric is then defined by

$$Q_P(e)^T \begin{pmatrix} \frac{1}{h_{1,P}^2} & 0 & 0 \\ 0 & \frac{1}{h_{2,P}^2} & 0 \\ 0 & 0 & \frac{1}{h_{3,P}^2} \end{pmatrix} Q_P(e), \tag{11}$$

where the desired mesh size at vertex $P$, $h_{1,P}$, $h_{2,P}$, $h_{3,P}$, is prescribed in order to satisfy (10). More precisely, we go back to (6) and split the estimator on triangle $K$ in the three directions of stretching corresponding to $r_{i,K}$ :

$$(\eta_{i,K})^4 = (\rho_K(u_h))^2 \lambda_{i,K}^2 \left( r_{i,K}^T G_K(e) r_{i,K} \right) \qquad i = 1, 2, 3,$$

and then compute the corresponding quantity at each vertex $P$ of the mesh

$$(\eta_{i,P})^4 = \sum_{\substack{K \in \mathscr{T}_h \\ P \in K}} (\eta_{i,K})^4 \qquad i = 1, 2, 3.$$

Let $N_P$ is the number of mesh vertices. Since

$$\sum_{i=1}^{3} \sum_{P \in \mathscr{T}_h} (\eta_{i,P})^4 = 4 \sum_{i=1}^{3} \sum_{K \in \mathscr{T}_h} (\eta_{i,K})^4 = 4 \sum_{K \in \mathscr{T}_h} (\eta_K)^4,$$

if

$$\left( \frac{4}{3N_P^2} \right)^{1/4} (1-\beta) TOL \|\nabla u_h\|_{L^2(\Omega)} \leq \eta_{i,P} \leq \left( \frac{4}{3N_P^2} \right)^{1/4} (1+\beta) TOL \|\nabla u_h\|_{L^2(\Omega)},$$

for $i = 1, 2, 3$ and for each vertex $P$ of the mesh, then (10) is satisfied. The desired mesh size at vertex $P$, $h_{1,P}$, $h_{2,P}$, $h_{3,P}$, is then computed as follows. If

$$\frac{4}{3N_P^2} (1-\beta)^4 TOL^4 \|\nabla u_h\|_{L^2(\Omega)}^4 > (\eta_{i,P})^4,$$

then the values of $h_{i,P}$ are set to $2\lambda_{i,P}$, $i = 1, 2, 3$, if

$$\frac{4}{3N_P^2} (1-\beta)^4 TOL^4 \|\nabla u_h\|_{L^2(\Omega)}^4 \leq (\eta_{i,p})^4 \leq \frac{4}{3N_P^2} (1+\beta)^4 TOL^4 \|\nabla u_h\|_{L^2(\Omega)}^4,$$

then the values of $h_{i,P}$ are set to $\lambda_{i,P}$, $i = 1, 2, 3$, if

$$(\eta_{i,P})^4 > \frac{4}{3N_P^2} (1+\beta)^4 TOL^4 \|\nabla u_h\|_{L^2(\Omega)}^4,$$

then the values of $h_{i,P}$ are set to $\lambda_{i,P}/2$, $i = 1, 2, 3$. Once new values of $h_{i,P}$ are obtained, the metric at each vertex is computed from (11) and the `feflo` software [13] is used to generate a new anisotropic mesh. The whole process is then repeated several times.

## 4 Anisotropic Mesh Generation

We give in the section some details on the algorithm and the mechanisms used in `feflo` [12] to refine the mesh according to the previous error estimate. The mesh generator fits the Riemannian metric framework of [4]. The goal is to generate a quasi-unit mesh with respect to the prescribed metric (11). The input of the mesh generator is the metric—a 3×3 symmetric positive definite matrix—thus we assume that, for each $x \in \Omega$, the metric $\mathcal{M}(x)$ is known.

The two fundamental operations in a mesh generator are the computation of length and volume. Let $E$ be the edge joining vertices $x_i$ and $x_j$. The length of $E$ and the volume of a tetrahedron $K$, with respect to the metric $\mathcal{M}$, are defined by:

$$\ell_{\mathcal{M}}(E) = \int_0^1 \sqrt{(x_j - x_i)^T \, \mathcal{M}(x_i + t(x_j - x_i)) \, (x_j - x_i)} \, \mathrm{d}t,$$

$$|K|_{\mathcal{M}} = \int_K \sqrt{\det(\mathcal{M}(x))} \, \mathrm{d}x.$$

From a discrete point view, the metric field needs to be interpolated [4] to compute an approximate edge length and tetrahedron volume. We consider a linear interpolation of the point-wise metric field $\mathcal{M}_i$ at the vertices $x_i$ of the mesh; the following approximations are then used:

$$\ell_{\mathcal{M}}(E) \approx \sqrt{(x_j - x_i)^T \, \mathcal{M}_i \, (x_j - x_i)} \, \frac{r-1}{r \ln(r)},$$

$$|K|_{\mathcal{M}} \approx \sqrt{\det\left(\frac{1}{4}\sum_{i=1}^{4} \mathcal{M}_i\right)} |K|,$$

where $|K|$ is the Euclidean volume of $K$ and $r$ stands for the ratio

$$\frac{\sqrt{(x_j - x_i)^T \, \mathcal{M}_i \, (x_j - x_i)}}{\sqrt{(x_j - x_i)^T \, \mathcal{M}_j \, (x_j - x_i)}},$$

see Example 1.3 in [9] for details. A mesh is said to be a unit-mesh with respect to $\mathcal{M}$ when the six edges of each tetrahedron $K$ satisfy

$$\ell_{\mathcal{M}}(E_i) \in \left[\frac{1}{\sqrt{2}}, \sqrt{2}\right] \qquad i = 1, \ldots, 6, \tag{12}$$

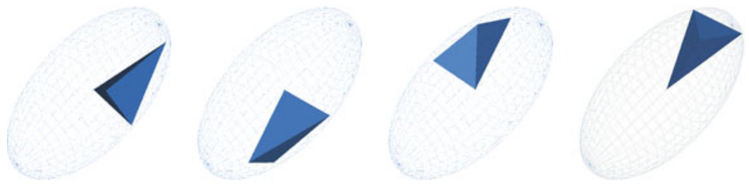


**Fig. 2** Some unit-elements with respect to a 3D metric represented by its unit-ball

and when the quality function $Q_{\mathcal{M}}$ satisfies

$$Q_{\mathcal{M}}(K) = \frac{36}{3^{\frac{1}{3}}} \frac{|K|_{\mathcal{M}}^{\frac{2}{3}}}{\sum_{i=1}^{6} \ell_{\mathcal{M}}^2(e_i)} \in [\alpha, 1], \tag{13}$$

where $\alpha > 0$ is a parameter. A classical and admissible value of $\alpha$ is 0.8. This value arises from some discussions on the possible tessellation of $\mathbb{R}^3$ with unit-elements [10]. Some unit-elements with respect to a 3D metric are depicted in Fig. 2.

To generate a unit-mesh in a given metric field $\mathcal{M}$, two operators are recursively used: edge collapse and point insertion on edge. The starting point for the insertion of a new point on an edge is to consider the shell of this edge composed of all elements sharing this edge. Each element of the shell is then divided into two new elements. The new point is accepted if each new tetrahedron has a positive volume. The edge collapse starts from the ball of the vertex to be deleted. Again, for the deletion of points inside the volume, the only possible rejection is the creation of a negative volume element.

We also combine the previous operators with a quality function $Q_{\mathcal{M}}$ together with the unit-length check. This supplementary check can be done at no cost since a lot of information can be re-used: the volume is already computed, as well as the length of the edges. By simply computing the quality function, we give to these operators the missing information on the orthogonal directions of the current scanned edge. For an optimal performance, two parameters are added in the rejection cases: a relative quality tolerance $q_r \geq 1$ and a global quality tolerance $q_a$. Rather than trying to increase $Q_{\mathcal{M}}$, a new configuration of elements is accepted if

$$q_r \, Q_{\mathcal{M}}^{ini} \leq Q_{\mathcal{M}}^{new} < q_a,$$

where $Q_{\mathcal{M}}^{ini}$ is the worse element quality of the initial configuration and $Q_{\mathcal{M}}^{new}$ is the worse quality of the new configuration. This approach is similar to the simulated annealing global optimization technique [7]. Note that the current version does not fully implement the classical metropolis algorithm where the rejection is based on a random probability. To ensure the convergence of the algorithm, the relative tolerance $q_r$ is decreased down to 1 after each pass of insertions and collapses. At the end of the process, the absolute tolerance $q_a$ is set up to the current worse quality among all elements.

This strategy yields a robust local remeshing procedure as a valid mesh is always provided on output. In particular, the volume and surface mesh generation are done simultaneously. Consequently, this procedure may be used advantageously for cases where global remeshing techniques become either unfeasible or unreliable. For the thin plate case, as a very high level of anisotropy is present on the surface, global remeshing approaches are likely to fail.

## 5 Numerical Experiments

Consider the case when $\Omega = (0,1) \times (0,1) \times (0,\varepsilon)$, with $\varepsilon = 0.01$ or $\varepsilon = 0.001$, $\Gamma_D$ are the lateral faces of the plate, $\Gamma_N$ the top and bottom faces corresponding to $z = 0$ and $z = \varepsilon$. Let $f = 1$ in (1), $\lambda = \mu = 1$ in (2). The exact solution is not known exactly but numerical experiments indicate that the maximum displacement up to three digits is between 0.0736 and 0.0737.

In Fig. 3, the mesh and vertical deformation are reported when $\varepsilon = 0.01$ or $\varepsilon = 0.001$ when using a $10 \times 10 \times 2$ mesh, each bloc being cup into six vertices. Starting from this $10 \times 10 \times 2$ mesh, convergence results with non-adapted meshes are presented in Table 1 when $\varepsilon = 0.01$. A locking effect can be observed when each bloc is cut into five tetrahedrons.

The adaptive procedure presented in the previous section is repeated 150 times, starting from the $10 \times 10 \times 2$ mesh considered above. At first, the *TOL* parameter which drives precision, see (10), is set to 1; it is halfed every 30 mesh generations, thus the adapted mesh number 29 corresponds to 30 mesh iterations with

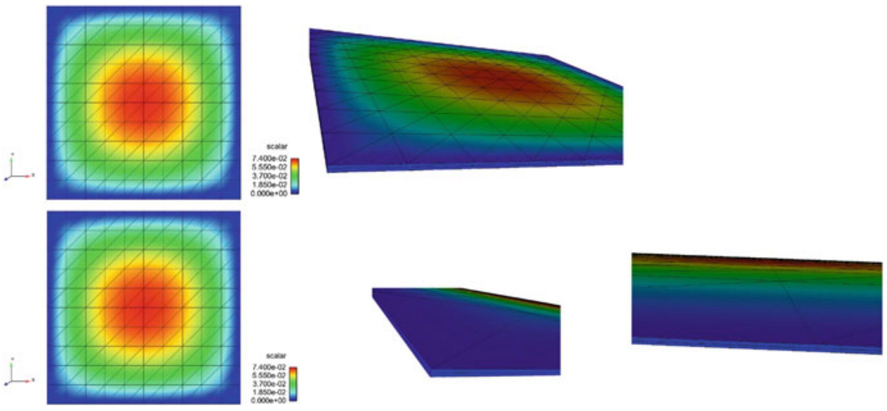


**Fig. 3** $10\times10\times2$ mesh, each bloc being cut in six tetrahedron, $\varepsilon = 0.01$ (*first row*) and $\varepsilon = 0.001$ (*second row*). *Left column*: xy plane, *Middle* and *Right column*: zoom

**Table 1** Convergence results with non-adapted meshes and $\varepsilon = 0.01$. Here tet/bloc denotes the number of tetrahedron per bloc

| Mesh | tet/bloc | nb. vert. | $CG$ | $u_{max}$ |
|---|---|---|---|---|
| $10 \times 10 \times 2$ | 6 | 363 | 103 | 0.0731 |
| $20 \times 20 \times 4$ | 6 | 2,205 | 401 | 0.0735 |
| $40 \times 40 \times 8$ | 6 | 15,129 | 1229 | 0.0736 |
| $80 \times 80 \times 16$ | 6 | 111,537 | 2294 | 0.0737 |
| $10 \times 10 \times 2$ | 5 | 363 | 45 | 0.0129 |
| $20 \times 20 \times 4$ | 5 | 2,205 | 229 | 0.0340 |
| $40 \times 40 \times 8$ | 5 | 15,129 | 879 | 0.0571 |
| $80 \times 80 \times 16$ | 5 | 111,537 | 2514 | 0.0687 |

Locking occurs when each bloc is cut into five tetrahedrons. CG denotes the number of iterations of the CG algorithm with diagonal preconditioner. The CG algorithm is stopped when the relative residual is less than $10^{-6}$

$TOL = 1$, the adapted mesh number 149 corresponds to 30 mesh iterations with $TOL = 0.0625$. The obtained adapted meshes corresponding to a plate thickness $\varepsilon = 0.01$ are shown in Fig. 4, those corresponding to a plate thickness $\varepsilon = 0.001$ are shown in Fig. 5. A careful examination of the adapted meshes reveals that there is only one layer of elements across the plate thickness and that mesh refinement is more important at the corners of the plate. Tables 2 and 3 contain the important numbers associated to these numerical experiments.

## 6 Conclusions

An adaptive, anisotropic finite element algorithm has been proposed to solve the 3D linear elasticity equations on a thin plate. Since the tetrahedrons are allowed to have large aspect ratio, the local mesh size can be small in the direction of the plate's thickness and coarse in the other directions. Numerical experiments show that adaptive computations can be performed on thin plates having geometrical aspect ratio 1:1000.

We are looking forward to perform numerical experiments on curved plates and/or plates having non homogenous thickness. In the case of curved plates, an error estimator for the geometry error has to be investigated. Also, the surface mesh should be reprojected onto the true geometry using the CAD data.

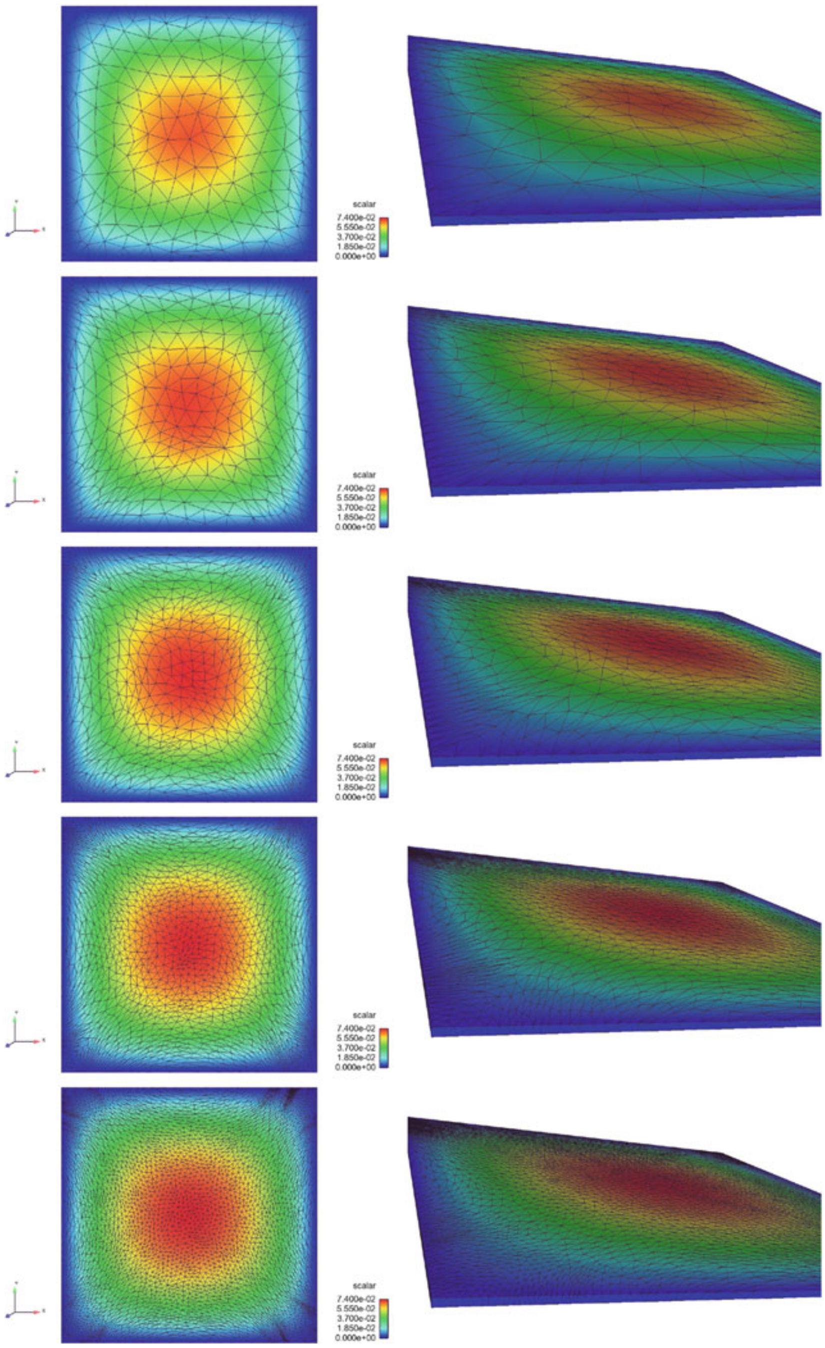


**Fig. 4** Adapted meshes with plate thickness $\varepsilon = 0.01$ and several values of *TOL* (*TOL* $= 1$ on row 1, *TOL* $= 0.5$ on row 2,…, *TOL* $= 0.0625$ on row 5). *Left column*: xy plane, *Right column*: zoom of the (0,0,0) corner

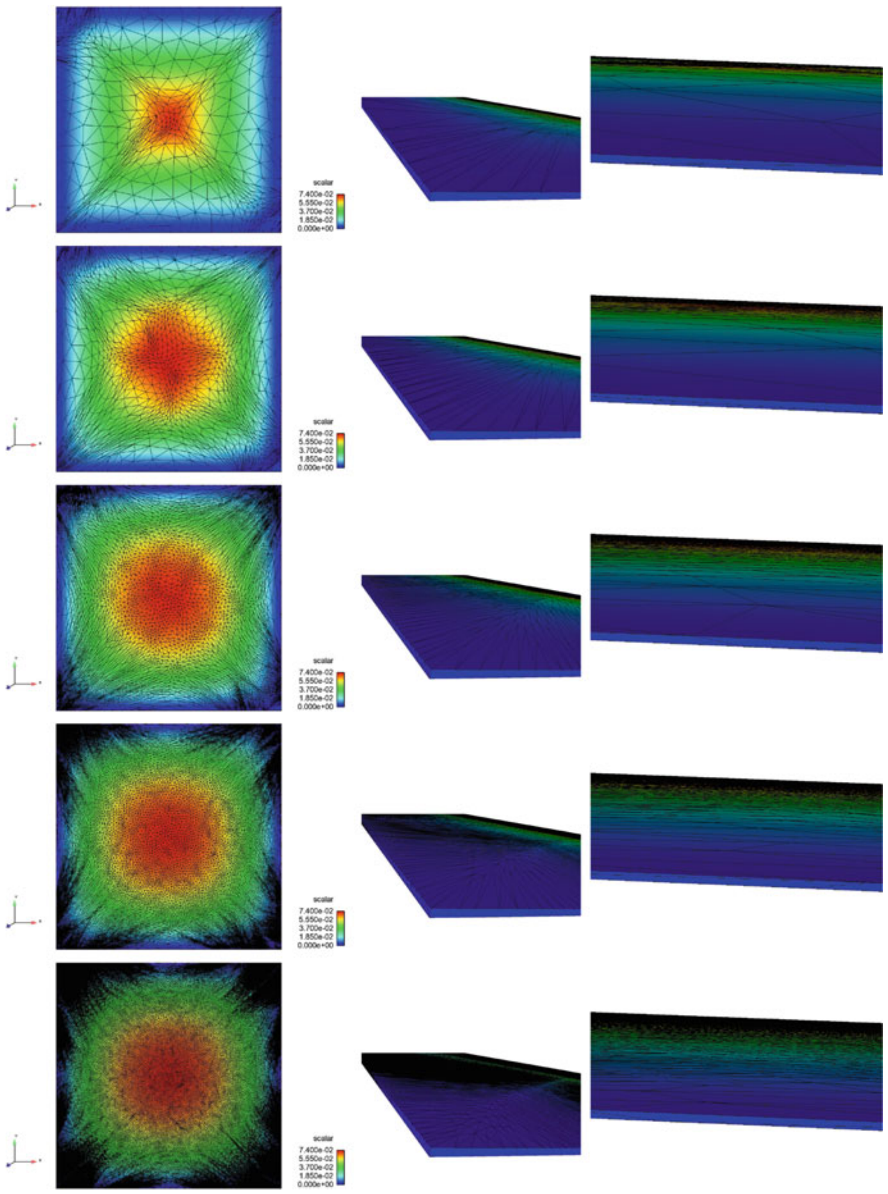


**Fig. 5** Adapted meshes with plate thickness $\varepsilon = 0.001$ and several values of *TOL* (*TOL* $= 1$ on row 1, *TOL* $= 0.5$ on row2,...,*TOL* $= 0.0625$ on row 5). *Left column*: xy plane. *Middle column*: zoom of the (0,0,0) corner. *Right column*: zoom at the center of an edge

**Table 2** Results with adapted meshes and plate thickness $\varepsilon = 0.01$ and several values of *TOL*

| Mesh nb. | *TOL* | nb. vert. | *CG* | $u_{max}$ | $\max \lambda_1$ | $\min \lambda_3$ |
|---|---|---|---|---|---|---|
| 0 | | 363 | 103 | 0.0731 | 0.16 | 0.0050 |
| 29 | 1 | 490 | 337 | 0.0697 | 0.17 | 0.0042 |
| 59 | 0.5 | 768 | 384 | 0.0713 | 0.19 | 0.0032 |
| 89 | 0.25 | 1419 | 401 | 0.0734 | 0.18 | 0.0030 |
| 119 | 0.125 | 3045 | 416 | 0.0736 | 0.15 | 0.0014 |
| 149 | 0.0625 | 7610 | 455 | 0.0736 | 0.11 | 0.00062 |

Here $\max \lambda_1 = \max_{K\in\mathscr{T}_h} \lambda_{1,K}$, $\min \lambda_3 = \min_{K\in\mathscr{T}_h} \lambda_{3,K}$

**Table 3** Results with adapted meshes and plate thickness $\varepsilon = 0.001$ and several values of *TOL*

| Mesh nb. | *TOL* | nb. vert. | *CG* | $u_{max}$ | $\max \lambda_1$ | $\min \lambda_3$ |
|---|---|---|---|---|---|---|
| 0 | | 363 | 117 | 0.0731 | 0.16 | 0.00050 |
| 29 | 1 | 1,274 | 1,656 | 0.0717 | 0.18 | 0.000049 |
| 59 | 0.5 | 2,790 | 2,680 | 0.0745 | 0.18 | 0.000061 |
| 89 | 0.25 | 7,069 | 3,375 | 0.0732 | 0.18 | 0.000072 |
| 119 | 0.125 | 17,402 | 4,090 | 0.0736 | 0.17 | 0.000054 |
| 149 | 0.0625 | 49,718 | 4,008 | 0.0737 | 0.16 | 0.000033 |

## References

1. Cao, W.: Superconvergence analysis of the linear finite element method and a gradient recovery postprocessing on anisotropic meshes. Math. Comp. **84**(291), 89–117 (2015)
2. Formaggia, L., Perotto, S.: New anisotropic a priori error estimates. Numer. Math. **89**(4), 641–667 (2001)
3. Formaggia, L., Perotto, S.: Anisotropic error estimates for elliptic problems. Numer. Math. **94**(1), 67–92 (2003)
4. Frey, P., George, P.L.: Mesh generation. Application to finite elements, 2nd edn. ISTE Ltd and John Wiley & Sons, New York (2008)
5. Hachem, E., Kloczko, T., Digonnet, H., Coupez, T.: Stabilized finite element solution to handle complex heat and fluid flows in industrial furnaces using the immersed volume method. Int. J. Numer. Methods Fluids **68**(1), 99–121 (2012)
6. Hassan, W.: Algorithmes d'adaptation de maillages anisotropes et application Ãă l'aérodynamique. Ph.D. thesis, EPFL, 1015 Lausanne, Switzerland (2012). Nb. 5304
7. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
8. Kunert, G.: A posteriori $L_2$ error estimation on anisotropic tetrahedral finite element meshes. IMA J. Numer. Anal. **21**(2), 503–523 (2001)
9. Loseille, A.: Adaptation de maillage anisotrope 3D multi-échelles et ciblée à une fonctionnelle pour la mécanique des fluides. Ph.D. thesis, University Paris VI (2008)
10. Loseille, A., Alauzet, F.: Continuous mesh framework part i: Well-posed continuous interpolation error. SIAM J. Numer. Anal. **49**(1), 38–60 (2011)
11. Loseille, A., Dervieux, A., Alauzet, F.: Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations. J. Comput. Phys. **229**(8), 2866–2897 (2010)
12. Loseille, A., Löhner, R.: Adaptive anisotropic simulations in aerodynamics. AIAA-10-0169 (2010)

13. Loseille, A., Löhner, R.: Anisotropic mesh generation for high-fidelity simulations in cfd. INRIA Preprint (2014)
14. Picasso, M.: An anisotropic error indicator based on Zienkiewicz-Zhu error estimator : application to elliptic and parabolic problems. SIAM J. Sci. Comp. **24**, 1328–1355 (2003)
15. Picasso, M.: Adaptive finite elements with large aspect ratio based on an anisotropic error estimator involving first order derivatives. Comput. Methods Appl. Mech. Eng. **196**(1-3), 14–23 (2006)
16. Piggott, M.D., Farrell, P.E., Wilson, C.R., Gorman, G.J., Pain, C.C.: Anisotropic mesh adaptivity for multi-scale ocean modelling. Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. **367**(1907), 4591–4611 (2009)

# Anisotropic Mesh and Time Step Adaptivity for Solute Transport Modeling in Porous Media

**Bahman Esfandiar, Giovanni Porta, Simona Perotto, and Alberto Guadagnini**

**Abstract** We assess the impact of space-time mesh adaptivity on the modeling of solute transport in porous media. This approach allows an automatic selection of both the spatial mesh and the time step on the basis of a suitable recovery-based error estimator. In particular, we deal with an anisotropic control of the spatial mesh. The solver coupled with the adaptive module deals with an advection-dispersion equation to model the transport of dissolved species, which are assumed to be convected by a Darcy flow field. The whole solution-adaptation procedure is assessed through two-dimensional numerical tests. A numerical convergence analysis of the spatial mesh adaptivity is first performed by considering a test-case with analytical solution. Then, we validate the space-time adaptive procedure by reproducing a set of experimental observations associated with solute transport in a homogeneous sand pack. The accuracy and the efficiency of the methodology are discussed and numerical results are compared with those associated with fixed uniform space-time discretizations. This assessment shows that the proposed approach is robust and reliable. In particular, it allows us to obtain a significant improvement of the simulation quality of the early solute arrivals times at the outlet of the medium.

## 1 Introduction

A wide set of physical processes involves transport of solutes in porous media. These include contamination of groundwater by inorganic and organic chemicals, petroleum generation and migration, reactive processes which can modify the

B. Esfandiar • G. Porta (✉) • A. Guadagnini
Dipartimento di Ingegneria Civile ed Ambientale, Politecnico di Milano,
Piazza Leonardo da Vinci 32, I-20133 Milano, Italy
e-mail: bahman.esfandiar@polimi.it; giovanni.porta@polimi.it; alberto.guadagnini@polimi.it

S. Perotto
MOX—Dipartimento di Matematica, Politecnico di Milano, Piazza Leonardo da Vinci 32,
I-20133 Milano, Italy
e-mail: simona.perotto@polimi.it

S. Perotto, L. Formaggia (eds.), *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, SEMA SIMAI Springer Series 5,
DOI 10.1007/978-3-319-06053-8_12

properties of soil and rock formations. Several modeling techniques and analytical solutions are proposed in the literature, while a variety of algorithms and software packages have been developed for the numerical simulation of solute transport in the subsurface (see, e.g., [3, 20, 52]). Transport of solutes in porous media is typically modeled through an advection-dispersion equation, where the dispersive coefficient embeds the effects of molecular diffusion and hydrodynamic dispersion. The advective term results from a velocity field, which is typically assumed to obey Darcy's law.

Different examples of mesh adaptivity applied to fluid flow and solute transport in homogeneous and heterogeneous porous media are available in the literature (see, e.g., the review in [34]). In the context of flow simulation in heterogeneous media, the advantages resulting from an a priori refinement of an otherwise static mesh have been quantified in field scale flow simulation [36] as well as in transport problems [23]. In [30], a moving mesh algorithm for the modeling of a three-dimensional flow in the subsurface is proposed. A two-dimensional technique based on local refinement of hierarchical meshes is presented in [9]. The mesh adaptivity is here driven by an a posteriori error estimator for the energy norm. An example of anisotropic adaptivity is provided in [51], where a dynamic mesh adaptation for a reactive transport problem is proposed. The directional features of the solution are taken into account via local refinement/coarsening error indicators. In particular, the authors deal with structured and rectangular grids in a hierarchical framework.

Time step adaptivity has also been applied to the error control in the context of transient transport phenomena in porous media. These include density driven flows (e.g., [16, 54, 55]), flow in unsaturated media (e.g., [29]) and reactive transport processes [49].

In this contribution, we aim at combining mesh with time step adaptivity for modeling solute transport in porous media. To the best of our knowledge, this represents a first attempt in this applicative context. To maximize the advantages deriving from adapted meshes, we resort to an anisotropic mesh adaptivity (see, e.g., [12, 26, 45]). Size, orientation and shape of the elements are optimized to match the directional features of the problem at hand. We base our work on the methodology used in [46] for simulating unsteady shallow water problems. Here, the adaptive procedure relies on a recovery-based a posteriori estimator for the global (i.e., space–time) error. The contribution of space and time approximation errors is kept separated following, e.g., [11, 37, 39, 50] so that the space and the time discretization grids are sequentially and independently adapted. The spatial mesh adaptivity is grounded on the a posteriori error estimate proposed in [40], which essentially represents the anisotropic counterpart of the error estimator originally proposed by Zienkiewicz and Zhu in [56]. This yields a computationally cheap, problem-independent error estimator, which has been already applied successfully to different two- and three-dimensional problem settings [18, 41, 47]. The time step adaptivity is derived through the recovery-based technique devised in [46]. This methodology allows us estimating the time approximation error upon relying on a higher-order local reconstruction of time derivatives. Finally, this paper is enriched by comparing the results provided by the proposed numerical approach with actual experimental data similar to those presented in [32]. This is a key contribution of this paper, and allows us discussing the impact of space-time adaptive simulation methodologies on the interpretation of experiments.

The paper is organized as follows. Section 2 introduces the flow and solute transport equations and the adopted finite-element discretization. In Sect. 3, we provide the reference anisotropic setting and define the recovery-based space-time error estimator. Section 4 provides the procedures adopted to predict the new space-time adapted mesh together with the coupling strategy used to combine the solver in Sect. 2.1 with the whole adaptive procedure. Finally, in Sect. 5 we deal with the numerical validation, by considering first a benchmark analytical test case and then by performing a comparison with the experimental results. A discussion with concluding remarks ends the paper.

## 2 Solute Transport Modeling

Solute transport in porous media is typically described through a standard advection-dispersion equation (ADE). While the transport phenomenon is intrinsically three-dimensional, we assume here that the variation of solute concentration along the vertical direction may be neglected. This approximation is typically acceptable in the context of the laboratory settings that we consider in this work and allows us casting the transport equation into a two-dimensional (planar) framework [3].

Let $\Omega \subset \mathbb{R}^2$ be a bounded polygonal domain with boundary $\partial\Omega$, and $[0, T]$ a time window of interest. The ADE reads

$$\frac{\partial C}{\partial t} + \nabla \cdot (\mathbf{v}C) - \nabla \cdot (\mathbf{D}\nabla C) = 0 \quad \text{in } \Omega \times (0, T], \tag{1}$$

where $C = C(\mathbf{x}, t)[\text{mol/m}^3]$ is the (unknown) vertically averaged solute concentration at location $\mathbf{x}$ and at time $t$, $\mathbf{v} = (v_1, v_2)^T[\text{m/s}]$ is the fluid velocity and $\mathbf{D} = \{D_{ij}\}$ is the symmetric positive definite dispersion tensor. Following [3], this tensor is typically defined as

$$D_{ij} = (\alpha_T \, \|\mathbf{v}\|_2 + D_m)\, \delta_{ij} + (\alpha_L - \alpha_T)\, \frac{v_i v_j}{\|\mathbf{v}\|_2} \quad \text{with } i, j = 1, 2 \,, \tag{2}$$

where $\alpha_T, \alpha_L[\text{m}]$ are the transverse and the longitudinal dispersivity, respectively, $\delta_{ij}$ is Kronecker's delta symbol, $D_m[\text{m}^2/\text{s}]$ is the molecular diffusion and $\|\mathbf{w}\|_2$ denotes the standard Euclidean norm of a generic vector $\mathbf{w} \in \mathbb{R}^2$. Equation (1) is completed with a suitable set of initial and boundary conditions which, in general, coincides with relations as

$$\begin{cases} C\left(\mathbf{x},0\right) = C_0\left(\mathbf{x}\right) & \text{for } \mathbf{x} \in \Omega, \\ C\left(\mathbf{x},t\right) = f_1\left(\mathbf{x},t\right) & \text{for } \mathbf{x} \in \Gamma_1,\ t \in (0,T], \\ -\left(\mathbf{D}\nabla C\right)\cdot\mathbf{n} = f_2\left(\mathbf{x},t\right) & \text{for } \mathbf{x} \in \Gamma_2,\ t \in (0,T], \\ \left(\mathbf{v}C - \mathbf{D}\nabla C\right)\cdot\mathbf{n} = f_3\left(\mathbf{x},t\right) & \text{for } \mathbf{x} \in \Gamma_3,\ t \in (0,T], \end{cases} \tag{3}$$

where $\Gamma_1, \Gamma_2$ and $\Gamma_3$, with $\cup_{i=1}^{3}\Gamma_i = \partial\Omega$, $\overset{\circ}{\Gamma_i} \cap \overset{\circ}{\Gamma_j} = \emptyset$, for $i, j = 1, 2, 3$ and $i \neq j$, represent a partition of the boundary $\partial\Omega$ associated with Dirichlet, Neumann and Robin boundary conditions, respectively, $C_0$ is the initial value of the solute concentration, $f_i$, with $i = 1, 2, 3$, are the boundary data, and $\mathbf{n}$ is the unit outward normal vector to $\partial\Omega$.

The velocity field $\mathbf{v}$ in (1) is typically obtained through the numerical approximation of the following equations

$$\begin{cases} \mathbf{v} = -\dfrac{k}{\mu\phi}\left(\nabla p + \rho g \mathbf{k}\right) & \text{for } \mathbf{x} \in \Omega, \\ \nabla\cdot\mathbf{v} = 0 & \text{for } \mathbf{x} \in \Omega, \\ \mathbf{v}\cdot\mathbf{n} = \psi & \text{for } \mathbf{x} \in \partial\Omega, \end{cases} \tag{4}$$

where $p$[Pa] is the pressure, $g$[m/s$^2$] is the gravity, $\mu > 0$[Pa·s] and $\rho > 0$[Kg/m$^3$] are the fluid viscosity and density, respectively, $k > 0$[m$^2$] is the porous medium permeability, $0 < \phi < 1$ is the porosity, $\psi$ is a flux imposed on the domain boundary and $\mathbf{k}$ denotes the unit vector aligned with the vertical direction. In particular, we assume $\mu$, $\rho$, $k$, $\phi$ real constants. Equation $(4)_1$ coincides with Darcy's law coupled with the continuity equation $(4)_2$, while equation $(4)_3$ models an imposed flux. Notice that, via the divergence theorem, we obtain $\int_{\partial\Omega} \psi \, ds = 0$. Since Eq. (4) is steady, we are assuming to deal with a time independent field $\mathbf{v}$ in (1).

## 2.1 The Finite Element Discretization

In this section, we provide the finite element formulation used to discretize problem (1)–(4). We first introduce the discretization of the flow problem (4) by resorting to a mixed two-field formulation (see, e.g., [15, 35, 44]). For this purpose, we consider the following function spaces

$$\begin{gathered} V = H\left(div, \Omega\right) = \left\{\mathbf{v} \in [L^2\left(\Omega\right)]^2 \ :\ \nabla\cdot\mathbf{v} \in L^2\left(\Omega\right), \text{trace}(\mathbf{v}\cdot\mathbf{n}) = \psi \in H^{-1/2}(\partial\Omega)\right\}, \\ W = H_0\left(div, \Omega\right) = \left\{\mathbf{w} \in [L^2\left(\Omega\right)]^2 \ :\ \nabla\cdot\mathbf{w} \in L^2\left(\Omega\right), \text{trace}(\mathbf{w}\cdot\mathbf{n}) = 0 \text{ on } \partial\Omega\right\}, \\ P = L_0^2\left(\Omega\right) = \left\{p \in L^2\left(\Omega\right) \ :\ \int_\Omega p \, d\Omega = 0\right\}, \end{gathered} \tag{5}$$

where trace$(\cdot)$ denotes the standard trace operator. The demand on the average of $p$ in $P$ is due to the fact that the pressure field in (4) is involved in a gradient form. As a consequence, to guarantee the uniqueness of $p$, we have to constrain it with a condition. In practice, we specify the value of $p$ at a certain point in $\Omega$ instead of implementing condition $\int_\Omega p\, d\Omega = 0$. For all the details concerning the spaces in (5), we refer to [6].

Thus, for given values of the physical parameters $\mu$, $k$, $\phi$, $\rho$ and $\psi$, the weak formulation of (4) reads: find $\mathbf{v} \in V$ and $p \in P$ such that, for any $\mathbf{w} \in W$ and $q \in P$,

$$\int_\Omega \left( \frac{\mu\phi}{k} \mathbf{v} \cdot \mathbf{w} - p \nabla \cdot \mathbf{w} + \nabla \cdot \mathbf{v} q \right) d\Omega = - \int_\Omega (\rho g \mathbf{k} \cdot \mathbf{w})\, d\Omega. \tag{6}$$

For sufficiently regular data, the weak formulation (6) is known to have a unique solution. To discretize problem (6), we introduce a conformal partition $\mathscr{T}_h = \{K\}$ of the domain $\Omega$ into triangles $K$ of diameter $h_K$ [13]. Then, we define the spaces

$$\begin{aligned} V_h &= \left\{ \mathbf{v}_h \in [L^2(\Omega)]^2 \ : \ \mathbf{v}_h\big|_K \in \mathbb{RT}_0(K), \forall K \in \mathscr{T}_h, \mathrm{trace}(\mathbf{v}_h \cdot \mathbf{n}) = \psi_h \right\}, \\ W_h &= \left\{ \mathbf{w}_h \in [L^2(\Omega)]^2 \ : \ \mathbf{w}_h\big|_K \in \mathbb{RT}_0(K), \forall K \in \mathscr{T}_h, \mathrm{trace}(\mathbf{w}_h \cdot \mathbf{n}) = 0 \right\}, \end{aligned}$$

where $\mathbb{RT}_0$ denotes the space of the zero-order Raviart–Thomas finite elements [6], while $\psi_h$ is a piecewise constant approximation of the trace $\psi$. The space $V_h$ is employed to discretize the velocity field, while $W_h$ is used to discretize the test functions. This choice leads us to select the space

$$P_h = \left\{ p_h \in P \ : \ p_h\big|_K \in \mathbb{P}_0(K), \forall K \in \mathscr{T}_h \right\},$$

for the pressure, $\mathbb{P}_0$ being the set of the polynomials of degree zero. Thus, the $\mathbb{RT}_0 - \mathbb{P}_0$ discretization of problem (6) reads: find $\mathbf{v}_h \in V_h$ and $p_h \in P_h$ such that, for any $\mathbf{w}_h \in W_h$ and $q_h \in P_h$,

$$\begin{aligned} &\sum_{K \in \mathscr{T}_h} \left\{ \int_K \left( \frac{\mu\phi}{k} \mathbf{v}_h \cdot \mathbf{w}_h - p_h \nabla \cdot \mathbf{w}_h + \nabla \cdot \mathbf{v}_h q_h \right) dK \right\} \\ &\qquad = \sum_{K \in \mathscr{T}_h} \left\{ - \int_K (\rho g \mathbf{k} \cdot \mathbf{w}_h)\, dK \right\}. \end{aligned} \tag{7}$$

Notice that, the combination $\mathbb{RT}_0 - \mathbb{P}_0$ of shape functions for the velocity and pressure does satisfy the Babuška-Brezzi stability condition. Moreover, since we assume to deal with a steady velocity field, we solve problem (7) only once, for a prescribed set of data and before dealing with ADE (1).

The discretization of the ADE moves from the weak formulation of Eq. (1): for any $t \in (0, T]$, find $C\ (t) \in Z$ such that, for any $z \in Z$,

$$\int_{\Omega} \left( \frac{\partial C(t)}{\partial t} z + (\mathbf{D}\nabla C(t) - \mathbf{v}C(t)) \cdot \nabla z \right) d\Omega = 0, \tag{8}$$

where $Z$ coincides with a subspace of the Sobolev space $H^1(\Omega)$, suitably modified to satisfy the essential boundary conditions assigned on $\partial\Omega$. On the other hand, the imposition of natural boundary conditions leads to modify accordingly the right-hand side in (8) by introducing suitable integral boundary terms.

The spatial discretization of (8) is obtained via a streamline upwind technique (see, e.g., [7]) to damp the spurious oscillations yielded by the standard Galerkin finite element approximation. Let $Z_h = \{z_h \in C^0(\overline{\Omega}) : z_h\big|_K \in \mathbb{P}^1(K),\ \forall K \in \mathcal{T}_h\} \cap Z$ be the space of the affine finite elements associated with the partition $\mathcal{T}_h$. Thus, the streamline upwind finite element discretization of (8) is: for any $t \in (0, T]$, find $C_h(t) \in Z_h$ such that, for any $z_h \in Z_h$,

$$\sum_{K \in \mathcal{T}_h} \left\{ \int_K \left( \frac{\partial C_h(t)}{\partial t} z_h + (\mathbf{D}_h \nabla C_h(t) - \mathbf{v}_h C_h(t)) \cdot \nabla z_h \right) dK \right. \\ \left. + Q_K \int_K (\mathbf{v}_h \cdot \nabla C_h(t))\,(\mathbf{v}_h \cdot \nabla z_h)\ dK \right\} = 0, \tag{9}$$

where $Q_K = \delta_K / |\mathbf{v}_h|$ represents the stabilization coefficient associated with element $K$, $\delta_K$ being a suitable coefficient proportional to the element dimension (we refer to Sect. 3.1 for an explicit expression of $\delta_K$). Notice that the dispersion tensor $\mathbf{D}$ and the advective field $\mathbf{v}$ in (8) are here replaced by the discrete counterparts $\mathbf{D}_h$ and $\mathbf{v}_h$, respectively. In particular, the discrete tensor $\mathbf{D}_h$ is computed via (2) after substituting $\mathbf{v}$ with $\mathbf{v}_h$. Moreover, the meshes employed to discretize (6) and (9) may differ. In such a case, $\mathbf{v}_h$ will be projected on the $\mathbb{P}_1$ degrees of freedom.

Finally, the full discretization of problem (8) is obtained by discretizing the time dependence in (9) via the standard $\theta$-method. For this purpose, we introduce a partition of the time window [0,$T$] by fixing the time levels $\{t^0, \ldots, t^n\}$, with $t^0 \equiv 0$ and $t^n \equiv T$, which identify the set $\{I_k\}_{k=0}^{n-1}$ of the time intervals $I_k$ of width $\Delta t^k = t^{k+1} - t^k$, for $k = 0, \ldots, n-1$. To guarantee the unconditionally absolute stability of the $\theta$-method, we set $\theta$ equal to $2/3$. This choice relieves us from any constraint in the choice of the time step in order to avoid the occurrence of spurious oscillations in the discrete solution. This is a crucial issue in view of the time adaptivity procedure in Sect. 4.2.

## 3 A Space-Time Recovery-Based Error Estimator

In this section, we provide the theoretical tools used to drive the space-time adaptive procedure. In particular, after introducing the framework used to settle the anisotropic mesh adaptivity, we furnish the a posteriori estimators for the control of the space and of the time discretization errors.

### 3.1 *The Anisotropic Setting*

Following the setting proposed in [21, 22], the anisotropic information is derived by introducing the standard invertible affine map $T_K : \hat{K} \to K$ which transforms the equilateral triangle $\hat{K}$ with vertices $(-\sqrt{3}/2, -1/2)$, $(\sqrt{3}/2, -1/2)$, $(0, 1)$ into the generic triangle $K \in \mathscr{T}_h$. The triangle $\hat{K}$ is inscribed in the unit circle centered at $(0, 0)$. The map $T_K$ changes this circle into an ellipse circumscribing $K$, as shown in Fig. 1. In particular, we have

$$\mathbf{x} = T_K(\hat{\mathbf{x}}) = M_K\hat{\mathbf{x}} + \mathbf{t}_K \quad \forall \mathbf{x} = (x_1, x_2)^T \in K,$$

with $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2)^T \in \hat{K}$, $M_K \in \mathbb{R}^{2\times 2}$ the Jacobian associated with the map $T_K$ and $\mathbf{t}_K \in \mathbb{R}^2$ a shift vector. We exploit the spectral properties of $M_K$ to describe the anisotropic features of the element $K$. To do this, first we introduce the polar decomposition $M_K = B_K Z_K$ of $M_K$, with $B_K \in \mathbb{R}^{2\times 2}$ a symmetric positive definite matrix and $Z_K \in \mathbb{R}^{2\times 2}$ an orthogonal matrix. Then, we consider the spectral decomposition $B_K = R_K^T \Lambda_K R_K$ of $B_K$, where $R_K^T = [\mathbf{r}_{1,K}, \mathbf{r}_{2,K}] \in \mathbb{R}^{2\times 2}$ is the matrix of the right eigenvectors of $B_K$ and $\Lambda_K = \text{diag}(\lambda_{1,K}, \lambda_{2,K})$ is the diagonal matrix of the corresponding eigenvalues, where we assume $\lambda_{1,K} \geq \lambda_{2,K}$.

Thus, shape, size and orientation of $K$ are fully described by the quantities $\mathbf{r}_{i,K}$ and $\lambda_{i,K}$ for $i = 1, 2$. In particular, $\mathbf{r}_{1,K}$ and $\mathbf{r}_{2,K}$ identify the directions of the two semi-axes of the ellipse circumscribing $K$, while $\lambda_{1,K}$ and $\lambda_{2,K}$ measure the length of these semi-axes (see Fig. 1). The aspect ratio $s_K = \lambda_{1,K}/\lambda_{2,K} \geq 1$ quantifies the

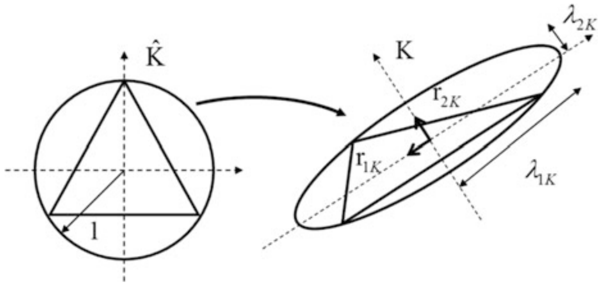


**Fig. 1** Geometric interpretation of the map $T_K$

deformation of triangle $K$. In particular, if $K$ coincides with an equilateral triangle (the isotropic case), $s_K = 1$ while $s_K$ increases as the shape of $K$ stretches.

Now, we provide the anisotropic interpolation error estimate that inspired the estimator introduced in Sect. 3.2.1 for the spatial discretization error. Let $I_h^1$ be the Clément quasi-interpolant [14] defined, in the case of linear finite elements, by

$$I_h^1(u)(\mathbf{x}) = \sum_{N_j \in \mathscr{N}} P_j u(N_j)\, \varphi_j(\mathbf{x}) \quad \forall u \in L^2(\Omega),$$

where $\mathscr{N}$ is the set of the mesh vertices except the ones belonging to the Dirichlet portion $\Gamma_1$, $\varphi_j$ is the Lagrangian basis function associated with the node $N_j$, and where $P_j u$ denotes the affine function associated with the patch $\Delta_j$ of the elements sharing node $N_j$ defined by the relations

$$\int_{\Delta_j} \big(P_j u - u\big)\, \vartheta \, d\Delta_j = 0 \quad \text{with } \vartheta = 1, x_1, x_2.$$

The local feature of the employed error estimator leads us to introduce the restriction $I_K^1$ of $I_h^1$ to $K$, such that $I_K^1(u\big|_K) = I_h^1(u)\big|_K$ for any $u \in L^2(\Omega)$ and for any $K \in \mathscr{T}_h$. Moving from [21], we can state the following result

**Lemma 1** *Let $\Delta_K = \{T \in \mathscr{T}_h \,:\, T \cap K \neq \emptyset\}$ be the patch of elements sharing at least a vertex with $K$, $\Delta_{\hat{K}} = T_K^{-1}(\Delta_K)$ be the reference patch obtained by mapping back the whole $\Delta_K$ via the map $T_K$ and let $u \in H^1(\Omega)$. Then, if card$(\Delta_K) \leq \mathscr{M}$ and diam$(\Delta_{\hat{K}}) \leq \mathscr{N}_\Delta$, there exists a constant $C = C(\mathscr{M}, \mathscr{N}_\Delta)$ such that*

$$\|u - I_K^1(u)\|_{L^2(K)} \leq C \Big[ \sum_{i=1}^{2} \lambda_{i,K}^2 \big(\mathbf{r}_{i,K}^T G_K(u) \mathbf{r}_{i,K}\big) \Big]^{1/2}, \tag{10}$$

*where $G_K(u) \in \mathbb{R}^{2\times 2}$ is the symmetric positive semi-definite matrix given by*

$$G_K(u) = \sum_{T \in \Delta_K} \begin{bmatrix} \displaystyle\int_T \Big(\frac{\partial u}{\partial x_1}\Big)^2 dT & \displaystyle\int_T \frac{\partial u}{\partial x_1}\frac{\partial u}{\partial x_2}\, dT \\ \displaystyle\int_T \frac{\partial u}{\partial x_1}\frac{\partial u}{\partial x_2}\, dT & \displaystyle\int_T \Big(\frac{\partial u}{\partial x_2}\Big)^2 dT \end{bmatrix}. \tag{11}$$

□

We highlight the explicit dependence of estimate (10) on the quantities $\mathbf{r}_{i,K}$ and $\lambda_{i,K}$, $i = 1, 2$, which allows us to characterize anisotropically the triangle $K$, i.e., to fix its size, shape and orientation. Moreover, the conditions on $\Delta_K$ and $\Delta_{\hat{K}}$ just avoid too distorted patches in the reference framework without introducing any limit on the anisotropic features of the mesh (we refer to [42] for examples of acceptable and not acceptable patches). When $\lambda_{1,K} \simeq \lambda_{2,K} \simeq h_K$, we recover the standard isotropic result [14]

$$\|u - I_K^1(u)\|_{L^2(K)} \leq C\, h_K\, |u|_{H^1(\Delta_K)}. \tag{12}$$

Note that the diameter $h_K$ in (12) is replaced by the lengths $\lambda_{i,K}$ in (10), while the components of the $H^1$-seminorm $|u|_{H^1(\Delta_K)}$ are projected along the anisotropic directions $\mathbf{r}_{i,K}$ via the terms $\mathbf{r}_{i,K}^T G_K(u)\mathbf{r}_{i,K}$.

Finally, we exploit the anisotropic spacing to fix the coefficient $\delta_K$ characterizing the stabilization coefficient $Q_K$ in (9). Following [42], we set $\delta_K = \lambda_{2,K}/2$.

## *3.2 Recovery-Based Error Estimators*

Recovery-based error estimators have been originally proposed by Zienkiewicz and Zhu in the framework of linear elasticity [56–58]. The key idea underlying this class of error estimators is to improve the accuracy of the gradient of a numerical solution through suitable interpolation or averaging techniques, generally known as gradient recovery procedures. In a displacement-based finite element approach, the demand for a sharper numerical gradient arises from the necessity to deal with derived fields (e.g., stresses or strain rates) at least as accurate as the primary ones (e.g., displacements), due to their significant physical meaning in practical applications.

As a byproduct of the gradient recovery procedure, in [58] Zienkiewicz and Zhu propose an a posteriori error estimator for the $H^1$-seminorm of the discretization error, simply defined as the $L^2$-norm of the difference between the recovered and the numerical gradient. This estimator is usually referred to as recovery-based. Many good properties characterize the recovery-based error estimators: they depend only on the adopted discrete space, being completely independent of the considered problem, of the governing equations and of the other details characterizing the adopted discrete formulation (e.g., the stabilization scheme); they are robust, easy to implement and cheap in terms of computational cost, since their definition involves only the numerical solution and the corresponding gradient. These properties make recovery-based error estimators a practical tool in view of an adaptive procedure and justify the extensive employment of these estimators in diverse applicative fields (see, e.g., [5, 8, 28, 33, 43]). More recently, an anisotropic version of the recovery-based error estimators has been proposed in [40] and successfully employed in two-dimensional as well as in three-dimensional settings [18, 41, 46, 47]. This anisotropic generalization allows us to combine the good properties of a recovery-based estimator with the richness of information needed for an anisotropic error analysis.

A theoretical investigation of the recovery-based error estimators is a recurrent but not yet very well understood issue in the literature, even in the simplest isotropic case [10, 31, 38, 53].

Following [46], in this section we resort to a recovery-based error estimator to adapt both the spatial mesh and the time step. For this purpose, we introduce an estimator $\eta_{ht}^A$ to control the global (i.e., in space and time) discretization error where the space and time contributions are separate (see also [11, 37, 39, 50]), namely, such that

$$\eta_{ht}^A = \eta_h^A + \eta_t, \tag{13}$$

with $\eta_h^A$ and $\eta_t$ the space and time recovery-based error estimator, respectively. In particular, our interest in anisotropic adapted meshes leads us to identify $\eta_h^A$ with an anisotropic error estimator. Estimators $\eta_h^A$ and $\eta_t$ are formulated separately in the two next sections.

### 3.2.1 The Spatial Error Estimator

Let us consider a generic time-dependent scalar variable $z$ and let $z_h$ be the corresponding linear finite element approximation. For any $t > 0$, we aim at providing an anisotropic estimate for the $H^1$-seminorm $\left|e_h^z(t)\right|^2_{H^1(\Omega)} = \int_\Omega |\nabla z(t) - \nabla z_h(t)|^2\, d\Omega$ of the discretization error $e_h^z(t) = z(t) - z_h(t)$. In an isotropic framework, the idea is to compute $e_h^z(t)$ by replacing the (generally) unknown exact gradient $\nabla z(t)$ with a suitable recovered gradient $P(\nabla z_h(t))$ [56–58], so that

$$\left|e_h^z(t)\right|^2_{H^1(\Omega)} \simeq \int_\Omega |P(\nabla z_h(t)) - \nabla z_h(t)|^2\, d\Omega = \left[\eta^{ZZ}(e_h^z(t))\right]^2. \tag{14}$$

To provide an anisotropic variant of estimator $\eta^{ZZ}(e_h^z(t))$, we refer to the anisotropic interpolation error estimate (10). In practice, by comparing estimates (10) and (12) we observe that the matrix $G_K(u)$ provides the anisotropic counterpart of the standard isotropic $H^1$-seminorm of $u$. This equivalence can be exploited for the definition of an anisotropic recovery-based error estimator, since $\eta^{ZZ}(e_h^z(t))$ in (14) coincides exactly with the $L^2$-norm of the *recovered error* on the gradient $E(z_h(t)) = P(\nabla z_h(t)) - \nabla z_h(t)$. Thus, following [18, 40] and moving from definitions (10) and (14) we identify the spatial recovery-based anisotropic estimator to be replaced in (13) with

$$\left[\eta_h^A(e_h^z(t))\right]^2 = \sum_{K\in\mathscr{T}_h} \left[\eta_{K,h}^A(z_h(t))\right]^2, \tag{15}$$

where

$$\left[\eta_{K,h}^A(z_h(t))\right]^2 = \frac{1}{\lambda_{1,K}\lambda_{2,K}} \sum_{i=1}^{2} \lambda_{i,K}^2 \left(\mathbf{r}_{i,K}^T G_K(E_K(z_h(t)))\mathbf{r}_{i,K}\right) \tag{16}$$

denotes the local error estimator, with $E_K(z_h(t)) = E(z_h(t))\big|_K$. The scaling factor $(\lambda_{1,K}\lambda_{2,K})^{-1}$ ensures the consistency with respect to the isotropic case, i.e., when we choose $\lambda_{1,K} = \lambda_{2,K}$, we get the isotropic estimator

$$\left[\eta_h^I(e_h^z(t))\right]^2 = \sum_{K\in\mathcal{T}_h} \left[\eta_{K,h}^I(z_h(t))\right]^2 \quad \text{with } \left[\eta_{K,h}^I(z_h(t))\right]^2 = \int_{\Delta_K} |E_K(z_h(t))|^2 \, d\Delta_K.$$

Finally, the quantity $z_h$ involved in (15) strictly depends on the problem at hand and identifies the physical quantity used to drive the spatial mesh adaptivity. Estimator (15) may be applied to more general problems involving not only scalar quantities, such as the elasticity or the Navier-Stokes equations [18].

We detail in the following the adopted gradient recovery procedure. Several recipes have been provided in the literature to compute the recovered gradient (see, e.g., [10, 33, 48, 57]). Following [40], we resort here to a very simple approach which defines a patchwise constant recovered gradient. We let

$$P_{\Delta_K}(\nabla z_h)(\mathbf{x},t) = \frac{1}{|\Delta_K|} \sum_{T\in\Delta_K} |T|\, \nabla z_h(t)\big|_T \quad \text{with } \mathbf{x} \in K,\ t > 0, \tag{17}$$

with $|\varpi|$ the measure of the generic set $\varpi \subset \mathbb{R}^2$, namely we compute the area-weighted average over the whole patch $\Delta_K$ of the gradients of the discrete solution and then we assign such a value to the single element $K$ (see Fig. 2, left for a sketch of the recovery procedure). Thus, for any element $K' \in \Delta_K$ with $K' \neq K$, $P_{\Delta_{K'}}(\nabla z_h)$ is, in general, different from $P_{\Delta_K}(\nabla z_h)$, and it is constant on the patch $\Delta_{K'}$. In view of a practical implementation, the time $t$ in (17) coincides with a time level $t^k$ of the time partition $\{t^0, \ldots, t^n\}$. The recovery procedure in (17) can be generalized to higher degree polynomials as shown in [40]. Here, to contain the computational costs, we adopt the simplest choice, i.e., $z_h$ coincides with a piecewise linear function.

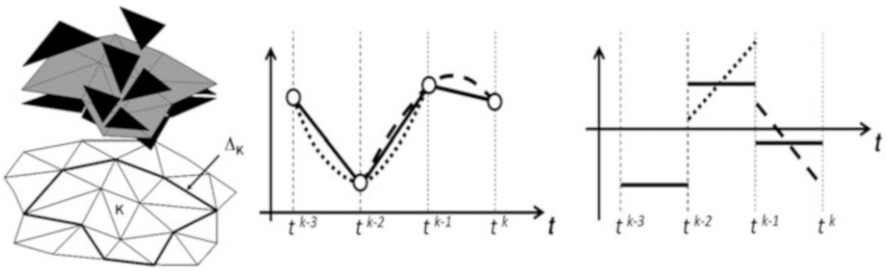


**Fig. 2** Spatial gradient recovery procedure (*left*): gradient of the discrete solution (*black*) and recovered gradient (*grey*); time gradient recovery (*center*): recovered solution $z^R$ (*dotted* and *dashed lines*) and linear interpolant of values $z_h^j$ (*continuous line*); time derivatives (*right*): $\partial z^R/\partial t$ (*dotted* and *dashed lines*), $\partial z_h/\partial t$ (*continuous line*)

Despite the heuristic nature of the anisotropic estimator $\eta_h^A(e_h^z(t))$, a corresponding theoretical background is furnished in [40], where a patch test on the estimator is provided and an equivalence relation between the local estimator $\eta_{K,h}^A(z_h(t))$ and the $H^1$-seminorm of the discretization error on the patch $\Delta_K$ is derived.

Finally, estimator $\eta_h^A(e_h^z(t))$ has been successfully extended and assessed in a three-dimensional setting in [18, 41].

### 3.2.2 The Time Error Estimator

The actual goal pursued in our time adaptive procedure is to predict, at the generic time level $t^k$, the time step $\Delta t^k$ identifying interval $I_k$. Following [46], this prediction is carried out by defining a local time error estimator instead of the global one $\eta_t$ in (13). As a consequence, we observe that the number $n$ of time intervals is not known a priori but it will be fixed by the adaptive procedure itself. As in Sect. 3.2.1, we consider a generic scalar variable $z$ together with the corresponding linear finite element approximation $z_h$. Then, in the spirit of a recovery-based error estimator, we move from the $H^1$-seminorm of the time discretization error on the time interval $I_{k-1}$

$$\left|e_h^z(\mathbf{x})\right|^2_{H^1(I_{k-1})} = \int_{I_{k-1}} \left|\frac{\partial z(\mathbf{x})}{\partial t} - \frac{\partial z_h(\mathbf{x})}{\partial t}\right|^2 dt. \tag{18}$$

Of course, at the generic time level $t^k$, we know $z_h$ at all the previous times $t^i$, with $i = 0, \ldots, k-1$. Now, the idea is to replace the two time derivatives in (18) with two easily computable quantities. We approximate the derivative of the discrete solution, $\partial z_h(\mathbf{x})/\partial t$, by replacing $z_h$ with the straight line interpolating $z_h$ at $t^{k-1}$ and $t^k$ (see Fig. 2, center), so that

$$\left.\frac{\partial z_h(\mathbf{x})}{\partial t}\right|_{I_{k-1}} \simeq \frac{z_h^k - z_h^{k-1}}{\Delta t^{k-1}},$$

with $z_h^j = z_h(\mathbf{x}, t^j)$ for $j = 0, \ldots, n$. To replace $\partial z(\mathbf{x})/\partial t$, we first substitute $z$ with a suitable recovered solution $z^R$ and then we compute $\partial z(\mathbf{x})/\partial t$ as $\partial z^R(\mathbf{x})/\partial t$. In particular, we select $z^R$ as the parabola interpolating the pairs of values $(t^{k-2}, z_h^{k-2})$, $(t^{k-1}, z_h^{k-1})$, $(t^k, z_h^k)$ (see Fig. 2, center). Notice that this choice leads to a piecewise linear recovered gradient in contrast to the piecewise constant discrete quantity $\partial z_h(\mathbf{x})/\partial t$ (see Fig. 2, right), in agreement with a standard recovery-based approach. Thus, the $H^1$-seminorm in (18) is approximated via the local time recovery-based error estimator $\eta_{k-1,t}(z_h(\mathbf{x}))$ as

$$\left|e_h^z(\mathbf{x})\right|^2_{H^1(I_{k-1})} \simeq \tilde{T} \int_{I_{k-1}} \left|\left.\frac{\partial z^R(\mathbf{x})}{\partial t}\right|_{I_{k-1}} - \frac{z_h^k - z_h^{k-1}}{\Delta t^{k-1}}\right|^2 dt = \left[\eta_{k-1,t}(z_h(\mathbf{x}))\right]^2, \tag{19}$$

with $\tilde{T}$ a suitable time scale factor typical of the problem at hand. Different choices are possible for $\tilde{T}$. It may just coincide with the time step $\Delta t^{k-1}$ or it may be related to geometrical and/or physical quantities, e.g., we can choose $\tilde{T} = L/w$, with $L$ a characteristic length of the domain $\Omega$ and $w$ a representative velocity. In the numerical validation of Sect. 5, we always choose $\tilde{T} = \Delta t^{k-1}$. Factor $\tilde{T}$ essentially makes the time estimator dimensionless, i.e., suited to be added to the dimensionless space estimator $\eta_h^A(e_h^z(t))$ in view of (13). Estimator $\eta_{k-1,t}(z_h(\mathbf{x}))$ can be evaluated at each vertex $N$ of the mesh, i.e., for any $\mathbf{x} \equiv N$. Nevertheless, we need in practice a unique value for such an estimator on the interval $I_{k-1}$. With this aim, we first obtain a unique value on the generic triangle $K \in \mathcal{T}_h$ simply by considering the mean of the values at the three vertices of $K$

$$\left[\eta_{k-1,t,K}(z_h)\right]^2 = \frac{1}{3} \sum_{N \in K} \left[\eta_{k-1,t}(z_h(N))\right]^2.$$

Then, we lump the information on the whole mesh by introducing the further area-weighted average

$$\left[\eta_{k-1,t}(z_h)\right]^2 = \frac{1}{\sum\limits_{K \in \mathcal{T}_h} |K|} \sum_{K \in \mathcal{T}_h} |K| \left[\eta_{k-1,t,K}(z_h)\right]^2. \tag{20}$$

In [46] we employ a standard sum over the mesh elements to get the value $\eta_{k-1,t}(z_h)$. The variant in (20) normalizes the time error estimator with respect to the domain dimension. This makes the element time estimators comparable even in the presence of strongly non-uniform spatial meshes. Finally, in view of the time adaptive procedure in the next section, we formally introduce the global time error estimator

$$\left[\eta_t(e_h^z)\right]^2 = \sum_{k=1}^{n} \left[\eta_{k-1,t}(z_h)\right]^2 \tag{21}$$

which can be introduced in (13). Notice that the number $n$ of time subintervals involved in (21) represents an unknown of the space-time adaptive procedure.

## 4 The Solution-Adaptation Procedure

We illustrate here how we combine the discretization of ADE (1) with the information provided by the error estimators $\eta_h^A(e_h^z(t))$ and $\eta_t(e_h^z(\mathbf{x}))$ to automatically adapt the spatial mesh $\mathcal{T}_h$ and the discretization of the time window $[0, T]$ to model an unsteady solute transport process. We aim at guaranteeing the total error below a certain global tolerance $\tau$. Following [39], we split this tolerance by setting a space and a time tolerance, $\tau_h$ and $\tau_t$, respectively such that $\tau = \tau_h + \tau_t$. Tolerance $\tau_h$

will drive the mesh adaptive procedure detailed in Sect. 4.1, while $\tau_t$ will lead us to identify the next time step via the predictive procedure in Sect. 4.2. The algorithm coupling these two adaptive procedures with the discrete solver for (1) is then provided in Sect. 4.3.

## *4.1 Spatial Mesh Adaptivity*

We resort to a well-established metric-based adaptive procedure, following the approach proposed in [22] and successfully employed in a number of works (see, e.g., [18, 39, 40, 47]). In particular, the adapted mesh is built starting from a metric induced by the error estimator $\eta_h^A(e_h^z(t))$ so that the number of mesh elements is minimized and the tolerance $\tau_h$ is guaranteed on $\eta_h^A(e_h^z(t))$ via an error equidistribution criterion.

According to the generic definition, a metric is a symmetric positive definite tensor field $\mathscr{M} : \Omega \to \mathbb{R}^{2\times 2}$, such that $\mathscr{M}(\mathbf{x}) = \tilde{R}^T(\mathbf{x})\tilde{\Lambda}^{-2}(\mathbf{x})\tilde{R}(\mathbf{x})$ for any $\mathbf{x} \in \Omega$, with $\tilde{\Lambda}(\mathbf{x}) = \mathrm{diag}(\tilde{\lambda}_1(\mathbf{x}), \tilde{\lambda}_2(\mathbf{x}))$ and $\tilde{R}^T(\mathbf{x}) = [\tilde{\mathbf{r}}_1(\mathbf{x}), \tilde{\mathbf{r}}_2(\mathbf{x})]$ a positive diagonal and an orthogonal tensor, respectively [24]. For an assigned mesh $\mathscr{T}_h$, it is a standard practice to approximate the pointwise tensors $\tilde{\Lambda}(\mathbf{x})$ and $\tilde{R}(\mathbf{x})$ via quantities which are piecewise constant on $\mathscr{T}_h$, so that $\tilde{\lambda}_i(\mathbf{x})\big|_K = \tilde{\lambda}_{i,K}$, $\tilde{\mathbf{r}}_i(\mathbf{x})\big|_K = \tilde{\mathbf{r}}_{i,K}$ for any $K \in \mathscr{T}_h$ and for $i = 1, 2$.

Now, we briefly explain how to associate a piecewise constant metric $\mathscr{M}^\eta$ with a background grid $\mathscr{T}_h^B$ by exploiting the anisotropic spatial error estimator in (15)–(16) evaluated on $\mathscr{T}_h^B$. For the sake of simplicity, we drop the time dependence in (15)–(16). In particular, we focus on the local estimator $\eta_{K,h}^A(z_h)$. We properly rewrite it, by collecting the area information in a unique multiplicative factor, i.e., as

$$\left[\eta_{K,h}^A(z_h)\right]^2 = |\Delta_{\hat{K}}|\,\lambda_{1,K}\lambda_{2,K}\left\{s_K\left(\mathbf{r}_{1,K}^T G_K^*(E_K(z_h))\mathbf{r}_{1,K}\right) + s_K^{-1}\left(\mathbf{r}_{2,K}^T G_K^*(E_K(z_h))\mathbf{r}_{2,K}\right)\right\} \tag{22}$$

where $G_K^*(E_K(z_h)) = G_K(E_K(z_h))/|\Delta_K|$, with $|\Delta_K| = |\Delta_{\hat{K}}|\,\lambda_{1,K}\lambda_{2,K}$ and $\Delta_{\hat{K}}$ defined as in Lemma 1. Now, the idea is to apply an error equidistribution criterion to guarantee that each element $K \in \mathscr{T}_h^B$ provides the same contribution to the global error, i.e., to ensure that $\left[\eta_{K,h}^A(z_h)\right]^2 = \tau_{loc}^2$, where $\tau_{loc} = \tau_h/\mathrm{card}(\mathscr{T}_h^B)$ is the local tolerance, with $\mathrm{card}(\mathscr{T}_h^B)$ the cardinality of the background grid. At the same time, we aim at minimizing the number of mesh elements, which is equivalent to maximizing the area of each triangle $K$. Consequently, we minimize the term in brackets in (22), i.e., for each $K \in \mathscr{T}_h^B$, we solve the minimization problem

$$\text{find } \{s_K, \mathbf{r}_{1,K}\} : s_K\left(\mathbf{r}_{1,K}^T G_K^*(E_K(z_h))\mathbf{r}_{1,K}\right) + s_K^{-1}\left(\mathbf{r}_{2,K}^T G_K^*(E_K(z_h))\mathbf{r}_{2,K}\right) \text{ is minimum,}$$

constrained by the following requirements $s_K \geq 1$, $\|\mathbf{r}_{1,K}\|_2 = \|\mathbf{r}_{2,K}\|_2 = 1$, $\mathbf{r}_{1,K} \cdot \mathbf{r}_{2,K} = 0$. As shown in [38], there exists a unique analytical solution to this problem which provides the *optimal* aspect ratio $\tilde{s}_K^{\eta} = \sqrt{\gamma_{1,K}/\gamma_{2,K}}$ and the *optimal* direction $\tilde{\mathbf{r}}_{1,K}^{\eta} = \mathbf{g}_{2,K}$, with $\gamma_{1,K} \geq \gamma_{2,K}$ the eigenvalues of $G_K^*(E_K(z_h))$ and $\mathbf{g}_{2,K}$ the eigenvector associated with $\gamma_{2,K}$. The metric $\mathscr{M}^{\eta}$ is then completely identified by deriving two separate *optimal* values $\tilde{\lambda}_{1,K}^{\eta}$, $\tilde{\lambda}_{2,K}^{\eta}$ from $\tilde{s}_K^{\eta}$. With this aim, we exploit the error equidistribution to get $\tilde{\lambda}_{1,K}^{\eta} = \sqrt{p\, \tilde{s}_K^{\eta}}$, $\tilde{\lambda}_{2,K}^{\eta} = \sqrt{p/\tilde{s}_K^{\eta}}$, with

$$p = \tilde{\lambda}_{1,K}^{\eta}\, \tilde{\lambda}_{2,K}^{\eta} = \tau_{loc}^2 \left[ |\Delta_{\hat{K}}| \, (\tilde{s}_K^{\eta}\, \gamma_{2,K} + (\tilde{s}_K^{\eta})^{-1}\, \gamma_{1,K}) \right]^{-1}. \tag{23}$$

To summarize, the metric $\mathscr{M}^{\eta}$ induced by $\eta_h^A(e_h^z(t))$ is univocally identified by the triplets $\{\tilde{\lambda}_{1,K}^{\eta}, \tilde{\lambda}_{2,K}^{\eta}, \tilde{\mathbf{r}}_{1,K}^{\eta}\}$, with $K \in \mathscr{T}_h^B$. This provides us with an *optimal* metric, which minimizes the number of elements, while guaranteeing the desired global accuracy $\tau_h$ and the equidistribution of the spatial discretization error.

Finally, the new adapted mesh is generated moving from the metric $\mathscr{M}^{\eta}$ and the background grid $\mathscr{T}_h^B$. To this end, we employ the two-dimensional metric-based mesh generator BAMG which performs a remesh on $\mathscr{T}_h^B$, although trying to preserve the original position of the mesh nodes [27].

The generation of the adapted mesh is constrained by some additional checks. In particular, we avoid an excessive element clustering, for instance where solution discontinuities occur, by setting a minimum value $p_{min}$ on the product $\tilde{\lambda}_{1,K}^{\eta}\, \tilde{\lambda}_{2,K}^{\eta}$, i.e., on the minimum area allowed for $K$. In practice, we predict $p$ via (23) and then we set $p = \max(p, p_{min})$. We also check the number of the triangles predicted by the metric $\mathscr{M}^{\eta}$. We impose that the cardinality of the adapted mesh belongs to a specific interval $[\mathscr{N}_{min}, \mathscr{N}_{max}]$, to prevent an extreme coarsening or refinement of the mesh elements. In particular, we predict the number of elements associated with the metric $\mathscr{M}^{\eta}$; if this number is smaller than $\mathscr{N}_{min}$ or greater than $\mathscr{N}_{max}$, we generate a new metric via a global and uniform scaling of the tensor $\mathscr{M}^{\eta}$. This check turns out to be crucial especially in the presence of unsteady phenomena characterized by a strong heterogeneity of space-time dynamics.

## 4.2 *Time Step Adaptivity*

Goal of the time step adaptivity here proposed is to predict, at each time $t^k$, the time step $\Delta t^k$ which identifies the next time level $t^{k+1}$. An error equidistribution criterion is pursued also in this case. Nevertheless, since the total number of time intervals is determined only at the end of the adaptive procedure, we fix a local tolerance $\tau_t^{\Delta t}$ to be associated with each $I_k$ instead of the global tolerance $\tau_t$ as in (13). Following [46], we rewrite the time error estimator in (20) as

$$\left[\eta_{k-1,t}(z_h)\right]^2 = \left[\Delta t^{k-1} \rho_{k-1,t}(z_h)\right]^2 \tag{24}$$

with

$$\left[\rho_{k-1,t}(z_h)\right]^2 = \frac{1}{\left[\Delta t^{k-1}\right]^2 \sum_{K\in\mathcal{T}_h^k} |K|} \sum_{K\in\mathcal{T}_h^k} |K| \left[\eta_{k-1,t,K}(z_h)\right]^2$$

and where $\mathcal{T}_h^k$ denotes the spatial mesh associated with the time $t^k$. Relation (24) is now exploited to predict $\Delta t^k$. In particular, after imposing $\eta_{k-1,t}(z_h) = \tau_t^{\Delta t}$, we solve (24) with respect to the time step and we obtain

$$\Delta t^k = \frac{\tau_t^{\Delta t}}{\rho_{k-1,t}(z_h)}. \tag{25}$$

Finally, we check that the time step predicted in (25) belongs to a suitable range of variation $[\Delta t_{min}, \Delta t_{max}]$, fixed a priori according to the temporal scales involved in the problem at hand. This control improves the global stability and accuracy of the whole adaptive procedure. Finally, since the time recovery procedure (19) involves the three times $t^k$, $t^{k-1}$, $t^{k-2}$, we are obliged to assign a priori the time steps $\Delta t^0$ and $\Delta t^1$, which are both set to $\Delta t_{min}$ in the numerical validation below.

## *4.3 Coupling Discretization with Adaptivity*

We detail here the strategy followed to combine the discretization in Sect. 2.1 with the space and time adaptive procedures of Sects. 4.1, 4.2, in view of a reliable and efficient modeling of unsteady solute transport processes. For this purpose, we identify the scalar variable $z$ driving the adaptive procedure with the concentration $C$.

To start the solution-adaptation procedure, we have to preliminarily assign the initial datum $C_0$ in (3) as well as the velocity field $\mathbf{v}$, solution to Darcy's problem (4), on a sufficiently fine initial grid $\mathcal{T}_h^0$. We focus now on the generic time $t^{k-1}$, i.e., we assume to know the discrete concentration field $C_h^{k-1} = C_h(t^{k-1})$ at time $t^{k-1}$, the mesh $\mathcal{T}_h^{k-1}$ and the time step $\Delta t^{k-1}$, both predicted at the previous time step. First of all, we discretize Eq. (1) on the interval $I_{k-1}$ via the stabilized finite element scheme (9), thus yielding the approximate solution $C_h^{*k}$ at time $t^k$. This solution is employed to generate the next adapted mesh, $\mathcal{T}_h^k$, following the anisotropic adaptive procedure detailed in Sect. 4.1, after identifying $z_h$ with $C_h^{*k}$ and $\mathcal{T}_h^B$ with $\mathcal{T}_h^{k-1}$. Then, all the quantities associated with $\mathcal{T}_h^{k-1}$ are projected on the new mesh $\mathcal{T}_h^k$. This last projection leads us to define the actual value, $C_h^k$, of the concentration field at the time $t^k$. Notice that, to contain the computational cost characterizing the whole time window, we do not resort to an iterative procedure to get the adapted mesh $\mathcal{T}_h^k$, by demanding, for instance, a stagnation of the number of mesh elements (see, e.g., [39]). On the contrary, the mesh identified by the optimal

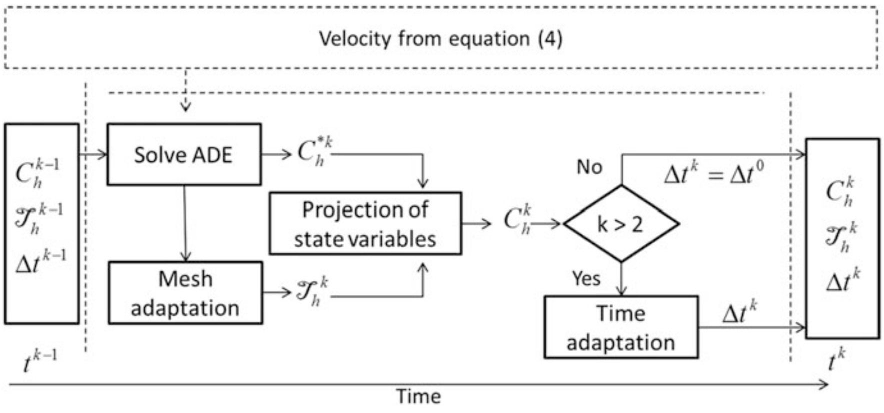


**Fig. 3** Sketch of the solution-adaptation procedure

metric in Sect. 4.1 is directly assumed as the mesh to be associated with time $t^k$. Finally, moving from the approximate solutions $C_h^k$, $C_h^{k-1}$ and $C_h^{k-2}$, we predict the next time step $\Delta t^k$ (i.e., the new time level $t^{k+1}$) via the time adaptive procedure in Sect. 4.2, after setting $z_h = C_h$ and a local time tolerance $\tau_t^{\Delta t}$. The whole procedure is sketched in Fig. 3.

Concerning the projection step, following [46], we resort to a standard $L^2$-projection which exhibits good conservation properties when applied, for instance, to the unsteady shallow water equations [47]. More sophisticated recipes to deal with such a projection are available in the literature (see, e.g., [1, 19]).

## 5 Numerical Results

We assess here the methodology introduced in Sects. 3–4. In particular, we first perform a quantitative investigation by considering a test case characterized by an analytical solution. Then, we model a solute transport experiment performed in a homogeneous sand box. For this test case, a comparison with experimental measurements is provided as well.

### *5.1 Analytical Test Case*

We consider a two-dimensional solute transport problem assigned in the semi-infinite rectangular domain $\Omega = (0, +\infty) \times (0, W)$[m]. A constant solute concentration $C^{in} = 1$ is introduced into the domain at the inlet section $\Gamma^{in}$ corresponding to the segment $\{(x_1, x_2) : x_1 = 0\,\text{m}, x_2 \in [Y_1, Y_2]\,\text{m}\}$. A zero-flux boundary condition is imposed on the remaining part of the left-side boundary as well as on the top and bottom edges. Then, we assume that a horizontal uniform velocity field $\mathbf{v} = (v_1, 0)^T$, with $v_1 \geq 0$, is applied to the system, while we choose $D_m = 0$, $\alpha_L = 0.1$ m and $\alpha_T = 0.05$ m in (2) and the initial value $C_0 = 0$. An implicit analytical expression for the concentration field $C$, solution to (1), is available in [52] and coincides with

$$C(x_1, x_2, t) = \sum_{n=0}^{+\infty} L_n P_n \cos(\eta x_2) \left\{ \exp\left[\frac{x_1(v_1-\beta)}{2D_{11}}\right] \operatorname{erfc}\left[\frac{x_1-\beta t}{2\sqrt{D_{11}t}}\right] + \exp\left[\frac{x_1(v_1+\beta)}{2D_{11}}\right] \operatorname{erfc}\left[\frac{x_1+\beta t}{2\sqrt{D_{11}t}}\right] \right\}, \tag{26}$$

with

$$L_n = \begin{cases} \frac{1}{2} & \text{if } n = 0 \\ 1 & \text{if } n > 0, \end{cases} \qquad P_n = \begin{cases} \dfrac{Y_2 - Y_1}{W} & \text{if } n = 0 \\ \dfrac{[\sin(\eta Y_2) - \sin(\eta Y_1)]}{n\pi} & \text{if } n > 0, \end{cases}$$

$$\eta = n\pi / W, \qquad \beta = \sqrt{v_1^2 + 4D_{11}(\eta^2 D_{22})},$$

where erfc is the complementary error function, while $D_{11}$, $D_{22}$ denote the diagonal components of the dispersion tensor $\mathbf{D}$ according to (2). In particular, we set $Y_1 = 0.13$ m, $Y_2 = 0.67$ m, $W = 1$m and $v_1 = 10^{-3}$m/s. These values, together with the choices made for $\alpha_T$ and $\alpha_L$, are representative of a typical laboratory scale transport setting.

The analytical solution (26) allows us to investigate the convergence properties of the proposed adaptive procedure. We perform this analysis for a fixed time level $t^* = 150$ s, when the solute concentration $C$ already exhibits a moderate spread within the domain as shown in Fig. 4-a. Figure 4-b shows the corresponding

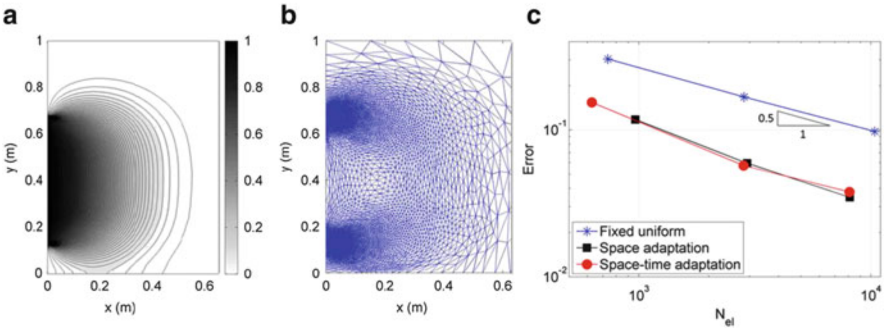


**Fig. 4** Analytical test case: (**a**) contour plot of the concentration field in (26) for $t^* = 150$s; (**b**) corresponding anisotropic adapted mesh; (**c**) trend of the $H^1(\Omega)$-seminorm of the relative error for different space-time meshes

**Table 1** Analytical test case: quantitative information associated with the convergence analysis

| | Fixed uniform | | | Space adaptivity | | | Space-time adaptivity | | |
|---|---|---|---|---|---|---|---|---|---|
| $N_{el}$ | 732 | 2,482 | 10,388 | 967 | 2,928 | 8,104 | 627 | 2,829 | 8,081 |
| $N_{\Delta_t}$ | 150 | 150 | 150 | 150 | 150 | 150 | 54 | 36 | 32 |
| $\lvert e_h^C(t^*)\rvert_{H^1}$ | $3 \cdot 10^{-1}$ | $1.70 \cdot 10^{-1}$ | $9.7 \cdot 10^{-2}$ | $1.18 \cdot 10^{-1}$ | $5.93 \cdot 10^{-2}$ | $3.47 \cdot 10^{-2}$ | $1.54 \cdot 10^{-1}$ | $5.72 \cdot 10^{-2}$ | $3.78 \cdot 10^{-2}$ |

anisotropic adapted mesh. The anisotropic features of the solution are not so marked. This is confirmed by the corresponding maximum value of the aspect ratio which is about equal to 8. We analyze the trend of the $H^1(\Omega)$-seminorm of the spatial discretization error $e_h^C(t^*) = C(t^*) - C_h(t^*)$ at time $t^*$. Concerning the evaluation of the exact solution, we truncate the series (26) at the hundredth term while, to mimic the semi-infinite domain, we identify $\Omega$ with a rectangular domain whose right side is sufficiently far from the left one so that it is never reached by the phenomenon at hand. Homogeneous Neumann condition are assigned along this fictitious boundary. In Fig. 4-c we compare the trend of the error $e_h^C(t^*)$ as a function of the number of elements. We consider three different space-time discretization strategies: i) fixed and uniform space-time grids; ii) meshes anisotropically adapted in space, but fixed and uniform in time; iii) meshes adapted both in space and time, with an anisotropic adaptivity in space. The slope of the three curves is comparable and consistent with the expected order of convergence, i.e., $-1/2$, while it is evident that, for a given number of mesh elements, the error associated with the fixed space-time mesh is the largest one.

More quantitative information are provided in Table 1. For each type of mesh, we furnish the mesh cardinality, $\#\mathscr{T}_h$, the number, $\mathscr{N}_{\Delta t}$, of time steps used to reach time $t^*$ and the $H^1(\Omega)$-seminorm of the error $e_h^C(t^*)$. Of course, a different tolerance $\tau_h$ and different values for $\mathscr{N}_{min}$ and $\mathscr{N}_{max}$ drive the mesh adaptive procedure through the columns of the table. The choice $p_{min} = 10^{-5}$ is common to all the simulations. The time step is set to $\Delta t = 1$s for the fixed uniform and space adaptive simulations (first two macro-columns in Table 1), while we set $\Delta t_{min} = 1$ s and $\Delta t_{max} = 20$ s for the space-time adaptive procedure (i.e., in the third macro-column in Table 1). An error of approximately 10 % is obtained with a fixed space-time grid of approximately 10,000 triangles and 150 time intervals. On the other hand, for the same $\mathscr{N}_{\Delta t}$, we are able to guarantee a better accuracy (i.e., an error of about 6 %) via an anisotropic adapted mesh consisting of about 3,000 triangles only. When also the time step is adapted, we obtain a comparable error by resorting to a similar number of elements but reducing $\mathscr{N}_{\Delta t}$ by a factor four. A comparison between the second and the third macro-column shows that for a comparable number of mesh elements, the number of time intervals predicted by the time adaptive procedure is significantly reduced (i.e., by a factor comprised between three and five).

## 5.2 Solute Transport in a Homogeneous Sand Box

Goal of this section is to compare the results provided by our numerical approach with experimental data. This provides an important added value to this work.

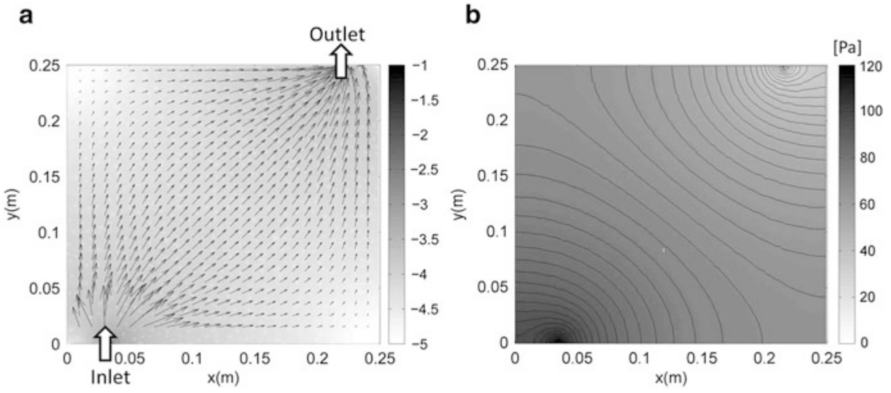


**Fig. 5** Experimental flow cell: (**a**) velocity vector field superimposed to the color map of $\log_{10} \|\mathbf{v}\|_2$; (**b**) pressure contour plot

The experiments are performed within a square flow cell with an extension of $0.249 \times 0.249\,\text{m}^2$ and a thickness of 1.5 cm. The inlet and outlet sections are located near the bottom-left and the top-right corner, respectively as shown in Fig. 5-a. The width of the inflow and outflow sections is very small, equal to 3 mm. The flow cell is packed with a homogeneous sand with porosity $\phi = 0.325$ and permeability $k = 1.4256 \cdot 10^{-10}\,\text{m}^2$ while, since the fluid is simply water, we set the viscosity $\mu$ and density $\rho$ to $10^{-3}$Pa·s and $1,000\,\text{Kg/m}^3$, respectively [32]. The solute concentration in the system is zero at the beginning of the experiment. A solution containing a constant concentration $C^{in}$ is then introduced as a step-input at the inlet, while a zero flux condition is imposed on the remaining part of $\partial\Omega$. The injection flow rate is constant in time and equal to $Q^{in} = 4$ ml/s. The available experimental measurements correspond to the time evolution of the average concentration at the outflow section, i.e., they coincide with the breakthrough curve

$$C_{out}(t) = \frac{1}{|\Gamma_{out}|} \int_{\Gamma_{out}} C(\mathbf{x}, t)\, d\Gamma \quad \forall t \in [0, T], \tag{27}$$

where $\Gamma_{out}$ is the outflow section and $|\Gamma_{out}|$ denotes its length. We consider here data from two identical experimental tests to increase the robustness of the results with respect to measurement errors.

The velocity field $\mathbf{v}$ is obtained by approximating (4) through the discrete formulation (7) on a fixed uniform unstructured grid of 22,108 elements. Figure 5 shows the pressure contour plot in panel b), and the velocity vector field superimposed to the contour plot of the corresponding modulus in logarithmic scale in panel a). A constant (atmospheric) pressure is imposed at the outlet boundary, while we set $\mathbf{v} \cdot \mathbf{n} = Q^{in}/A^{in}$ at the inlet, $A^{in}$ being the area of the inflow cross section. The remaining parts of the boundary of the flow cell are considered as impermeable, i.e., we set $\mathbf{v} \cdot \mathbf{n} = 0$.

Then, the solute transport is modeled by means of the space-time adaptive procedure detailed in Sect. 4 within the time window [0, 12,000] s. The dispersivities

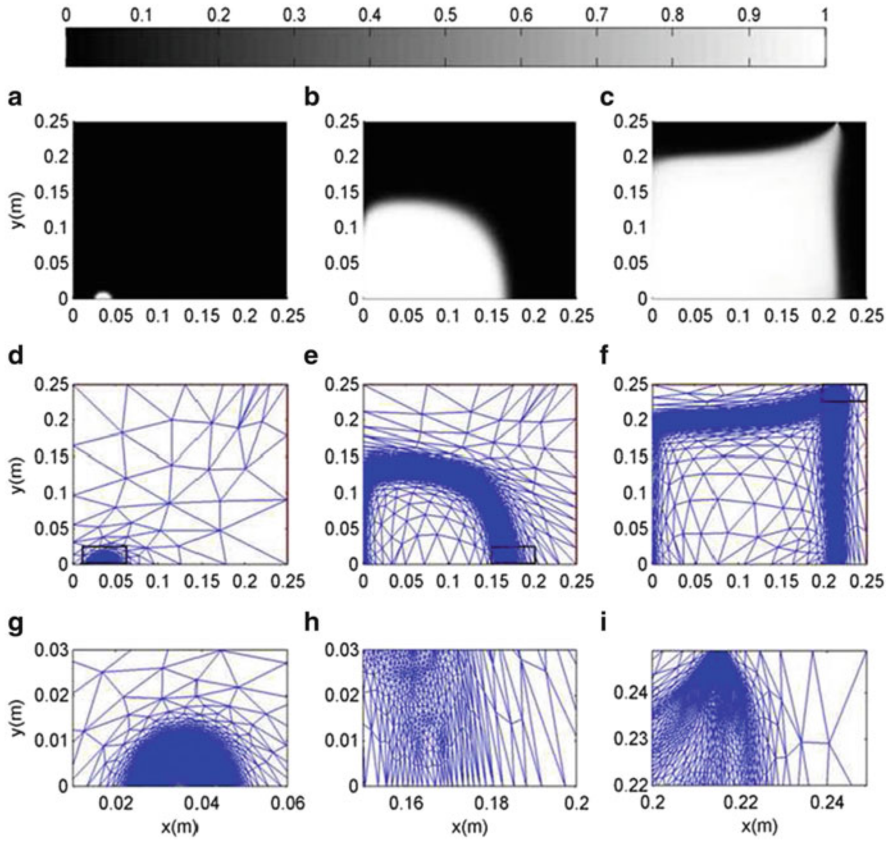


**Fig. 6** Experimental flow cell: solute concentration field (**a**)-(**c**); associated anisotropic adapted mesh (**d**)-(**f**); details of the adapted mesh (**g**)-(**i**) for $t = 10$s (*left*), $t = 1,500$s (*center*), $t = 3,000$s (*right*)

$\alpha_L$ and $\alpha_T$ are set to $10^{-4}$m and $10^{-5}$m, respectively, upon preliminary visual calibration against experimental data. The effect of molecular diffusion is here embedded in the dispersion coefficients, i.e., we set $D_m = 0$. In the following, we describe the process evolution, we provide a comparison with experimental measurements and, finally, we discuss the sensitivity of the numerical results to the parameters involved in the space-time adaptive technique.

Figure 6-a-c shows some snapshots of the concentration evolution. During the first instants the solute spreads radially into the cell, around the inflow section (Fig. 6-a). Then, the advective field and the dispersive processes deform and displace the concentration front towards the center of the cell (Fig. 6-b). At time $t \approx 3,000$ s, the solute reaches the outlet boundary following a preferential flow path (Fig. 6-c). For longer times, the solute slowly spreads towards the top-left and bottom-right corners of the cell. At the final time $T = 12,000$ s, the cell is characterized by a constant concentration equal to one. In Fig. 6-d-f we gather the corresponding anisotropic adapted meshes, while a zoom in on the boxed areas is provided in the last row of the figure. A good matching between solution and mesh can be observed. The refinement of the mesh essentially follows the advancing front, where the gradient variations are significant, while a general coarsening occurs where the concentration is basically uniform. The anisotropic features of the adapted mesh become more significant when the front develops and gradually spreads the cell (see the enlarged views in Fig. 6-h and 6-i). On the contrary, at the initial times the triangles are clustered around the inlet section but they exhibit a mild anisotropy, due to extremely reduced size of the concentration front. Thus, the maximum value of the aspect ratio increases in time, and corresponds to 26.5, 27.8 until 49.82 for the meshes displayed in Fig. 6-d, -e, -f, respectively. The maximum value reached by $s_K$ on the whole time window is 407.18. To obtain the results in Fig. 6, we have employed the following setting for the parameters involved in the space-time adaptive procedure: concerning the adaptivity in space we choose $\tau_h = 0.7$, $\mathscr{N}_{min} = 400$ and $\mathscr{N}_{max} = 10{,}000$ while the adaptivity in time is performed by assigning the values $\tau_t^{\Delta t} = 0.17$, $\Delta t_{min} = 1$ s and $\Delta t_{max} = 50$ s.

The numerical validation demonstrates that the value $p_{min}$ proportional to the minimum allowed element area plays a critical role in the adaptive procedure. In principle, the choice $p_{min} = 10^{-6}\text{m}^2$ should allow to capture all the meaningful details of the process at hand since the whole flow cell area is equal to $0.062\text{m}^2$. Nevertheless, this value is still too large to provide an accurate approximation of the solute behaviour at the outlet and inlet cross sections, which are 3mm wide. This represents a critical issue for two important reasons: (i) a sharp discretization of the concentration field at the inlet is crucial to capture the initial times of the solute evolution and, consequently, to allow the correct development of the phenomenon; (ii) a proper discretization of the concentration at the outlet is essential to compare the numerical results with the experimental measurements that we have at our disposal, i.e., with the breakthrough curve (27). To take into account both these demands, we introduce an adaptive choice of the quantity $p_{min}$ during the simulation time window, following this two-value strategy: for $t < 20$s, we set $p_{min} = 10^{-9}\text{m}^2$ to properly model the solution near the inflow section (see Fig. 6-d and 6-g); in the next phase, we increase $p_{min}$ to $10^{-6}\text{m}^2$ to save computational resources (see Fig. 6-e and 6-h); then, as soon as we detect at the outlet concentration values above a given threshold (set to $10^{-3}$ in this application), we set again $p_{min} = 10^{-9}\text{m}^2$ (see Fig. 6-f and 6-i).

We deal now with the comparison between the numerically computed and experimentally measured breakthrough curves. As for the previous test case, we consider three different space-time discretization strategies, namely, a fixed uniform space-time mesh, a mesh anisotropically adapted in space but fixed and uniform in time, a fully adapted mesh, i.e., adapted both in space and time. The fixed uniform mesh is characterized approximatively by $10{,}000$ elements, while the discretization step is $\Delta t = 1$ s. When we adapt the spatial mesh only, we set $\tau_h = 0.7$ and preserve the constant time step $\Delta t = 1$ s. Finally, the fully adaptive procedure is associated with the values $\tau_h = 0.7$ and $\tau_t^{\Delta t} = 0.14$.

Figure 7 compares the corresponding results with the experimental measurements. As shown in Fig. 7-a, the three simulations seem to correctly reproduce

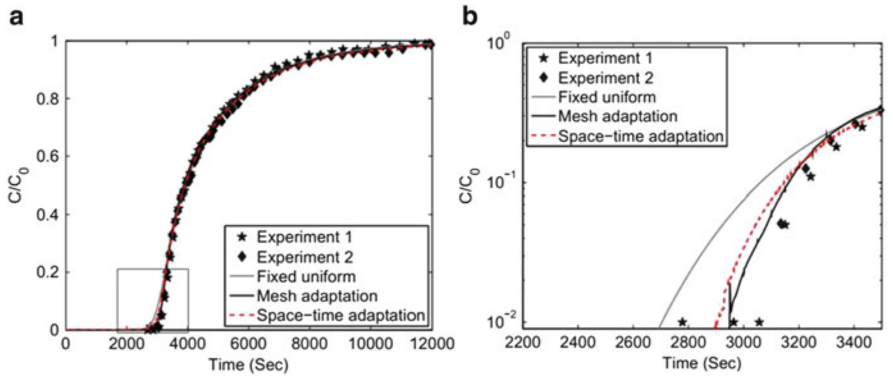


**Fig. 7** Experimental flow cell: (**a**) comparison of experimental measurements with the numerical breakthrough curves for three different choices of the space-time mesh; (**b**) enlarged view on the early solute breakthrough in semi-logarithmic scale

the general trend of the experimental measurements within the considered time window. However, a detailed inspection of the concentration trend at the initial times shows a significant difference among the three approximations (see Fig. 7-b, where data are plotted in a semi-logarithmic scale to emphasize the small values of the concentration characterizing the early tail of the solute breakthrough curve). Note that accurate modeling of the early solute arrivals is often crucial, for instance when dealing with risk assessment analysis in contaminant transport scenarios [2]. The data of the experiment 1 show two isolated positive concentration values at $t < 3{,}000$ s; then, a sudden increase of the concentration is observed at $t \approx 3{,}000$ s. In our analysis we assume that the early arrival time of the solute can be identified with $t \approx 3{,}000$ s, since the earlier values can be considered as oscillations which typically cannot be rendered by a continuum scale approximation such as the ADE (1). The solution obtained through a fixed space-time discretization under-estimates the solute early arrival time of about 400 s (i.e., of about 15 % of the experimental value). When the mesh is spatially adapted and the time step is fixed to $\Delta t = 1$ s, the difference between the numerical and the experimental breakthrough curves significantly reduces. However, this approach is computationally expensive, since the spatial mesh is adapted at each of the 12,000 time steps. The space-time adaptivity still provides a sufficiently accurate solution, while containing the whole computational cost as the total number of time steps reduces to 1, 232. To provide a quantitative assessment of the results, we compute the mean squared error (MSE) between the computed and the observed breakthrough concentrations depicted in Fig. 7. The MSE is defined as

$$MSE = \frac{\sum_{i=1}^{N_O} \left[C_{out,h}(t_i) - C^*_{out}(t_i)\right]^2}{N_O}, \tag{28}$$

where $C_{out}^*$ and $C_{out,h}$ are the experimentally measured and the numerically computed breakthrough concentrations, respectively, $N_O$ is the number of available experimental observations and $t_i$ denotes the time corresponding to the $i$-th observation. For the fixed uniform meshes we obtain a MSE of $6.417 \cdot 10^{-4}$. The MSE decreases to $2.983 \cdot 10^{-4}$ and $2.811 \cdot 10^{-4}$ when resorting to space and to space-time adaptive procedures, respectively. We observe that, while all three approximation strategies yield acceptable results in terms of MSE, the adaptive discretization strategies enable us to halve the MSE with respect to the fixed uniform discretization. Moreover, consistent with results presented in Sect. 5.1, we emphasize that the space-time discretization technique enables us to obtain a slight improvement of the accuracy with respect to the space adaptive methodology, while significantly reducing the computational cost.

Now, we assess the sensitivity of the numerical results to the tolerances, $\tau_h$ and $\tau_t^{\Delta t}$. Figure 8 compares the experimental data with the solutions associated with the three choices $\tau_h = 1, 1.4, 1.8$ for $\tau_t^{\Delta t} = 0.17$. In particular, Fig. 8-a provides the corresponding early breakthrough curves analogously to Fig. 7-b, while Fig. 8-b shows the evolution in time of the cardinality of the adapted meshes. The capability of the numerical breakthrough curves to reproduce experimental observations decreases as the tolerance increases while the trend becomes more and more irregular (see Fig. 8-a). In particular, the numerical curves locate leftward which means that the solute arrival time is underestimated by the numerical procedure. This can be justified by the considerable difference in terms of mesh cardinality among the three simulations, as shown in Fig. 8-b. For $\tau_h = 1$ the number of elements increases for $t \in [0, 2, 600]$ s; then, the maximum threshold $\mathcal{N}_{max} = 10,000$ is reached and maintained for the remaining part of the simulation time window. The increase in the number of elements is also associated with the decrease of the parameter $p_{min}$, from $10^{-6}$m$^2$ to $10^{-9}$m$^2$, as previously discussed. For $\tau_h = 1.4$, we get a similar oscillation in the mesh cardinality for

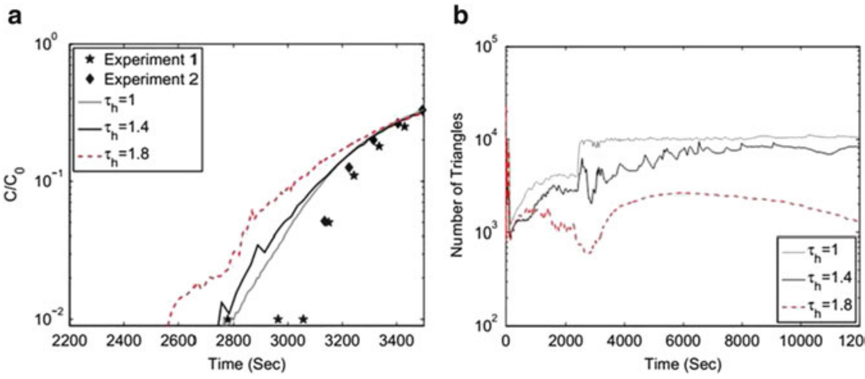


**Fig. 8** Experimental flow cell: (**a**) comparison of experimental measurements with the numerical breakthrough curves in a semi-logarithmic scale and (**b**) time evolution of the mesh cardinality for three different choices of the spatial tolerance $\tau_h$

$t < 2,600$ s. Nevertheless, in this case, the number of elements smoothly increases up to approximately 8,000 triangles at the end of the simulation and the maximum number $\mathcal{N}_{max}$ of elements is never reached. For $\tau_h = 1.8$, the number of mesh elements is around 1,000 for the entire simulation time. This significant difference with respect to the number of elements predicted for $\tau_h = 1$ is reflected in the computed breakthrough curve (Fig. 8-a). The strong oscillations exhibited by the numerical solution can be also ascribed to the reinterpolation step which involves coarse meshes for $\tau_h = 1.8$. The amplitude of the oscillations reduces for $\tau_h = 1.4$ and $\tau_h = 1$, namely in the presence of finer meshes.

Finally, we perform a sensitivity analysis to the local time tolerance $\tau_t^{\Delta t}$. Figure 9-a provides a comparison among the breakthrough curves and the experimental data for $\tau_h = 0.7$ and by choosing $\tau_t^{\Delta t} = 0.14, 0.17, 0.2$. As expected, the accuracy of the numerical solution decreases as the tolerance increases. In Fig. 9-b we show the evolution of the time step predicted for the three different choices of $\tau_t^{\Delta t}$. For $\tau_t^{\Delta t} = 0.2$, the time step quickly changes from the initial value $\Delta t_{min}$ to $\Delta t_{max}$ and then it remains constant for the rest of the simulation. For $\tau_t^{\Delta t} = 0.17$, the

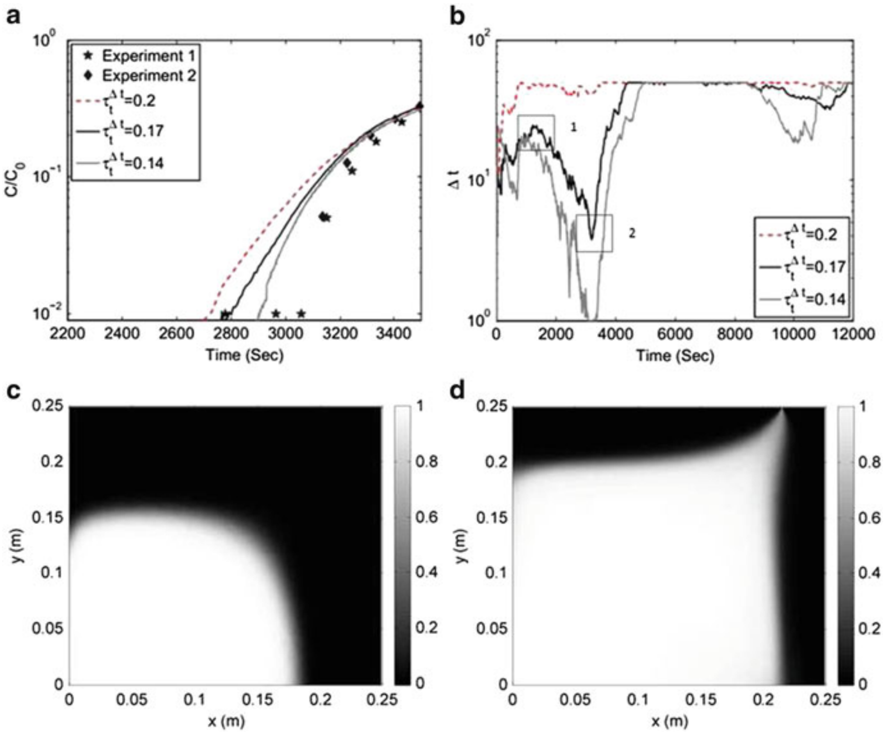


**Fig. 9** Experimental flow cell: (**a**) comparison of experimental measurements with the numerical breakthrough curves in a semi-logarithmic scale and (**b**) evolution of the time step $\Delta t$ for three different choices of the local time tolerance $\tau_t^{\Delta t}$; (**c**) concentration fields corresponding to box 1 and (**d**) to box 2

time step initially increases until reaching $\Delta t \approx 20\,\text{s}$ at $t \approx 1,500\,\text{s}$. The numerical solution corresponding to this time level is shown in Fig. 9-c. In this initial phase the concentration front has a reduced amplitude and the concentration is equal to zero in a considerable portion of the domain. At the same time, the concentration gradient at the front decreases due to the effect of dispersion. These features justify the trend of the time step observed in Fig. 9-b. As the front expands, we observe a progressive reduction of the time step,which attains a minimum value $\Delta t \approx 5\,\text{s}$ at $t \approx 3,000\,\text{s}$ when the solute reaches the outlet section (see Fig. 9-d), i.e., when the breakthrough curve starts to sharply increase in time. As time advances, the concentration field smoothly evolves and, for $t > 3,000\,\text{s}$, $\Delta t$ tends to the maximum allowed value $\Delta t_{max} = 50\,\text{s}$. A similar trend can be observed for $\tau_t^{\Delta t} = 0.14$, even though the predicted time steps are now, in general, smaller with respect to the ones associated with $\tau_t^{\Delta t} = 0.17$. For instance, $\Delta t$ reaches the minimum value $\Delta t_{min} = 1\,\text{s}$ around $t \approx 3,000\,\text{s}$ for a limited time interval.

## 6 Conclusions

Accurate numerical approximations are required to obtain reliable predictions of solute transport processes in porous media. In this paper, we detail an adaptive numerical methodology, where the computational space and time discretizations are automatically selected on the basis of a suitable error control. In particular, we implement an anisotropic mesh adaptation technique which allows us to optimize the spatial computational grid according to the directional features of the numerical solution. Both the adaptive procedures are grounded on recovery-based error estimators, which typically guarantee robust and computationally cheap error estimates.

A spatial mesh adaptivity significantly increases the accuracy of a finite element approximation for a fixed number of elements. This has been quantitatively verified in Sect. 5 by comparing the numerical results with an analytical solution as well as with experimental measurements. However, the proposed metric-based adaptive procedure may become computationally intensive, especially due to the prediction of the new metric. As a consequence, we have tried to contain the number of solution-adaptation iterations throughout each simulation, by combining space with time adaptivity. Results in Sect. 5 show that coupling mesh with time step adaptivity leads to a degree of accuracy similar to the one provided by the spatial adaptivity only, but it significantly reduces the number of demanded time steps.

The comparison against experimental data has allowed to assess the influence of the space-time discretization strategy on the accuracy of the simulation, quantified in terms of solute concentration at the outlet section (i.e., the solute breakthrough curve). In particular, we have focused on the analysis of the early solute arrival times, which are relevant in practical applications, such as risk assessment in contaminant transport analysis. It is shown that an adaptive space-time discretization is able to greatly improve the prediction of early solute arrival times with respect to a fixed uniform space-time discretization, clearly upon setting the same physical parameters in the differential problems describing flow and solute transport, respectively. This result suggests that the impact of an adaptive space-time discretization within parameter calibration and inverse modeling schemes may be relevant in laboratory and field scale applications.

An ad-hoc tuning of the parameters involved in the solution-adaptation procedure proved to be a key issue in view of meaningful results. In more detail, we constrain both the space and time adaptivity via a set of criteria which improve the robustness of the solution-adaptation approach. We resort to both global and local controls, by fixing the minimum and maximum number of mesh elements and a minimum value for the element area. Our results show that these controls have to be tuned according to the characteristic space-time scales of the problem at hand, as well as to the desired accuracy. In particular, a coarse mesh typically yields an oscillating numerical breakthrough curve, due to successive interpolations between coarse meshes at the outlet section. On the contrary, when the tolerance and the minimum element area are properly tuned, these oscillations attain a negligible amplitude (i.e., $O(10^{-2})$ or smaller).

Future investigations may lead to combine different information within the space-time adaptive strategy by properly intersecting distinct metrics, as proposed, e.g., in [47]. These metrics may be related to the numerical solution as well as to a target output of the simulation (e.g., the breakthrough curve), in the spirit of a goal-oriented approach [4, 17, 25]. Further extensions of this research will involve solute transport modeling in the presence of both block heterogeneous and random permeability fields, with a view to field scale applications.

**Acknowledgements** We kindly acknowledge Prof. Brian Berkowitz for sharing with us the experimental data. The financial support of MIUR (Project "Innovative methods for water resources under hydro-climatic uncertainty scenarios", PRIN 2010/2011) is also gratefully acknowledged.

## References

1. Alauzet, F., Mehrenberger, M.: P1-conservative solution interpolation on unstructured triangular meshes. Int. J. Numer. Methods Eng. **84**(13), 1552–1588 (2010)
2. Andričević, R., Cvetkovic, V.: Evaluation of risk from contaminants migrating by groundwater. Water Resour. Res. **32**(3), 611–621 (1996)
3. Bear, J., Cheng, A.H.-D.: Modeling Groundwater Flow and Contaminant Transport. Springer Dordrech (2010)
4. Becker, R., Rannacher, R.: An optimal control approach to a posteriori error estimation in finite element methods. Acta Numerica **10**, 1–102 (2001)
5. Bouillard, P., Allard, J.-F., Warzée, G.: Superconvergent patch recovery technique for the finite element method in acoustic. Commun. Numer. Methods Eng. **12**(9), 581–594 (1996)
6. Brezzi, F., Fortin, M.: Mixed and Hybrid Finite Element Methods. Springer, New York (1991)
7. Brooks, A.N., Hughes, T.J.R.: Streamline upwind/Petrov Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier Stokes equations. Comput. Methods Appl. Mech. Eng. **32**(1–3), 199–259 (1982)

8. Cai, Z., Zhang, S.: Recovery-based error estimators for interface problems: mixed and nonconforming finite elements. SIAM J. Numer. Anal. **48**(1), 30–52 (2010)
9. Cao, J., Kitanidis, P.K.: Adaptive-grid simulation of groundwater flow in heterogeneous aquifers. Adv. Water Resour. **22**(7), 681–696 (1999)
10. Carstensen, C., Bartels, S.: Each averaging technique yields reliable a posteriori error control in FEM on unstructured grids. I: low order conforming, nonconforming, and mixed FEM. Math. Comp. **71**, 945–969 (2001)
11. Cascón, J.M., Ferragut, L., Asensio, M.I.: Space-time adaptive algorithm for the mixed parabolic problem. Numer. Math. **103**(3), 367–392 (2006)
12. Castro-Diaz, M.J., Hecht, F., Mohammadi, B., Pironneau, O.: Anisotropic unstructured mesh adaptation for flow simulations. Int. J. Numer. Meth. Fluids **25**(4), 475–491 (1997)
13. Ciarlet, P.G.: The Finite Element Method for Elliptic Problems. North Holland, Amsterdam (1978)
14. Clément, Ph.: Approximation by finite element functions using local regularization. RAIRO Anal. Numér. **2**, 77–84 (1975)
15. Correa, M.R., Loula, A.F.D.: Unconditionally stable mixed finite element methods for Darcy flow. Comput. Methods Appl. Mech. Eng. **197**(17), 1525–1540 (2008)
16. Diersch, H.-J.: Finite element modelling of recirculating density-driven saltwater intrusion processes in groundwater. Adv. Water Resour. **11**(1), 25–43 (1988)
17. Eriksson, K., Estep, D., Hansbo, P., Johnson, C.: Introduction to adaptive methods for differential equations. Acta Numerica **4**, 105–158 (1995)
18. Farrell, P.E., Micheletti, S., Perotto, S.: An anisotropic Zienkiewicz-Zhu-type error estimator for 3D applications. Int. J. Numer. Methods Eng. **85**(6), 671–692 (2011)
19. Farrell, P.E., Piggott, M.D., Pain, C.C., Gorman, G.J., Wilson, C.R.: Conservative interpolation between unstructured meshes via supermesh construction. Comput. Methods Appl. Mech. Eng. **198**(33–36), 2632–2642 (2009)
20. Fetter, C.W.: Contaminant Hydrogeology, Waveland Prentice Hall, Englewood Cliffs (2008)
21. Formaggia, L., Perotto, S.: New anisotropic a priori error estimates. Numer. Math. **89**(4), 641–667 (2001)
22. Formaggia, L., Perotto, S.: Anisotropic error estimates for elliptic problems. Numer. Math. **94**(1), 67–92 (2003)
23. Gedeon, M., Mallants, D.: Sensitivity analysis of a combined groundwater flow and solute transport model using local-grid refinement: a case study. Math. Geosci. **44**(7), 881–899 (2012)
24. George, P.L., Borouchaki, H.: Delaunay Triangulation and Meshing: Application to Finite Elements. Hermes, Paris (1998)
25. Giles, M.B., Suli, E.: Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality. Acta Numerica **11**, 145–236 (2002)
26. Gruau, C., Coupez, T.: 3D tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric. Comput. Methods Appl. Mech. Engrg. **194**(48–49), 4951–4976 (2005)
27. Hecht, F.: New development in FreeFem++. J. Numer. Math. **20**(3–4), 251–265 (2012)
28. Katragadda, P., Grosse, I.R.: A posteriori error estimation and adaptive mesh refinement for combined thermal-stress finite element analysis. Comput. Struct. **59**(6), 1149–1163 (1996)
29. Kavetski, D., Binning, P., Sloan, S.W.: Adaptive backward Euler time stepping with truncation error control for numerical modelling of unsaturated fluid flow. Int. J. Numer. Methods Eng. **53**(6), 1301–1322 (2002)
30. Knupp, P.: A moving mesh algorithm for 3-D regional groundwater flow with water table and seepage face. Adv. Water Resour. **19**(2), 83–95 (1996)
31. Krìzek, M., Neittaanmäki, P.: Superconvergence phenomenon in the finite element method arising from average gradients. Numer. Math. **45**(1), 105–116 (1984)
32. Levy, M., Berkowitz, B.: Measurement and analysis of non-Fickian dispersion in heterogeneous porous media. J. Contam. Hydrol. **64**(3), 203–226 (2003)
33. Maisano, G., Micheletti, S., Perotto, S., Bottasso, C.L.: On some new recovery-based a posteriori error estimators. Comput. Methods Appl. Mech. Eng. **195**, 4794–4815 (2006)

34. Mansell, R.S., Ma, L., Ahuja, L.R., Bloom, S.A: Adaptive grid refinement in numerical models for water flow and chemical transport in soil. Vadose Zone J. **1**(2), 222–238 (2002)
35. Masud, A., Hughes, T.J.R.: A stabilized mixed finite element method for Darcy flow. Comput. Methods Appl. Mech. Eng. **191**(39), 4341–4370 (2002)
36. Mehl, S., Hill, M.C.: Development and evaluation of a local grid refinement method for block-centered finite-difference groundwater models using shared nodes. Adv. Water Resour. **25** (5), 497–511 (2002)
37. Meidner, D., Vexler, B.: Adaptive space-time finite element methods for parabolic optimization problems. SIAM J. Control Optim. **46**(1), 116–142 (2007)
38. Micheletti, S., Perotto, S.: Reliability and efficiency of an anisotropic Zienkiewicz-Zhu error estimator. Comput. Methods Appl. Mech. Eng. **195**(9), 799–835 (2006)
39. Micheletti, S., Perotto, S.: Anisotropic mesh adaption for time-dependent problems. Int. J. Numer. Methods Fluids. **58**(9), 1009–1015 (2008)
40. Micheletti, S., Perotto, S.: Anisotropic adaptation via a Zienkiewicz-Zhu error estimator for 2D elliptic problems. In: Kreiss, G., Lötstedt, P., Malqvist, A, Neytcheva, M. (eds.) Numerical Mathematics and Advanced Applications, pp. 645–653. Springer, New York (2010)
41. Micheletti, S., Perotto, S., Farrell, P.E.: A recovery-based error estimator for anisotropic mesh adaptation in CFD. SeMA J. **50**(1), 115–137 (2010)
42. Micheletti, S., Perotto, S., Picasso, M.: Stabilized finite elements on anisotropic meshes: a priori error estimates for the advection-diffusion and the Stokes problems. SIAM J. Numer. Anal. **41**(3), 1131–1162 (2003)
43. Mu, L., Jari, R.: A recovery-based error estimate for nonconforming finite volume methods of interface problems. Appl. Math. Comput. **220**, 63–74 (2013)
44. Nakshatrala, K.B., Turner, D.Z., Hjelmstad, K.D., Masud, A.: A stabilized mixed finite element method for Darcy flow based on a multiscale decomposition of the solution. Comput. Methods Appl. Mech. Eng. **195**(33–36), 4036–4049 (2006)
45. Peraire, J., Vahdati, M., Morgan, K., Zienkiewicz, O.: Adaptive remeshing for compressible flow computations. J. Comput. Phys. **72**(2), 449–466 (1987)
46. Porta, G.M., Perotto, S., Ballio F.: A space-time adaptation scheme for unsteady shallow water problems. Math. Comput. Simulat. **82**(12), 2929–2950 (2011)
47. Porta, G.M., Perotto, S., Ballio, F.: Anisotropic mesh adaptation driven by a recovery-based error estimator for shallow water flow modeling. Int. J. Numer. Methods Fluids **70**(3), 269–299 (2012)
48. Rodríguez, R.: Some remarks on Zienkiewicz-Zhu estimator. Numer. Methods Partial Differ. Equ. **10**, 625–635 (1994)
49. Saaltink, M.W., Carrera, J., Olivella S.: Mass balance errors when solving the convective form of the transport equation in transient flow problems. Water Resour. Res. **40**(5), W05107 (2004). doi:10.1029/2003WR002866
50. Schmich, M., Vexler, B.: Adaptivity with dynamic meshes for space-time finite element discretizations of parabolic equations. SIAM J. Sci. Comput. **30**(1), 369–393 (2008)
51. Sun, S., Wheeler, M.F.: Anisotropic and dynamic mesh adaptation for discontinuous Galerkin methods applied to reactive transport. Comput. Methods Appl. Mech. Eng. **195**(25), 3382–3405 (2006)
52. Wexler, E.J.: Analytical Solutions for One-, Two-, and Three-Dimensional Solute Transport in Ground-Water Systems with Uniform Flow. U.S. Geological Survey, vol. 3, Techniques of Water-Resources Investigations, Book, Chapter 7 (2005)
53. Yan, N.N., Zhou, A.: Gradient recovery type a posteriori error estimates for finite element approximations on irregular meshes. Comput. Methods Appl. Mech. Eng. **190** (32), 4289–4299 (2001)
54. Younes, A., Ackerer, P.: Empirical versus time stepping with embedded error control for density-driven flow in porous media. Water Resour. Res. **46**(8), W08523 (2010). doi:10.1029/2009WR008229

55. Younes, A., Ackerer, P., Lehmann, F.: A new efficient Eulerian-Lagrangian localized adjoint method for solving the advection-dispersion equation on unstructured meshes. Adv. Water Resour. **29**(7), 1056–1074 (2006)
56. Zienkiewicz, O.C., Zhu, J.Z.: A simple error estimator and adaptive procedure for practical engineerng analysis. Int. J. Numer. Methods Eng. **24** (2), 337–357 (1987)
57. Zienkiewicz, O.C., Zhu, J.Z.: The superconvergence patch recovery and a posteriori error estimates, Part 1: The recovery technique. Int. J. Numer. Methods Eng. **33** (7), 1331–1364 (1992)
58. Zienkiewicz, O.C., Zhu, J.Z.: The superconvergence patch recovery and a posteriori error estimates, Part 2: Error estimates and adaptivity. Int. J. Numer. Methods Eng. **33** (7), 1368–1382 (1992)

# A 2D Topology-Adaptive Mesh Deformation Framework for Mesh Warping

**Jibum Kim, David McLaurin, and Suzanne M. Shontz**

**Abstract** We propose a framework for performing anisotropic mesh deformations. Our goal is to produce high quality meshes with no inverted elements on domains which undergo large deformations. To the greatest extent possible, the meshes should have similar element shape; however, topological changes are performed as necessary in order to improve mesh quality. Our framework is based upon the previous work of two of the authors and their collaborators (Kim et al., Int. J. Numer. Methods Eng. 94(1):20–42, 2013; Kim et al., Computer and Mathematics with Applications, Submitted, November 2014) and consists of four steps. The first step is to perform anisotropic finite element-based mesh warping to estimate the interior vertex positions based upon an appropriate choice of the PDE coefficients. The second step is to perform multiobjective mesh optimization in order to eliminate inverted elements and improve element shape. Edge swaps are then performed to further improve the mesh quality. A final mesh smoothing pass is then performed. Our numerical results show that our framework can be used to generate high quality meshes with no inverted elements for very large deformations. In particular, the addition of topological changes to our hybrid mesh deformation algorithm (Kim et al., Computer and Mathematics with Applications, Submitted, November 2014) proved to be an extremely efficient way of improving the mesh quality.

J. Kim
Department of Computer Science and Engineering, Incheon National University, Incheon, South Korea
e-mail: jibumkim@incheon.ac.kr

D. McLaurin
CD-adapco, Austin, TX 78750, USA
e-mail: david.mclaurin@cd-adapco.com

S.M. Shontz (✉)
University of Kansas, Lawrence, KS 66045, USA
e-mail: shontz@ku.edu

© Springer International Publishing Switzerland 2015
S. Perotto, L. Formaggia (eds.), *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, SEMA SIMAI Springer Series 5,
DOI 10.1007/978-3-319-06053-8_13

## 1 Introduction

There are numerous scientific applications for which the geometric domains deform as a function of time. Such applications include simulations of Arbitrary-Lagrangian-Eulerian (ALE) fluid flow [1, 2], fluid-structure interaction [3, 4], ALE plasticity [5], crack propagation [6], biomedical applications [7, 8], patient and medical devices [9–11], and computer graphics [12]. Whenever such deformations occur, the meshes must be updated with respect to time in order to remain valid approximations of the geometry. There are two main types of strategies for updating the mesh in response to a deforming geometric domain. The first strategy is to remesh the domain whenever necessary in response to the deforming domain. This typically creates a different mesh, and the associated numerical partial differential equation (PDE) solution must be interpolated from the initial mesh to the new mesh since their topology is different. Another issue with this approach is that frequent remeshing can lead to loss of data resolution and accumulation of round-off errors leading to inaccuracies [13]. Alternatively, a mesh deformation (i.e., mesh warping or mesh morphing) strategy can be used to move the mesh from the source domain onto the target domain; such techniques recompute interior mesh vertex positions after the boundary has been deformed. Mesh warping is preferred over remeshing based on accuracy (as described above) as well as efficiency.

There are numerous mesh warping techniques in the literature. We give a summary of relevant mesh warping techniques here; however, this list is not comprehensive. Mesh warping techniques are typically based on the solution of a PDE which describes the motion of the interior mesh vertices or on the solution of an optimization problem which guides the mesh motion based on desired properties. Researchers have proposed various mesh warping techniques [14] based on Laplace's equation, e.g., finite element-based mesh warping (FEMWARP) [15, 16], weighted Laplacian smoothing [17], biharmonic PDEs [1], elasticity [18–20], and an inverse distance function [21]. Researchers have also proposed techniques which combine the solution of PDEs with techniques for altering the mesh topology [22–25] in order to yield high quality deformed meshes in simulations with large deformations.

Researchers have also proposed several optimization-based mesh deformation techniques. For example, techniques have been developed based on a log-barrier techniques [17] and the target matrix paradigm [26]. Many of the mesh deformation techniques which are guided by optimization also involve PDE solutions. For example, the nonlinear elasticity-based Untangling before Newton (UBN) method [20] mentioned above can also be thought of as an optimization-based mesh warping technique in that it solves a variational problem (which is equivalent to a minimization problem) to achieve static equilibrium. An adjoint-based optimization procedure for mesh warping was developed in [27] in order to improve the robustness and extend the range of linear elasticity-based mesh warping techniques. FEMWARP has also been combined with PDE-based level set techniques [9, 10], the purpose of which is to first predict the mesh deformation using an evolving level set and then FEMWARP to deform the mesh to the location computed by the level set method. Another example of a hybrid mesh deformation algorithm which employs both optimization and PDEs is found in [28]. This particular hybrid algorithm employs an anisotropic version of FEMWARP for the mesh deformation followed by multiobjective mesh optimization [29] for smoothing and untangling of the deformed mesh.

All of the methods described thus far, with the exception of the hybrid mesh deformation algorithm in [28], suffer from one or more of the following problems: a tendency to produce inverted elements for large boundary deformations, or an inability to preserve features of the initial mesh in the deformed mesh.

In the current work, we consider larger boundary deformations than those in [28]. In particular, the hybrid mesh deformation method in [28] is unable to produce high quality deformed meshes for these test cases.

In this paper, we propose a topology-adaptive mesh deformation framework for mesh warping. In particular, we combine the hybrid mesh deformation algorithm of Kim, Miller, and Shontz in [28] with topological changes in order to generate high quality meshes for extremely large boundary deformations. We describe our topology-adaptive mesh deformation framework in Sect. 2. In Sect. 3, we describe several numerical experiments in which we test the ability of our algorithm to generate high quality deformed meshes on several extremely large boundary deformations. The numerical results from the first two experiments can be compared against those from the same experiments in [28] in order to obtain a comparison against the method by Kim, Miller, and Shontz. In Sect. 4, we summarize our work and describe several possibilities for future work.

## 2 Algorithmic Framework

In this section, we describe our topology-adaptive mesh deformation framework for use in mesh warping applications. Our mesh deformation framework is composed of four steps: (1) anisotropic FEMWARP, (2) multiobjective mesh optimization, (3) topological changes, and (4) mesh smoothing. Our algorithmic framework builds upon the hybrid mesh deformation algorithm [28] of Kim, Miller, and Shontz, which performs only step 1 (anisotropic FEMWARP) and step 2 (multiobjective mesh optimization), and the multiobjective mesh optimization framework [29] of Kim, Panitanarak, and Shontz.

Initially, the deformation that the user provides is applied to the boundary vertices. This prescribes the final positions of the boundary vertices in the deformed mesh. We then perform the four steps in our topology-adaptive mesh deformation framework in order to compute the positions of the interior vertices.

The first step in our framework is to compute initial, approximate locations for the interior vertices in the deformed mesh by performing one step of anisotropic finite element-based mesh warping (i.e., anisotropic FEMWARP) with an appropriate choice of PDE coefficients as proposed in [28]. In particular, anisotropic

FEMWARP solves the following anisotropic version of Laplace's equation

$$-\alpha \frac{\partial^2 u}{\partial x^2} - \beta \frac{\partial^2 u}{\partial y^2} = 0 \text{ on } \Omega\,, \tag{1}$$

where $u = u_0$ on $\partial\Omega$. We assume that $\alpha > 0$ and $\beta > 0$ and solve the PDE using an elliptic PDE solver. The mesh topology is held fixed during anisotropic FEMWARP.

We adaptively choose $\alpha$ and $\beta$ with respect to the direction of the boundary deformation. In [28], we demonstrated that $\alpha$ and $\beta$ control the strength of the x- and y-axis couplings between adjacent vertices, respectively. Therefore, $\alpha$ should be larger than $\beta$ if more deformation occurs along the x-axis. Similarly, $\beta$ should be larger than $\alpha$ if more deformation occurs along the y-axis. We compute the cumulative boundary vertex displacements in the x and y directions to decide in which direction more deformation occurs. Let $x_k$ and $y_k$ be the $k$th vertex coordinates on the initial mesh and $\hat{x_k}$ and $\hat{y_k}$ be the $k$th vertex coordinates on the deformed mesh. A particular choice of the anisotropic PDE coefficients that worked well for our experiments in [28] was

$$\begin{cases} \alpha = \sum_{k=1}^{N_B} \left(\hat{x_k} - x_k\right),\ k \in B \\ \beta = \sum_{k=1}^{N_B} \left(\hat{y_k} - y_k\right),\ k \in B, \end{cases} \tag{2}$$

where $B$ is the set of boundary vertices and $N_B$ is the number of boundary vertices. Here, the relative ratio between $\alpha$ and $\beta$ can be understood as the angle of the direction of the deformation [28].

Our second step is the multiobjective mesh optimization. The deformed mesh using anisotropic FEMWARP could include inverted elements with poor element qualities on the deformed domain when a huge deformation occurs. Similar to the hybrid deformation algorithm [28], we employ the target matrix paradigm (TMP) shape metric to improve the element quality on the deformed domain. The TMP shape metric is useful when our goal is to preserve good element qualities (shapes) on the deformed domain. Let $(A_{\text{def}})_i$ and $(A_{\text{init}})_i$ be the Jacobians of the mappings from the reference element to the actual elements in the deformed and initial domains, respectively. The TMP shape metric in 2D is defined as

$$q_i = \left|T_i - (\text{adj}(T_i)^T)\right|_F^2,$$

where $T_i = (A_{\text{def}})_i (A_{\text{init}})_i^{-1}$. The TMP shape metric ($q_i$) is zero when the quality of the element on the deformed mesh is same as the one on the initial mesh. In order to eliminate inverted elements on the deformed domain, we employ the untangling beta quality metric [29]. The untangling beta quality metric is defined as

$$q_j = \left|V_j - \beta\right| - (V_j - \beta),$$

where $V_j$ is the signed area of the $j$th element and $\beta$ is a user defined small constant value. The untangling beta quality metric is zero when the deformed mesh does not have any inverted elements.

Let the overall mesh quality computed by the TMP shape metric and the overall mesh quality computed by the untangling beta quality metric be $F_1$ and $F_2$, respectively. Then, $F_1 = \sum_{i=1}^{|E|} q_i^2$ and $F_2 = \sum_{j=1}^{|E|} q_j^2$, where $|E|$ is the number of elements on the mesh.

In order to simultaneously untangle inverted elements and smooth the deformed mesh, we employ the exponential sum multiobjective mesh optimization method in [29]. The exponential sum multiobjective mesh optimization method utilizes the min-max property, which minimizes the worst (maximum) objective function using the exponential penalty function. The exponential sum multiobjective mesh optimization function is denoted as

$$F = c\ ln\left[e^{F_1/c} + e^{F_2/c}\right].$$

Our goal is to minimize $F$ to find the optimal vertex positions on the deformed domain. Similar to [29], we employ the Fletcher-Reeves nonlinear conjugate gradient method to find a locally optimal point. The combination of the first two steps described above is the hybrid mesh deformation algorithm in [28]. Similar to other mesh untangling algorithms [26, 29], there is no guarantee that our hybrid mesh deformation algorithm in [28] is able to always untangle the deformed mesh.

Once the mesh is untangled, we perform edge swaps (step 3) as indicated in [30] in order to further improve the quality of the mesh. Even if we eliminate all inverted elements after step 2, the output after step 2 still suffer from poor element qualities when a huge deformation occurs. From [30],

> For an edge $uv$ with opposite vertices $p$ and $q$, we flip $uv$ if the Delaunay flipping criterion (i.e., $\angle upv + \angle uqv > \pi$) is satisfied... We used a greedy strategy to flip edges in decreasing order of maximum opposite angle....

This work differs from [30] in the manner in which infinite loops cause by repeated edge flipping are avoided. There, a constraint was added to the algorithm to only allow a given edge to be flipped once. Here, a suitably small tolerance, $tol$ (a scalar multiple of the run-time-calculated round-off error), is used as a "buffer" for angle comparison. This changes the edge-flip criteria to: $(\angle upv + \angle uqv > (\pi + tol))$ This buffer lessens the likelihood of numerical errors causing infinite loops. Since an infinite is loop is still possible (but improbable), a limit of 100 flips per edge was enforced—but never encountered in practice. This developed strategy of limiting edge-flips produced results that were more favorable for this application than those in [30]. Additionally, any edge flips that would create invalid topology (cannot flip boundary edges) or geometry (inverted/tangled triangles) were not allowed. GRX5 [31], a topology repair and feature removal toolkit and library, was used to make the topological changes. The data structures and algorithms are optimized for performing topological operations on triangular surface grids. Currently it is available as a stand-alone tool and is also incorporated into SolidMesh [32].

The final step in our mesh deformation framework is to perform a final pass of mesh smoothing according to the inverse mean ratio (IMR) metric in order to obtain further improvement in the mesh quality. Let the coordinates of the three vertices of a triangle denoted by $a$, $b$ and $c$. Then, an incidence matrix, $C$ is given by $[b-a, c-a]$. For an equilateral triangle, the incidence matrix for an ideal element, $W$, is denoted by

$$W = \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} \end{bmatrix},$$

The IMR quality metric measures how similar the current element to the ideal element (equilateral element). The IMR quality metric is defined as

$$q_{IMR} = \frac{\left\| CW^{-1} \right\|_F^2}{2\left|\det(CW^{-1})\right|}.$$

The IMR quality metric has the value of 1 for the ideal element (equilateral triangle) and a smaller value indicates a better element quality. Similar to the TMP shape metric, we compute the overall mesh quality by computing, $F_{IMR} = \sum_{i=1}^{|E|} q_{IMR}^2$. We minimize $F_{IMR}$ using the the Fletcher-Reeves nonlinear conjugate gradient method to find a locally optimal point. For our test meshes, roughly 10–30 nonlinear conjugate gradient iterations are needed to reach the locally optimal points. Numerical results show that our final mesh smoothing step is able to improve the average and worst mesh quality up to 52.2 % and 96.2 %, respectively for the test meshes.

## 3 Numerical Experiments

Table 1 summarizes a description of each step and language/software we used to perform each step. We use Mesquite [33] to perform steps 2 and 4. We stop to perform step 2 and move to step 3 when we eliminates all inverted elements on the deformed mesh. We use GRX5 [31] to perform step 3. For step 3, GRX5 was used

**Table 1** Description of each step

| | Description of each step | Language/software |
|---|---|---|
| Step 1 | Perform anisotropic FEMWARP | C/C++ |
| Step 2 | Perform multiobjective mesh optimization | Mesquite [33] |
| Step 3 | Perform topological changes | C/C++ |
| Step 4 | Perform mesh smoothing | Mesquite [33] (C++) |

to swap interior edges that did not meet Delaunay criteria. The resultant mesh is a constrained Delaunay triangulation.

In our framework, we first perform multiobjective mesh optimization (step 2) and perform topological changes (step 3) as a next step. This is because preliminary numerical experiments show that the reverse order (first perform step 3 and second perform step 2) results in much slower multiobjective mesh optimization time compared with the proposed order. When we perform multiobjective mesh optimization, we use the TMP shape metric such that the initial element and the deformed element have the similar element shape. However, we noticed that initially performing edge swaps often result in output meshes which are less similar to the initial mesh and therefore time to perform multiobjective mesh optimization could be very slow.

We compare our mesh deformation framework with Knupp's mesh deformation algorithm [26]. Knupp's mesh deformation algorithm sets the initial mesh to be a reference mesh and deforms the given mesh to be similar to the reference mesh based on a target matrix paradigm framework. We use Mesquite [33] to implement Knupp's mesh deformation algorithm.

The machine employed for this study is equipped with an AMD Opteron processor 6174 (2.2 GHz) and 6.5 GB of RAM.

## *3.1 Moving Bar Domain for Anisotropic Boundary Deformation Along the Y-axis*

We consider a moving bar domain where the deformation occurs along the y-axis. Since the deformation only occurs along the y-axis, we chose the PDE coefficients, $\alpha$=0 and $\beta$=1. Figures 1, 2, 3, 4, and 5 show the initial mesh and output meshes after each step. The initial mesh (Fig. 1) has no inverted elements and the output mesh after performing anisotropic FEMWARP (Fig. 2) has 84 inverted elements. Most of

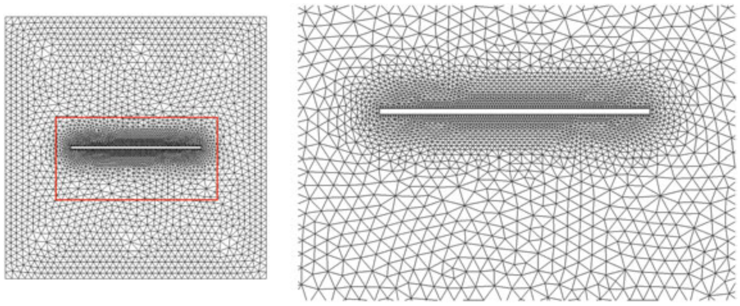


**Fig. 1** Moving bar domain for anisotropic boundary deformation: Initial mesh (*left*) and zoomed-in mesh (*right*) on the bar domain

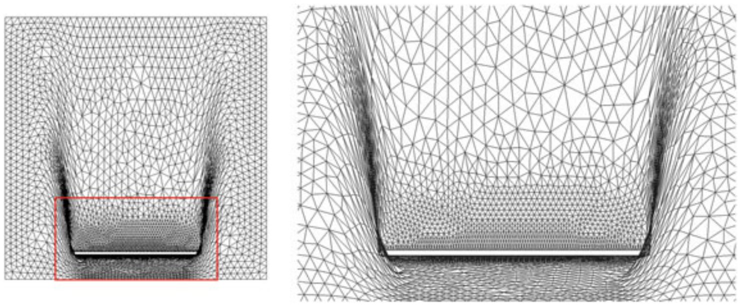


**Fig. 2** Moving bar domain for anisotropic boundary deformation: Output mesh after step 1 (*left*) and zoomed-in output mesh after step 1 (*right*)

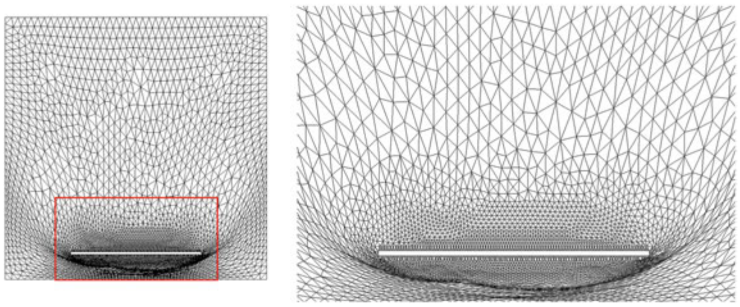


**Fig. 3** Moving bar domain for anisotropic boundary deformation: Output mesh after step 2 (*left*) and zoomed-in output mesh after step 2 (*right*)

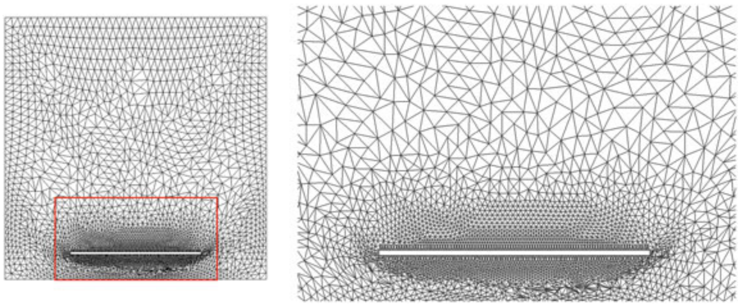


**Fig. 4** Moving bar domain for anisotropic boundary deformation: Output mesh after step 3 (*left*) and zoomed-in output mesh after step 3 (*right*)

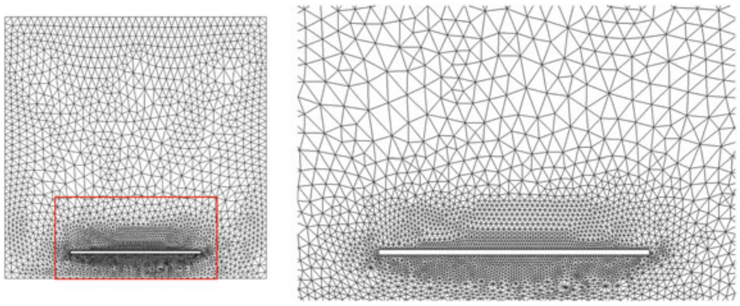


**Fig. 5** Moving bar domain for anisotropic boundary deformation: Output mesh after step 4 (*left*) and zoomed-in output mesh after step 4 (*right*)

**Table 2** Mesh quality statistics of moving bar domain measured by the inverse mean ratio quality metric and the number of inverted elements after each step

| Mesh quality | Minimum | Avg | rms | Maximum | Std.dev. | # of inverted elements |
|---|---|---|---|---|---|---|
| Initial | 1 | 1.076 | 1.081 | 1.649 | 0.110 | 0 |
| Step 1 | 1 | 11,628.3 | 107,803 | 1e+06 | 107,174 | 84 |
| Step 2 | 1 | 2.135 | 3.669 | 71.127 | 2.970 | 0 |
| Step 3 | 1 | 1.268 | 1.348 | 10.682 | 0.456 | 0 |
| Step 4 | 1 | 1.081 | 1.086 | 2.019 | 0.0969 | 0 |
| Knupp [26] | −0.000471 | 2.478 | 33.826 | 2,822.830 | 33.736 | 27 |

Here, "each step" indicates the output mesh after each step

the inverted elements occur around the bar due to the large deformation. The output mesh after performing multiobjective mesh optimization is shown in Fig. 3. After this step, all inverted elements are eliminated but many elements are still distorted due to the large deformation. The output mesh after performing topological changes is shown in Fig. 4. Here we observe that mesh quality is improved—especially the poorest quality elements. The output mesh after performing final mesh smoothing is shown in Fig. 5. Here we observe that the final output mesh has a good element quality and similar isotropy relative to the initial mesh.

Table 2 shows mesh quality statistics and the number of inverted elements of the initial and output meshes after each step. The IMR quality metric was used to measure the element quality. Here, a smaller value indicates a better element quality. The final output mesh is exhibits similar element quality to the initial mesh (near isotropy). Knupp's mesh deformation algorithm [26] results in an output mesh with 27 inverted elements and poor element qualities. Knupp's mesh deformation algorithm does not include the mesh untangling step and only tries to keep similar element qualities. Therefore, we observe that Knupp's mesh deformation algorithm fails to produce an output mesh with no inverted elements and good element qualities.

## 3.2 Moving Cylinder Domain for Anisotropic Boundary Deformation Along the X-axis

We consider a moving cylinder domain where the deformation occurs along the x-axis. We choose the PDE coefficients $\alpha$=1 and $\beta$=0, since the deformation only occurs along the x-axis. Figures 6, 7, 8, 9, and 10 show the initial mesh (Fig. 6) and output meshes after performing each step. Overall results are similar to the previous moving bar example. Many poor quality and inverted elements (19 inverted elements) occur after the large deformation (Fig. 7). After performing multiobjective mesh optimization, we are able to eliminate inverted elements; but poor quality elements persist around the cylinder (Fig. 8). We observe that topological changes are effective in that they adjust the edges so that an increased amount of vertex movement can be performed as compared to before. For this example, topological changes reconnected the edges in the crowded area around the cylinder (Fig. 9). The final output mesh has a good element quality after performing mesh smoothing (Fig. 10).

Table 3 shows mesh quality statistics and the number of inverted elements of the initial mesh and output meshes after performing each step. The output mesh, after performing multiobjective mesh optimization, still suffers from poor element quality. However, performing topological changes significantly improves the element quality. The worst element quality improves approximately 99 % after performing topological changes. Note that the step 3 is significantly faster than the step 2 as we will discuss later. Similar to the previous moving bar example, the final output mesh has a similar element quality to the initial mesh. Similar to the previous example, Knupp's mesh deformation algorithm [26] fails to eliminate inverted elements and results in an output mesh with poor element qualities.

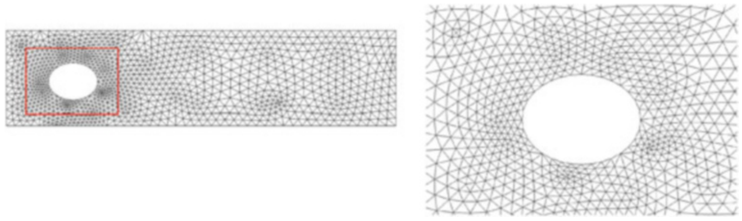


**Fig. 6** Moving cylinder domain for anisotropic boundary deformation: Initial mesh (*left*) and zoomed-in mesh (*right*) on the cylinder domain

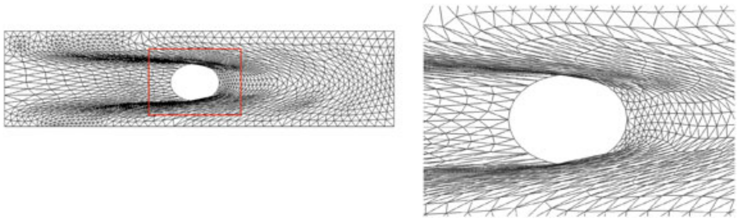


**Fig. 7** Moving cylinder domain for anisotropic boundary deformation: Output mesh after step 1 (*left*) and zoomed-in output mesh after step 1 (*right*)

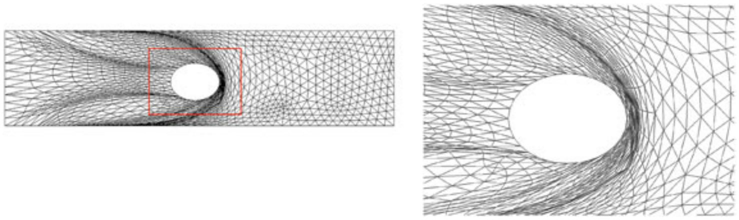


**Fig. 8** Moving cylinder domain for anisotropic boundary deformation: Output mesh after step 2 (*left*) and zoomed-in output mesh after step 2 (*right*)

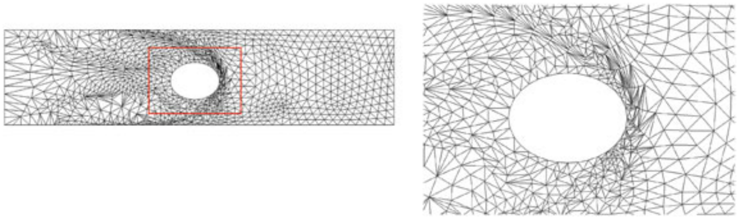


**Fig. 9** Moving cylinder domain for anisotropic boundary deformation: Output mesh after step 3 (*left*) and zoomed-in output mesh after step 3 (*right*)

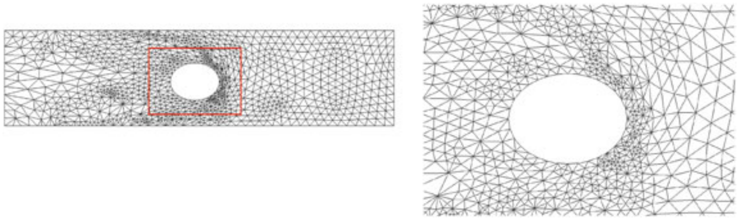


**Fig. 10** Moving cylinder domain for anisotropic boundary deformation: Output mesh after step 4 (*left*) and zoomed-in output mesh after step 4 (*right*)

**Table 3** Mesh quality statistics of moving cylinder domain measured by the inverse mean ratio quality metric and the number of inverted elements after each step

| Mesh quality | Minimum | Avg | rms | Maximum | Std.dev. | # of inverted elements |
|---|---|---|---|---|---|---|
| Initial | 1 | 1.040 | 1.040 | 1.940 | 0.0594 | 0 |
| Step 1 | 1 | 8,490.440 | 92,098.6 | 1e+06 | 91,706.4 | 19 |
| Step 2 | 1 | 9.437 | 148.19 | 6,979.570 | 147.889 | 0 |
| Step 3 | 1 | 1.761 | 2.358 | 17.131 | 1.568 | 0 |
| Step 4 | 1 | 1.159 | 1.171 | 1.948 | 0.170 | 0 |
| Knupp [26] | −0.000441 | 15.843 | 7,296.871 | 345,349 | 7,295.152 | 248 |

Here, "each step" indicates the output mesh after each step

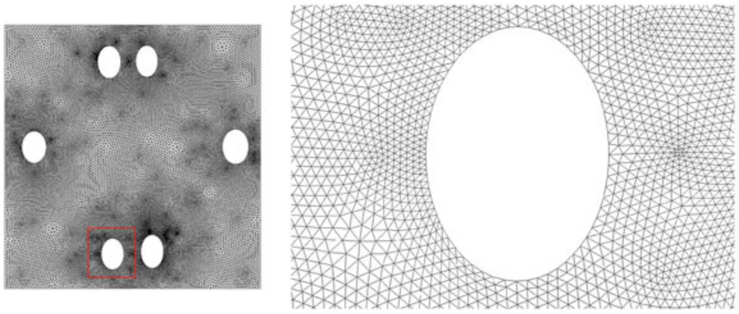


**Fig. 11** Moving particles domain for anisotropic boundary deformation: Initial mesh (*left*) and zoomed-in mesh on the particles domain (*right*)

## 3.3 *Moving Particles Domain for Anisotropic Boundary Deformation Along the X-axis*

We consider a geometry that simulates moving particles—where several particles (cylinders) are moving in anisotropic ways. This example is more challenging than the previous moving cylinder example, since particles with different size are moving in different directions along the x-axis. The PDE coefficients are chosen as $\alpha$=1 and $\beta$=0 because the deformation occurs along the x-axis. Figure 11 shows the initial mesh and the output meshes (Figs. 12, 13, 14, and 15) after performing each step. The bottom two particles are initially close each other and move toward the boundary. Therefore, movement of the vertices in elements around each particle is highly constrained after the deformation occurs (Fig. 12). After deformation, 218 inverted and (or) anisotropic elements are generated—mostly around each particle. After performing multiobjective mesh optimization, all inverted elements are removed, but many of the elements are still anisotropic, and the movement of the corresponding vertices is highly constrained (Fig. 13). However, we are able to more uniformly distribute the positions of these elements after performing

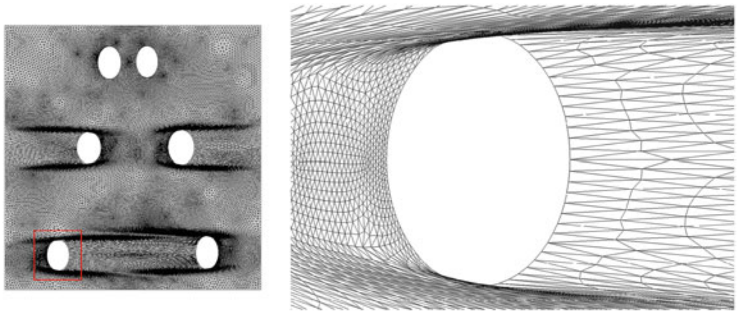


**Fig. 12** Moving particles domain for anisotropic boundary deformation: Output mesh after step 1 (*left*) and zoomed-in output mesh after step 1 (*right*)

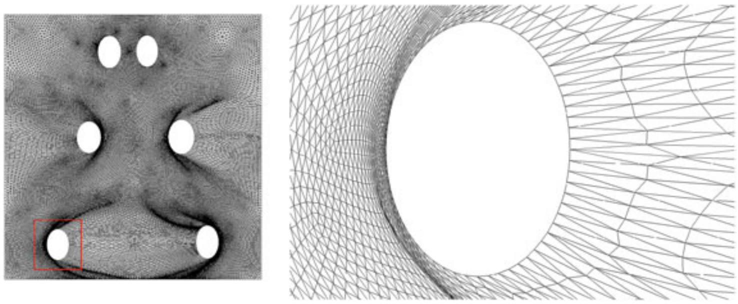


**Fig. 13** Moving particles domain for anisotropic boundary deformation: Output mesh after step 2 (*left*) and zoomed-in output mesh after step 2 (*right*)

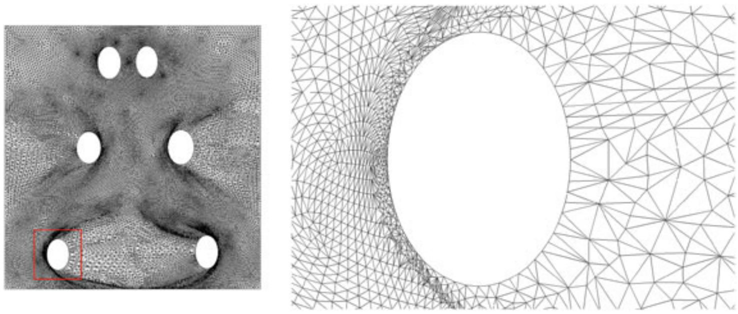


**Fig. 14** Moving particles domain for anisotropic boundary deformation: Output mesh after step 3 (*left*) and zoomed-in output mesh after step 3 (*right*)

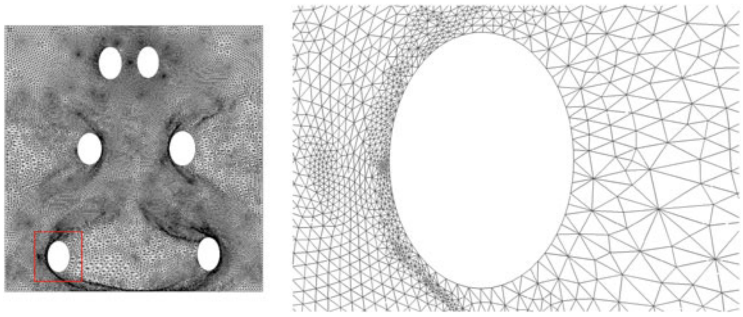


**Fig. 15** Moving particles domain for anisotropic boundary deformation: Output mesh after step 4 (*left*) and zoomed-in output mesh after step 4 (*right*)

**Table 4** Mesh quality statistics of moving particles domain measured by the inverse mean ratio quality metric and the number of inverted elements after each step

| Mesh quality | Minimum | Avg | rms | Maximum | Std.dev. | # of inverted elements |
|---|---|---|---|---|---|---|
| Initial | 1 | 1.017 | 1.018 | 2.548 | 0.0430 | 0 |
| Step 1 | 1 | 4,893.13 | 69,907.3 | 1e+06 | 69,735.9 | 218 |
| Step 2 | 1 | 2.889 | 17.421 | 1,346.45 | 17.180 | 0 |
| Step 3 | 1 | 1.348 | 1.774 | 93.742 | 1.154 | 0 |
| Step 4 | 1 | 1.087 | 1.095 | 3.609 | 0.132 | 0 |
| Knupp [26] | −0.00520 | 2.850 | 28.434 | 4,631.670 | 28.290 | 647 |

Here, "each step" indicates the output mesh after each step

topological changes. After performing topological changes, the worst element quality is improved approximately 93 % (Fig. 14). After the final mesh smoothing pass, we are able to restore element quality similar to the input mesh (Fig. 15).

Table 4 shows mesh quality statistics and the number of inverted elements for the initial mesh and the output meshes after each step. We observe that the overall mesh quality of the final output mesh is similar to the initial mesh. Similar to previous examples, Knupp's mesh deformation algorithm is not able to remove inverted elements in the deformed mesh and instead increases the number of inverted elements.

## 3.4 Summary of Numerical Results

**Choice of PDE Coefficients** We compare FEMWARP [15, 16] with anisotropic FEMWARP in terms of both the number of inverted elements and efficiency. FEMWARP always fixes the PDE coefficient as $\alpha$=1 and $\beta$=1; however, anisotropic FEMWARP adaptively chooses the appropriate PDE coefficients with respect to the direction of deformation. Figure 16 shows the number of inverted elements

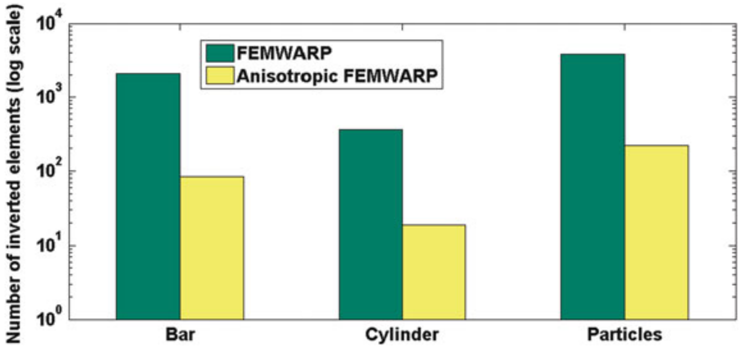


**Fig. 16** Number of inverted elements after performing FEMWARP and anisotropic FEMWARP

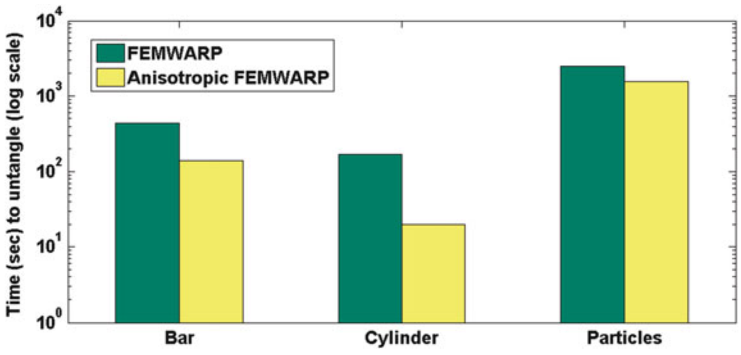


**Fig. 17** Time (s) to untangle inverted elements using multiobjective mesh optimization

after performing FEMWARP and anisotropic FEMWARP. The output mesh with anisotropic FEMWARP has up to 95.9 % fewer number of inverted elements than the output mesh with isotropic FEMWARP. Since mesh untangling step is a relatively time-consuming step, these results indicate the importance of choosing appropriate PDE coefficients. Figure 17 shows the timing results for using FEMWARP and anisotropic FEMWARP to eliminate inverted elements using multiobjective mesh optimization (step 2). The FEMWARP algorithm by itself does not include the mesh untangling step. Therefore, for the purposes of comparison, we apply our step 2 to untangle the output meshes after applying FEMWARP. We observe that the use of anisotropic FEMWARP results in a decrease in the untangling time of up to 67.3 % compared with using FEMWARP. Note that the multiobjective optimization step, which performs untangling, is the slowest individual step of the developed algorithm. Therefore, it is desirable to have as fewer inverted elements as possible by choosing appropriate PDE coefficients.

**Mesh Quality** Figures 18 and 19 shows the average and the worst element quality of the initial mesh and the final output mesh (after step 4), respectively, for each

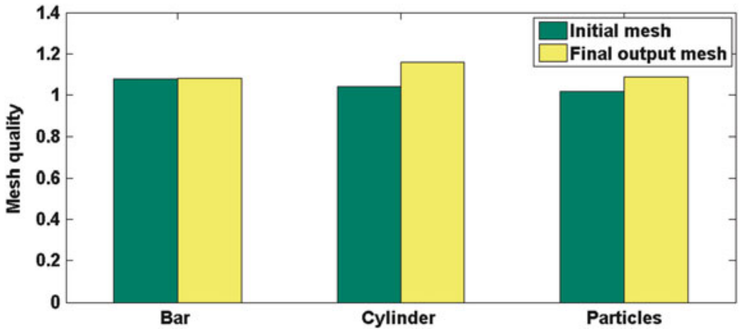


**Fig. 18** Average element quality of the initial mesh and the final output mesh for each geometric domain measured by the IMR quality metric

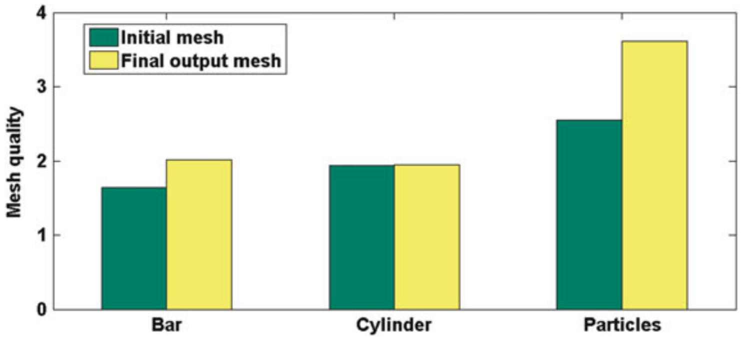


**Fig. 19** Worst element quality of the initial mesh and the final output mesh for each geometric domain measured by the IMR quality metric

geometric domain. We observe that our framework is able to maintain good element quality even when large deformations were performed. For a moving bar and cylinder domain, the initial and the final output meshes exhibit nearly identical mesh quality, which is highly desirable for PDE-based applications. For a challenging moving particles domain, our algorithm is able to maintain similar average mesh quality; although a slight increase in the worst element quality was noticed.

**Timing Results** Table 5 shows timing results of each step for three geometric domains. We compare our timing results with Knupp's deformation algorithm [26]. We observe that step 2 and step 3 are the slowest and fastest steps of the entire procedure, respectively. Step 3 significantly improves the overall element quality and distributes the elements whose vertex movement is highly constrained, and it takes less than 1 s to perform. This timing result justifies the motivation of using step 3 to rather than keep performing step 2 to improve the element quality. Note that our algorithm performs topological changes (step 3) right after all inverted elements are eliminated. Knupp's mesh deformation algorithm is slightly faster to converge than

**Table 5** Timing results (s) of each geometric domain

| Timing (s) | Step 1 | Step 2 | Step3 | Step 4 | Knupp [26] |
|---|---|---|---|---|---|
| Bar (7K elements) | 1.512 | 141.714 | 0.003 | 16.854 | 131.902 |
| Cylinder (2K elements) | 0.625 | 20.586 | 0.003 | 6.151 | 20.443 |
| Particles (44K elements) | 4.183 | 540.674 | 0.062 | 53.742 | 534.124 |

our framework, but it results in an output mesh with inverted elements and poor element qualities.

## 4 Conclusions

We propose a mesh deformation framework for anisotropic mesh deformations. Our framework is composed of four steps: (1) anisotropic FEMWARP, (2) multiobjective mesh optimization, (3) topological changes, (4) mesh smoothing. Numerical results show that our framework successfully eliminates inverted elements and keeps good element qualities on the deformed domain—even when applied to a large deformation. By choosing appropriate PDE coefficients, anisotropic FEMWARP is able to decrease the number of inverted elements up to 95.9 %. Our second step, multiobjective mesh optimization successfully eliminates inverted elements while keeping good element quality. We observe that performing topological changes is an extremely efficient and effective way of increasing the amount of vertex movement possible on the final smoothing pass and hence improving the element quality on the deformed domain. Our final mesh smoothing step is able to further improve the element quality. Numerical results show that our mesh deformation framework significantly outperforms Knupp's mesh deformation algorithm based on a target matrix paradigm framework [26].

Our topology-adaptive mesh deformation framework expands on our earlier hybrid mesh deformation framework by incorporating topological changes for additional mesh quality improvement as is required by large, anisotropic deformations. Because edge swaps yield a large improvement in the mesh quality very efficiently, we developed our initial framework in two dimensions. However, the hybrid mesh deformation algorithm upon which our work builds also works for three-dimensional mesh warping. Thus, our framework can be extended to three dimensions by the consideration of edge swaps and/or face swaps in three dimensions.

It is likely the case that the addition of other operations to alter the topology (e.g., edge splits, edge contractions, and multi-face removal) will lead to even further improvements in the mesh quality. In fact, such a strategy was used to evolve surface meshes in [34]. We plan to investigate the addition of such topological operations for our future research. Discrete optimization algorithms [35] can then be developed which improve the quality of the mesh by altering the mesh topology.

We also plan to compare the performance of our topology-adaptive mesh deformation framework with meshless techniques for mesh deformation (e.g., [36]). Meshless techniques do not use the mesh topology and hence are much faster than either PDE or optimization-based techniques for mesh deformation. However, more insight is needed as to how to appropriately choose a kernel function for use with meshless techniques for various types of mesh deformation.

**Acknowledgements** The work of the third author is supported in part by NSF grant CAREER Award ACI-1330054 (formerly OCI-1054459). The authors wish to thank the anonymous referee for his/her careful reading of the paper and for the helpful suggestions which strengthened it.

## References

1. Helenbrook, B.T.: Mesh deformation using the biharmonic operator. Int. J. Numer. Methods Eng. **56**, 1007–1021 (2003)
2. Villone, M.M., Hulsen, M.A., Anderson, P.D., Maffettone, P.L.: Simulations of deformable systems in fluids under shear flow using an arbitrary Lagrangian Eulerian technique. Comput. Fluids **90**, 88–100 (2014)
3. Pan, F., Kubby, J., Chen, J.: Numerical simulation of fluid-structure interaction in a MEMS diaphragm drop ejector. J. Micromech. Microeng. **12**, 70–76 (2002)
4. Crosetto, P., Reymond, P., Deparis, S., Kontaxakis, D., Stergiopulos, N., Quateroni, A.: Fluid-structure interaction simulation of aortic blood flow. Comput. Fluids **43**(1), 46–57 (2011)
5. Armero, F., Love, E.: An arbitrary Lagrangian-Eulerian finite element method for finite strain plasticity. Int. J. Numer. Meth. Eng. **57**, 471–508 (2003)
6. Kaczmarczyk, L., Nezhad, M.M., Pearce, C.: Three-dimensional brittle fracture: configurational force-driven crack propagation. Int. J. Numer. Methods Eng. **97**, 531–550 (2013)
7. Bah, M.T., Nair, P.B., Browne, M.: Mesh morphing for finite element analysis of implant positioning in cementless total hip replacements. Med. Eng. Phys. **31**, 1235–1243 (2009)
8. Baldwin, M.A., Langenderfer, J.E., Rullkoetter, P.J., Laz, P.J.: Development of subject-specific and statistical shape models of the knee using an efficient segmentation and mesh-morphing approach. Comput. Methods Programs Biomed. **97**, 232–240 (2010)
9. Park, J., Shontz, S.M., Drapaca, C.S.: A combined level set/mesh warping algorithm for tracking brain and cerebrospinal fluid evolution in hydrocephalic patients. In: Image-Based Geometric Modeling and Mesh Generation. Lecture Notes in Computational Vision and Biomechanics, vol. 3, pp. 107–141. Springer, Amsterdam (2013)
10. Park, J., Shontz, S.M., Drapaca, C.S.: Automatic boundary evolution tracking via a combined level set method and mesh warping technique: Application to hydrocephalus. In: Proc. of the MICCAI Workshop on Mesh Processing in Medical Image Analysis, pp. 122–133 (2012)
11. Sastry, S.P., Kim, J., Shontz, S.M., Craven, B.A., Lynch, F.C., Manning, K.B., Panitanarak, T.: Patient-specific model generation and simulation for pre-operative surgical guidance for pulmonary embolism treatment. In: Image-Based Geometric Modeling and Mesh Generation. Lecture Notes in Computational Vision and Biomechanics, vol. 3, pp. 223–301. Springer, Amsterdam (2013)
12. Lee, A.W.F., Dobkin, D., Sweldens, W., Schroder, P.: Multiresolution mesh morphing. In: Proc. of the 26th SIGGRAPH Conference, pp. 343–350 (1999)
13. Klingner, B.: Tetrahedral Mesh Improvement. Ph.D. Thesis, University of California at Berkeley (2009)

14. Staten, M.L., Owen, S.J., Shontz, S.M., Salinger, A.G., Coffey, T.S.: A comparison of mesh morphing techniques for 3D shape optimization. In: Proc. of the 2011 International Meshing Roundtable, pp. 293–312 (2011)
15. Baker, T.J.: Mesh movement and metamorphosis. In: Proc. of the 10th International Meshing Roundtable, pp. 387–396 (2001)
16. Shontz, S.M., Vavasis, S.A.: Analysis of and workarounds for element reversal for a finite element-based algorithm for warping triangular and tetrahedral meshes. BIT Numer. Math. **50**, 863–884 (2010)
17. Shontz, S.M., Vavasis, S.A.: A mesh warping algorithm based on weighted Laplacian smoothing. In: Proc. of the 12th International Meshing Roundtable, pp. 147–158 (2003)
18. Stein, K., Tezduyar, T., Benney, R.: Mesh moving techniques for fluid-structure interactions with large displacements. Trans. ASME **70**, 58–63 (2003)
19. Stein, K., Tezduyar, T., Benney, R.: Automatic mesh update with the solid-extension mesh moving technique. Comput. Methods Appl. Mech. Eng. **193**, 2019–2031 (2004)
20. Shontz, S.M., Vavasis, S.A.: A robust solution procedure for hyperelastic solids with large boundary deformation. Eng. Comput. **28**, 135–147 (2012)
21. Luke, E., Collins, E., Blades, E.: A fast mesh deformation method using explicit interpolation. J. Comput. Phys. **231**, 586–601 (2012)
22. Antaki, J., Blelloch, G., Ghattas, O., Malcevic, I., Miller, G., Walkington, N.: A parallel dynamic mesh Lagrangian method for simulation of flows with dynamic interfaces. In: Proc. of the 2000 Supercomputing Conference, p. 26 (2000)
23. Cardoze, D., Cunha, A., Miller, G., Phillips, T., Walkington, N.: A Bézier-based approach to unstructured moving meshes. In: Proc. of the 20th ACM Symposium on Computational Geometry (2004)
24. Cardoze, D., Miller, G., Olah, M., Phillips, T.: A Bézier-based moving mesh framework for simulation with elastic membranes. In: Proc. of the 13th International Meshing Roundtable, pp. 71–80. Sandia National Laboratories, Williamsburg (2004)
25. Alauzet, F., Marcum, D.: A closed advancing-layer method with changing topology mesh movement for viscous mesh generation. In: Proc. of the 22nd International Meshing Roundtable, pp. 241–262 (2013)
26. Knupp, P.: Updating meshes on deforming domains: an application of the target-matrix paradigm. Commun. Numer. Methods Eng. **24**, 467–476 (2007)
27. Yang, Z., Mavripilis, D.J.: Mesh deformation strategy optimized by the adjoint method on unstructured meshes. AIAA J. **45**(12), 2885–2896 (2007)
28. Kim, J., Miller, B.J., Shontz, S.M.: A hybrid mesh deformation algorithm using anisotropy and multiobjective mesh optimization. Computer and Mathematics with Applications, Submitted (November 2014)
29. Kim, J., Panitanarak, T., Shontz, S.M.: A multiobjective framework for mesh optimization. Int. J. Numer. Methods Eng. **94**(1), 20–42 (2013)
30. Jiao, X., Colombi, A., Ni, X., Hart, J.C.: Anisotropic mesh adaptation for evolving triangulated surfaces. In: Proc. of the $15^{th}$ International Meshing Roundtable, pp. 173–190 (2006)
31. McLaurin, D., Marcum, D., Remotigue, M., Blades, E.: Algorithms and Methods for Discrete Surface Repair. Ph.D. Thesis, Mississippi State University (2010)
32. McLaurin, D.: Discrete Mesh Intersection Tutorial. http://www.simcenter.msstate.edu/docs/solidmesh/discretegridintersection.html (2011)
33. Brewer, M., Freitag Diachin, L., Knupp, P., Leurent, T., Melander, D.: The mesquite mesh quality improvement toolkit. In: Proc. of the $12^{th}$ International Meshing Roundtable, Sandia National Laboratories, pp. 239–250 (2003)
34. Zaharescu, A., Boyer, E., Horaud, R.: Topology-adaptive mesh deformation for surface evolution, morphing, and multiview reconstruction. IEEE Trans. Pattern Anal. Mach. Intell. **33**(4), 823–837 (2011)
35. Shewchuk, J.R.: Two discrete optimization algorithms for the topological improvement of tetrahedral meshes, Unpublished (2002)
36. Sieger, D., Menzel, S., Botsch, M.: High quality mesh morphing using triharmonic radial basis functions. In: Proc. of the $21^{st}$ International Meshing Roundtable, pp. 1–15 (2013)

# On Shape Deformation Techniques for Simulation-Based Design Optimization

**Daniel Sieger, Stefan Menzel, and Mario Botsch**

**Abstract** We present an in-depth analysis and benchmark of shape deformation techniques for their use in simulation-based design optimization scenarios. We first introduce classical free-form deformation, its direct manipulation variant, as well as deformations based on radial basis functions. We compare the techniques in a series of representative synthetic benchmarks, including computational performance, numerical robustness, quality of the deformation, adaptive refinement, as well as precision of constraint satisfaction. As an application-oriented benchmark we investigate the ability to adapt an existing volumetric simulation mesh according to an updated surface geometry, including unstructured tetrahedral, structured hexahedral, and arbitrary polyhedral example meshes. Finally, we provide a detailed assessment of the methods and give concrete advice on choosing a suitable technique for a given optimization scenario.

## 1 Introduction

Simulation-based design optimization is a key aspect of the product development process of automotive industry, aircraft construction, and naval architecture. The overall goal is to explore alternative and novel designs with improved physical or aesthetic properties. The development process typically starts with the creation of an initial design prototype using a computer aided design (CAD) tool. Subsequent steps create a polygon surface mesh from the CAD model as well as a volumetric simulation mesh for physical performance evaluation, e.g., using computational fluid dynamics (CFD) simulations for aerodynamic performance calculation, or finite element methods (FEM) for structural mechanics simulations. Design variations are

D. Sieger (✉) • M. Botsch
Bielefeld University, Postfach 100 131, D-33501 Bielefeld, Germany
e-mail: dsieger@techfak.uni-bielefeld.de; botsch@techfak.uni-bielefeld.de

S. Menzel
Honda Research Institute Europe GmbH, Carl-Legien-Str. 30, D-63073 Offenbach/Main, Germany
e-mail: stefan.menzel@honda-ri.de

S. Perotto, L. Formaggia (eds.), *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, SEMA SIMAI Springer Series 5,
DOI 10.1007/978-3-319-06053-8_14

then created based on physical performance during simulation. In this paper, we are concerned with efficient means to create such alternate designs.

The obvious approach of changing the CAD model directly is prohibitive in many cases, since both the surface and volume meshing steps would have to be repeated. For complex geometries and precise physical simulations the meshing process might even require manual interaction by an expert. An alternative is to use *shape deformation techniques* to adapt both the surface and the volume mesh of the initial design prototype directly. This way, the design optimization can be performed in a fully automatic and parallel manner, which is of particular importance when using stochastic optimization techniques—such as evolutionary algorithms—which typically require the creation and evaluation of a large number of design variations in order to find a feasible solution.

This paper is organized as follows: We begin with an investigation of the fundamental requirements a deformation method should satisfy in order to be suitable for common design optimization scenarios (Sect. 2). Based on these requirements we introduce state-of-the-art shape deformation methods including classical free-form deformation (FFD), direct manipulation FFD (DM-FFD), and deformations based on radial basis functions (RBFs). We compare the different methods in a series of synthetic and application oriented benchmarks in Sect. 3. Finally, we perform a detailed assessment of the methods in Sect. 4 and give concrete guidance for choosing a suitable technique for a particular design optimization scenario.

## 2 Shape Deformation Methods

In this section we introduce state-of-the-art shape deformation methods for their use in simulation-based design optimization. Before describing the individual techniques in detail, we briefly review related work, motivate our selection of methods, and introduce the concept of a space deformation. Shape deformation methods have been an area of continuous and extensive research within the fields of computer graphics and geometric modeling. Consequently, a wide variety of techniques has been proposed during recent years. Since providing an overview of the complete field is beyond the scope of this work we refer to existing introductions and surveys. Detailed references for the individual methods covered in this work are provided in the corresponding sections. A general introduction to shape deformation techniques is provided by [5]. Surveys on space deformation techniques have been presented in [1, 8]. While the former concentrates on building a mathematical formalism for the different methods, the latter is focused on the interactive manipulation of a model by a designer. In contrast, a survey of shape parametrization techniques in the context of design optimization is given in [27]. Staten and coworkers recently proposed and evaluated several mesh morphing techniques, which they compared with respect to computational performance and element quality on different tetrahedral and hexahedral meshes [38]. This evaluation was later extended by Sieger and colleagues [35, 36]. Linear variational surface deformation methods have been investigated in detail in [4].

The selection of deformation methods considered in our comparison is highly driven by our application domain—design optimization. In this context, one may formulate several requirements a deformation technique should satisfy. A fundamental one is the ability to transparently deal with different object representations such as triangular or quadrilateral surface meshes, volumetric meshes, polygon soups, as well as point-based representations. On the one hand, this requirement stems from the desire to be able to optimize a wide variety of designs. On the other hand, it is particularly important when the evaluation of the objective function involves computationally expensive CFD or FEM simulations. As already outlined in Sect. 1, the volumetric simulation meshes typically are very time-consuming to generate. Therefore, in order to avoid the costly mesh generation process for each design variation created during optimization, one typically aims at adapting an initial simulation mesh alongside with the surface. A second requirement is the ability to robustly deal with defects in the input geometry. Especially when the surface mesh is the result of an automatic conversion process from the CAD model the resulting mesh might contain degeneracies such as badly shaped triangles, non-manifold configurations, or disconnected components.

A type of deformation methods that naturally fulfills the above requirements are so-called *space deformations*. The fundamental idea behind these methods is to deform the embedding space around an object, thereby deforming the object implicitly. From a mathematical point of view a space deformation is a function $\mathbf{d}\colon \mathbb{R}^3 \to \mathbb{R}^3$ that maps each point in space to a certain displacement. Given a deformation function, a model $\mathcal{M}$ can be transformed to a deformed model $\mathcal{M}'$ by computing updated point locations $\mathbf{x}' = \mathbf{x} + \mathbf{d}(\mathbf{x})$ for each original point $\mathbf{x} \in \mathcal{M}$. The basic procedure is illustrated in Fig. 1, where a space deformation for the DrivAer body [12] is shown. Naturally, space deformation techniques differ in how the function $\mathbf{d}$ is constructed. Typically, a control structure such as a volumetric lattice or a set of points is blended with some form of basis functions, as we will describe for the individual methods in the sections that follow. Since the deformations applied during optimization are typically relatively small, we focus on linear space deformation methods.

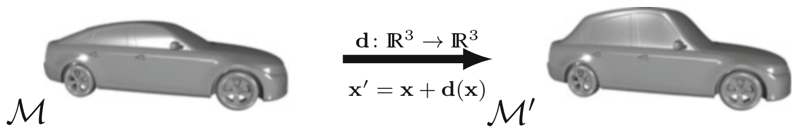

$\mathcal{M}$
$\mathbf{d}\colon \mathbb{R}^3 \to \mathbb{R}^3$
$\mathbf{x}' = \mathbf{x} + \mathbf{d}(\mathbf{x})$
$\mathcal{M}'$

**Fig. 1** Deformation of the DrivAer model. The model $\mathcal{M}$ is warped by the space deformation function $\mathbf{d}$. Each point $\mathbf{x} \in \mathcal{M}$ is transformed to updated locations $\mathbf{x}' = \mathbf{x} + \mathbf{d}(\mathbf{x}) \in \mathcal{M}'$

### 2.1 Free-Form Deformation

Free-form deformation (FFD) is a well-established deformation technique that has been widely used in both academia and industry. Since it also has been employed within shape optimization [18, 28] and simulation-based design optimization [19–21, 34] it forms the basis for our comparison. Before describing the method in detail we first review important variants of the technique. Free-form deformations using Bézier basis functions have been originally introduced in [31]. Local deformations using B-spline basis functions have been introduced in [11]. An extension to more flexible control lattices, in particular cylindrical ones, has been proposed in [6]. This approach was later extended to control lattices of arbitrary topology [17]. Free-form deformations using non-uniform rational B-splines are described in [16]. A highly flexible but computationally involved variant of FFD based on a 3D-Delaunay triangulation, its Voronoi dual, and Sibson coordinates [33] has been presented in [23]. A variant of FFD using T-splines [32] as basis functions—thereby allowing for local refinement of the control lattice—has been presented in [37].

The basic idea of FFD is based on embedding the object to be deformed in a parallelepiped lattice and deforming it using a trivariate tensor-product Bézier or B-spline function. The deformation procedure to perform free-form deformation of an object can be divided into several steps. First, a control lattice has to be generated and adapted to the deformation scenario at hand. Then the local coordinates with respect to the control lattice have to be computed for each point to be deformed. After this embedding each object point $\mathbf{x} \in \mathcal{M}$ can be expressed as a linear combination of lattice control points $\mathbf{c}_{ijk}$ and basis functions $N_i$:

$$\mathbf{x} = \sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n} \mathbf{c}_{ijk} N_i(u_1) N_j(u_2) N_k(u_3), \qquad (1)$$

where $(u_1, u_2, u_3)$ are the local coordinates of $\mathbf{x}$ with respect to the control lattice, and $l, m, n$ are the numbers of control points in each direction. For the sake of simplicity we define

$$\mathbf{u}(\mathbf{x}) := (u_1, u_2, u_3), \quad N_p(\mathbf{u}(\mathbf{x})) := N_i(u_1) N_j(u_2) N_k(u_3),$$

as well as

$$\boldsymbol{\delta}\mathbf{c}_p := \boldsymbol{\delta}\mathbf{c}_{ijk} = \mathbf{c}'_{ijk} - \mathbf{c}_{ijk},$$

where $\mathbf{c}'_{ijk}$ denotes an updated control point location. We can then define the FFD space deformation function as

$$\mathbf{d}_{\text{ffd}}(\mathbf{x}) = \sum_p \boldsymbol{\delta}\mathbf{c}_p N_p(\mathbf{u}(\mathbf{x})). \qquad (2)$$

Finally, the deformation is performed by moving the control points and computing the updated object point locations. An example deformation using FFD is illustrated in Fig. 2.

In our implementation of FFD we use cubic B-splines with a uniform knot vector. While this type of basis functions requires an iterative root-finding technique such as a Newton method [26] for computing the local coordinates, the important advantage is the capability to perform deformations with local support.

## *2.2 Direct Manipulation FFD*

In an interactive modeling system the manipulation of control points to perform a deformation becomes a tedious task—especially when using a complex control lattice with a large number of control points. A more flexible and intuitive interface for controlling a deformation is offered by *direct manipulation* approaches, which have been introduced for FFD in [13], referred to as DM-FFD throughout this paper. Instead of moving control points, the user *directly* moves the object points. The modeling system then computes control point displacements so that the new object point positions are matched as precise as possible. An example deformation of the DrivAer model using direct manipulation is shown in Fig. 3.

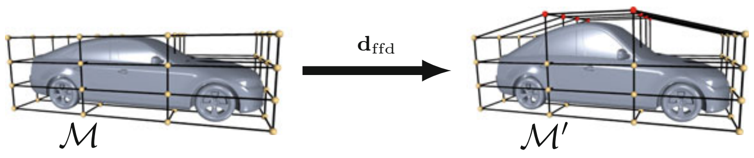


**Fig. 2** Free-form deformation applied to the DrivAer model. The original model $\mathcal{M}$ is embedded in a regular lattice of $4 \times 4 \times 4$ control points (*golden*). After moving the selected control points (*red*) the updated object point locations $\mathbf{x}'$ are computed by evaluating the FFD space deformation function $\mathbf{d}_{\mathrm{ffd}}$ for the local coordinates $\mathbf{u}$ of the point $\mathbf{x}$

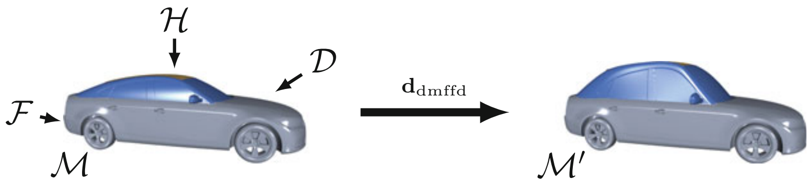


**Fig. 3** Direct manipulation FFD on the DrivAer model using a handle-based direct manipulation interface. The vertices of the model $\mathcal{M}$ are classified into three distinct sets: Handle vertices ($\mathcal{H}$, *golden*) can be directly displaced, fixed vertices ($\mathcal{F}$, *gray*) are kept in place, and deformable vertices ($\mathcal{D}$, *blue*) are updated according to the deformation method

Direct manipulation interfaces are not only beneficial within an interactive modeling scenario, they can also be used effectively within simulation-based design optimization, as has been shown for direct manipulation FFD in [21]. Due to the more direct influence of the parameters determined during optimization on the design, using such an interface can result in a drastically faster convergence of the optimization. In contrast to classical FFD, the ability to choose an arbitrary object point for optimization offers a higher degree of flexibility. Furthermore—due to the automatic computation of control point displacements—this approach also reduces the need to pre-deform the control lattice to a certain degree.

Within a direct manipulation interface the user—be it an engineer or an optimization algorithm—prescribes a set of $m$ displacement constraints at so-called *handle points* $\{\mathbf{h}_1, \ldots, \mathbf{h}_m\}$, where the deformation function has to attain certain displacement values $\mathbf{d}(\mathbf{h}_i) = \mathbf{h}'_i - \mathbf{h}_i = \boldsymbol{\delta}\mathbf{h}_i$. The displacements $\{\boldsymbol{\delta}\mathbf{c}_1, \ldots, \boldsymbol{\delta}\mathbf{c}_n\}$ of the $n$ control points satisfying the prescribed displacements can be computed by solving the linear system

$$\underbrace{\begin{bmatrix} N_1(\mathbf{u}(\mathbf{h}_1)) & \ldots & N_n(\mathbf{u}(\mathbf{h}_1)) \\ \vdots & \ddots & \vdots \\ N_1(\mathbf{u}(\mathbf{h}_m)) & \ldots & N_n(\mathbf{u}(\mathbf{h}_m)) \end{bmatrix}}_{\boldsymbol{\Phi}} \underbrace{\begin{pmatrix} \boldsymbol{\delta}\mathbf{c}_1^T \\ \vdots \\ \boldsymbol{\delta}\mathbf{c}_n^T \end{pmatrix}}_{\mathbf{C}} = \underbrace{\begin{pmatrix} \boldsymbol{\delta}\mathbf{h}_1^T \\ \vdots \\ \boldsymbol{\delta}\mathbf{h}_m^T \end{pmatrix}}_{\mathbf{H}}. \qquad (3)$$

Since the linear system (3) can be over-determined as well as under-determined, it is typically solved by computing the *pseudo-inverse* $\boldsymbol{\Phi}^+$ of the basis function matrix $\boldsymbol{\Phi}$. This is typically done by performing a singular value decomposition (SVD) [10, 13] so that $\boldsymbol{\Phi} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$, where $\mathbf{U}$ is a $m \times m$ orthogonal matrix, $\boldsymbol{\Sigma}$ is a $m \times n$ diagonal matrix containing the singular values of $\boldsymbol{\Phi}$, and $\mathbf{V}^T$ is a $n \times n$ orthogonal matrix. The pseudo-inverse of $\boldsymbol{\Phi}$ then is $\boldsymbol{\Phi}^+ = \mathbf{V}\boldsymbol{\Sigma}^+\mathbf{U}^T$, where the pseudo-inverse of the diagonal matrix $\boldsymbol{\Sigma}$ can be computed as

$$\boldsymbol{\Sigma}_{ij}^+ = \begin{cases} \frac{1}{\sigma_i}, & \text{if } i = j \wedge \sigma_i \neq 0, \\ 0, & \text{otherwise}, \end{cases} \qquad (4)$$

where $\sigma_i$ is the $i$-th singular value of $\boldsymbol{\Phi}$. We note that for values close to zero $\sigma_i$ has to be clamped in order to prevent numerical instabilities. Once $\boldsymbol{\Phi}^+$ has been computed the control point displacements can be computed by

$$\mathbf{C} = \boldsymbol{\Phi}^+\mathbf{H}, \qquad (5)$$

where $\mathbf{C}$ is the matrix of control point displacements and $\mathbf{H}$ is the matrix of constraint displacements. However, solving for $\mathbf{C}$ using the pseudo-inverse has its drawbacks. If the system is under-determined, a *least-norm* solution is found, i.e., the amount of movement of the control points $\|\boldsymbol{\delta}\mathbf{c}\|$ is minimized. If the system is overdetermined, a *least-squares* solution is found, i.e., the error in satisfying the specified constraints is minimized. This means that depending on the resolution of the control lattice the system might not be able to satisfy the constraints specified by the user in an exact manner. In both cases, however, the solution does not necessarily result in a physically plausible deformation.

## 2.3 RBF Deformation

Mesh deformation using radial basis functions (RBFs) has been proposed by several authors [2, 3, 15, 22]. This method improves upon FFD and DM-FFD in two significant aspects: First, due to its point-based or kernel-based nature, introducing additional degrees of freedom in regions of interest is highly flexible, without the need to maintain a complicated control structure. Second, the deformation function can be constructed in such a way that it directly minimizes a physically inspired energy—resulting in a smooth and physically plausible deformation. An example deformation based on radial basis functions using the same handle-based interface as described in Fig. 3 for DM-FFD is given in Fig. 4.

On a more abstract level, we can treat mesh deformation as a scattered data interpolation problem: We search for a function $\mathbf{d}\colon \mathbb{R}^3 \to \mathbb{R}^3$ that (1) exactly interpolates the prescribed displacements $\mathbf{d}(\mathbf{h}_i) = \boldsymbol{\delta}\mathbf{h}_i$ and (2) smoothly interpolates these displacements through space. Radial basis functions are well known to be suitable for solving this type of problem [39]. Using RBFs we define the deformation function as a linear combination of radially symmetric kernel functions $\varphi_j(\mathbf{x}) = \varphi(\|\mathbf{x} - \mathbf{c}_j\|)$, located at centers $\mathbf{c}_j \in \mathbb{R}^3$ and weighted by $\mathbf{w}_j \in \mathbb{R}^3$, plus a linear polynomial to guarantee linear precision:

$$\mathbf{d}(\mathbf{x}) = \sum_{j=1}^{m} \mathbf{w}_j \varphi_j(\mathbf{x}) + \sum_{k=1}^{4} \mathbf{q}_k \pi_k(\mathbf{x}), \tag{6}$$

where $\{\pi_1, \pi_2, \pi_3, \pi_4\} = \{x, y, z, 1\}$ is a basis of the space of linear trivariate polynomials, weighted by coefficients $\mathbf{q}_k \in \mathbb{R}^3$. Note that the polynomial term is important, since it guarantees to find the optimal affine motion (translation, rotation, scaling) contained in the prescribed displacements $\boldsymbol{\delta}\mathbf{h}_i$.

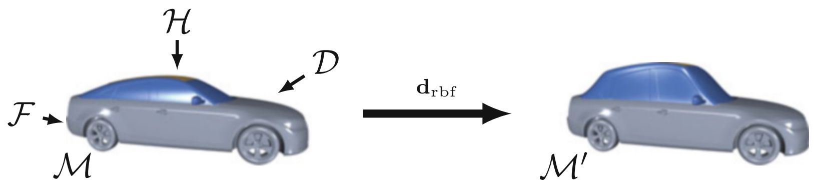


**Fig. 4** Deformation of the DrivAer model using a handle-based direct manipulation interface for RBFs

**Table 1** Commonly used radial basis functions

| | |
|---|---|
| Gaussian | $\varphi(r) = \mathrm{e}^{-(\epsilon r)^2}$ |
| Multiquadric | $\varphi(r) = \sqrt{1 + (\epsilon r)^2}$ |
| Inverse multiquadric | $\varphi(r) = 1/\sqrt{1 + (\epsilon r)^2}$ |
| Polyharmonic spline in $\mathbb{R}^d$ | $\varphi_k(r) = \begin{cases} r^{2k-d}, & d \text{ odd}, \\ r^{2k-d}\log(r), & d \text{ even}. \end{cases}$ |

For Gaussians and (inverse) multiquadrics $\epsilon$ denotes the shape parameter. For polyharmonic splines $k$ denotes the order of smoothness

The choice of the kernel function $\varphi\colon \mathbb{R} \to \mathbb{R}$ basically determines the shape of the interpolant. Commonly used kernels include Gaussians, (inverse) multiquadrics, and polyharmonic splines (see Table 1 for an overview). In our application we aim for high quality deformations minimizing the distortion of mesh elements. To meet this goal, we have to use a sufficiently smooth kernel function. While Gaussian and multiquadric basis functions provide infinite smoothness, i.e., they are $C^\infty$, they require the choice of an additional shape parameter (the $\epsilon$ in Table 1). Small values of $\epsilon$ increase approximation accuracy, but lead to numerically instabilities, and *vice versa*. Therefore, finding the optimal shape parameter for a given radial basis function and the particular application is a non-trivial task on its own (see [7] for an overview of different strategies).

In contrast, polyharmonic splines are free of shape parameters, but only of finite smoothness. Depending on the application scenario, we have to choose a sufficiently high degree of smoothness. In $\mathbb{R}^3$ the polyharmonic spline $\varphi_k(r) = r^{2k-3}$ is a fundamental solution of the $k$-th order Laplacian $\Delta^k$, such that also the RBF deformation (6) is $k$-harmonic, i.e., $\Delta^k \mathbf{d} = 0$. Being the strong form of a variational energy minimization, this is equivalent [39] to $\mathbf{d}$ minimizing the weak form

$$\iiint_{\mathbb{R}^3} \left\| \frac{\partial^k \mathbf{d}}{\partial x^k} \right\|^2 + \left\| \frac{\partial^k \mathbf{d}}{\partial x^{k-1} \partial y} \right\|^2 + \ldots + \left\| \frac{\partial^k \mathbf{d}}{\partial z^k} \right\|^2 \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}z. \tag{7}$$

In order to preserve mesh quality during deformation, we should construct a deformation function that at least minimizes the change of first-order derivatives of the mesh elements [38], and therefore the first-order derivatives of the deformation function. With $k = 1$ in (7), this is achieved by the harmonic RBF $\varphi(r) = 1/r$, but those basis functions are singular at their centers. The biharmonic spline $\varphi(r) = r$ is well defined, but not differentiable at the center and therefore not smooth enough for our application (see Fig. 5). By choosing $\varphi(r) = r^3$, we obtain a deformation function that is triharmonic, therefore penalizes third-order derivatives in (7), and is globally $C^2$ smooth. With these properties, it is the lowest-order polyharmonic RBF suitable for our application. Since for numerical robustness a low order is preferable, we eventually chose triharmonic RBFs for our deformation method.

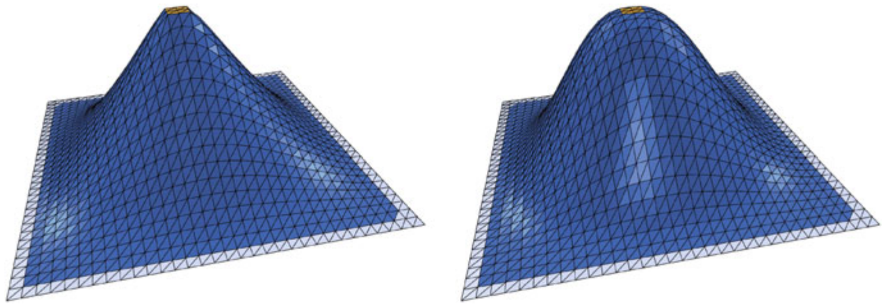


**Fig. 5** Comparison between a biharmonic (*left*) and a triharmonic (*right*) deformation of a plane. We displace the *golden* region, keep the *gray* region fixed, and deform the *blue* region. We place RBF kernels on all vertices in the *golden* and *gray* regions

We can *exactly* satisfy the interpolation constraints $\mathbf{d}(\mathbf{h}_i) = \boldsymbol{\delta}\mathbf{h}_i$ by placing RBF kernels at the constraint positions (i.e., $\mathbf{c}_j = \mathbf{h}_j$) and finding the coefficients $\mathbf{w}_j$ and $\mathbf{q}_k$ by solving the $(m+4) \times (m+4)$ linear system

$$\underbrace{\begin{pmatrix} \varphi_1(\mathbf{h}_1) & \cdots & \varphi_m(\mathbf{h}_1) & \pi_1(\mathbf{h}_1) & \cdots & \pi_4(\mathbf{h}_1) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \varphi_1(\mathbf{h}_m) & \cdots & \varphi_m(\mathbf{h}_m) & \pi_1(\mathbf{h}_m) & \cdots & \pi_4(\mathbf{h}_m) \\ \pi_1(\mathbf{h}_1) & \cdots & \pi_1(\mathbf{h}_m) & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \pi_4(\mathbf{h}_1) & \cdots & \pi_4(\mathbf{h}_m) & 0 & \cdots & 0 \end{pmatrix}}_{\boldsymbol{\Phi}} \cdot \underbrace{\begin{pmatrix} \mathbf{w}_1^T \\ \vdots \\ \mathbf{w}_m^T \\ \mathbf{q}_1^T \\ \vdots \\ \mathbf{q}_4^T \end{pmatrix}}_{\mathbf{W}} = \underbrace{\begin{pmatrix} \boldsymbol{\delta}\mathbf{h}_1^T \\ \vdots \\ \boldsymbol{\delta}\mathbf{h}_m^T \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix}}_{\mathbf{H}} . \tag{8}$$

After solving (8) we can compute the morphed mesh $\mathcal{M}'$ by simply evaluating the RBF deformation at each mesh vertex: $\mathbf{x}'_i = \mathbf{x}_i + \mathbf{d}(\mathbf{x}_i)$. The computationally most expensive part is the solution of the linear system (8), which is dense due to the global support of $\varphi(r)$. We discuss the performance and the scalability of our method in Sect. 3.1.

## 3 Benchmarks

In this section we evaluate the different deformation methods based on a set of synthetic benchmarks. The goal of these benchmarks is to capture basic properties of the different deformation methods which are relevant for the use in design optimization scenarios. We perform our evaluation based on the following criteria: computational performance, numerical robustness, adaptivity and precision, as well as quality of the deformation. For each criterion we first describe our tests and methodology, and then present the results for the individual deformation methods.

We performed all tests on a Dell T7500 workstation with an Intel Xeon E5645 2.4 GHz CPU and 18GB RAM running Ubuntu Linux 12.04 x86_64. We compiled all code with `gcc` 4.6.3, optimization turned on (using `-O3`) and debugging checks disabled (`-DNDEBUG`). In order to rule out caching and power saving issues, we averaged the timings over five morphing steps.

In many of the benchmarks a direct comparison with FFD based on control point manipulation is not really feasible, i.e., it is not possible to compare the methods on a solid and representative basis. In these cases, we only compare DM-FFD and RBFs.

## *3.1 Performance*

While the impact of the performance of a deformation method is often negligible when used within an design optimization loop, it is still an important and fundamental characteristic. Furthermore, it is crucial for usage in an interactive modeling application. Within control point-based FFD, the only performance-critical component is the computation of the local coordinates of each object point with respect to the control lattice. When using B-spline basis functions, this computation requires the use of a numerical technique such as a golden section search or a Newton method [26]. However, since the local coordinate computation is independent for each object point, this part is trivial to parallelize.

Naturally, direct manipulation FFD also requires the local coordinate computation discussed above. In addition, however, the linear system (3) has to be solved. The standard approach for this is based on computing the pseudo-inverse using singular value decomposition, which has a computational cost of $4m^2n + 22n^3$ floating point operations [10]. Additional computational costs come from the matrix multiplications required to actually compute the pseudo-inverse $\mathbf{\Phi}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^T$ from the SVD.

Within the RBF deformation technique the most expensive part is the solution of the linear system (8), which is dense due to the global support of the chosen radial basis functions $\varphi(r) = r^3$. The resulting asymptotic complexity is of $O(m^3)$ when using standard solvers for dense linear systems. Since the linear system (8) is symmetric but not positive definite, efficient Cholesky-type solvers are not applicable. However, the system can still be solved efficiently by using a $\mathrm{LDL}^\mathrm{T}$ factorization, which has computational costs of $\frac{1}{3}m^3$ floating point operations. For a more comprehensive investigation of different solvers for RBF deformation we refer to [36].

However, benchmarking the performance by simply measuring the time it takes to deform a given mesh is not really meaningful since the methods pre-compute different amounts of information. Comparing the performance of control point-based FFD to directly manipulated DM-FFD or RBFs is not feasible, since there is no way to perform the same deformation with all three methods. In order to facilitate a representative and objective comparison between DM-FFD and RBFs, we present an alternative formulation of both deformation methods which allows us to fully pre-compute the deformation. The deformation methods we investigated are linear, i.e., they require solving a linear problem in one form or another. Therefore, the deformations can be pre-computed by solving a sequence of $m$ linear system (see, e.g., [3]). Even more, the methods can be handled in a uniform manner by expressing the deformation in terms of handle basis functions.

The $m$ displacement constraints $\boldsymbol{\delta}\mathbf{h}_i$ are given as prescribed values of the deformation function $\mathbf{d}(\mathbf{h}_i) = \boldsymbol{\delta}\mathbf{h}_i$. In case of DM-FFD, the control point displacements $\boldsymbol{\delta}\mathbf{c}_j$ satisfying these constraints are found by solving the linear system of (3). In case of RBFs, we find the weights $\mathbf{w}_j$ for the deformation function (6) by solving (8). What we are searching for are the displacements $\mathbf{x}_i' - \mathbf{x}_i = \boldsymbol{\delta}\mathbf{x}_i$ for each deformable vertex $\mathbf{x}_i$. Written in matrix form this becomes $\mathbf{X} = (\boldsymbol{\delta}\mathbf{x}_1, \ldots, \boldsymbol{\delta}\mathbf{x}_k)^T$, where $k$ is the number of deformable vertices. In case of DM-FFD, $\mathbf{X}$ can be computed using

$$\mathbf{X} \;=\; \mathbf{M} \cdot \mathbf{C}, \quad \mathbf{M}_{ij} = N_j(\mathbf{u}(\mathbf{x}_i)), \tag{9}$$

where $N_j(\mathbf{u}(\mathbf{x}_i))$ is the trivariate tensor-product B-spline basis function of control point $\mathbf{c}_j$ evaluated at point $\mathbf{x}_i$, and $\mathbf{C}$ is the matrix of control point displacements $\boldsymbol{\delta}\mathbf{c}_j$. By substituting $\mathbf{C}$ using (5) we can rewrite (9) as

$$\mathbf{X} \;=\; \underbrace{\mathbf{M} \cdot \boldsymbol{\Phi}^{+}}_{\mathbf{B}} \cdot \mathbf{H},$$

where $\mathbf{H}$ is the $m \times 3$ matrix of prescribed handle displacements. Using the $k \times m$ matrix $\mathbf{B}$ we can then directly evaluate the vertex displacements in terms of handle displacements.

The corresponding formulation for RBFs is similar: $\mathbf{X}$ can be computed by

$$\mathbf{X} \;=\; \mathbf{M} \cdot \mathbf{W}, \quad \mathbf{M}_{ij} = \varphi_j(\mathbf{x}_i), \tag{10}$$

where $\mathbf{W}$ is the matrix of radial basis function weights. Based on (8) the weight matrix $\mathbf{W}$ can be computed by inverting $\boldsymbol{\Phi}$, i.e., as $\mathbf{W} = \boldsymbol{\Phi}^{-1}\mathbf{H}$. This yields

$$\mathbf{X} \;=\; \underbrace{\mathbf{M} \cdot \boldsymbol{\Phi}^{-1}}_{\mathbf{B}} \cdot \mathbf{H}.$$

Then $\mathbf{B}$ is the desired $k \times m$ basis function matrix that can be used to compute the vertex displacements from the given handle displacements.

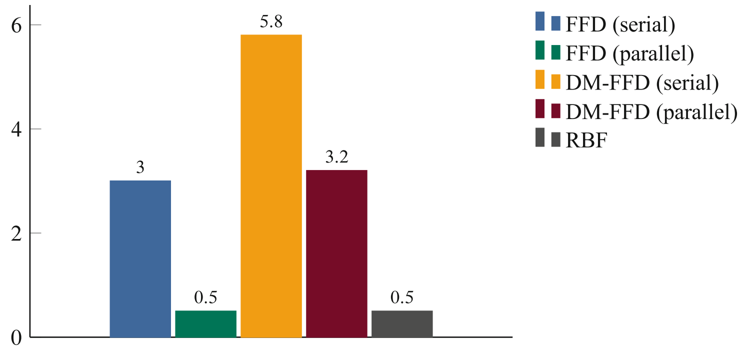


**Fig. 6** Performance comparison of the deformation methods. Times in seconds. For FFD methods a control grid of resolution $8^3$ was used, which results in a comparable number of DoFs as the RBF setup

Based on this formulation, we compare the performance of the methods by pre-computing a deformation with 427 constraints. In Fig. 6 we present the results comparing both FFD variants in their serial and parallel (using OpenMP [25]) versions as well as RBFs. As to be expected from theory, DM-FFD offers the worst performance. While parallel local coordinate computation clearly improves (DM-)FFD performance, RBFs require the same amount of time to solve the full problem.

## 3.2 Robustness

The robustness of a deformation method describes its robustness towards defects in the input data. Such defects can include low-quality triangles with very large or very small angles, such as caps or needle elements, non-manifold configurations, or self-intersections in the input mesh (see Fig. 7 some for common examples). Due to their space-based nature, FFD and RBFs are highly robust with respect to defects in the input data. However, in the direct manipulation variant of FFD the singular value decomposition used to compute the pseudo-inverse might also be a source for numerical instabilities. In order to prevent division by zero, artifacts in the deformation, as well as extreme distortions of the control lattice, one has to clamp the singular values $\sigma_i$ in (4). Figure 8 presents an example of the unwanted artifacts in the deformation depending on different clamping values. Since a suitable clamping value for a given deformation setup is not known a-priori, it has to be determined heuristically by the user—thereby constituting a source of increased effort and potential failure.

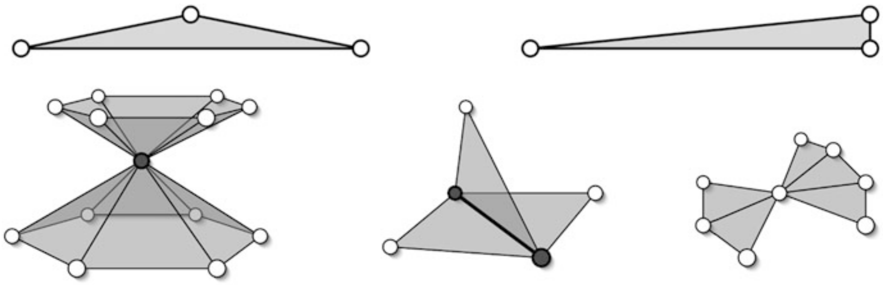


**Fig. 7** Examples of common surface mesh degeneracies. *Top row*: low quality triangles. *Bottom row*: non-manifold connectivity

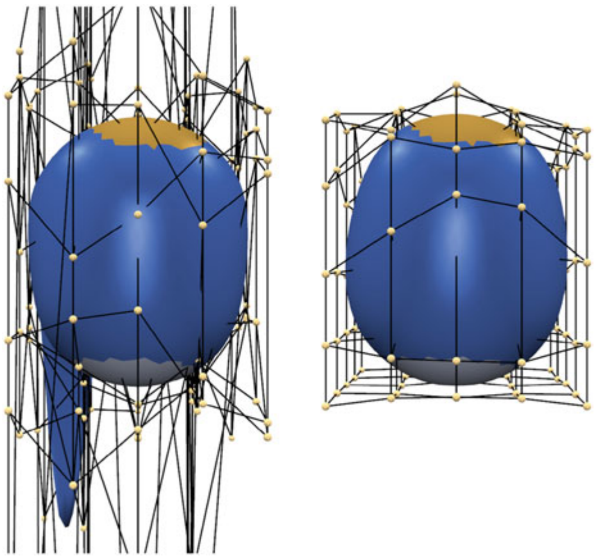


**Fig. 8** Artifacts in the deformation of a sphere due to lack of clamping of singular values. DM-FFD with a $5 \times 5 \times 5$ control lattice. Different clamping values: $10^{-10}$ (*left*) and $10^{-2}$ (*right*). The handle displacements are exactly the same in both examples

Non-manifold configurations are problematic in general, since in this case the 1-ring neighborhood of a vertex can no longer be traversed reliably. While this does not pose a direct problem for the methods investigated, it might prevent the use of a more efficient reduced constraint direct manipulation interface as described in [3].

## 3.3 Quality of Deformation

The quality of the deformation includes several aspects. On the most general level, the deformation should be free of any unexpected oscillations or artifacts. Following the principle of *simplest shape* [29], the deformation function should be smooth, fair, and physically plausible. Furthermore, we want the deformation to maintain mesh element quality as much as possible in order to allow for as large as possible deformations. We note, however, that the methods we consider do not incorporate additional mesh optimization procedures that are eventually required for particularly large deformations.

As a first benchmark we investigate the smoothness of the deformation techniques by analyzing the curvature of a surface mesh after deformation. More specifically, we consider mean curvature defined as

$$H = \frac{\kappa_1 + \kappa_2}{2},$$

where $\kappa_1$ and $\kappa_2$ are the principal (maximum and minimum) curvatures of the surface. Using the cotangent weight discretization of the Laplace-Beltrami operator [5] we compute the mean curvature on for a given vertex $\mathbf{x}_i$ of the mesh by

$$H(\mathbf{x}_i) = \frac{1}{2} \|\Delta \mathbf{x}_i\| .$$

For more details we refer the reader to [5]. A color-coded mean curvature visualization is shown in Fig. 9 after performing a pre-defined deformation with both RBFs and DM-FFD. As can be seen from the visualization, DM-FFD suffers from aliasing artifacts due to its lattice-based nature. The same artifacts occur in control point FFD. In contrast, RBFs result in highly smooth deformations due to their built-in minimization of physically-inspired energies as described in Sect. 2.3.

FFD results in deformations that are not necessarily physically plausible. Especially its direct manipulation variant does not optimize for a high quality deformation, but for minimization of control point movement. In general, using a lattice-based method the shape of the deformation strongly depends on the resolution and form of the control lattice, as shown in Fig. 10. Therefore, it becomes highly difficult to predict the shape resulting from a particular deformation setup in advance.

Another problem with lattice-based methods is the continuity in case of partial control grids. If the control grid covers only a subset of the object, non-smooth transitions between object points inside the control volume and those outside may occur (see Fig. 11, center). In such cases additional sheets of control points have to be inserted in order to assure a smooth transition (Fig. 11, right). This not only complicates the setup process of FFD, it also introduces unnecessary degrees of freedom due to bad adaptivity (see Sect. 3.4).

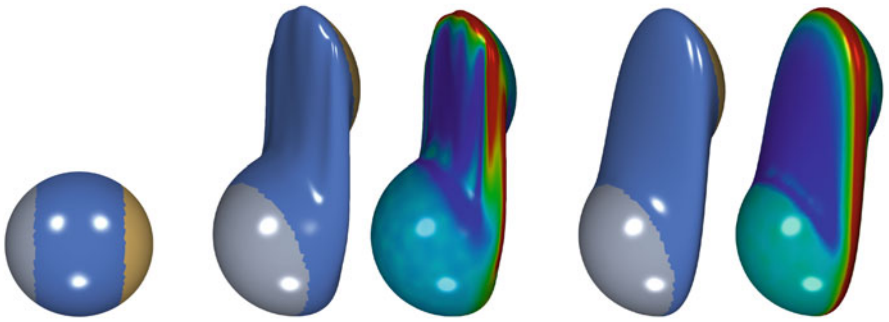


**Fig. 9** Comparison of mesh smoothness after deformation based on mean curvature visualizations. *Red* indicates high curvature, *blue* low curvature. *From left to right*: Setup, deformed mesh and curvature visualizations for DM-FFD (729 control points) and RBFs (792 kernels)

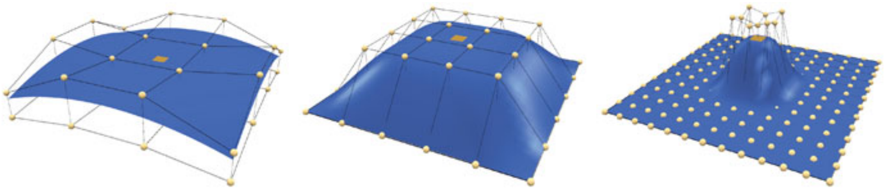


**Fig. 10** Dependency of the deformation on the control lattice resolution. For all examples the same handle region was moved by the same translation

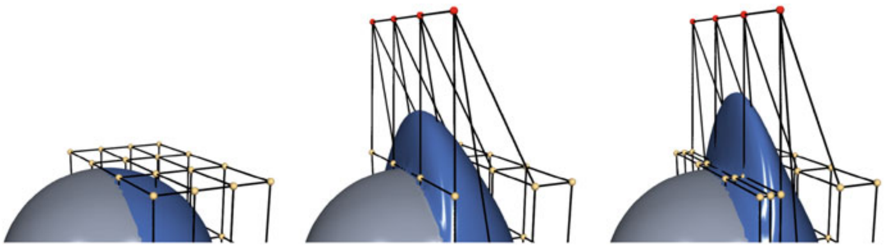


**Fig. 11** Continuity problems in FFD in case of partial control volumes. *From left to right*: Original setup, non-smooth transition, smooth transition

## 3.4 Adaptivity

In general, the adaptivity of a deformation method describes how well the method is capable of approximating a certain shape with an as low as possible number of degrees of freedom (DoFs). In the context of shape optimization the ability to dynamically add additional DoFs in regions of high interest is particularly important.

In order to evaluate the adaptivity we implemented a benchmark test that matches a source shape to a given target shape. In this test case, all vertices are prescribed as constraints, and the deformation method has to match the shape as closely as possible. For each of the methods we start with a low number of DoFs and successively refine the method to include more and more DoFs. We stop refinement once the number of DoFs is equal to the number of constraints.

Adaptivity can be measured best when approximating a target shape that is identical to the source shape for most vertices while having sharp local features in another region. The target shape is shown in Fig. 12. This shape is particularly demanding since the transition from the plane to the feature area is very steep.

In case of DM-FFD we perform adaptive refinement by inserting additional control point planes in $x$- and $y$-directions in those cells containing the vertex with the largest error. We do not perform refinement in $z$-direction, since in our example this would only result in wasted degrees of freedom. As becomes clear from Fig. 12, the adaptivity of DM-FFD is generally poor, since it depends on the resolution of the lattice being used. While increasing the resolution of the lattice leads to sufficient degrees of freedom to approximate fine details as well, at the same time the insertion process also alters the deformation itself. An alternative to the current control point refinement would be to use knot insertion.

In case of RBFs we use straightforward adaptive greedy refinement [30]. Initially, we uniformly sample the plane with a given number of kernels. We then successively add additional kernels at the vertices of the mesh having the largest errors. The results in Fig. 12 clearly confirm that RBFs provide superior approximation accuracy compared to DM-FFD.

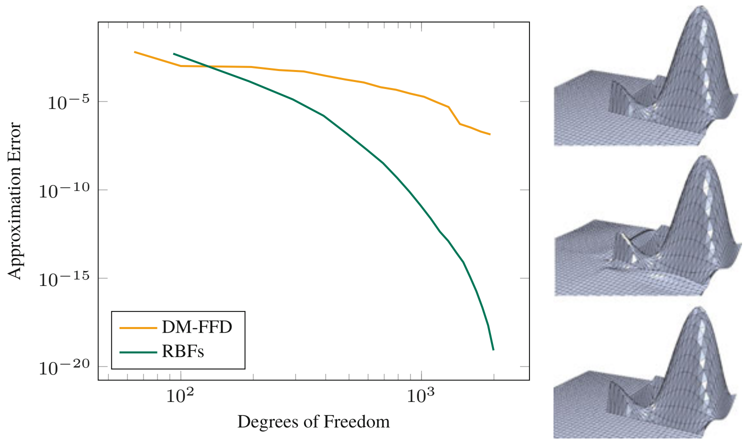


**Fig. 12** Adaptive refinement benchmark results. *Left*: degrees of freedom vs. approximation error. *Right*: example results. *From top to bottom* : target shape, DM-FFD (900 DoF), and RBFs (993 DoF)

## *3.5 Precision*

The precision of a deformation method describes the accuracy in satisfying the positional constraints as prescribed by the user or optimization method. Typically, the accuracy is either *exact*, only provided in a *least-squares* sense, or only in a *qualitative* manner. Manipulating control points of a lattice as in case of FFD can only provide qualitative precision. Directly manipulated FFD improves on this by providing precision in a least-squares sense through the solution of (3). Finally, by solving (8) RBFs allow for exact satisfaction of constraints, thereby offering the highest level of precision. The quantitative results of Sect. 3.4 underline the differences in precision.

## *3.6 Volume Mesh Morphing*

The ability to morph an existing volumetric simulation mesh according to a updated CAD geometry or alongside with a changed surface mesh is a crucial feature for deformation techniques in simulation-based design optimization: By avoiding costly (re-)meshing, it drastically reduces the computational cost, and it enables the construction of fully automatic optimization loops. This benchmark is particularly meaningful, since it accumulates results from the previous synthetic benchmarks. Even though both FFD and RBFs allow for volume mesh morphing due to their space-based nature, there are significant differences in the resulting mesh element quality. We present two different test scenarios involving three different mesh types: We investigate morphing of unstructured tetrahedral and structured hexahedral meshes according to an updated CAD geometry in Sect. 3.6.1. Finally, present a test-case of the DrivAer model involving an arbitrary polyhedral mesh to used CFD computations in Sect. 3.6.2.

In case of DM-FFD we generate a uniform control lattice enclosing the complete volume mesh. Unfortunately, the resolution required to satisfy given deformation constraints as precisely as possible is not known in advance and heavily depends on the complexity of the deformation and the geometry to be deformed. To accommodate for this, we investigate different grid resolutions, namely from $5^3$, $10^3$, $15^3$, and $25^3$ control points (referred to as DM-FFD-5/10/15/25 below). Therefore, the problem of automatic control grid generation is largely unsolved and a serious obstacle for fully automatic optimization procedures.

### **3.6.1 Pipe Model**

In this section we investigate mesh quality based on the morphing benchmarks introduced in [36, 38]. Given an initial CAD surface and a volume mesh, shape variations are created by changing geometric parameters of the CAD model and computing updated surface nodes to match the new geometry. The surface nodes are then used as input to the morphing technique computing updated interior volume nodes. For a more detailed description of the benchmarks we refer to [36, 38]. We choose absolute morphing of the unstructured tetrahedral and structured hexahedral Pipe meshes as a representative example and present the results by means of minimum Scaled Jacobian in the volume mesh vs. percentage of parameter change in the CAD model in Fig. 14. The plots show that RBFs better preserve element quality. Note that while the low resolution DM-FFD-5 test case results in more or less reasonable mesh quality, the constraints are not fulfilled exactly, i.e., the boundary nodes of the volume mesh do not match the update CAD surface (see Fig. 13, right for an example). In contrast, higher resolutions are not capable of dealing with large changes due to the increasing locality of the deformation (Fig. 14).

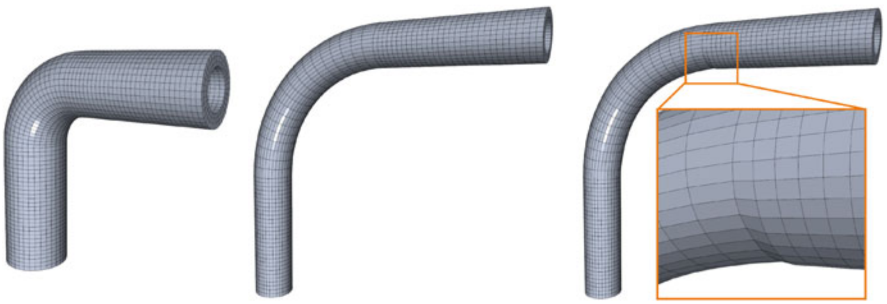


**Fig. 13** Pipe model morphing examples. *From left to right*: Initial mesh, RBF, DM-FFD-5

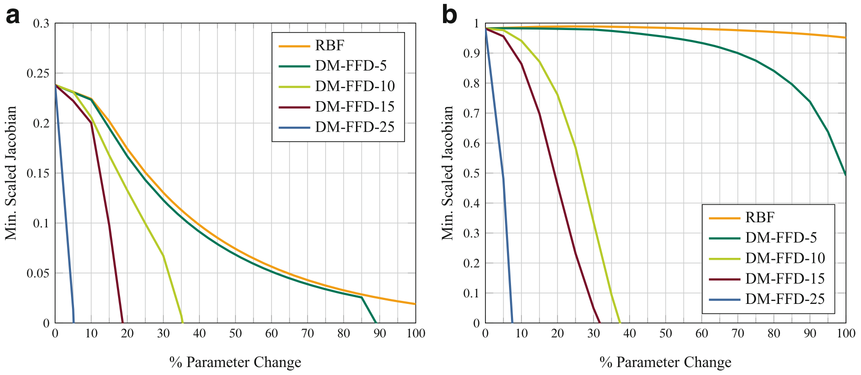


**Fig. 14** Pipe model morphing results. Mesh quality vs. parameter change. (**a**) Pipe Tet model-absolute; (**b**) pipe Hex model-absolute

### 3.6.2 DrivAer Model

As an application-oriented benchmark we investigate an exemplary CFD test case for the DrivAer model. We use OpenFOAM [24] for the CFD setup and generate the volume mesh using the `snappyHexMesh` utility. The resulting arbitrary polyhedral mesh contains 1.2M cells and 1.6M points. To investigate the resulting mesh quality we use OpenFOAM's `checkMesh` tool, which analyzes general mesh properties, such as connectivity, ordering, and orientation, but also essential mesh quality characteristics, such as cell orthogonality, aspect ratio, and face skewness. For the deformation setup we select three handle vertices on the car roof while keeping the outer boundary of the volume mesh fixed. For the deformation itself we simply lift the handle vertices upwards.

A cut view of the resulting volume mesh and car surface patch is shown in Fig. 15. A summary of results as obtained by OpenFOAM's `checkMesh` tool is given in Table 2. For a detailed description of the individual mesh checks we refer to the OpenFOAM [24] documentation. In case of RBFs the volume mesh is still usable and all mesh quality checks succeed. In case of DM-FFD-10 and DM-FFD-15 the meshes are still usable, but the cell orthogonality check warns about one non-orthogonal cell that might spoil the accuracy and/or convergence of the simulation. Furthermore, we note that for more complex deformations the $10^3$ and $15^3$ resolutions might not be sufficient to satisfy the displacement constraints with acceptable precision. In case of the higher resolution DM-FFD-25 setup several mesh quality checks fail and the mesh is no longer usable for simulation at all:

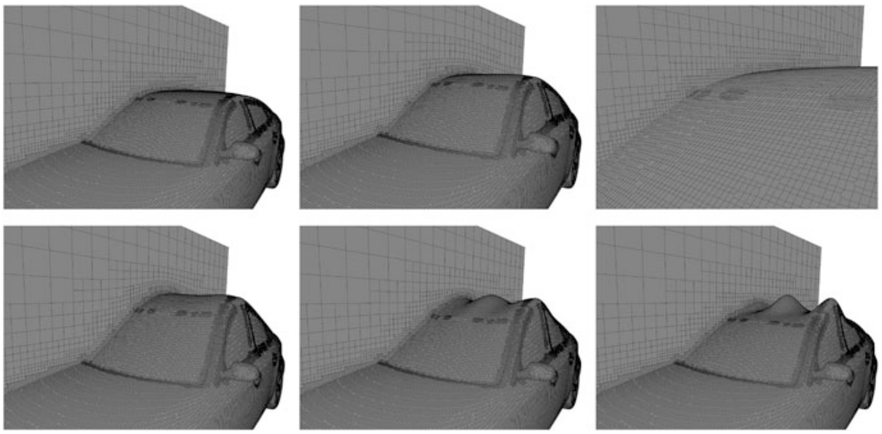


**Fig. 15** Cut-view and car surface patch of the resulting volume mesh after deformation. Original (*top left*), RBF (*top center*), zoom to the surface (*top right*), DM-FFD (*bottom row*). In case of DM-FFD the results for control grid resolutions $10^3$, $15^3$, and $25^3$ are shown

**Table 2** Results reported by OpenFOAM's `checkMesh`

| | Aspect ratio | Cell orthogonality | Face skewness | Face pyramids |
|---|---|---|---|---|
| Original | 6.9 ✓ | 64.7 ✓ | 3.4 ✓ | ✓ |
| RBF | 6.6 ✓ | 68.6 ✓ | 3.7 ✓ | ✓ |
| DM-FFD-10 | 7.0 ✓ | 71.3 ! | 3.6 ✓ | ✓ |
| DM-FFD-15 | 7.0 ✓ | 70.7 ! | 3.4 ✓ | ✓ |
| DM-FFD-25 | 2.5e+195 ✗ | 179.7 ✗ | 1,031.8 ✗ | ✗ |

Successful tests are indicated by a ✓, warnings by !, and errors by ✗. Numbers are given for the worst quality element in the mesh

The mesh contains 151 high aspect ratio cells, 1,353 non-orthogonal faces, 1,414 incorrectly oriented face pyramids, and 62 highly skewed faces. For all DM-FFD setups, Fig. 15 again demonstrates the strong dependence of the resulting shape on the chosen control grid resolution.

## 4 Conclusions

The results of the individual benchmarks show that there are significant differences between the deformation methods. A compact and simplified summary of the results is presented in Table 3. For each of the benchmarks and methods we assign either a positive ( + ), neutral ( ○ ), or a negative ( – ) assessment. Both FFD and DM-FFD achieve mixed results. While DM-FFD improves upon FFD in terms of precision, computational costs increase and robustness issues might occur. Both FFD techniques expose significant weaknesses with regards to adaptive refinement and quality of the deformation. In contrast, RBFs score the largest number of positive and only one neutral assessment.

However, the choice of a deformation method heavily depends on the needs of a given design optimization scenario. If the scenario neither demands for precise constraint satisfaction nor for adaptive refinement but only aims for general exploration of the design space, FFD offers a simple and robust deformation technique. In many cases, however, exact control is highly important in order obtain valid designs that meet production limitations such as keeping critical components fixed or deforming them only rigidly. In such cases, we clearly recommend RBFs over both FFD and its direct manipulation variant.

In some optimization scenarios the locality of the deformation might also be an important aspect. Both FFD methods allow for local deformations—depending on control grid resolution and setup as well as basis function degree. In contrast, our RBF deformations are global due to our choice of triharmonic basis functions. While there exist compactly supported RBFs [39], these basis functions lack the built-in

**Table 3** Summary of results

| | Performance | Robustness | Quality | Adaptivity | Precision |
|---|---|---|---|---|---|
| FFD | ○ | + | ○ | − | − |
| DM-FFD | − | ○ | ○ | − | ○ |
| RBF | ○ | + | + | + | + |

For each benchmark test and deformation method we assign a negative ( − ), neutral ( ○ ), or a positive ( + ) assessment

energy minimization of (7). However, in many cases a proper setup of fixed and handle regions in the direct manipulation interface eventually provides a sufficient degree of locality.

Naturally, all of three methods can be enhanced in several ways. In case of both FFD methods the use of more flexible basis functions such as T-splines [32] or truncated hierarchical B-splines [9] would drastically improve the adaptivity of the respective methods. As for RBFs, constraining the deformation function to be positive—similar to the bounded biharmonic weights introduced in [14]—offers an interesting perspective for future work.

**Acknowledgements** Daniel Sieger gratefully acknowledges the financial support from Honda Research Institute Europe (HRI-EU). Mario Botsch is supported by the German National Research Foundation (DFG CoE 277: CITEC). The authors kindly thank Matthew Staten from Sandia National Laboratories for originally providing us with Pipe models from [38].

## References

1. Bechmann, D.: Space deformation models survey. Comput. Graph. **18**(4), 571–586 (1994)
2. de Boer, A., van der Schoot, M., Bijl, H.: Mesh deformation based on radial basis function interpolation. Comput. Struct. **85**, 784–795 (2007)
3. Botsch, M., Kobbelt, L.: Real-time shape editing using radial basis functions. Comput. Graph. Forum (Proc. Eurograph.) **24**(3), 611–621 (2005)
4. Botsch, M., Sorkine, O.: On linear variational surface deformation methods. IEEE Trans. Vis. Comput. Graph. **14**(1), 213–230 (2008)
5. Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Levy, B.: Polygon Mesh Processing. AK Peters, Natick (2010)
6. Coquillart, S.: Extended free-form deformation: a sculpturing tool for 3D geometric modeling. In: Proceedings of ACM SIGGRAPH, pp. 187–196 (1990)
7. Fasshauer, G.E.: Meshfree Approximation Methods with MATLAB. World Scientific Publishing, Singapore (2007)
8. Gain, J., Bechmann, D.: A survey of spatial deformation from a user-centered perspective. ACM Trans. Graph. **27**, 107:1–107:21 (2008)
9. Giannelli, C., Jüttler, B., Speleers, H.: THB–splines: the truncated basis for hierarchical splines. Comput. Aided Geom. Des. **29**, 485–498 (2012)
10. Golub, G.H., van Loan, C.F.: Matrix Computations. Johns Hopkins University Press, Baltimore (1989)

11. Griessmair, J., Purgathofer, W.: Deformation of solids with trivariate B-splines. In: Proceedings of Eurographics (1989)
12. Heft, A.I., Indinger, T., Adams, N.A.: Introduction of a new realistic generic car model for aerodynamic investigations. In: SAE 2012 World Congress (2012)
13. Hsu, W.M., Hughes, J.F., Kaufman, H.: Direct manipulation of free-form deformations. In: Proceedings of ACM SIGGRAPH, pp. 177–184 (1992)
14. Jacobson, A., Baran, I., Popović, J., Sorkine, O.: Bounded biharmonic weights for real-time deformation. ACM Trans. Graph. **30**, 78:1–78:8 (2011)
15. Jakobsson, S., Amoignon, O.: Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization. Comput. Fluids **36**(6), 1119–1136 (2007)
16. Lamousin, H., Waggenspack, N.: NURBS-based free-form deformations. IEEE Comput. Graph. Appl. **14**(6), 59–65 (1994)
17. MacCracken, R., Joy, K.I.: Free-form deformations with lattices of arbitrary topology. In: Proceedings of ACM SIGGRAPH, pp. 181–188 (1996)
18. Manzoni, A., Quarteroni, A., Rozza, G.: Shape optimization for viscous flows by reduced basis methods and free-form deformation. Int. J. Numer. Meth. Fluids **70**, 646–670 (2011)
19. Menzel, S., Sendhoff, B.: Representing the change-free form deformation for evolutionary design optimization. Stud. Comput. Intell. **88**, 63–86 (2008)
20. Menzel, S., Olhofer, M., Sendhoff, B.: Application of free form deformation techniques in evolutionary design optimisation. In: Proceedings of the 6th World Congress on Structural and Multidisciplinary Optimization (2005)
21. Menzel, S., Olhofer, M., Sendhoff, B.: Direct manipulation of free form deformation in evolutionary design optimisation. In: International Conference on Parallel Problem Solving from Nature (PPSN), pp. 352–361 (2006)
22. Michler, A.K.: Aircraft control surface deflection using RBF-based mesh deformation. Int. J. Numer. Meth. Eng. **88**(10), 986–1007 (2011)
23. Moccozet, L., Thalmann, N.: Dirichlet free-form deformations and their application to hand simulation. In: Computer Animation '97, pp. 93–102 (1997)
24. OpenFOAM: Open source field operation and manipulation C++ libraries. http://www.openfoam.org (2012)
25. OpenMP Architecture Review Board: OpenMP application program interface version 3.1. URL http://www.openmp.org/mp-documents/OpenMP3.1.pdf (2011)
26. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes: The Art of Scientific Computing, 2nd edn. Cambridge University Press, Cambridge (1992)
27. Samareh, J.A.: A survey of shape parameterization techniques. Technical Report NASA/CP-1999-209136/PT1, NASA Langley Research Center (1999)
28. Samareh, J.A.: Aerodynamic shape optimization based on free-form deformation. In: Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference (2004)
29. Sapidis, N.S.: Designing Fair Curves and Surfaces: Shape Quality in Geometric Modeling and Computer-Aided Design. Society for Industrial and Applied Mathematics, Philadelphia (1994)
30. Schaback, R., Wendland, H.: Adaptive greedy techniques for approximate solution of large RBF systems. Numer. Algorithms **24**(3), 239–254 (2000)
31. Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. In: Proceedings of ACM SIGGRAPH, pp. 151–159 (1986)
32. Sederberg, T.W., Zheng, J., Bakenov, A., Nasri, A.: T-splines and T-NURCCs. ACM Trans. Graph. **22**(3), 477–484 (2003)
33. Sibson, R.: A vector identity for the dirichlet tessellation. Math. Proc. Camb. Philos. Soc. **87**, 151–155 (1980)
34. Sieger, D., Menzel, S., Botsch, M.: A comprehensive comparison of shape deformations in evolutionary design optimization. In: Proceedings of the 3rd International Conference on Engineering Optimization (2012)
35. Sieger, D., Menzel, S., Botsch, M.: High quality mesh morphing using triharmonic radial basis functions. In: Proceedings of the 21st International Meshing Roundtable (2012)

36. Sieger, D., Menzel, S., Botsch, M.: RBF morphing techniques for simulation-based design optimization. Eng. Comput. **30**(2), 161–174 (2014)
37. Song, W., Yang, X.: Free-form deformation with weighted T-spline. Vis. Comput. **21**, 139–151 (2005)
38. Staten, M.L., Owen, S.J., Shontz, S.M., Salinger, A.G., Coffey, T.S.: A comparison of mesh morphing methods for 3D shape optimization. In: Proceedings of the 20th International Meshing Roundtable, pp. 293–311 (2011)
39. Wendland, H.: Scattered Data Approximation. Cambridge University Press, Cambridge (2005)

# Creating Free-Surface Flow Grids with Automatic Grid Refinement

**Jeroen Wackers, Ganbo Deng, Emmanuel Guilmineau, Alban Leroyer, Patrick Queutey, and Michel Visonneau**

**Abstract** The objective of this work is to create grids for free-surface water flow simulation entirely with automatic grid refinement. It is shown why it is necessary to refine the mesh iteratively as the solution converges and why refinement and derefinement of hexahedral cells must be treated anisotropically.

The proposed refinement criterion is a combination of the pressure Hessian with refinement at the free surface, in order to capture the flow which drives the surface motion and the position of the surface itself. Smoothing is needed in the computation of the Hessian in order to remove oscillations in the pressure, the pressure Hessian is extrapolated through the free surface to remove its discontinuity there.

Two test cases confirm that effective fine meshes for wave computation can be created with the proposed automatic refinement procedure.

## 1 Introduction

Free-surface flows with gravity waves, such as the water flow around ships and offshore structures, are created through the interaction between the turbulent viscous flows on both sides of the interface and the motion of the interface. To simulate such flows, the position of the free surface needs to be modelled as well as the velocity and pressure fields. An attractive and very robust model is the surface capturing mixture-fluid approach [8]. Here, the entire fluid is modelled as a numerical mixture of the pure fluids on the two sides of the interface. The proportion of both fluids in the mixture is computed with a convection equation for the volume fraction of one of them, having a discontinuous inflow condition. This discontinuity is convected through the flow, implicitly giving the position of the free surface without any specific treatment of the surface region (as opposed to, for example,

J. Wackers (✉) • G. Deng • E. Guilmineau • A. Leroyer • P. Queutey • M. Visonneau
Laboratoire de recherche en Hydrodynamique, Energétique et Environnement Atmosphérique, CNRS UMR 6598, Ecole Centrale de Nantes, 1 rue de la Noë, B.P. 92101, 44321 Nantes Cedex 03, France
e-mail: jeroen.wackers@ec-nantes.fr

S. Perotto, L. Formaggia (eds.), *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, SEMA SIMAI Springer Series 5,
DOI 10.1007/978-3-319-06053-8_15

the Volume-of-Fluid or Level-Set approaches). However, for accuracy, special care must be taken to keep the numerical interface sharp.

Adaptive local grid refinement is particularly well suited for these simulations. Free-surface water flows have many features which are local in nature and whose exact position is difficult to estimate beforehand, so their precision can be increased with adaptive grid refinement. For example, refinement around the surface strongly increases the resolution of the volume fraction equation, so the modeling of the free surface is improved. But also other aspects of these flows, such as wakes and trailing vortices, are resolved with greater precision when grid refinement is used.

The unstructured Reynolds-averaged Navier-Stokes solver ISIS-CFD which we develop contains an automatic grid refinement method [10–13]. This flow solver, distributed by NUMECA Int. as part of the FINE/Marine software, is aimed at the simulation of realistic flow problems in all branches of marine hydrodynamics. Therefore, the grid refinement method is general and flexible, featuring anisotropic refinement on unstructured hexahedral grids, derefinement of previous refinements to enable unsteady flow computation, and full parallelisation including integrated dynamic load balancing. The anisotropic refinement is based on metric tensors. To our knowledge, it is the first grid adaptation method included with success in a general-purpose hydrodynamic flow solver.

In our earlier work on grid refinement for free-surface flows, the multiphysics character of the flows was not explicitly taken into account for the grid refinement. Instead, the original grid was chosen sufficiently fine to get a reasonable resolution of the flow, then automatic grid refinement was used to improve the accuracy of one particular flow feature. Thus, gravity waves at the water surface were computed with refinement based on the discontinuity in the volume fraction [12] and wake flows with refinement based on the pressure [10, 13].

The objective of this paper is to go beyond these earlier works and create fine meshes for free-surface flow simulation entirely with adaptive grid refinement. This removes the need for original grids with finer cells in all the possible positions of the water surface and other important flow features, that are difficult to generate and very costly if strong waves appear or if a simulated object is free to move. The new refinement approach simplifies the mesh generation for users and can be much more efficient, since all fine cells are placed only there where they are really needed.

For such refinement, the technical part of the algorithm has to be modified to treat both refinement and derefinement of cells in an anisotropic way, allowing division or merging of cells in one direction only. The first part of the paper highlights the aspects of the existing algorithm that are essential for free-surface flows, such as the need to continuously modify the refined mesh as the solution converges, and shows why anisotropic derefinement must be introduced. The most important evolution however is in the refinement criterion, which must be a combination of different physical sensors due to the multiphysics character of two-fluid flows. A criterion is proposed which combines refinement at the free surface with a pressure Hessian criterion, modified to remove the singularity in the pressure gradient at the surface.

The paper is organised as follows. Section 2 describes the ISIS-CFD flow solver and the meshes that we use. Section 3 then discusses the aspects of its grid refinement method which are relevant for free-surface flow. Section 4 looks at anisotropic derefinement, which is currently under development. The combined refinement criterion is introduced in Sect. 5. Finally, Sect. 6 shows a test on two ship flow cases, with a new case which highlights the interest of grid refinement for a flow containing both free surface and strong vortices.

## 2 Solver and Meshes

ISIS-CFD is an incompressible unsteady Reynolds-averaged Navier-Stokes method. The solver is based on the finite-volume method to build the spatial discretisation of the transport equations. The velocity field is obtained from the momentum conservation equations and the pressure field is extracted from the mass conservation constraint transformed into a pressure equation. These equations are similar to the Rhie and Chow SIMPLE method [9], but have been adapted for flows with discontinuous density fields. As mentioned, free-surface flow is simulated with a mixture flow approach: the water surface is captured with a conservation equation for the volume fraction of water, discretised with specific compressive discretisation schemes. A detailed description of the solver is given by [3, 8].

The unstructured discretisation is face-based: fluxes are computed face by face, the reconstructions of the cell-centred state variables to the face centres are made with interpolations that use the two cells next to a face and their neighbour cells without a-priori assumptions about the cell topologies. And while the linearised systems used to solve the momentum and pressure equations are formulated in the cell-centred unknowns, these systems are constructed by summing the contributions of the faces to each cell. Thus, no cell topology assumptions are made anywhere, which means that cells with any number of arbitrarily shaped faces are accepted.

The solver is mostly used with unstructured hexahedral grids generated by the HEXPRESS grid generator which is also part of FINE/Marine. The grid in Fig. 1 shows the typical features of such meshes: several semi-structured regions, with body-fitted boundary grids near the walls in order to ensure the best possible grid quality in the boundary. The grid consists purely of hexahedral cells, with mesh size variations obtained by placing one large cell next to two or four smaller neighbour cells. Due to its face-based nature, the ISIS-CFD solver treats these grids just the same as any other type of mesh. With these meshes, good solution accuracy is obtained from the semi-structured parts and the body-fitted boundary meshes, combined with the flexibility to mesh complex geometries.

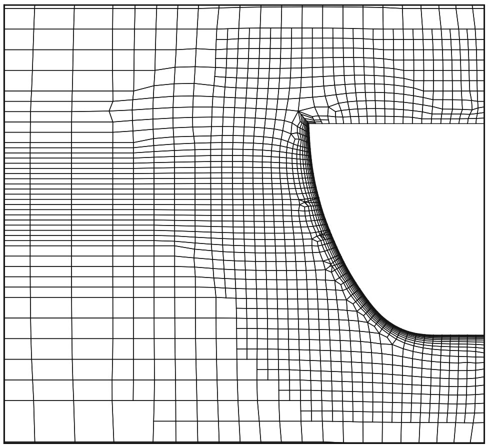


**Fig. 1** Cut through an unstructured hexahedral mesh for a ship geometry. The objective of the present work is to generate the fine grid around the water surface with adaptive refinement

## 3 Grid Refinement Method

The natural method of grid adaptation for unstructured pure-hexahedral meshes is local grid refinement by dividing cells into smaller cells, as this is how the original grids themselves are constructed. Thus, adaptively refined grids are of the same type as all other grids so the flow solver can use them without modifications. This section shows our existing grid refinement method and how refinement criteria in a metric context are used to pilot the refinement, focusing on those aspects which are relevant for free-surface flows. More details of the metric criteria are found in [12].

### *3.1 Refinement Algorithm*

For free-surface flows the simulated position of flow features often depends strongly on the level of grid refinement; for example, waves may become steeper when the flow is resolved on finer and finer grids. This means either that automatic grid refinement should be applied in a large buffer zone around the flow features of interest to take into account their displacement once the grid is refined, or that the refinement procedure must be iterative, continuously changing the grid as the solution converges. Thus, in each refinement iteration, new cells must be refined if the features of interest have moved, while previous refinements of other cells may need to be undone in positions which the features of interest have left. For such an approach, the derefinement of previously refined cells is a necessity.

Our refinement algorithm takes this second approach. ISIS-CFD computes both steady and unsteady flows with a time integration technique. For grid refinement, after a given number of time steps (usually 25 for steady flows and 2–4 for unsteady flows), the grid is adapted to the current solution, after which the time integration continues until the grid is adapted again. For steady flows, this procedure eventually converges (typically after 40–50 refinement cycles). If the refinement criterion (Sect. 3.4) indicates that the grid is well adapted to the flow and the flow solution itself has converged, then the refinement procedure will no longer modify the mesh.

## *3.2 Anisotropy*

Anisotropic refinement is the possibility to divide a hexahedral cell in its three directions independently, either in two, four or eight smaller cells. This is essential for our refinement procedure, since isotropic refinement (division in eight cells only) is much too costly in three dimensions if very fine cells are needed to accurately resolve a local flow feature. By applying anisotropic refinement for features that require a fine grid in only one direction (notably, the water surface!), the total number of cells required can be greatly reduced, or much finer flow details can be resolved.

A second reason for directional refinement is, that the original grids (Fig. 1) already contain anisotropic refinement with cells of different aspect ratios lying side by side. Therefore, when refining, we need to control the size of the fine cells in all their directions independently, otherwise refined grids may have smoothly varying sizes in one direction, but repeated changes from fine to coarse and back to fine in another. Isotropic refinement cannot prevent this (see Fig. 2 for an example), so directional refinement is the mandatory choice.

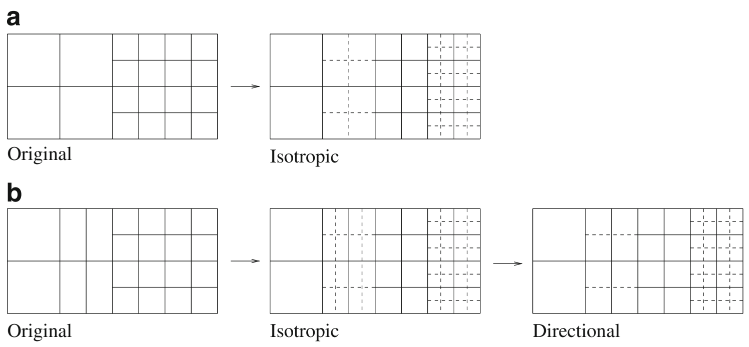


**Fig. 2** Isotropic grid refinement is satisfactory on an original grid where all cells have the same aspect ratios (**a**), otherwise directional refinement is needed (**b**). (**a**) Locally isotropic grid; (**b**) Locally anisotropic grid

### 3.3 Quality of Refined Cells

Since the cells in the refined grid are formed by the subdivision of the cells in the original grid, the shape of the refined cells is entirely determined by the original grid. If these original cells are close to rectangular, then the refined cells will be rectangular as well (this is the case in Fig. 2). However, the quality of bad original cells is deteriorated by grid refinement. Especially if an original cell is arrow-shaped, its subdivision may result in cells that are turned inside out.

Thus, the quality of the refined grids can be assured by making sure that the original grid is as regular as possible. In HEXPRESS meshes, bad quality cells only appear near the surface of objects, typically at inward-facing angles in the geometry. Usually, bad original grids can be prevented by locally imposing fine original cells near these surface features. This is not contrary to the idea of automatic grid refinement; if the original grid is made to capture the geometry well, the rest of the fine grid can be created by automatic refinement.

### 3.4 Metric-Based Refinement Criteria

The use of metric tensors as refinement criteria allows us to specify different cell sizes in different directions. This technique was first developed for the generation and refinement of unstructured tetrahedral meshes [4]. It is also an extremely useful and flexible framework for the anisotropic refinement of hexahedral meshes.

In the metric context, the refinement criterion is a smoothly varying tensor field whose values at every point in the flow domain indicate what the ideal size for a cell in that position would be. In each cell, the criterion is a $3 \times 3$ symmetric positive definite matrix $\mathscr{C}$, which is interpreted as a geometric transformation of the cell in the physical space to a deformed space. The refinement of the cells is decided as follows. Let the criterion tensors $\mathscr{C}$ in each cell be known (their computation from the flow solution is described in Sect. 5). In each hexahedral cell, the cell sizes $\mathbf{d}_j$ $(j = 1, 2, 3)$, which are the vectors between the opposing face centres in the three cell directions, are determined. Next, the modified sizes are computed as:

$$\tilde{\mathbf{d}}_j = \mathscr{C}\mathbf{d}_j. \tag{1}$$

Finally, a cell is refined in the direction $j$ when the modified size exceeds a given, constant threshold value $T_r$:

$$\|\tilde{\mathbf{d}}_j\| \geq T_r, \tag{2}$$

while a previously refined group of cells is derefined back into one cell if:

$$\|\tilde{\mathbf{d}}_j\| \leq T_r/d \qquad \forall j = 1 \cdots 3, \tag{3}$$

for all cells in the group. $d$ is chosen slightly larger than 2, to prevent cells being alternately derefined and re-refined (because $\tilde{\mathbf{d}}_j$ doubles when cells are derefined). Since the procedure only uses the vectors $\mathbf{d}_j$ to characterise a cell, it can be used for any shapes and sizes of cells.

The objective of the refinement is thus to create a uniform grid in the deformed space. The tensors $\mathscr{C}$ are direct specifications of the desired cell sizes: in a converged refined grid, the cell sizes are inversely proportional to the magnitude of the $\mathscr{C}$. The threshold $T_r$ functions as a global specification of the fineness of the grid; sensible choices for $T_r$ in different situations are discussed in Sect. 6 (see also Sect. 5.3).

## 4 The Need for Directional Derefinement

The derefinement of previously refined cells, like the refinement, ought to be treated in an anisotropic way. Until now, the refinement of a cell is decided separately for each direction [Eq. (2)] but the derefinement only allows the complete undoing of a refinement step; if a cell was divided in eight, the eight cells have to be derefined back into one which requires the criterion (3) to be satisfied in all three directions $j$. This section shows why this limitation is much too restrictive and how we will modify our method to anisotropic derefinement. As the transformation of an existing grid refinement method like ours is difficult compared with developing it correctly from the beginning, this section may serve as a warning to others. . . .

For free-surface flows, isotropic derefinement is inefficient because cells refined initially in different directions often need to remain refined in at least one direction. A typical example is the simulation of a travelling wave with grid refinement around the free surface (see Sect. 5.1). When the wave passes, the free surface is inclined so the grid is refined both in horizontal and in vertical direction. Then after the wave has passed and the free surface has returned to its rest state, the vertical refinement must remain in place. Thus, with isotropic derefinement, the horizontal refinement cannot be removed either. This leads to large clusters of unnecessary fine cells and the more the grid is refined, the more this is evident. Since the creation of free-surface grids entirely with automatic refinement requires much refinement of the original cells, this problem has to be solved.

A situation which shows the problem of isotropic derefinement particularly well is the initialisation of the refined grid. Usually the mesh is adapted to the undisturbed position of the free surface before starting the simulation, to have a good initial resolution of the volume fraction. When the original grid contains no specific refinement at the free surface, it has varying cell sizes at the surface position (Fig. 3a): the grid near a solid wall is much finer than elsewhere. Thus, the discontinuity in the volume fraction is thinner near solid walls than far away, which means that its top and bottom are inclined! This, in turn, leads to horizontal refinement of cells. At the end of the initialisation, when the vertical cell sizes around the surface have become equal due to the automatic refinement,

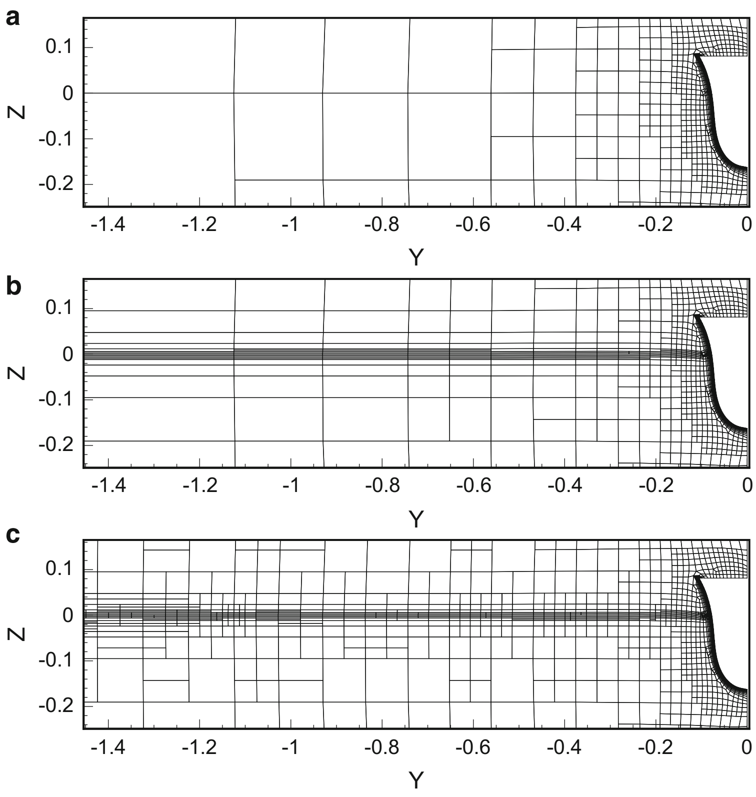


**Fig. 3** Initialisation of the free-surface refined grid from the original grid (**a**): as it should be (**b**) and with only isotropic derefinement (**c**). The middle figure was created artificially by initialising without allowing refinement in $x$- and $y$-directions

this horizontal refinement is no longer required (Fig. 3b). However, with isotropic derefinement it cannot go away because the vertical refinement must remain in place (Fig. 3c).

Thus, for efficient derefinement it is necessary to remove previous refinements in one direction only (in Fig. 3 the horizontal direction), for example by changing a cell divided in four into one divided in two. The problem that we have is that our data structure is not suited for this, since the history of the refinement is stored in the refined cells as 'mother' pointers towards the cells that were divided and 'sister' pointers to one of the other refined cells, forming a loop. This structure does not preserve the relative position of the fine cells. The solution that we are working on consists of adaptively modifying the refinement history after the fact in order to suit the required directional derefinement. Clearly, it would be much more elegant to work with a suitable data structure, for example one which directly indicates the position of a small cell within the original cell.

## 5 Combined Criteria

Suitable refinement criteria for free-surface flows must take into account gravity waves, which propagate through a cyclic exchange of potential (gravity) and kinetic energy, caused by the interaction of the free-surface motion with the pressure and velocity fields below the surface. So in order to correctly simulate free-surface flows, a good resolution is needed of the volume fraction equation which gives the position of the free surface, as well as the pressure and velocity variations below the surface.

A grid refinement criterion for free-surface wave simulation must therefore be based both on the pressure and velocity field and on the volume fraction. For these two, different indicators are used because the volume fraction $\alpha$ is discontinuous at the free surface and constant everywhere else, while the pressure and the velocity are smooth in the whole flow field except at the surface. Spatial derivative based error indicators can identify the regions of importance for the flow field below the surface, but do not work at the water surface itself since any derivative of $\alpha$ goes to infinity when the grid is refined. Also, the grid specified by the criterion must be as uniform as possible near the surface, since transitions from fine to coarse cells lead to large errors in the volume fraction. Basing a criterion only on the value of $\alpha$ itself gives the most stable values for the criterion and thus the best meshes.

This section describes a criterion which is based on the Hessian of the pressure, combined with refinement where $\alpha$ is neither 0 nor 1. These criteria are first introduced individually, then we indicate how they are combined. The section is a continuation of the work in [11].

### *5.1 Free-Surface Criterion*

To resolve accurately the volume fraction field $\alpha$ which is a discontinuity that is convected with the flow, it is sufficient to refine the grid at and around the free surface, in the direction normal to the surface. When the surface is locally aligned with the cell directions, anisotropic refinement can be used to keep the total number of cells as low as possible. It is important that cells locally have the same size, to prevent distortions of the volume fraction.

The free-surface criterion $\mathscr{C}_S$ is non-zero when $\alpha$ is neither 0 nor 1. The normal direction to the surface is computed from a field $\alpha_s$ which corresponds to $\alpha$, smoothed out by averaging over a cell and its neighbours a given number of times. The gradient of this field gives the normal directions. The criterion is then derived from vectors $\mathbf{v}_\alpha$ in each cell which are unit vectors in this normal direction for those cells where the smoothed $\alpha_s$ field is non-zero:

$$\mathbf{v}_\alpha = \begin{cases} \frac{\nabla \alpha_s}{\|\nabla \alpha_s\|} & \text{if } 0.1 \le \alpha_s \le 0.9, \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{4}$$

Using the smoothed field guarantees that the normals are well defined and also that the mesh is refined in a certain zone around the surface to create a margin of safety.

In tensor form, the free-surface criterion is computed as matrices having only one non-zero eigenvalue, associated with the direction of the vector $\mathbf{v}_\alpha$. The tensors $\mathscr{C}_S$ are computed as follows (with $\otimes$ representing the tensor product):

$$\mathscr{C}_S = \mathbf{v}_\alpha \otimes \mathbf{v}_\alpha. \tag{5}$$

In the directions normal to the vector $\mathbf{v}_\alpha$, the eigenvalues are zero. This implies a modified cell size of zero [Eq. (1)]. As a consequence, the grid is not refined in these directions. Since the $\mathbf{v}_\alpha$ are unit vectors, the only non-zero eigenvalues of $\mathscr{C}_S$ are equal to 1. Thus, Eq. (2) shows that the threshold value $T_r$ directly indicates the desired cell size at the surface. Also, the specified cell size normal to the surface is *exactly* the same in all surface cells, as required. The free-surface criterion has been used on its own, with good results, in our earlier work [10, 12].

## 5.2 *Computing the Pressure Hessian*

The Hessian matrix of second spatial derivatives can be linked to the interpolation error of a smooth solution, thus it is a well-known refinement criterion for anisotropic refinement (see for example [6, 7]). Here this criterion is based on the pressure, since we prefer a criterion which is insensitive to boundary layers [12]. The number of layers in the boundary layer grid should be the same everywhere for solution accuracy and the required layer thickness is known, so these layers can be created in the original grid and do not have to come from adaptive refinement.

For the numerical computation of the Hessian (see also [11]) we need discretised second-derivative operators. A particular complication for this discretisation is that our meshes always contain places where the grid size changes abruptly, as small cells lie next to twice larger cells (see Sect. 2). The number of these places increases significantly when automatic refinement is used so a suitable technique for computing the Hessian must be insensitive to cell size variations.

### 5.2.1 Definition of the Hessian Criterion

The pressure Hessian matrix is:

$$\mathscr{H}(p) = \begin{bmatrix} (p)_{xx} & (p)_{xy} & (p)_{xz} \\ (p)_{xy} & (p)_{yy} & (p)_{yz} \\ (p)_{xz} & (p)_{yz} & (p)_{zz} \end{bmatrix}. \tag{6}$$

Assuming, heuristically, that an indication of the local truncation error is given by $\mathscr{H}$ times the cell sizes to a power $b$ (where $b$ depends on the numerical method) and requiring equidistribution of this error indicator leads to a refinement criterion where the Hessian matrix is modified with a power law:

$$\mathscr{C}_H = (\mathscr{H}(p)^a), \tag{7}$$

where $\mathscr{H}^a$ has the same eigenvectors as $\mathscr{H}$ and eigenvalues that are those of $\mathscr{H}$ (in absolute value) to the power $a = \frac{1}{b}$. In general, we use $a = \frac{1}{2}$ which is appropriate for a second-order accurate discretisation.

### 5.2.2 Smoothed Gauss (SG) Method

Unfortunately, due to the variations in cell sizes, the numerical pressure is non-smooth. The SIMPLE-based pressure equation in ISIS-CFD contains a Laplace operator in finite-volume form for which normal derivatives on the cell faces are constructed with central interpolation. On non-uniform meshes these interpolations are first-order accurate, so the Laplace operator itself in the worst case has a truncation error of order zero. The pressure still converges because these local errors cancel, but the second derivatives of the pressure have the same order of accuracy as the Laplace operator, i.e. they are inconsistent. This is not due to the numerical evaluation of the second derivatives, but inherent in the pressure solution itself. The consequence for grid refinement is, that refining cells creates large errors in the Hessian on the boundaries between finer and coarser cells. Thus, the grid is not only refined where the solution dictates it, but also in places where it has already been refined. This leads to irregular meshes.

As the error in the Hessian is related to small-scale irregularities in the pressure field, which can be reduced by smoothing, we define a smoothed Gauss (SG) Hessian. Let the Gauss approximation to the gradient of a field $q$ be given as:

$$\vec{\nabla}_G(q) = \frac{1}{V}\sum_f q_f S_f \mathbf{n}_f, \tag{8}$$

where the face values $q_f$ are computed with central interpolation from the two neighbour cells. $V$ is the volume of the cell, $S_f$ are the areas of the faces. With the same face reconstructions, a Laplacian smoothing $\mathscr{L}$ is defined as:

$$\mathscr{L}(q) = \frac{\sum_f q_f S_f}{\sum_f S_f}. \tag{9}$$

Then the SG Hessian is computed as follows:

1. Compute the gradient of $p$ using $\overrightarrow{\nabla}_G$.
2. Smooth each component of the gradient by applying $N$ times the smoothing $\mathscr{L}$, where $N = 4$ is sufficient in most cases.
3. Compute the gradients of the smoothed gradient components using $\overrightarrow{\nabla}_G$.
4. Symmetrize the resulting Hessian matrix by setting $(\mathscr{H})_{ij} = \frac{1}{2}((\mathscr{H})_{ij} + (\mathscr{H})_{ji})$.
5. Smooth the Hessian by applying $N$ times $\mathscr{L}$ to each component.

The reason for this procedure is, that the second derivatives of the pressure have zeroth-order oscillatory errors which means first-order errors in the derivatives and second-order wiggles in the solution itself. The smoothing operator uses the same type of discretisation as the original Laplace equation, so the smoothing itself introduces second-order wiggles. Therefore, it is useless to smooth the pressure. However, the gradient of the pressure has first-order errors; these are still small compared with the gradient itself but they are large enough to be removed by the smoother. Therefore, step 2 is the core of the procedure. The smoothing of the second derivative cannot further remove errors since they are of the same order as the solution now, but it creates a more regular refinement criterion and thus better meshes.

The smoothing procedure decreases the spurious oscillations in the refinement criterion but also reduces the intensity of physical small-scale features. This limitation of the criterion is the reason that all smoothing should be kept to a minimum.

### 5.2.3 Hessian at the Free Surface

The Hessian criterion cannot be directly evaluated at the free surface. Due to the presence of gravity, there exists a pressure gradient proportional to the density $\rho$, which is discontinuous in the normal direction at the free surface. Therefore, the second derivative is a Dirac $\delta$ function. For numerical solutions, the second derivative in the zone of varying $\alpha$ has a peak which grows as the grid becomes finer. Numerical differentiation produces large errors in this case.

As a result, no correct values can be computed for the pressure Hessian around the surface so an approximative procedure is needed. In the cells where $0.0001 \leq \alpha \leq 0.9999$, the gradient smoothing (step 2 in the SG algorithm) is not performed. The Hessian smoothing (step 5) is replaced by the following algorithm:

5a. Smooth the Hessian in all cells except those where $0.0001 \leq \alpha \leq 0.9999$ plus two layers of cells around those, to take into account that the perturbed pressure gradient in the cells at the surface influences the Hessian in their neighbours as well.

5b. Copy the computed values of the Hessian from outside the zone in (5a) across the zone, following the vertical direction (this removes the peak at the surface).

The criterion is copied in the upward direction, so the Hessian values computed in the water are used across the surface region.

5c. Smooth the Hessian only in the zone of (5a).

The idea of this approach is, that the SG Hessian at the surface must not be used. Therefore, sensible values must be copied from elsewhere. While this procedure is heuristic, it works well in practice as will be shown in Sect. 6.

## 5.3 The Combined Criterion

The final criterion is a combination of the two criteria above. Considering the problems of the Hessian criterion at the surface, it is tempting to select the free-surface criterion there and the Hessian everywhere else. However, the free-surface criterion specifies no refinement in the direction parallel to the surface, while this refinement may be needed if only to ensure that the grid at the surface is not less refined than just below it. The criterion in each cell is thus computed from both criteria.

The criteria are combined with a weighted maximum of the two tensors. We want the threshold $T_r$ to indicate directly the desired cell size at the surface (as for the free-surface criterion), so a weighting factor $c$ is applied only to the Hessian criterion:

$$\mathscr{C}_C = \max\left(\mathscr{C}_S, c\,\mathscr{C}_H\right). \tag{10}$$

Computing the approximate maximum of the two tensors is described in [12], a modification of the procedure presented by [4].

There are (as yet) no universal guidelines for choosing $c$, since appropriate values depend on the type of problem the results of interest. However, for the specification of guidelines for computations which are similar, a non-dimensional $\bar{c}$ is introduced as:

$$c = \bar{c}\left(\frac{L^2}{\frac{1}{2}\rho V_\infty^2}\right)^a. \tag{11}$$

where $L$, $V_\infty$, and $\rho$ are respectively the reference length, velocity, and the density. The power $a$ is the one in Eq. (7). If two computations are geometrically identical except for a scaling in $L$ and $V_\infty$, then choosing the same $\bar{c}$ will result in identical refinement for both cases.

## 6 Test Cases

This section presents two test cases which evaluate the capacity of the refinement method to create effective fine meshes for typical ship flow cases and discuss the choice of the weighting factor $c$. The cases are the Series 60 ship (Sect. 6.1) and the Delft Catamaran in drift condition (Sect. 6.2).

### *6.1 Series 60 Wave Pattern*

The Series 60 hull is studied in straight-ahead motion and calm water. Detailed experiments for this case are available from IIHR [5] at $Fr = 0.316$ and $Re = 5.3 \cdot 10^6$. Apart from this Froude number, we also compute the flow at $Fr = 0.16$ and $Fr = 0.41$ with $Re = 2.7 \cdot 10^6$ and $Re = 6.9 \cdot 10^6$ respectively.

The computations are started from an original grid (242k cells) without any refinement around the free surface. Since directional derefinement is not yet available, the refinement is initialised as in Fig. 3b by allowing only refinement in vertical direction. The computation is then continued with refinement in all directions. The threshold (equivalent to the desired cell size at the surface) is the same for all cases, $T_r = 0.001L$ which is the usual cell size at the surface for ISIS-CFD. For each Froude number, the flow is computed with $\bar{c} = 0.016$, 0.024 and 0.032 which gives the grid sizes in Table 1. Reference results are obtained without grid refinement on a fine mesh of 3.45M cells. Turbulence is modelled with the Menter $k - \omega$ SST model.

The wave pattern at the three Froude numbers is shown in Fig. 4. The wave strength varies strongly with $Fr$; for the two highest $Fr$, the bow wave breaks. Figure 5 shows cross-sections of the mesh for the three Froude numbers at the largest weighting factor $\bar{c} = 0.032$. Around the position of the free surface, the meshes have directional refinement. The cell size below the surface decreases gradually from the bottom up, the finest cells are concentrated in the bow wave (left) and the stern wave (right). The refined cells below the waves are predominantly square, although some cells near the surface are smaller in the vertical direction. The size of the cells in the original grid can be seen in the upper right corner of Fig. 5b, so the entire fine grid is actually created by automatic refinement.

**Table 1** Number of cells in the refined meshes for the Series 60 test cases

| $\bar{c}$ | 0.016 | 0.024 | 0.032 |
|---|---|---|---|
| $Fr = 0.16$ | 698k | 1169k | 1793k |
| $Fr = 0.316$ | 597k | 892k | 1360k |
| $Fr = 0.41$ | 757k | 973k | 1348k |

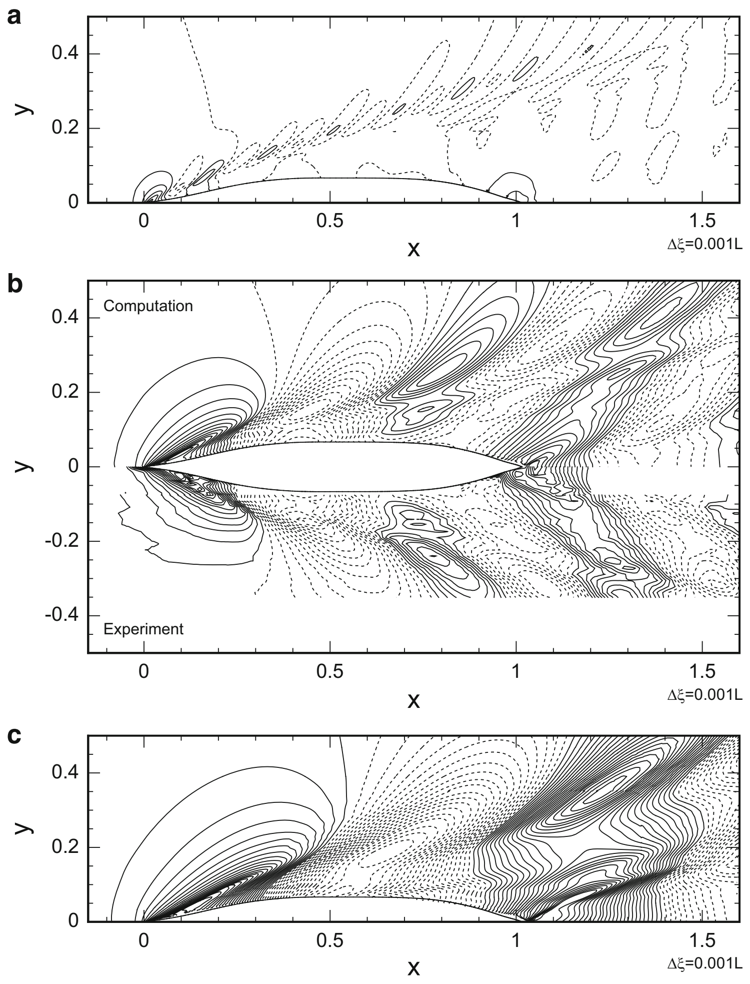


**Fig. 4** Wave patterns for the Series 60 at $Fr = 0.16$ (**a**), $Fr = 0.316$ (**b**), and $Fr = 0.41$ (**c**). The isoline distance is the same for all figures and corresponds to $L/1000$. The $Fr = 0.316$ result is compared with experiments from IIHR [5]

The solutions for all cases are compared in Fig. 6, which shows the free surface in three $X$-cuts when $\bar{c}$ is varied, compared with the non-adapted fine grid. For the highest Froude number (Fig. 6c), all solutions are close to the fine-grid solution, only some discrepancy is seen for the $\bar{c} = 0.024$ solution behind the ship. However, at $Fr = 0.316$ (Fig. 6b) notable differences exist for $\bar{c} = 0.016$. Finally, at $Fr = 0.16$ (Fig. 6a) the computation for $\bar{c} = 0.016$ is the closest to the fine-grid solution, which means that both are questionable! For very small waves, the wave heights may in fact

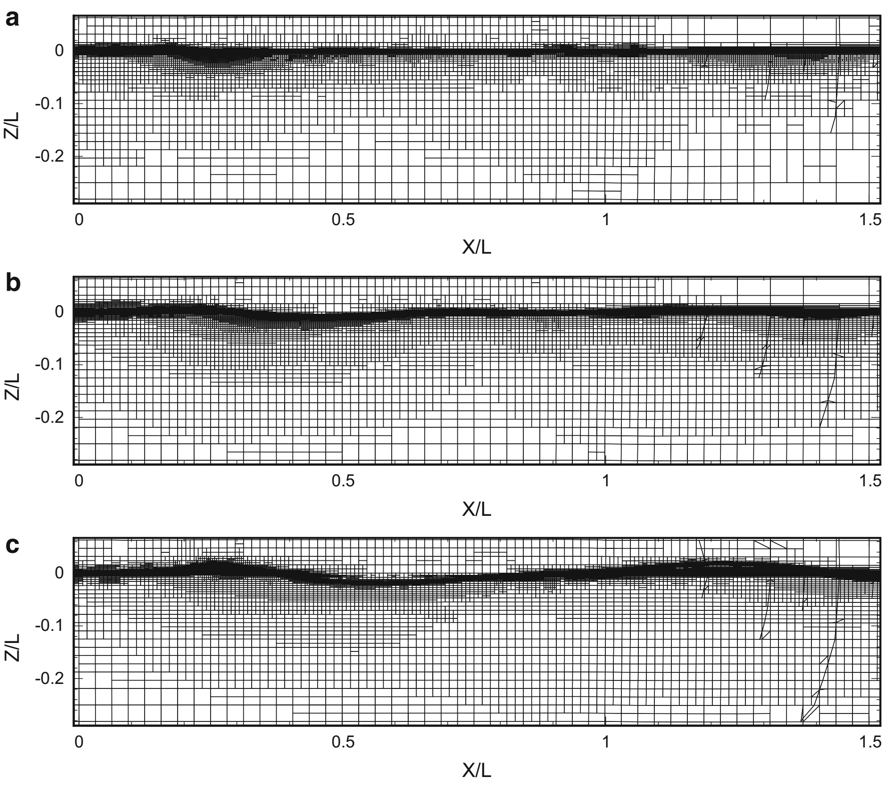


**Fig. 5** Series 60, cross-sections of the grid at $Y/L = 0.118$ for $Fr = 0.16$ (**a**), $Fr = 0.316$ (**b**), and $Fr = 0.41$ (**c**)

be overpredicted on too coarse grids [2]. The solutions for $\bar{c} = 0.024$ and $\bar{c} = 0.032$ are reasonably close. Figure 6d shows velocity profiles on a horizontal line below the water at the stern. Only $Fr = 0.316$ is shown here, since the results for other $Fr$ are similar. Results for all $\bar{c}$ are close to the fine-grid solution, the discrepancy with the experiments may be due to the isotropic $k - \omega$ SST turbulence model which is not always well adapted to the simulation of ship wake flows [1, 3].

In conclusion, to accurately model the wave pattern, higher values for $\bar{c}$ are needed at low $Fr$ than at high $Fr$. However, considering the weak influence of the waves on the rest of the flow field at low $Fr$, a constant choice for $\bar{c}$ is justified, which gives the added advantage that the grid on the hull below the surface is refined in the same way for all $Fr$. For slender hulls like the Series 60, setting $T_r = 0.001L$ with a value of $\bar{c}$ around 0.024 gives sufficient accuracy. Higher $\bar{c}$ increase significantly the total number of cells (Table 1).

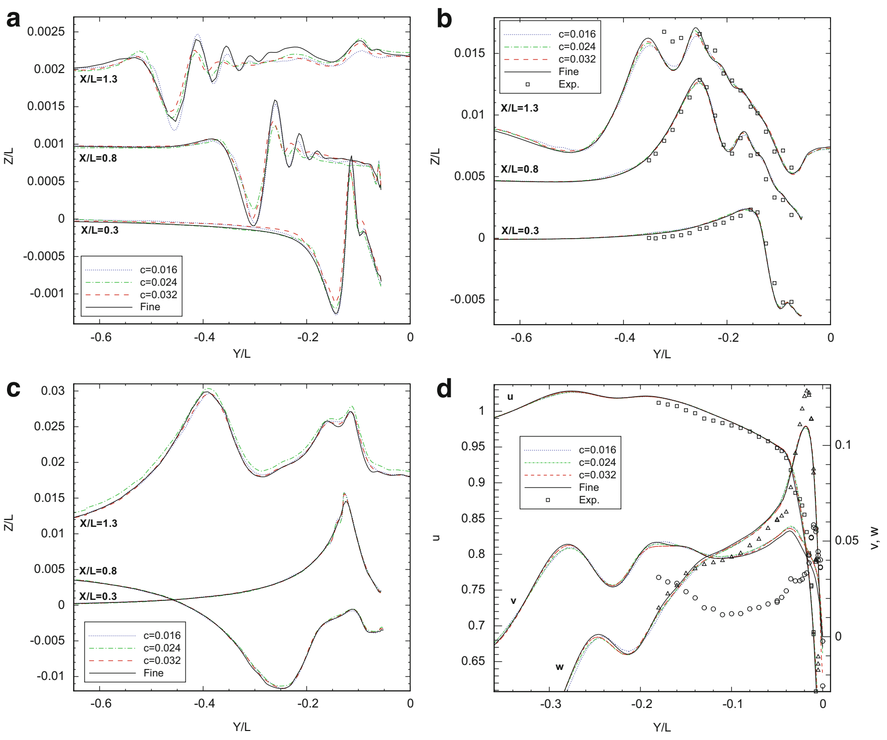


**Fig. 6** Series 60, comparisons of the water surface in three $X$-cuts at $Fr = 0.16$ (**a**), $Fr = 0.316$ (**b**), and $Fr = 0.41$ (**c**). Comparison of the velocities on the line $X/L = 1$, $Z/L = -0.02$ (**d**) for $Fr = 0.316$. Experiments from IIHR [5]

## 6.2 Delft Catamaran in Sideslip

As a second case, the flow around the Delft-327 Catamaran in sideslip is computed. The particularity of this case is that it has both strong waves and longitudinal vortices created below the hull, so the combined refinement criterion is of great interest since it can capture both these features. The case concerns a motion in steady drift ($\beta = 6^{\circ}$) at $Fr = 0.4$. Automatic grid refinement is started from an original grid of 1.0M cells which has some limited refinement around the surface, it is computed with $T_r = L/500$ and with $\bar{c} = 0.064$. This higher value of $\bar{c}$ than for the Series 60 case comes mostly from the higher $T_r$; the combination creates refinement that is concentrated in the trailing vortices. The converged refined grid has 2.96M cells, the results are compared with a fine-grid solution of 20M cells. The anisotropic EASM turbulence model [1] is used.

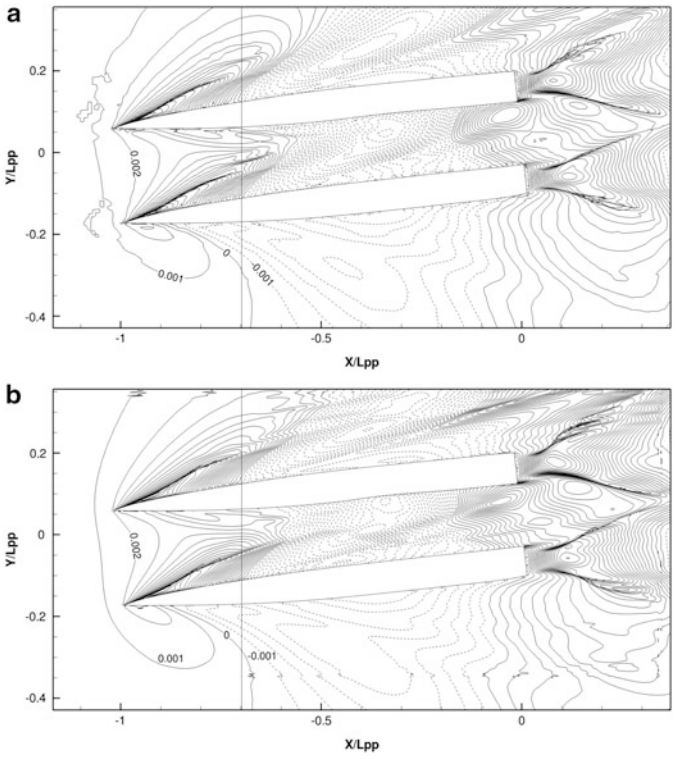


**Fig. 7** Delft Catamaran, free surface on adaptively refined (**a**) and fine (**b**) grid. The *vertical line* shows the position of the cut in Fig. 8

A cut through the mesh is shown in Fig. 8a, with refinement around the deformed surface and in the cores of two vortices below the hulls. Figure 7 compares the free surface for the refined-grid and fine-grid solution. The two are similar, although the far-field waves are damped slightly more on the refined grid due to the high threshold. However, the breaking bow waves have more details on the refined grid, because the cells are locally smaller in $y$-direction than on the fine grid. These small waves (especially between the hulls) are also seen in Fig. 8b, c, which show the axial velocity and the free surface. The vortices are computed on very fine cells in the refined mesh, so they are notably stronger than on the fine mesh.

Thus, also for this case the relevant flow features can be computed well on an adaptively refined mesh. The value for $\bar{c}$, if $T_r = L/1000$ had been used, would have been 0.032. This is still higher than for the Series 60 case, which is justified by the objective to capture the trailing vortices particularly well.

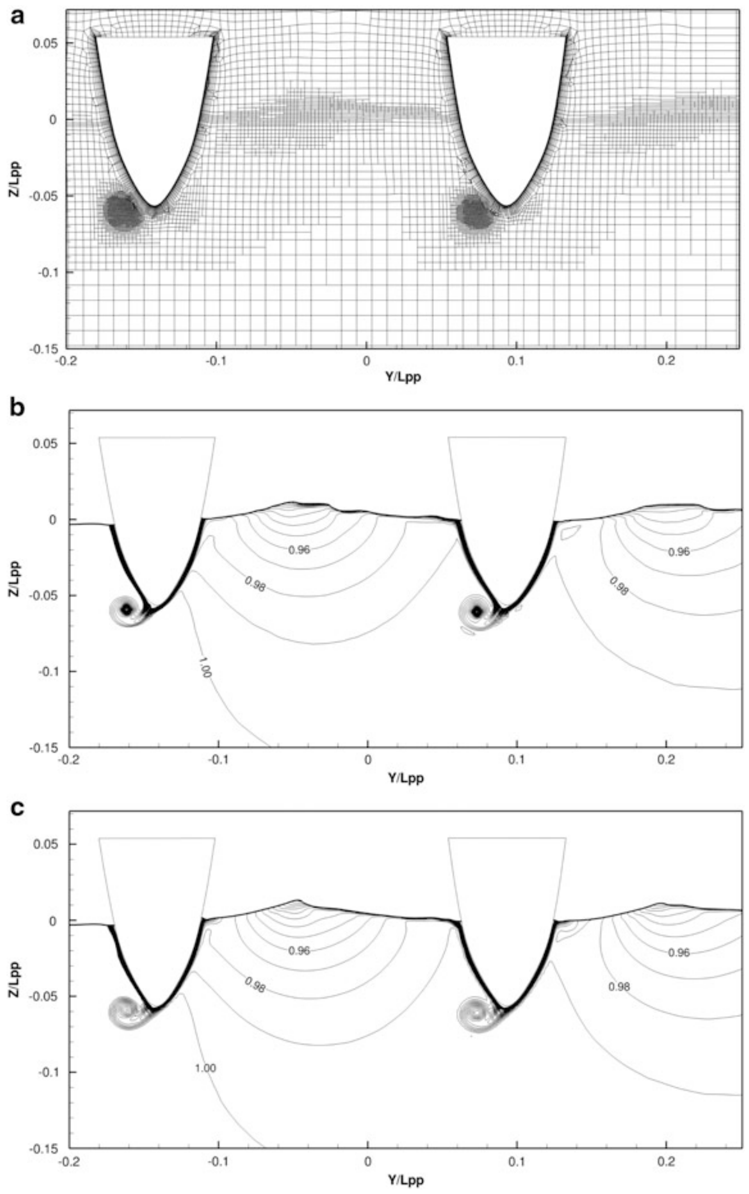


**Fig. 8** Delft Catamaran, cut at $x/L = 0.3$. Refined mesh (**a**), axial velocity on refined (**b**) and fine (**c**) mesh

## 7 Conclusions

The goal of this paper is the simulation of free-surface water flows. For these flows, fine grids are needed around the water surface and the surface position is not known beforehand. Therefore, automatic grid refinement can be very useful, placing fine cells only exactly where they are needed and removing from the user the task of estimating the surface position before the computation in order to generate the mesh.

To create the fine grid needed for flows with waves entirely using grid refinement, the refinement technique must adapt the grid often to the solution as it evolves, since the surface position for example may change position when the grid becomes finer. For refinement by subdivision of cells, this means that both refinement and derefinement are needed even for steady cases, the latter to remove refined cells which are no longer needed as the simulation converges. Refinement and derefinement must be performed in an anisotropic way to limit the total number of cells in three dimensions. The anisotropic derefinement is especially important to allow cells which were refined in several directions and need to remain refined in only one, to get rid of the refinement in the other directions.

Suitable refinement criteria must create refinement both around the surface, to resolve the convection equation for the volume fraction, and in the region below the surface in order to capture the orbital flow fields. To ensure a regular mesh at the surface, a derivative-based criterion suitable for detecting the velocity and pressure fields is combined with a robust free-surface capturing criterion. We choose a criterion which uses directional refinement in the region where the volume fraction is between 0.1 and 0.9 and refinement based on the pressure Hessian. On unstructured hexahedral meshes, the Hessian is found with Gauss derivation twice to compute first the gradient of the pressure and then the second derivatives, followed by smoothing to remove irregularities. The numerical pressure Hessian has a peak at the free surface, due to the discontinuity in the pressure gradient. Therefore, the free-surface region is not smoothed and the computed Hessian outside the surface zone is extrapolated through this zone. To combine the two criteria, their relative weights must be chosen; a non-dimensional scale-independent form for the Hessian criterion is introduced with a weight $\bar{c}$.

Two different ship flow test cases show that the automatic grid refinement is able to create effective fine meshes from original meshes which are neither much refined near the surface, nor around other significant flow features such as vortices. These features were simulated with the same precision as on uniformly fine meshes using 50 % to 85 % less cells. For the weight of the Hessian, the Series 60 case shows that it is acceptable to choose $\bar{c}$ independently of $Fr$. Sensible values for slender ships are $\bar{c} = 0.02$–$0.035$ with a refinement threshold set around $T_r = L/1000$, a high must be chosen if wake features are of major importance. Based on these tests, the perspective of generating complete free-surface meshes with automatic refinement seems entirely realistic.

**Acknowledgements** Computations were performed using HPC resources from GENCI (Grant 2013-21308), which is gratefully acknowledged.

## References

1. Deng, G.B., Visonneau, M.: Comparison of explicit algebraic stress models and second-order turbulence closures for steady flows around ships. In: 7th International Conference on Numerical Ship Hydrodynamics, Nantes, France (1999)
2. Deng, G.B., Guilmineau, E., Leroyer, A., Queutey, P., Visonneau, M., Wackers, J.: Verification and validation for unsteady computation. In: Larsson, L., Stern, F., Visonneau, M. (eds.) A Workshop on Numerical Ship Hydrodynamics, Gothenburg, Sweden (2010)
3. Duvigneau, R., Visonneau, M.: On the role played by turbulence closures in hull shape optimization at model and full scale. J. Mar. Sci. Technol. **8**, 11–25 (2003)
4. George, P.L., Borouchaki, H.: Delaunay Triangulation and Meshing – Application to Finite Elements. Hermes, Paris (1998)
5. Longo, J., Stern, F.: Effects of drift angle on model ship flow. Exp. Fluids **32**, 558–569 (2002)
6. Loseille, A., Dervieux, A., Alauzet, F.: Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations. J. Comput. Phys. **229**, 2866–2897 (2010)
7. Majewski, J.: Anisotropic adaptation for flow simulations in complex geometries. In: 36th CFD/ADIGMA Course on hp-Adaptive and hp-Multigrid Methods. Lecture Series 2010–01. Von Karman Institute, Sint-Genesius-Rode, Belgium (2009)
8. Queutey, P., Visonneau, M.: An interface capturing method for free-surface hydrodynamic flows. Comput. Fluids **36**(9), 1481–1510 (2007)
9. Rhie, C.M., Chow, W.L.: A numerical study of the turbulent flow past an isolated airfoil with trailing edge separation. AIAA J. **17**, 1525–1532 (1983)
10. Wackers, J., Ait Said, K., Deng, G.B., Queutey, P., Visonneau, M., Mizine, I.: Adaptive grid refinement applied to RANS ship flow computation. In: 28th Symposium on Naval Hydrodynamics, Pasadena, California (2010)
11. Wackers, J., Deng, G.B., Visonneau, M.: Combined tensor-based refinement criteria for anisotropic mesh adaptation in ship wave simulation. In: ADMOS 2011, Paris, France (2011)
12. Wackers, J., Leroyer, A., Deng, G.B., Queutey, P., Visonneau, M.: Adaptive grid refinement for hydrodynamic flows. Comput. Fluids **55**, 85–100 (2012)
13. Wackers, J., Deng, G.B., Queutey, P., Visonneau, M., Guilmineau, E., Leroyer, A.: Sliding grids and adaptive grid refinement applied to ship hydrodynamics. In: 29th Symposium on Naval Hydrodynamics, Gothenburg, Sweden (2012)