METRIC-AWARE OPTIMIZATION OF HIGH-ORDER MESHES FOR CURVED ADAPTIVITY

Guillermo Aparicio-Estrems

Doctoral Thesis
Advisor: Xevi Roca Navarro
Co-advisor: Abel Gargallo-Peiró
Barcelona, March 2023

Departament de Matemàtiques
Programa de Doctorat en Matemàtica Aplicada

Universitat Politècnica de Catalunya
Programa de Doctorat en Matemàtica Aplicada

Barcelona Supercomputing Center
Computer Applications in Science and Engineering Department

Metric-aware optimization of high-order meshes for curved adaptivity

by

Guillermo Aparicio-Estrems

PhD dissertation
Advisor: Xevi Roca Navarro
Co-advisor: Abel Gargallo-Peiró

Barcelona, March 2023

Dedicat als meus germans, als meus pares, i a l'Alícia.

Abstract

Metric-aware optimization of high-order meshes for curved adaptivity

Guillermo Aparicio-Estrems

To enhance the simulation accuracy when the solution presents sharp curved features, the community of high-order methods has started to curve not only the boundary but also the interior of unstructured high-order meshes. Many of these approaches contribute to curved high-order adaptivity based on error estimators. The error estimators determine a discrete metric that is used to modify the curved high-order mesh. Unfortunately, although all these approaches must modify the mesh coordinates, no approach considers metric-aware optimization of curved high-order meshes for a high-polynomial degree. Furthermore, the existing approaches neither explicitly enforce unitary Riemannian measures for all mesh entities nor specifically devise a specific-purpose solver for curved sharp features.

To address these issues, this thesis aims to demonstrate metric-aware optimization of high-order meshes on curved geometry with the coordinates as design variables. To this end, it proposes the following contributions. First, to verify and optimize the stretching and alignment deviation between the mesh and an analytic metric, we define a differentiable shape distortion measure for curved high-order meshes. Second, to enforce unitary Riemannian measures of the mesh entities, we define a differentiable size-shape distortion measure for curved high-order meshes. Third, to efficiently minimize with tight tolerances a point-wise metric-aware distortion measure for curved high-order meshes, we devise a specific-purpose solver. Fourth, to apply Newton's method for the distortion minimization of meshes equipped with a discrete target metric, we derive up to second-order the derivatives for a high-order metric interpolation. Finally, to also match curved boundaries, we derive up to second-order the derivatives for an implicit CAD representation.

In conclusion, this thesis demonstrates metric-aware optimization of high-order meshes on curved geometry. To this end, it proposes a novel metric- and geometry-aware mesh optimization framework and a specific-purpose optimization solver. These novelties will contribute to error-driven curved high-order adaptivity. Hence, they will help to enhance the simulation accuracy for solutions presenting sharp curved features.

Acknowledgments

During the thesis, many people have contributed to achieving the objectives and promoting a healthy environment. It is for this reason, that I consider it necessary to thank everyone who has made this thesis possible.

I want to thank Xevi Roca for his professional and personal guidance. Thanks for sharing his ideas and for his comprehension when dealing with several proposals. For the fixed objectives, his flexibility allowed me to freely develop this thesis, while his ideas instructed me on how to be a professional researcher. In addition, I would like to thank Abel Gargallo Peiró, my second advisor. His advice has promoted my concerns, an important ingredient to drive novel objectives for the thesis.

Thank the members of the Tesseract group for promoting a pleasant and cheerful atmosphere. In particular, I would like to thank Dr. Guillem Belda for his business-worker knowledge, which allowed me to have new perspectives on life and the Ph.D. In addition, I thank Albert Jiménez for many conversations sharing ideas and perspectives, either mathematical or personal. Keep the path Albert, "now it's your bureaucracy turn". Finally, I am also grateful to Eloi Ruiz for sharing his knowledge about the computational technicalities of video games, which are surprisingly related to the project objectives.

I appreciate the institutions involved in the development of the thesis. I thank the Computer Applications in Science and Engineering (CASE) group at Barcelona Supercomputing Center (BSC) for the possibility of doing and sharing my research. As well, I would thank to Facultat de Matemàtiques i Estadística (FME) for the Ph.D. program and the courses assisted during the thesis.

I appreciate the effort of the tribunal members and the referees. I thank them for deciding to spend their time on this thesis, and Pep Sarrate for being part of this thesis committee in its beginnings.

Agraeixo molt la companyia i les conversacions amb amics de l'àmbit científic: Jordi Font, Andy Ravenna, i Mireia Masias. El fet de compartir les nostres inquietuds i dificultats en l'àmbit de la recerca m'han permès tenir una visió més enllà de la meva bombolla.

Ha estat imprescindible el suport d'un company, en Xavier Veloy. La seva comprensió ha permès aconsellar-me en decisions importants durant la tesi. També, m'ha compartit una visió independent però acurada de la recerca, l'educació, i la comunicació. Encara més, és la seva influència, des dels aspectes més filosòfics fins als més

quotidians, la que m'ha permès ser qui soc i qui esdevindrà.

Finalment, donar les gràcies a la meva família. Els moments compartits amb els meus germans, el Jaumet i l'Alejandro, han donat forma a la meva manera de veure les coses. Agraeixo molt la col·laboració de l'Alejandro, la Luisa, i la meva mare a casa. Sobretot durant la pandèmia de la COVID, són ells amb qui he compartit uns dies agradables. També agraeixo als meus pares, per estar al meu costat en les inquietuds sobre la meva formació, per donar-me la llibertat d'escollir, per haver lluitat per obtenir els mitjans per fer-ho realitat. Finalment, agraeixo a l'Alícia, qui s'ha mantingut al meu costat, per les experiències compartides i per les que vindran.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation and background

To enhance the simulation accuracy in problems where the solution presents sharp curved features, the community of unstructured high-order methods has started to curve not only the boundary but also the interior (Knupp et al., 2021; Barrera et al., 2023; Dobrev et al., 2019, 2021; Sanjaya and Fidkowski, 2016; Rochery and Loseille, 2021; Zhang, 2022; Coupez, 2017; Marcon, 2019; Zahr et al., 2020; Ekelschot et al., 2019; Feuillet et al., 2019; Feuillet, 2019) of unstructured high-order meshes. These methods aim to match the sharp curved features of the solution by exploiting the non-constant Jacobian of curved high-order elements (Fidkowski and Darmofal, 2011). To this end, they modify the high-order mesh topology and coordinates (Dobrev et al., 2021; Rochery and Loseille, 2021; Zhang, 2022; Ekelschot et al., 2019; Feuillet et al., 2019; Feuillet, 2019) or only the coordinates (Knupp et al., 2021; Barrera et al., 2023; Dobrev et al., 2019; Sanjaya and Fidkowski, 2016; Coupez, 2017; Marcon et al., 2017; Zahr et al., 2020). In both families, the modification of the mesh coordinates is a crucial ingredient.

Many of these approaches are contributing to enable error-driven curved high-order adaptivity. Specifically, to exploit existent high-order goal-oriented (Yano and Darmofal, 2012; Fidkowski and Darmofal, 2011) and interpolation-oriented (Loseille and Alauzet, 2011; Coulaud and Loseille, 2016a) error estimators, some curved high-order mesh modification approaches consider an objective function that accounts for the discrete metric obtained from an error estimator (Rochery and Loseille, 2021;

Feuillet et al., 2019; Feuillet, 2019; Zhang et al., 2018; Ekelschot et al., 2019; Sanjaya and Fidkowski, 2016). These metric-aware modifications are performed in two different manners. First, optimizing element-by-element for high-polynomial degree (Sanjaya and Fidkowski, 2016; Sanjaya, 2019). Second, through local cavity modification of curved quadratic meshes (Rochery and Loseille, 2021; Feuillet et al., 2019; Feuillet, 2019; Zhang et al., 2018; Zhang, 2022; Coupez, 2017). Alternatively, instead of a metric, it is possible to match a point-wise target deformation matrix (Dobrev et al., 2019, 2018, 2020; Camier et al., 2023). Unfortunately, no approaches consider metric-aware optimization of curved high-order meshes for high-polynomial degree.

Developing a metric-aware optimization of curved high-order meshes for high-polynomial degree is the main challenge of this thesis. This development is relevant because it will enable error-driven curved r-adaptivity and metric-aware smoothing for local cavity operators.

1.1.1 Unitary Riemannian measures

In the metric-aware approaches (Rochery and Loseille, 2021; Feuillet et al., 2019; Feuillet, 2019; Zhang et al., 2018; Ekelschot et al., 2019; Sanjaya and Fidkowski, 2016), the target metric encodes the curved geometric features of the solution, features such as the point-wise stretching, alignment, and sizing. Using this encoding, these methods enforce either curved edges with unitary Riemannian lengths (Rochery and Loseille, 2021; Zhang, 2022) or average independent curved elements featuring the stretching, alignment, and sizing of the target metric element-by-element (Sanjaya and Fidkowski, 2016; Ekelschot et al., 2019). However, no approaches explicitly enforce unitary Riemannian measures for the mesh edges as well as for the face areas and the cell volumes.

Enforcing unitary Riemannian measures for all mesh entities is the first research problem of this thesis. It might be critical when the metric varies point-wise. Without this feature, the resulting mesh might not reduce the error as expected. We should expect this issue because the differential measure at each point of the curved high-order mesh is not explicitly enforced to match the stretching, alignment, and sizing of the prescribed point-wise metric.

1.1.2 Optimization

Regarding the optimization of the objective function, we can iteratively modify the coordinates of either all the free nodes (all nodes) or one free node (one node) per non-linear iteration by using either gradient-based (first-order) or Hessian-based (second-order) optimization methods. For linear elements, there are several studies on the performance of local (Diachin et al., 2006, 2004; Sastry and Shontz, 2009; Diachin et al., 2004) first and second-order optimization methods. One common conclusion is that when highly optimized and accurate meshes are required, especially in isotropic meshes featuring high gradations of the element size, a specific-purpose all-nodes globalized Newton method (Steihaug, 1983) outperforms local optimization methods (Diachin et al., 2006, 2004; Sastry and Shontz, 2012). Unfortunately, no approach has devised or studied a specific-purpose all-nodes globalized Newton method for metric-aware optimization of curved high-order meshes.

Devising a globalized and preconditioned Newton solver for metrics with curved sharp features is the second research problem of this thesis. For curved high-order mesh optimization, we know that standard globalized and preconditioned Newton-Krylov solvers have robustness and efficiency issues (Ruiz-Gironés and Roca, 2022). These issues are especially triggered by non-uniform sizing, stretching ratios, and curved alignment. When more remarkable these characteristics are, more difficult the convergence with a general-purpose optimization solver (Ruiz-Gironés and Roca, 2022). First, in each non-linear step, highly non-uniform mesh gradation stiffens the validity of the mesh deformations and the corresponding linear systems. Second, for high stretching ratios, the deformations in some directions are locally stiffer than in other directions. Third, curved alignment requires curved high-order elements. For these elements, when higher is the order, stiffer is the corresponding linear system. We should expect similar issues for metric-aware curved high-order optimization.

1.2 Research opportunity and questions

The previous overview *identifies a research opportunity to enable curved r-adaptivity driven by an error estimator.* Although there are methods for curved r-adaption of high-order meshes, *there is no known method that simultaneously matches a target metric and geometry using second-order optimization.* The overview also identifies the following key research questions:

(Q1) *How to formulate a metric-aware optimization problem that enforces unitary Riemannian measures of the mesh edges, faces, and cells?*

(Q2) *How to solve a metric-aware optimization problem for target metrics featuring non-uniform sizing, high stretching ratios, and curved alignments?*

The combination of the answers to questions (Q1) and (Q2) enables curved r-adaptivity driven by the metric obtained from an error estimator. The answer to question (Q1) provides the optimization formulation. This formulation can be solved with the answer of question (Q2).

1.3 Aim and objectives

To enable curved r-adaption driven by an error estimator metric, this thesis aims to demonstrate metric-aware optimization of high-order meshes on curved geometry with the mesh coordinates as design variables. To this end, this thesis develops the following *objectives*:

(O1) *To evaluate in an optimizable manner the shape and orientation matching between a curved high-order mesh and a target metric*, Aparicio-Estrems et al. (2018).

(O2) *To evaluate in an optimizable manner the shape, orientation, and size matching between a curved high-order mesh and a target metric*, Chapter 2.

(O3) *To optimize a curved high-order mesh to tightly match a non-uniform anisotropic target metric*, Chapter 3.

(O4) *To account and optimize for a discrete metric*, Aparicio-Estrems et al. (2022).

(O5) *To account and optimize for a discrete metric and a curved geometry*, Chapter 4.

The aim of this thesis addresses the research opportunity. Moreover, the combination of the objectives addresses the research questions. For the research question (Q1), objective (O1) accounts for the metric shape and orientation, objective (O2) accounts not only for the metric shape and orientation but also the size, objective

(O4) accounts for a discrete metric, and objective (O5) accounts not only for a discrete metric but also a curved boundary. For the research question (Q2), objective (O3) addresses the solution of the optimization problem, objective (O4) optimize for a discrete metric, and (O5) optimize for a discrete metric and the curved geometry.

1.4 Methodology

To gradually meet the aim of this thesis, the research methodology approaches the different requirements. In (O1) it proposes a distortion formulation to match stretching and alignment of analytic metrics. In (O2), exploiting the previous formulation, it proposes a distortion formulation to match the stretching, alignment, and sizing of target metrics. In (O3), for a distortion formulation, it proposes a specific-purpose minimization solver tested for analytic metrics. In (O4), to enable practical metrics, it proposes a high-order interpolation approach compatible with second-order optimization. Also in (O5), to enable curved boundaries, it combines the previous approaches with an implicitization of the boundary representation of the geometry.

The methodology approaches are based on mathematical formulations and derivations, design of computational methods, heuristics, computer implementations, runtime checks, and verification approaches. First, the mathematical formulation and derivations allow stating the base problem formulation. Second, the proposed computational methods are the base of the computer implementations. Third, heuristics are used to devise globalization, stopping and switching tolerances, and the preconditioner. Fourth, the computer implementations of the proposed methods allow showing empirical evidence of the advantages for curved r-adaption. Fifth, the computer implementation checks at runtime that the determinants of the Jacobians are positive during the whole process. Finally, to verify the results we compare with the metric unit to measure the lengths, areas, and volumes of the resulting curved elements.

1.5 Contributions and novelty

The main contribution is to demonstrate the optimization of curved high-order meshes that match a target metric and a curved boundary. The optimization enforces unitary Riemannian measures of the mesh edges, lengths, and cells (Q1). Moreover, it deals

with target metrics featuring non-uniform sizing, high stretching ratios, and curved alignments (Q2). To this end, addressing the previously stated objectives, *this thesis contributes with novel methods*:

(C1) *Defining a differentiable shape distortion measure for curved high-order meshes accounting for the alignment and stretching of the target metric.* This contribution addresses the research question (Q1) only for stretching and alignment. The main novelty is to account for a target metric because alternative shape distortion measures use a target deformation matrix. *This contribution corresponds to the peer-reviewed conference paper Aparicio-Estrems et al. (2018).*

(C2) *Defining a differentiable size-shape distortion measure for curved high-order meshes accounting not only for the alignment and stretching but also the sizing of the target metric*, Chapter 2. This contribution addresses the research question (Q1). Regarding size-shape distortions, there are two main novelties. First, the definition uses a new differentiable surrogate of the standard non-differentiable sizing measure. Second, the definition does not use a deformation matrix but a metric matrix. *This contribution corresponds to the journal paper in preparation Aparicio-Estrems et al. (2023b).*

(C3) *Proposing a solver to efficiently minimize with tight tolerances a point-wise metric-aware distortion measure for curved high-order meshes with the coordinates as design variables*, Chapter 3. This contribution addresses the efficiency aspect of the research question (Q2). The main novelty is to propose a second-order mesh optimization solver specific for high degrees and non-uniform anisotropic metrics. Alternative solvers use either a surrogate for the second-order derivatives of a matrix deformation or first-order optimization only for the mesh edges. *This contribution corresponds to the journal paper in preparation Aparicio-Estrems et al. (2023c).*

(C4) *Computing up to second-order the derivatives of an objective function accounting for a high-order metric interpolation.* This contribution addresses aspects of the research questions (Q1) and (Q2). For question (Q1), it provides the formulation that accounts for discrete metrics. For question (Q2), it addresses the computation up to second derivatives of the formulation. For an existent log-Euclidean metric interpolation, the main novelty is to compute the derivatives on a pseudo-inverse decomposition. Alternative approaches only compute first

derivatives for curved quadratic mesh edges. *It corresponds to the peer-reviewed conference paper Aparicio-Estrems et al. (2022).*

(C5) *Computing up to second-order the derivatives of an objective function accounting not only for a high-order metric interpolation but also for a curved boundary representation*, Chapter 4. This contribution addresses aspects of the research questions (Q1) and (Q2). For question (Q1), it provides not only the formulation that accounts for discrete metrics but also CAD curved geometries. For question (Q2), it addresses the computation up to second derivatives of the formulation. The main novelty is to account not only for a high-order metric interpolation but also for the implicitation of a CAD curved boundary representation. Alternative approaches neither compute up to second-order derivatives nor use a CAD implicitation. *This contribution corresponds to the journal paper Aparicio-Estrems et al. (2023a).*

There are two key central findings in this thesis. First, to enforce unitary Riemannian lengths for all the mesh entities on point-wise varying metrics, *it is key to define a point-wise metric-aware distortion measure accounting for the shape, orientation, and size,* Chapter 2. Using an entity-wise metric-aware measure we could only enforce unitary Riemannian measures for that type of entity. Second, to solve problems with non-uniform anisotropic point-wise metrics featuring curved sharp features, *it is key to define a specific-purpose non-linear solver,* Chapter 3. Without this solver we could only demonstrate metric-aware optimization of curved high-order meshes for simpler metrics.

1.6 Layout

In Chapter 2, we define a regularized size-shape distortion (quality) measure for curved high-order elements on a Riemannian space. To this end, we measure the deviation of a given element, straight-sided or curved, from the stretching, alignment, and sizing determined by a target metric. The defined distortion (quality) is suitable to check the validity and the quality of straight-sided and curved elements on Riemannian spaces determined by constant and point-wise varying metrics. The examples illustrate that the distortion can be minimized to curve (deform) the elements of a given high-order (linear) mesh and try to match with curved (linear) elements

the point-wise stretching, alignment, and sizing of a discrete target metric tensor. In addition, the resulting meshes simultaneously match the curved features of the target metric and boundary. Finally, to verify if the minimization of the metric-aware size-shape distortion leads to meshes approximating the target metric, we compute the Riemannian measures for the element edges, faces, and cells. The results show that, when compared to anisotropic straight-sided meshes, the measures of the curved high-order mesh entities are closer to unit Riemannian measures.

In Chapter 3, we present a specific-purpose globalized and preconditioned Newton-CG solver to minimize a metric-aware curved high-order mesh distortion. The solver is specially devised to optimize curved high-order meshes for high polynomial degrees with a target metric featuring non-uniform sizing, high stretching ratios, and curved alignment. To this end, we consider two ingredients: a specific-purpose globalization and a specific-purpose Jacobi-ILDL preconditioning with dynamic forcing terms. First, to enhance the global convergence of the non-linear solver, the globalization strategy modifies Newton's direction to a feasible step. In particular, our specific-purpose strategy enables a step-length continuation evolution during the optimization process while ensuring sufficient decrease and progress. Second, to compute Newton's direction in second-order optimization problems, we consider a conjugate-gradient iterative solver with specific-purpose preconditioning and dynamic forcing terms. To account for the metric stretching and alignment, the preconditioner uses specific orderings for the mesh nodes and the degrees of freedom. We also present a preconditioner switch between Jacobi and ILDL preconditioners to control the numerical ill-conditioning of the preconditioner. In addition, the dynamic forcing terms determine the required accuracy for the Newton direction approximation. Specifically, they control the residual tolerance and enforce sufficient positive curvature for the conjugate-gradients method. Finally, to analyze the performance of our method, the results compare the specific-purpose solver with standard optimization methods. For this, we measure the matrix-vector products indicating the solver computational cost and the line-search iterations indicating the total amount of objective function evaluations. When we combine the globalization and the linear solver ingredients, we conclude that the specific-purpose Newton-CG solver reduces the total number of matrix-vector products by one order of magnitude. Moreover, it also reduces the number of non-linear and line-search iterations.

In Chapter 4, we detail how to use Newton's method for distortion-based curved

r-adaption to a discrete high-order metric field while matching a target geometry. Specifically, we combine two terms: a distortion measuring the deviation from the target metric; and a penalty term measuring the deviation from the target boundary. For this combination, we consider four ingredients. First, to represent the metric field, we detail a log-Euclidean high-order metric interpolation on a curved (straight-edged) mesh. Second, for this metric interpolation, we detail the first and second derivatives in physical coordinates. Third, to represent the domain boundaries, we propose an implicit representation for 2D and 3D NURBS models. Fourth, for this implicit representation, we obtain the first and second derivatives. The derivatives of the metric interpolation and the implicit representation allow minimizing the objective function with Newton's method. For this second-order minimization, the resulting meshes simultaneously match the curved features of the target metric and boundary. Matching the metric and the geometry using second-order optimization is an unprecedented capability in curved (straight-edged) r-adaption. This capability will be critical in global and cavity-based curved (straight-edged) high-order mesh adaption.

Chapter 2

Defining a metric-aware size-shape distortion measure

2.1 Introduction

Recently, there has been an increased interest to modify the coordinates and topology of a high-order mesh to match curved anisotropic solution features with high-order meshes. This interest has been awakened because these modified curved high-order meshes promise to reduce the error of the approximation to solution for the same number of degrees of freedom, especially when the solution has curved anisotropic features. To this end, existing interior mesh curving approaches exploit the non-constant Jacobian of high-order meshes to match the target curved anisotropic features of the solution using coordinate modifications (Dobrev et al., 2019; Sanjaya and Fidkowski, 2016; Coupez, 2017; Marcon et al., 2017; Zahr et al., 2020) and local cavity modifications (Dobrev et al., 2021; Rochery and Loseille, 2021; Zhang, 2022; Ekelschot et al., 2019; Feuillet, 2019).

To exploit existent high-order goal-oriented (Yano and Darmofal, 2012; Fidkowski and Darmofal, 2011) and interpolation-oriented (Loseille and Alauzet, 2011; Coulaud and Loseille, 2016a) error estimators, curved high-order mesh optimization approaches (Rochery and Loseille, 2021; Zhang et al., 2018; Ekelschot et al., 2019; Sanjaya and Fidkowski, 2016; Aparicio-Estrems et al., 2018, 2022, 2023a) consider an objective function that accounts for the discrete metric obtained from the error estimator. These approaches enforce either curved edges with unitary lengths (Rochery and Lo-

seille, 2021; Zhang, 2022) or curved elements featuring the stretching and alignment of the target metric at a reference (Sanjaya and Fidkowski, 2016; Ekelschot et al., 2019) or a physical (Aparicio-Estrems et al., 2018, 2022, 2023a) mesh. Alternatively, instead of a metric, it is possible to match a point-wise target deformation matrix (Dobrev et al., 2019). Unfortunately, no approaches enforce unitary Riemannian measures for the mesh edges as well as for the face areas and the cell volumes.

Enforcing unitary Riemannian metrics for all mesh entities is critical when the metric varies point-wise. Without this feature, the resulting mesh might not reduce the error as expected. This issue is so because the differential measure at each point of the curved high-order mesh might not match the stretching, alignment, and sizing of the prescribed point-wise metric.

2.1.1 Aim and contribution

Accordingly, we aim to enforce unitary Riemannian measures for all the mesh entities. To this end, the main contribution of this chapter is to define a differentiable point-wise size-shape distortion measure that accounts for the stretching, alignment, and sizing of the target metric. Moreover, to check if the Riemannian measures are unitary, we detail how to compute metric-aware measures of the mesh entities, *i.e.*, Riemannian lengths, areas, and volumes. Finally, we verify whether minimizing the metric-aware size-shape distortion leads to meshes with Riemannian measures closer to unity for the element edges, faces, and cells.

To define the differentiable metric-aware size-shape distortion, the main novelty is to propose a differentiable multiplicative combination of an existent metric-aware shape distortion (Aparicio-Estrems et al., 2018) and a new differentiable metric-aware size distortion. Regarding size-shape distortion measures, there are related works for linear and curved-high-order meshes yet targeting a deformation matrix. For linear meshes, to obtain a distortion measure that accounts for shape and size, it is standard to multiply a shape and a non-differentiable size distortion (Knupp, 2001). The size distortion considers dilation volumes. For curved high-order meshes targeting a deformation matrix, existing differentiable distortion measures account for stretching, alignment, and sizing (Dobrev et al., 2019). It is also possible to use a weighted sum of a shape and a reciprocal of a size quality surrogate that depends on a parameter. The quality surrogate considers a normalized difference of volume dilation and its reciprocal. The main difference between these approaches and our approach is that

we use a target metric to exploit existent error estimators. Another difference is that our differentiable size distortion considers squares of the d roots of a normalized summation of the volume dilation and its reciprocal.

The rest of the chapter is organized as follows. First, in Section 2.2, we introduce the shape measures for high-order Euclidean elements. Next, in Section 2.3, we present the new size-shape measures for linear elements equipped with constant metrics. Then, in Section 2.4, we extend the size-shape measures to curved high-order elements equipped with point-wise varying metrics. Following, in Section 2.5, we present several examples to illustrate the capabilities of the proposed measure. To finalize, in Section 2.6, we present the main conclusions.

2.2 Preliminaries: shape measures for high-order Euclidean elements

In this section, we present the Jacobian-based shape quality measures for linear and high-order elements defined in the Euclidean space (Knupp, 2001; Roca et al., 2012; Gargallo-Peiró et al., 2015a). In addition, we introduce the required notation for Riemannian elements that is, elements equipped with a metric.

To define and compute a Jacobian-based measure for linear Euclidean elements in $\mathbb{R}^d$, three elements are required (Knupp, 2001): the master, the ideal, and the physical, see Figure 2.1 for 2D simplices. The master (E^M) is the element from which the iso-parametric mapping is defined. The ideal element $\left(E^I\right)$ represents the target configuration which, in the Euclidean case, is an equilateral element ($E^\triangle$). The physical (E^P) is the element to be measured.

First, we obtain the mappings between the ideal and the physical elements through the master element. By means of these mappings, we determine a mapping between the ideal and physical elements by the composition

$$\boldsymbol{\phi}_E : E^\triangle \xrightarrow{\boldsymbol{\phi}_\triangle^{-1}} E^M \xrightarrow{\boldsymbol{\phi}_P} E^P.$$

The Jacobian of the affine mapping $\boldsymbol{\phi}_E$, denoted by $\mathbf{D}\boldsymbol{\phi}_E$, encodes the deviation of the physical element with respect to the equilateral one.

We define the shape distortion measure η_{shape} of the physical element as (Knupp, 2001)

$$\eta_{\text{shape}}(\mathbf{D}\boldsymbol{\phi}_E) := \frac{1}{d}\frac{\mathbf{S}^2}{\sigma^{2/d}}, \tag{2.1}$$

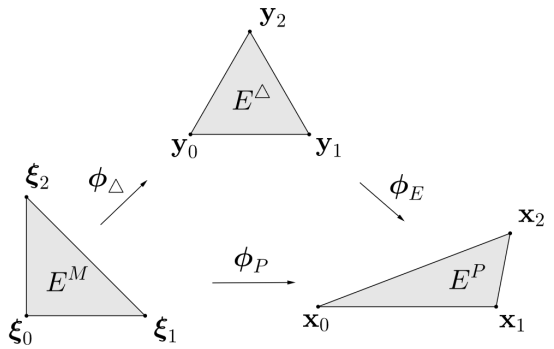


Figure 2.1: Mappings between the master, the ideal, and the physical elements in the linear case.

where $\mathbf{S}$ and σ are the Frobenius norm and the determinant of $\mathbf{D}\boldsymbol{\phi}_E$, respectively. This distortion measure quantifies the shape deviation between the physical and ideal elements.

The matrix $\mathbf{D}\boldsymbol{\phi}_E$ is computed for linear triangles as

$$\mathbf{D}\boldsymbol{\phi}_E = \mathbf{D}\boldsymbol{\phi}_P \, \mathbf{D}\boldsymbol{\phi}_{\triangle}^{-1} = \begin{pmatrix} x_1 - x_0 & \frac{2x_2 - x_1 - x_0}{\sqrt{3}} \\ y_1 - y_0 & \frac{2y_2 - y_1 - y_0}{\sqrt{3}} \end{pmatrix},$$

where

$$\mathbf{D}\boldsymbol{\phi}_P = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix}, \quad \text{and} \quad \mathbf{D}\boldsymbol{\phi}_{\triangle} = \begin{pmatrix} 1 & \frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} \end{pmatrix}, \tag{2.2}$$

being $\mathbf{x}_i = (x_i, y_i)$ the coordinates of the physical element E^P. These matrices are written for the master element E^M with node coordinates $\{\boldsymbol{\xi}_0 = (0,0)\,,\ \boldsymbol{\xi}_1 = (1,0)\,,\ \boldsymbol{\xi}_2 = (0,1)\}$, and the ideal element E^I determined by the nodes $\{\mathbf{y}_0 = (0,0)\,,\ \mathbf{y}_1 = (1,0)\,,\ \mathbf{y}_2 = \big(1/2, \sqrt{3}/2\big)\big\}$.

The shape distortion measure, Equation (2.1), quantifies the shape deviation between the physical and ideal shapes. The measure gets value 1 when the physical element is a scaled equilateral element. It is important to note that it is invariant under translations, rotations, and symmetries. Moreover, it can be regularized to detect inverted elements. From the distortion measure, we define the shape quality measure of an element as

$$q_{\text{shape}} := \frac{1}{\eta_{\text{shape}}}, \tag{2.3}$$

which takes values in the interval $[0, 1]$, being 0 for degenerated elements and 1 for the ideal element and its symmetric analogs.

For high-order (Gargallo-Peiró et al., 2015c,a,b) and multi-linear (Gargallo-Peiró et al., 2015) elements E^P with non-constant Jacobian, we reinterpret a distortion measure η as a point-wise measure $\mathcal{N}\boldsymbol{\phi}_E$. In particular, we define

$$\mathcal{N}\boldsymbol{\phi}_E(\mathbf{y}) := \eta(\mathbf{D}\boldsymbol{\phi}_E(\mathbf{y})), \quad \forall \mathbf{y} \in E^{\triangle}.$$

Furthermore, we define the elemental distortion (Roca et al., 2012; Gargallo-Peiró et al., 2015a) as

$$\eta_{E^P} := \frac{\int_{E^{\triangle}} \mathcal{N}\boldsymbol{\phi}_E(\mathbf{y}) \, d\mathbf{y}}{\int_{E^{\triangle}} 1 \, d\mathbf{y}}, \tag{2.4}$$

and its quality q_{E^P} follows from Equation (2.3).

2.3 Size and size-shape measures for linear elements and constant metric

Herein, we present the measures for linear elements equipped with constant metrics. First, in Section 2.3.1, we define a quality measure that quantifies the size deviation of Euclidean elements. Second, in Section 2.3.2, we extend the quality measure to linear simplices equipped with a constant metric. Finally, in Section 2.3.3, we illustrate the behavior of the proposed measure.

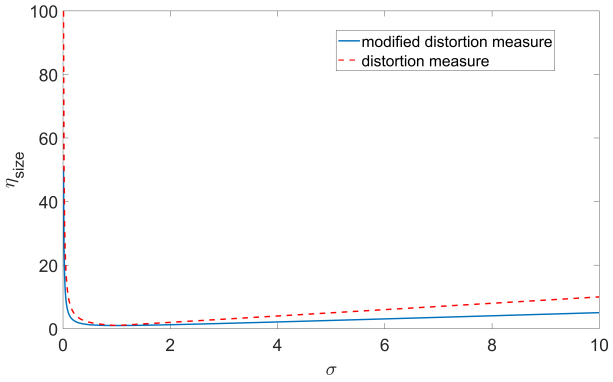
2.3.1 Differentiable size and size-shape distortion for linear Euclidean elements

The shape distortion measure of Section 2.2 quantifies the shape deviation between the physical and ideal elements. However, it does not take into account the size deviation between the physical and ideal elements. For this reason, we define an additional distortion measure that takes into account sizing. In particular, we define the size distortion measure η_{size} of the physical element as

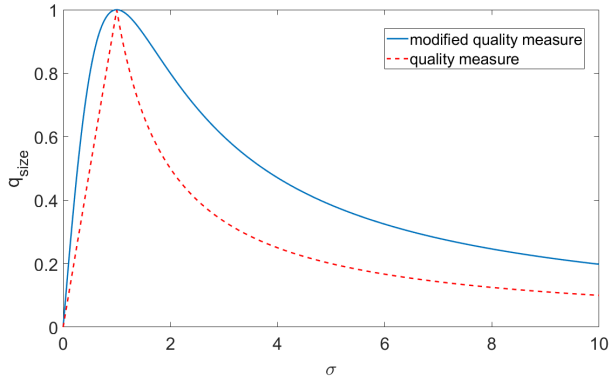
$$\eta_{\text{size}}(\mathbf{D}\boldsymbol{\phi}_E) = \left(\frac{1}{2}\left(\sigma + \frac{1}{\sigma}\right)\right)^{2/d}, \tag{2.5}$$

where $\sigma := \det(\mathbf{D}\boldsymbol{\phi}_E)$. This distortion measure quantifies the size deviation between the physical and ideal elements. We expect the size distortion measure to behave as

$$\mu(\sigma) = \max(\sigma, \sigma^{-1})^{2/d}.$$



(a)



(b)

Figure 2.2: Plots of the original and modified: (a) size distortion measure and (b) size quality measure.

Note that, the base $\max(\sigma, \sigma^{-1})$ is the standard size measure of Knupp (2001).

However, we cannot use the function μ in a continuous optimization procedure since it is not differentiable. To overcome this drawback, we propose to replace $\mu(\sigma)$ by the size distortion measure, see Equation (2.5), a continuous and differentiable function that holds the same minimum and the same asymptotic behavior.

Figure 2.2 shows the size distortion and the size quality measures using the original, $\mu(\sigma)$, and the modified function, η_{size}, in terms of σ. It is worth to notice that using the modification presented in Equation (2.5), the size distortion measure η_{size} is still a distortion measure that is, orientation-invariant, positive, and transpose-invariant (Knupp, 2001).

Finally, we define the distortion measure η of the physical element by

$$\eta(\mathbf{D}\boldsymbol{\phi}_E) = \eta_{\text{shape}}(\mathbf{D}\boldsymbol{\phi}_E)\ \eta_{\text{size}}(\mathbf{D}\boldsymbol{\phi}_E). \tag{2.6}$$

The distortion measure combines η_{shape} and η_{size}, see Knupp (2001) for more details. Thus, it quantifies both the size and the shape of the element.

2.3.2 Size-shape distortion for linear elements and constant metric

To define a measure that quantifies the quality of a given element, we need to define an ideal element that represents the desired configuration, as detailed in Section 2.2. In the unitary-Euclidean case, where the metric $\mathbf{M}$ is represented by the identity matrix $\mathbf{Id}$, the ideal element E^I corresponds to the equilateral element $E^{\triangle}$, the one with unit length edges. For non-unitary metrics, we describe how to obtain the ideal configuration. Then, we measure the distortion of the physical element by comparing it with the ideal element.

We define the ideal element as the element with edges of unit length under the desired metric. To compute this configuration, we first decompose $\mathbf{M}$ as follows

$$\mathbf{M} = \mathbf{F}^{\mathrm{T}}\ \mathbf{F}. \tag{2.7}$$

Matrix $\mathbf{F}$ can be interpreted as a linear mapping between the space with metric $\mathbf{M}$ and the space with unitary metric $\mathbf{Id}$. Thus, we define the anisotropic ideal E^I as the preimage by $\mathbf{F}$ of the equilateral element, see Figure 2.3. In particular, let $\mathbf{u}_i,\ i = 0, 1, 2$ be the nodes of the equilateral element $E^{\triangle}$. Then, we define the nodes $\mathbf{y}_i,\ i = 0, 1, 2$ of the ideal element E^I as

$$\mathbf{y}_i = \mathbf{F}^{-1}\ \mathbf{u}_i, \quad i = 0, 1, 2.$$

A direct consequence of the above definition is that the ideal triangle has unit edge lengths in the metric sense. Once the ideal triangle is defined, we measure the deviation between the ideal and physical elements. Similarly to the approaches for a unitary metric, see Section 2.2, in this section we define the distortion between the ideal E^I and physical E^P elements in terms of the mapping between those elements, $\boldsymbol{\phi}_E$.

A priori, we do not know how to compare elements considering the target metric. Nevertheless, we know how to compare elements in the unitary sense, see Section 2.2,

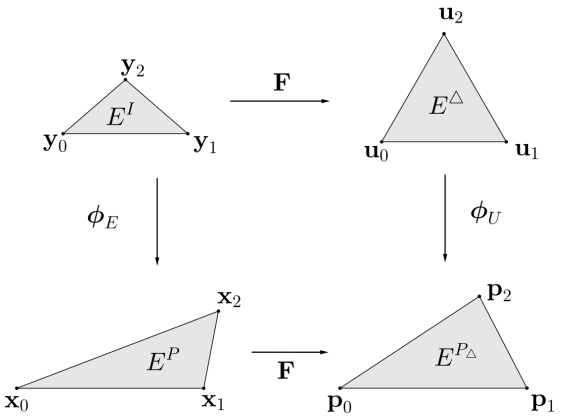


Figure 2.3: Mappings between the equilateral, the ideal, the physical, and the unitary physical triangles.

and thus, we map both elements E^I and E^P to the same Euclidean space using $\mathbf{F}$, see Figure 2.3. Then, we compare the image elements $E^\triangle$ and $E^{P_\triangle}$ using the distortion measure presented in Equation (2.6).

Let $E^{P_\triangle}$ be the image of the physical triangle E^P by $\mathbf{F}$. By construction, the image by $\mathbf{F}$ of the ideal triangle is the equilateral triangle. We measure the distortion between the ideal E^I and physical E^P elements in terms of the distortion of the mapping between the $E^\triangle$ and $E^{P_\triangle}$.

Finally, we define the distortion between the physical triangle E^P and the ideal triangle E^I with respect to the desired metric as the distortion of the matrix $\mathbf{D}\boldsymbol{\phi}_U$:

$$\eta_{\mathbf{M}}(\mathbf{D}\boldsymbol{\phi}_E) := \eta(\mathbf{D}\boldsymbol{\phi}_U). \tag{2.8}$$

The distortion presented in Equation (2.8) is well defined. This is because the measure does not depend on the symmetries of $E^{P_\triangle}$. We show first the case for rotations. The rotation of angle θ of $E^{P_\triangle}$ is the triangle $\tilde{E}^{P_\triangle}$ composed by the nodes $\tilde{\mathbf{y}}_i = \mathbf{R}(\theta)\ \mathbf{y}_i,\ i = 0, 1, 2$. Then

$$\mathbf{D}\tilde{\boldsymbol{\phi}}_U = \mathbf{R}(\theta)\ \mathbf{D}\boldsymbol{\phi}_U,$$

where $\tilde{\boldsymbol{\phi}}_U$ is the mapping between the equilateral triangle $E^\triangle$ and $\tilde{E}^{P_\triangle}$. Consequently, we have

$$\mathbf{D}\tilde{\boldsymbol{\phi}}_U^{\mathrm{T}}\ \mathbf{D}\tilde{\boldsymbol{\phi}}_U = \mathbf{D}\boldsymbol{\phi}_U^{\mathrm{T}}\ \mathbf{R}(\theta)^{\mathrm{T}}\ \mathbf{R}(\theta)\ \mathbf{D}\boldsymbol{\phi}_U = \mathbf{D}\boldsymbol{\phi}_U^{\mathrm{T}}\ \mathbf{D}\boldsymbol{\phi}_U. \tag{2.9}$$

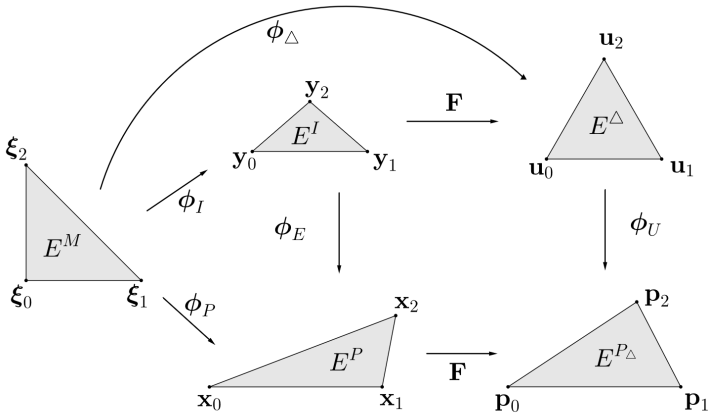


Figure 2.4: Mappings between the master, the equilateral, the ideal, the physical, and the unitary physical triangles.

By Equations (2.9), (2.8), and (2.6) we conclude that the corresponding distortions are equal. The case for reflections follows analogously since any symmetry $\boldsymbol{\Sigma}$ satisfies that $\boldsymbol{\Sigma}^{\mathrm{T}}\ \boldsymbol{\Sigma} = \mathbf{Id}$.

Next, we show how to compute the distortion presented in Equation (2.8) without decomposing it using matrix $\mathbf{F}$. First, in Figure 2.4, we include the master element in the diagram of mappings of Figure 2.3. Let $\boldsymbol{\phi}_{\triangle}$ be the mapping between the master and the equilateral triangle. This mapping is equivalent to the composition of the mappings $\boldsymbol{\phi}_I$ and $\mathbf{F}$, but it can be directly computed from the coordinates of the master and equilateral triangles, as previously done for the isotropic case in Section 2.2. Taking into account the computation of $\mathbf{D}\boldsymbol{\phi}_{\triangle}$ in terms of the node coordinates in Equation (2.2), the distortion measure $\eta_{\mathbf{M}}(\mathbf{D}\boldsymbol{\phi}_E)$ can be rewritten without decomposing $\mathbf{M}$. We note that, *a priori*, the right-hand side in Equation (2.8) depends on $\mathbf{F}$ since

$$\mathbf{D}\boldsymbol{\phi}_U = \mathbf{D}\boldsymbol{\phi}_{P^{\triangle}}\ \mathbf{D}\boldsymbol{\phi}_{\triangle}^{-1} = \mathbf{F}\ \mathbf{D}\boldsymbol{\phi}_P\ \mathbf{D}\boldsymbol{\phi}_{\triangle}^{-1}. \tag{2.10}$$

Manipulating Equation (2.8), one realizes that there is no explicit dependence on $\mathbf{F}$:

$$\begin{aligned}\mathbf{D}\boldsymbol{\phi}_U^{\mathrm{T}}\ \mathbf{D}\boldsymbol{\phi}_U &= \left(\mathbf{D}\boldsymbol{\phi}_{\triangle}\right)^{-\mathrm{T}}\ \mathbf{D}\boldsymbol{\phi}_P^{\mathrm{T}}\ \mathbf{F}^{\mathrm{T}}\ \mathbf{F}\ \mathbf{D}\boldsymbol{\phi}_P\ \left(\mathbf{D}\boldsymbol{\phi}_{\triangle}\right)^{-1} \\ &= \left(\mathbf{D}\boldsymbol{\phi}_{\triangle}\right)^{-\mathrm{T}}\ \mathbf{D}\boldsymbol{\phi}_P^{\mathrm{T}}\ \mathbf{M}\ \mathbf{D}\boldsymbol{\phi}_P\ \left(\mathbf{D}\boldsymbol{\phi}_{\triangle}\right)^{-1}.\end{aligned}$$

Thus, we obtain an expression for the distortion that does not require to decompose the metric $\mathbf{M}$. In particular, we define the a Riemannian analog for the Frobenius

norm $\mathbf{S}_\mathbf{M}$ and determinant $\sigma_\mathbf{M}$ as follows

$$\mathbf{S}_\mathbf{M} := \sqrt{\mathrm{tr}\left(\left(\mathbf{D}\boldsymbol{\phi}_P\ \mathbf{D}\boldsymbol{\phi}_\triangle^{-1}\right)^{\mathrm{T}}\ \mathbf{M}\ \mathbf{D}\boldsymbol{\phi}_P\ \mathbf{D}\boldsymbol{\phi}_\triangle^{-1}\right)}, \quad \text{and}$$
$$\sigma_\mathbf{M} := \sqrt{\det\left(\left(\mathbf{D}\boldsymbol{\phi}_P\ \mathbf{D}\boldsymbol{\phi}_\triangle^{-1}\right)^{\mathrm{T}}\ \mathbf{M}\ \mathbf{D}\boldsymbol{\phi}_P\ \mathbf{D}\boldsymbol{\phi}_\triangle^{-1}\right)}.$$

Finally, analogously to the Euclidean size-shape distortion measure of Equation (2.6), we define the Riemannian size-shape distortion as

$$\eta_\mathbf{M}(\mathbf{D}\boldsymbol{\phi}_E) = \eta_{\mathbf{M},\mathrm{shape}}(\mathbf{D}\boldsymbol{\phi}_E)\ \eta_{\mathbf{M},\mathrm{size}}(\mathbf{D}\boldsymbol{\phi}_E), \tag{2.11}$$

where the corresponding shape, $\eta_{\mathbf{M},\mathrm{shape}}$, and size, $\eta_{\mathbf{M},\mathrm{size}}$, distortion measures are given by

$$\eta_{\mathbf{M},\mathrm{shape}}(\mathbf{D}\boldsymbol{\phi}_E) = \frac{1}{d}\frac{\mathbf{S}_\mathbf{M}^2}{\sigma_\mathbf{M}^{2/d}}, \quad \text{and} \quad \eta_{\mathbf{M},\mathrm{size}}(\mathbf{D}\boldsymbol{\phi}_E) = \left(\frac{1}{2}\left(\sigma_\mathbf{M} + \frac{1}{\sigma_\mathbf{M}}\right)\right)^{2/d}.$$

2.3.3 Behavior of the quality measures: shape, size, and size-shape

In this section, we illustrate the behavior of the shape quality measure corresponding to the distortion measure, presented in Equation (2.8), for linear anisotropic triangles equipped with a constant metric. We first show the level curves of the quality measure of a triangle when we fix two nodes and we let the third node to move in $\mathbb{R}^2$, in Section 2.3.3.1. Second, in Section 2.3.3.2, we analyze the behavior of the measure with respect to the alignment of the element with the metric.

2.3.3.1 Level sets for one moving vertex

To show the behavior of the level curves of the shape, size, and size-shape quality measures we consider two cases, the Euclidean or isotropic case when $\mathbf{M} = \mathbf{Id}$ and the anisotropic case when $\mathbf{M}$ has two different eigenvalues.

For each quality measure and each metric we apply a test to a triangle. We illustrate the behavior by plotting the level sets in terms of a free node of the triangle. We consider the anisotropic metric given by

$$\mathbf{M} = \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{h^2} \end{pmatrix}, \quad h = 1/3. \tag{2.12}$$

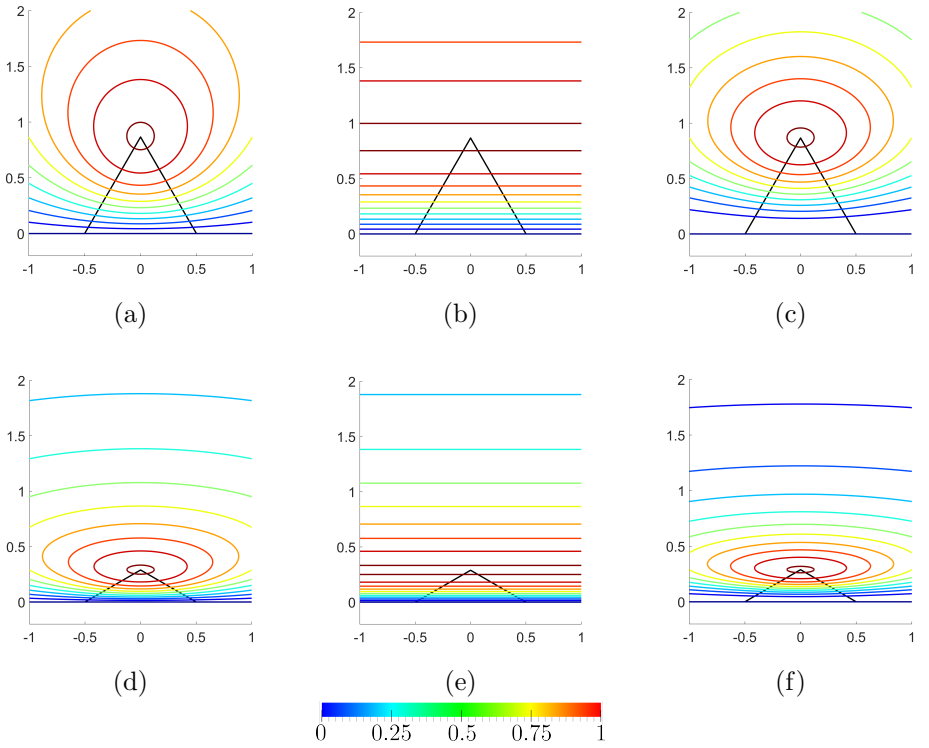


Figure 2.5: Level sets for the quality measures with different metrics: **(a,d)** shape, **(b,e)** size, and **(c,f)** size-shape; **(a,b,c)** isotropic and **(d,e,f)** anisotropic metrics.

This metric is aligned with the canonical axes and features a stretching ratio of 1 against 3. Specifically, it is devised to ensure that vectors $(1, 0)$ and $(0, h)$ have unit length. The ideal element E^I is expected to be an element of height h and base 1. In each test, we consider a free node, keeping the rest of nodes fixed at their original location, and we compute the quality of the element in terms of the location of this node. The free node considered is the vertex node $\mathbf{x}_2$.

In Figure 2.5, we show the contour plots of the quality for each test when the free node is allowed to move in a region of $\mathbb{R}^2$. The locus of the points where the element has positive Jacobian, the feasible region, is independent of the metric and corresponds to the half-plane $y > 0$.

As expected, for each metric the optimal node location is different. Furthermore, we can observe that the level sets and the height of the ideal triangle corresponding to the metric of Equation (2.12) are more stretched than in the isotropic case. Similarly, the level sets of the quality measure become more stretched as the anisotropy of the metric increases.

Similarly, for each quality measure, the optimal node location is also different. First, for the shape quality, the level curves are circular in the Euclidean case and elliptic in the metric case. Second, for the size quality, we observe that the level sets are straight horizontal lines. This is because the size quality depends only on the height of the triangle since the base is fixed. In the metric case, the spacing between the straight lines are more stretched than in the Euclidean case. Third, for the size-shape quality, the level curves are more stretched in the metric case than in the Euclidean case. Moreover, we observe that the level curves of the size-shape quality are more stretched than the ones of the shape quality. This indicates that the size-shape quality is more restrictive, in terms of variation, than the shape one.

2.3.3.2 Influence of element alignment

In the second test, we illustrate how the quality measure depends on the alignment between the anisotropy axes and the element. We compute the quality measure of a sequence of physical elements generated rotating the ideal element. We consider the metric presented in Equation (2.12).

Let $\mathbf{R}(\theta)$ be the rotation at the origin of angle $\theta \in [0, 2\pi)$ which is given by

$$\mathbf{R}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

We define the physical element as the ideal element rotated θ radians, with nodes $\mathbf{x}_i = \mathbf{R}(\theta) \quad \mathbf{y}_i,\ i = 0, 1, 2$. For each θ we compute the quality of the corresponding physical element.

In Figure 2.6, we plot the quality of each physical element with respect to the angle of the rotation applied to the ideal element to generate it. We represent the angle of rotation θ in the x-axis and the quality measure in the y-axis. We mark the cases $\theta = 0,\ \pi/2,\ \pi,\ 3\pi/2,$ and 2π with a black dot and we show the corresponding rotations of the ideal element in Figures 2.6(a), (b), (c), (d), and (e), respectively. We map a rotation of the unit circle in the Euclidean space to the same ellipse in the metric space, see Figures (a)-(e). We highlight that independently of the applied rotation, the ellipse remains constant. An element with quality one must have the nodes on the ideal ellipse.

In the isotropic case, rotations of the equilateral triangle have quality 1. In the anisotropic case, when two axes correspond to different eigenvalues of the metric, we

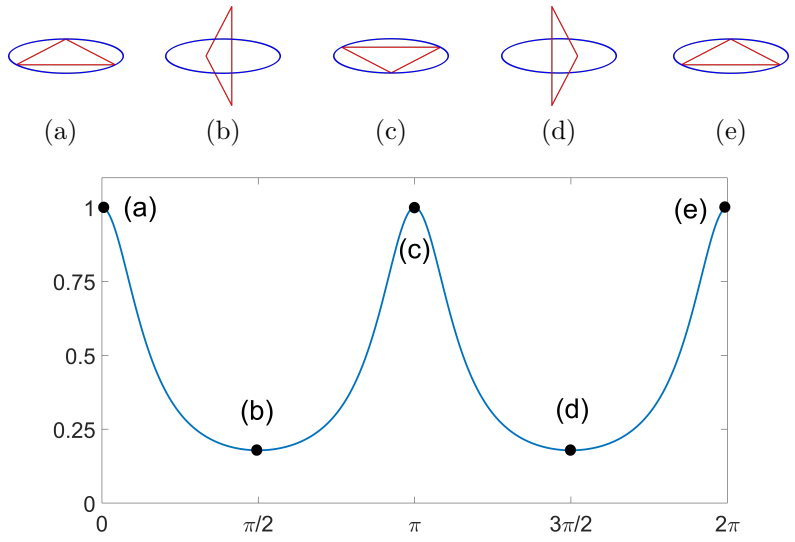


Figure 2.6: Influence of alignment in the shape quality measure. First row, physical elements which are rotations of the ideal element in radians: **(a)** 0; **(b)** $\pi/2$; **(c)** π; **(d)** $3\pi/2$; and **(e)** 2π. Second row, shape quality measure in terms of the rotation angle and corresponding mark for rotated elements **(a,b,c,d,e)**.

observe that the quality oscillates having two maxima and two minima in $[0, 2\pi)$. The maxima are obtained in $\theta = 0$ and $\theta = \pi$ and the minima at $\theta = \frac{\pi}{2}$ and $\theta = \frac{3\pi}{2}$. When $\theta = 0$ the rotation $\mathbf{R}(\theta)$ is the identity and $E^P = E^I$. When $\theta = \frac{\pi}{2}$ then the axes are interchanged (up to sign) and the quality at $\theta = \frac{\pi}{2}$ attains a minimum. The minima are attained when both axes are interchanged (up to sign) and the maxima are attained when the axes coincide with the eigenvectors of the metric (up to sign).

2.4 Measures for curved high-order meshes with varying metric

Herein, we define the point-wise measures for curved high-order meshes equipped with point-wise varying metrics. First, in Section 2.4.1, we present the point-wise size-shape distortion measure for high-order elements equipped with point-wise varying metrics. Then, in Section 2.4.2, we present the Riemmanian measure of mesh entities.

2.4.1 Size-shape distortion for curved high-order elements on varying metric

In Section 2.3, we presented the distortion measure for linear elements equipped with a constant metric. For high-order elements, the Jacobian of the mapping is not constant. In this section, we describe the analogous formulation for linear and curved high-order elements equipped with a non-constant metric field.

The point-wise distortion measure for an element E^P equipped with a metric $\mathbf{M}$, at a point $\mathbf{u} \in E^\triangle$ is defined as

$$\mathcal{N}\boldsymbol{\phi}_U(\mathbf{u}) := \eta(\mathbf{D}\boldsymbol{\phi}_U(\mathbf{u})).$$

Following Equation (2.4), the distortion measure for an element E^P equipped with a metric $\mathbf{M}$ is defined as

$$\eta_{(E^P,\mathbf{M})} = \frac{\int_{E^\triangle} \mathcal{N}\boldsymbol{\phi}_U(\mathbf{u}) \, \mathrm{d}\mathbf{u}}{\int_{E^\triangle} 1 \, \mathrm{d}\mathbf{u}}. \tag{2.13}$$

Equation (2.13) can be written in terms of $\boldsymbol{\xi}$ on the master element. That is, the Jacobian of the map $\boldsymbol{\phi}_U$ can be written in terms of $\boldsymbol{\xi}$ as:

$$\mathbf{D}\boldsymbol{\phi}_U(\boldsymbol{\phi}_\triangle(\boldsymbol{\xi})) = \mathbf{F}(\boldsymbol{\phi}_P(\boldsymbol{\xi})) \, \mathbf{D}\boldsymbol{\phi}_P(\boldsymbol{\xi}) \, \left(\mathbf{D}\boldsymbol{\phi}_\triangle(\boldsymbol{\xi})\right)^{-1},$$

where

$$\mathbf{M}(\boldsymbol{\phi}_P(\boldsymbol{\xi})) = \mathbf{F}(\boldsymbol{\phi}_P(\boldsymbol{\xi}))^{\mathrm{T}} \, \mathbf{F}(\boldsymbol{\phi}_P(\boldsymbol{\xi})).$$

Then, Equation (2.13) reads

$$\eta_{(E^P,\mathbf{M})} = \frac{\int_{E^M} \mathcal{N}\boldsymbol{\phi}_U(\boldsymbol{\phi}_\triangle(\boldsymbol{\xi})) \, |\det \mathbf{D}\boldsymbol{\phi}_\triangle(\boldsymbol{\xi})| \, \mathrm{d}\boldsymbol{\xi}}{\int_{E^M} |\det \mathbf{D}\boldsymbol{\phi}_\triangle(\boldsymbol{\xi})| \, \mathrm{d}\boldsymbol{\xi}}.$$

Similarly to Equation (2.11), the decomposition of the metric is not required:

$$\mathbf{D}\boldsymbol{\phi}_U(\boldsymbol{\phi}_\triangle(\boldsymbol{\xi}))^{\mathrm{T}} \, \mathbf{D}\boldsymbol{\phi}_U(\boldsymbol{\phi}_\triangle(\boldsymbol{\xi})) = \mathbf{A}(\boldsymbol{\xi})^{\mathrm{T}} \, \mathbf{M}(\boldsymbol{\phi}_P(\boldsymbol{\xi})) \, \mathbf{A}(\boldsymbol{\xi}),$$

where

$$\mathbf{A}(\boldsymbol{\xi}) := \mathbf{D}\boldsymbol{\phi}_P(\boldsymbol{\xi}) \, \left(\mathbf{D}\boldsymbol{\phi}_\triangle(\boldsymbol{\xi})\right)^{-1}.$$

Using the above equation we obtain the final expression on each point $\boldsymbol{\xi}$ of the master element:

$$\mathcal{N}\boldsymbol{\phi}_U(\boldsymbol{\phi}_\triangle(\boldsymbol{\xi})) = \eta_{\mathbf{M}}\left(\mathbf{A}(\boldsymbol{\xi})\right), \tag{2.14}$$

where

$$\mathbf{S}_{\mathbf{M}}\left(\mathbf{A}(\boldsymbol{\xi})\right) := \sqrt{\mathrm{tr}\left(\mathbf{D}\boldsymbol{\phi}_E(\boldsymbol{\xi})^{\mathrm{T}}\ \mathbf{M}(\boldsymbol{\phi}_P(\boldsymbol{\xi}))\ \mathbf{D}\boldsymbol{\phi}_E(\boldsymbol{\xi})\right)}, \quad \text{and}$$
$$\sigma_{\mathbf{M}}\left(\mathbf{A}(\boldsymbol{\xi})\right) := \det \mathbf{D}\boldsymbol{\phi}_P(\boldsymbol{\xi}) \det \mathbf{D}\boldsymbol{\phi}_{\triangle}(\boldsymbol{\xi})^{-1} \sqrt{\det \mathbf{M}(\boldsymbol{\phi}_P(\boldsymbol{\xi}))}.$$

In order to detect inverted elements (Branets and Garanzha, 2002; López et al., 2008; Escobar et al., 2003; Gargallo-Peiró et al., 2015c) we regularize the determinant $\sigma_{\mathbf{M}}$ to

$$\sigma_{0,\mathbf{M}} := \frac{1}{2}(\sigma_{\mathbf{M}} + |\sigma_{\mathbf{M}}|).$$

Then, we define the point-wise regularized distortion measure of a physical element E^P as

$$\mathcal{N}_0\boldsymbol{\phi}_U(\mathbf{u}) := \eta_0(\mathbf{D}\boldsymbol{\phi}_U(\mathbf{u})) := \frac{1}{d}\frac{\mathbf{S}_{\mathbf{M}}^2}{\sigma_{0,\mathbf{M}}^{2/d}} \left(\frac{1}{2}\left(\sigma_{0,\mathbf{M}} + \frac{1}{\sigma_{0,\mathbf{M}}}\right)\right)^{2/d},$$

and its corresponding quality

$$\mathcal{Q}_0\boldsymbol{\phi}_U(\mathbf{u}) := \frac{1}{\mathcal{N}_0\boldsymbol{\phi}_U(\mathbf{u})}. \tag{2.15}$$

Finally, we regularize the elemental distortion of Equation (2.13) as

$$\eta_{0,(E^P,\mathbf{M})} := \frac{\int_{E^{\triangle}} \mathcal{N}_0\boldsymbol{\phi}_U(\mathbf{u})\ \mathrm{d}\mathbf{u}}{\int_{E^{\triangle}} 1\ \mathrm{d}\mathbf{u}},$$

and its corresponding quality as

$$q_{0,(E^P,\mathbf{M})} := \frac{1}{\eta_{0,(E^P,\mathbf{M})}}. \tag{2.16}$$

We can improve the mesh configuration by means of relocating the nodes of the mesh according to a given distortion measure (Chapter 3). For example, in Aparicio-Estrems et al. (2018) it is proposed the optimization of the distortion (quality) of a mesh $\mathcal{M}$ equipped with a target metric $\mathbf{M}$ that describes the desired alignment and stretching of the mesh elements. To optimize the given mesh $\mathcal{M}$, we define the mesh distortion by

$$\mathcal{F}\left(\mathcal{M}\right) := \sum_{E^P \in \mathcal{M}} \int_{E^{\triangle}} \left(\mathcal{N}_0\boldsymbol{\phi}_E(\mathbf{y})\right)^2\ \mathrm{d}\mathbf{y}, \tag{2.17}$$

which allows to pose the following global minimization problem

$$\mathcal{M}^* := \mathrm{argmin}_{\mathcal{M}}\mathcal{F}\left(\mathcal{M}\right), \tag{2.18}$$

to improve the mesh configuration according to $\mathcal{F}$. In particular, herein, the degrees of freedom of the minimization problem in Equation (2.18) correspond to the spatial coordinates of the mesh nodes.

2.4.2 Riemmanian measure of mesh entities

Next, we propose a method to compare how a mesh matches a target metric. To this end, we present the point-wise and element-wise size measure of the mesh entities according to a Riemannian metric.

We consider the Riemannian measure of the mesh entities relative to the target metric $\mathbf{M}$. On the one hand, we define the point-wise relative measure $\rho_{\mathbf{M}}$ of a physical element E^P with respect to a reference element E^M. Specifically, for a k-dimensional physical element E^P embedded in the Riemannian space $(\mathbb{R}^n, \mathbf{M})$, the *point-wise metric-aware density* of E^P respect to a k-dimensional master element E^M is given by $\sqrt{\det\ [\mathbf{D}\boldsymbol{\phi}_P(\boldsymbol{\xi})^{\mathrm{T}}\ \mathbf{M}\left(\boldsymbol{\phi}_P(\boldsymbol{\xi})\right)\ \mathbf{D}\boldsymbol{\phi}_P(\boldsymbol{\xi})]}$. We also consider the *metric-aware normalized density* as the quotient of the physical density by the ideal density

$$\rho_{\mathbf{M}}(\boldsymbol{\xi}) := \sqrt{\frac{\det\ [\mathbf{D}\boldsymbol{\phi}_P(\boldsymbol{\xi})^{\mathrm{T}}\ \mathbf{M}\left(\boldsymbol{\phi}_P(\boldsymbol{\xi})\right)\ \mathbf{D}\boldsymbol{\phi}_P(\boldsymbol{\xi})]}{\det\ [\mathbf{D}\boldsymbol{\phi}_I(\boldsymbol{\xi})^{\mathrm{T}}\ \mathbf{M}\left(\boldsymbol{\phi}_P(\boldsymbol{\xi})\right)\ \mathbf{D}\boldsymbol{\phi}_I(\boldsymbol{\xi})]}}, \quad \text{for} \quad \boldsymbol{\xi} \in E^M. \tag{2.19}$$

Accordingly, we say that an element E^P is *unitary* if $\rho_{\mathbf{M}} \equiv 1$ is satisfied for the element measure and for the measure of all its sub-entities. Considering the commutative diagram in Figure 2.4, we compute the metric-aware density as

$$\rho_{\mathbf{M}}(\boldsymbol{\xi}) = \sqrt{\frac{\det\ [\mathbf{D}\boldsymbol{\phi}_P(\boldsymbol{\xi})^{\mathrm{T}}\ \mathbf{M}\left(\boldsymbol{\phi}_P(\boldsymbol{\xi})\right)\ \mathbf{D}\boldsymbol{\phi}_P(\boldsymbol{\xi})]}{\det\ \left[\mathbf{D}\boldsymbol{\phi}_\triangle(\boldsymbol{\xi})^{\mathrm{T}}\ \mathbf{D}\boldsymbol{\phi}_\triangle(\boldsymbol{\xi})\right]}}, \quad \text{for} \quad \boldsymbol{\xi} \in E^M, \tag{2.20}$$

where the unit element $E^\triangle$ is a k-dimensional regular element with all the edges of unit length. Note that, any sub-entity of a unitary element $E^\triangle$ is also unitary, because all edges have unit length. On the other hand, we define the *metric-aware normalized measure* of E^P according to the metric $\mathbf{M}$ as

$$V_{\mathbf{M}}\left(E^P\right) := \frac{1}{V\left(E^M\right)} \int_{E^M} \rho_{\mathbf{M}}(\boldsymbol{\xi})\ \mathrm{d}\boldsymbol{\xi}, \tag{2.21}$$

where $V\left(E^M\right) = \int_{E^M} 1\ \mathrm{d}\boldsymbol{\xi}$.

While it is common to consider only the element-wise length of the element edges, this does not illustrate if the element is unitary or not, specially for non-constant metric or curved elements. In contrast, an element is unitary if the point-wise measure of all its sub-entities is constant equal to one. For this reason, we measure how a mesh is unitary according to the metric by measure all the mesh entities in the point-wise sense. In particular, we do this in terms of the measures of the mesh entities. That is, lengths of edges, areas of faces, and volumes of cells. For example, only when the mesh

matches the metric, the lengths, areas, and volumes are unit and vice-versa. That is, they match the length, area, and volume of the equilateral element, respectively. In contrast, a higher stretching or non-unit volume of the intrinsic metric indicates that the mesh does not match the stretching or the volume of the metric, respectively. As a consequence, the lengths, areas, and volumes are non-unit and vice-versa.

2.5 Results

In this section, we apply the size-shape distortion minimization for curved r-adaption. Specifically, we first compare the behavior of the size-shape distortion minimization with the shape distortion minimization presented in Aparicio-Estrems et al. (2018). Then, we illustrate how the size-shape distortion minimization can be used for the improvement of the interpolation error of an input function.

First, in Section 2.5.1, we compare the shape and size-shape distortion measures for an analytic target metric. Second, in Sections 2.5.2, 2.5.3, and 2.5.4, we apply the size-shape distortion minimization for high-order interpolation. Specifically, in Section 2.5.2, we consider a 2D case with varying degrees and a quadratic 3D example. Then, in Section 2.5.3, we minimize the size-shape distortion for initial isotropic and initial adapted straight-edged quartic meshes. The results show that the size-shape distortion minimization improves the interpolation and approximation errors of the input function. Finally, in Section 2.5.4, we minimize the size-shape distortion for an initial adapted straight-edged cubic mesh according to a curved boundary. The results show that the size-shape distortion minimization improves the interpolation and approximation errors of the input function while targeting a curved boundary.

Because our goal is to optimize the mesh distortion, instead of including mathematical proofs of mesh validity, we detail how we numerically enforce the positiveness of the element Jacobians. Specifically, we use a numerical valid-to-valid approach that uses four ingredients. First, because we want numerically valid results, we enforce mesh validity on the integration points. Second, to initialize the optimization, we start from a numerically valid mesh. Third, to penalize inverted elements, we modify the point-wise distortion to be infinity for non-positive Jacobians. Specifically, we regularize the element Jacobians to be zero for non-positive Jacobians, so their reciprocals are infinite, see Section 2.4. Note that these reciprocals appear in the distortion expression, and thus, they determine the infinite distortion value. Fourth,

to enforce numerically valid mesh displacements, we equip Newton's method with a backtracking line-search. Specifically, if the mesh optimization update is invalid in any integration point, the objective function is infinite. In that case, the step is divided by two until it leads to a valid mesh update.

The Julia prototyping code is sequential, it corresponds to the implementation of the method presented in this chapter and the one presented in Chapter 3. In all the examples, the optimization corresponds to finding a minimum of a nonlinear unconstrained multi-variable function f, see Equations (2.17) and (2.18). In particular, the mesh optimizer uses an unconstrained line-search globalization with an iterative preconditioned conjugate gradients linear solver. The stopping condition is set to reach an absolute root mean square residual, defined as $\frac{\|\nabla f(x)\|_{\ell^2}}{\sqrt{n}}$ for $x \in \mathbb{R}^n$, smaller than 10^{-4} or a length-step smaller than 10^{-4}. Each optimization process has been performed in a node featuring two Intel Xeon Platinum 8160 CPU with 24 cores, each at 2.10 GHz, and 96 GB of RAM memory.

Although we generate meshes adapted to a target metric with MMG (Dobrzynski, 2012), our goal is not to compare the distortion minimization with the MMG package. Actually, we acknowledge MMG because it generates an initial straight-edged mesh that matches the stretching and alignment of the target metric.

2.5.1 Shape versus size-shape distortion minimization: curved high-order mesh and analytic metric

In what follows, we compare the shape and size-shape distortion measures presented in Section 2.4. Specifically, we do this for a curved high-order mesh and an analytic metric. For this, we first define the target metric. Then, we illustrate the initial and optimized meshes. Finally, we compare the distortion measures from the distribution and statistics of the Riemannian measures (length and area), see Section 2.4.2 for the details.

We consider the quadrilateral domain $\Omega = [-0.5, 0.5]^2$ equipped with a metric matching a boundary layer. In particular, our target metric $\mathbf{M}$ is characterized by a boundary layer metric with a diagonal matrix $\mathbf{D}$, a deformation map φ, and the characteristic length $h_{\mathrm{m}} := 0.25$ by the following expression

$$\mathbf{M} = \frac{1}{h_{\mathrm{m}}^2} \nabla\varphi^{\mathrm{T}} \, \mathbf{D} \, \nabla\varphi. \tag{2.22}$$

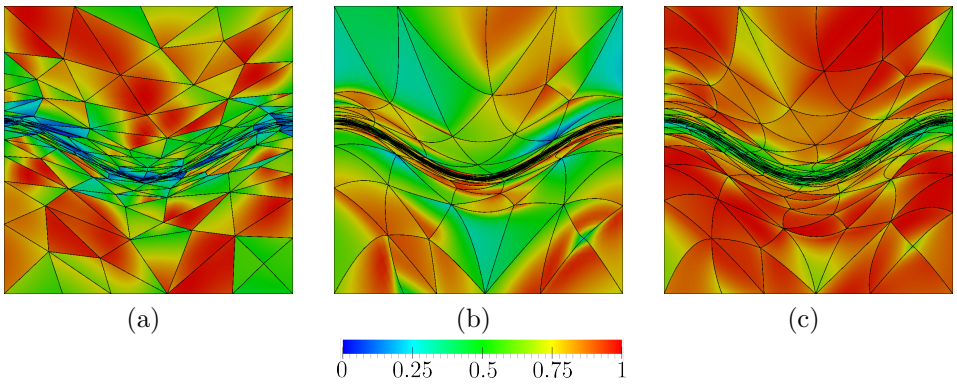


Figure 2.7: Point-wise size-shape quality measure for (a) initial and quadratic meshes optimized according to the (b) shape and (c) size-shape distortion measure, respectively.

In what follows, we first detail the boundary layer metric $\mathbf{D}$ and then the deformation map φ, see Chapter 4 for more details.

On the one hand, the boundary layer aligns with the x-axis. It determines a constant unit element size along the x-direction, and a non-constant element size along the y-direction. This vertical element size grows linearly with the distance to the x-axis, with a factor $\alpha = 2$, and starts with the minimal value $h_{\min} = 0.01$. Thus, the stretching ratio blends from $1 : 100$ to $1 : 1$ between $y = -0.5$ and $y = 0.5$. Specifically, we define the metric as:

$$\mathbf{D} := \begin{pmatrix} 1 & 0 \\ 0 & 1/h(y)^2 \end{pmatrix}, \tag{2.23}$$

where the function h is defined by

$$h(x) := h_{\min} + \alpha|x|.$$

On the other hand, the deformation map φ in Equation (2.22) aligns the stretching of $\mathbf{D}$ according to a given curve. In this case, we define the map φ by

$$\varphi(x, y) := \left(x, \frac{10y - \cos(2\pi x)}{\sqrt{100 + 4\pi^2}} \right).$$

Finally, the metric $\mathbf{M}$ of Equation (2.22) attains the highest level of anisotropy close to the curve described by the points $(x, y) \in \Omega$ such that $\varphi(x, y) = (x, 0)$.

In Figure 2.7, we illustrate the initial and optimized quadratic meshes equipped with the input metric of Equation (2.22). The meshes are colored according to the

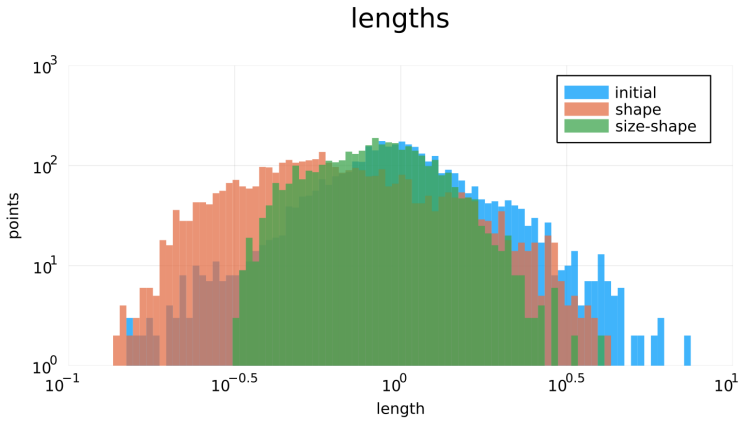
Table 2.1: Size-shape quality and geometry statistics for the initial and optimized meshes according to the shape and size-shape distortion measures.

Measure	Mesh	Minimum	Maximum	Mean	Standard deviation
Quality	Initial	0.0156	0.9694	0.4276	0.2687
	Shape	0.0950	0.9104	0.4927	0.2207
	Size-shape	0.3136	0.9882	0.6337	0.1976
Length	Initial	0.2241	3.8578	1.1369	0.5451
	Shape	0.1964	3.1337	0.7926	0.5445
	Size-shape	0.3471	2.3952	0.9131	0.3383
Area	Initial	0.0724	1.8048	0.5593	0.2931
	Shape	0.0417	2.6045	0.5594	0.5843
	Size-shape	0.2148	1.1885	0.5593	0.2310

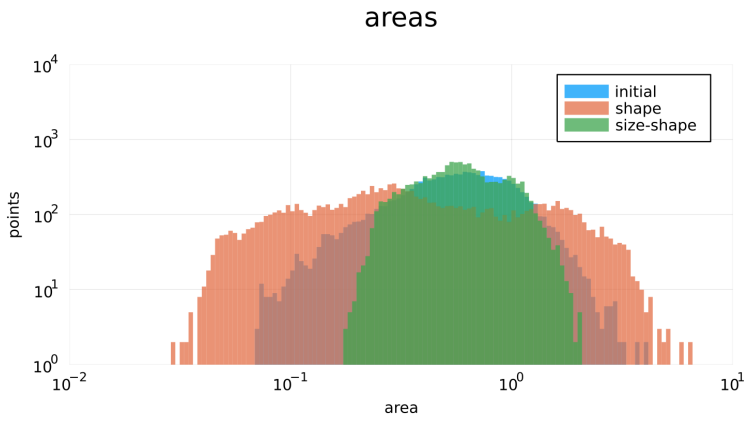
point-wise size-shape quality measure of Equation (2.15). With MMG, we generate an initial anisotropic straight-edged mesh $\mathcal{M}$ according to the target metric of Equation (2.22). The obtained mesh is composed by 254 triangles and 553 nodes. From this initial mesh, we observe that the straight-edged elements are stretched, aligned, and scaled approximating the target metric. Then, we optimize the initial mesh $\mathcal{M}$ according to the shape and size-shape distortion measures to obtain the corresponding optimized meshes $\mathcal{M}^*_{\text{shape}}$ and $\mathcal{M}^*$. Finally, we observe that the elements are curved according to the point-wise metric stretching and alignment for the mesh $\mathcal{M}^*_{\text{shape}}$, and according to the point-wise metric stretching, alignment, and sizing for the mesh $\mathcal{M}^*$.

From this example, we qualitatively compare the shape and size-shape distortion measures. For this, in Figure 2.8, we illustrate the logarithmic point-wise distributions of Riemannian length and area for the associated metric, see Section 2.4.2. In particular, the shape minimization distorts the distribution of length and area. In contrast, when compared to the initial mesh and the shape minimization, the size-shape optimization concentrates more the distribution of length and area around unit values. From this, we conclude that the size-shape minimization matches more faithfully the target metric than the shape optimization.

We quantitatively compare the shape and size-shape distortion measures. For this, in Table 2.1, we show the statistics of the elemental size-shape quality (Equation (2.16)) and Riemannian length and area (Equation (2.21)). They allow us to compare the geometric quantities between the initial and optimized meshes, and between the shape and the size-shape quality measures. On the one hand, we observe that the shape minimization does not improve the length and area statistics from the



(a)



(b)

Figure 2.8: Logarithmic point-wise (first row) length and (second row) area for (blue) initial and optimized quadratic meshes according to the (orange) shape and (green) size-shape distortion measure.

initial mesh. This is because, the shape distortion does not take into account the local element size. In contrast, when compared from the initial mesh and the shape minimization, the size-shape optimization substantially improves the length and area statistics. This can be explained from the coupling between the size and shape distortion measures, which takes into account the local element size and shape deviation, see Section 2.4. From this, we conclude that the size-shape distortion minimization homogeneously matches more the geometric features of the input metric than the shape optimization.

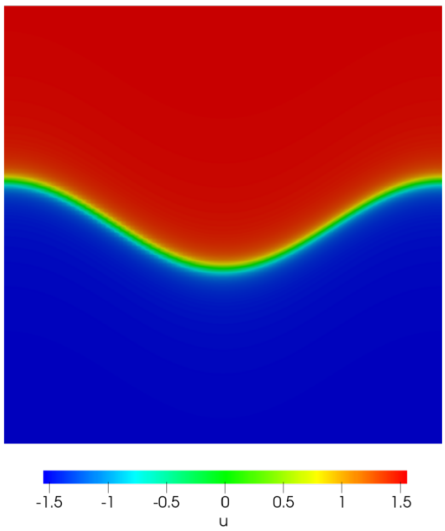


Figure 2.9: Values of the function u for $\gamma = 10$.

2.5.2 Size-shape distortion minimization for high-order interpolation: 2D varying degrees and 3D quadratic

Herein, we apply the size-shape distortion minimization for high-order interpolation. In particular, we consider a 2D case with varying degrees and a 3D quadratic example. For this, in Section 2.5.2.1, we set a discrete target metric from the higher-order derivatives of the input function. From this discrete metric, we minimize the size-shape distortion, in Section 2.5.2.2. To verify that the stretching, alignment, and sizing match the discrete metric, we measure the Riemannian lengths, areas, and volumes, in Section 2.5.2.3. Then, to illustrate the potential of curved r-adaption, we measure how the mesh represents the input function. In particular, we measure the interpolation and approximation L^2-errors, in Section 2.5.2.4.

2.5.2.1 Discrete high-order metric: high-order interpolation

Herein, we compute a discrete metric from the input function as in Coulaud and Loseille (2016a). Specifically, for each polynomial interpolation degree, we obtain a discrete metric approximating the high-order derivatives of the function.

In the 2D case, we consider a square domain $\Omega = [-0.5, 0.5]^2$ and a function $u : \Omega \to \mathbb{R}$ given by

$$u(x, y) := \arctan\left(\gamma\ \varphi(x, y)\right), \quad \varphi(x, y) := 10y + \cos(2\pi x). \tag{2.24}$$

In Figure 2.9, we show the values of u for $\gamma = 10$. We observe that, near the curve $\varphi(x, y) = 0$ there is a sharp transition. Far away from such curve, the function is almost constant.

In the 3D case, we consider a square domain $\Omega = [-0.5, 0.5]^3$ and a function $u : \Omega \to \mathbb{R}$ given by

$$u(x, y, z) := \arctan\left(\gamma\ \varphi(x, y, z)\right), \quad \varphi(x, y, z) := 10z + \cos(2\pi x)\cos(2\pi y). \quad (2.25)$$

Analogously to the 2D case, near the surface $\varphi(x, y, z) = 0$ there is a sharp transition. Far away from such surface, the function is almost constant.

To approximate the function u we consider an error indicator represented by a discrete target metric $\hat{\mathbf{M}}$. We obtain the metric $\hat{\mathbf{M}}$ from the high-order derivatives of the function u (Coulaud and Loseille, 2016a). In particular, for a mesh polynomial degree q, we consider the $(q + 1)$th derivatives of u, $\nabla^{q+1}u$. Then, we obtain the discrete metric $\hat{\mathbf{M}}$ in terms of $\nabla^{q+1}u$. To do this, we generate a background isotropic mesh $\hat{\mathcal{M}}$ of polynomial degree q and we evaluate the high-order derivatives, $\nabla^{q+1}u$, at the background mesh nodes. Finally, we obtain the values of an approximative discrete metric $\hat{\mathbf{M}}$ at the background mesh nodes and, we regularize this metric according to an L^p-norm and a fixed size h (Loseille and Alauzet, 2011).

2.5.2.2 Size-shape distortion minimization: straight-edged anisotropic mesh adapted to the discrete metric

Herein, we minimize the size-shape distortion according to the discrete metric $\hat{\mathbf{M}}$ of Section 2.5.2.1. To do this, we apply the methodology presented in Chapter 4. The method considers two meshes: a background mesh $\hat{\mathcal{M}}$ and a physical mesh $\mathcal{M}$. First, we generate a background mesh $\hat{\mathcal{M}}$ to interpolate the metric values $\mathbf{M}$ in terms of the discrete metric $\hat{\mathbf{M}}$. Then, we generate and optimize a physical mesh $\mathcal{M}$ according to the interpolated metric $\mathbf{M}$. This results in a triangular (tetrahedral) mesh $\mathcal{M}^*$ with Riemannian lengths and areas (and volumes) closer to the metric unit, see Section 2.5.2.3. As a consequence, the interpolation and approximation error are improved, see Section 2.5.2.4.

For this, we consider a background mesh $\hat{\mathcal{M}}$ and a physical mesh $\mathcal{M}$ of the same polynomial degree q, and the same characteristic size h. We first generate an isotropic background mesh $\hat{\mathcal{M}}$ and we equip it with the discrete target metric $\hat{\mathcal{M}}$. From this mesh, we generate an initial anisotropic physical mesh $\mathcal{M}$ with the MMG mesh gen-

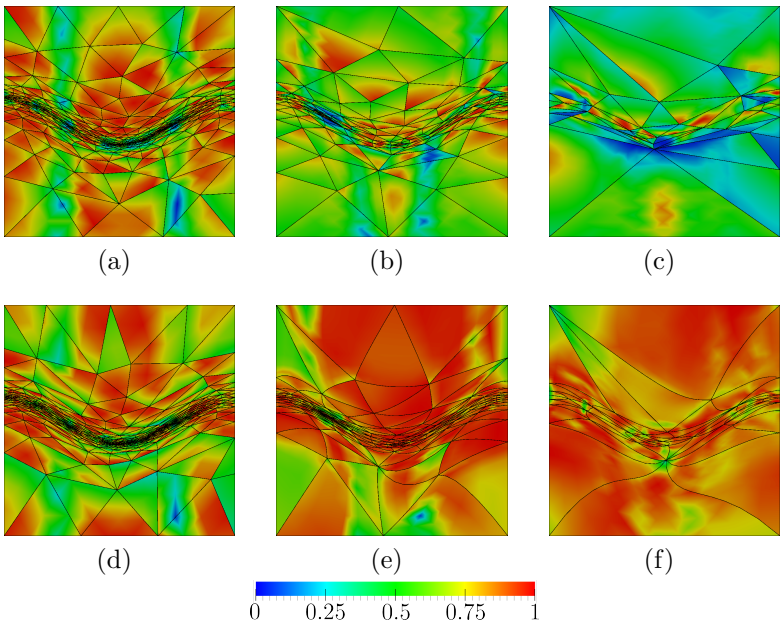


Figure 2.10: Point-wise size-shape quality measure for (rows) initial and optimized triangular meshes of (columns) polynomial degree 1, 2, and 4.

erator (Dobrzynski, 2012). In this situation, using a high-order background mesh is not possible. Instead, we consider the linear metric interpolation in a uniformly subdivided linear background mesh $\hat{\mathcal{M}}'$ from the generated one $\hat{\mathcal{M}}$. We expect that both, the high-order $\hat{\mathcal{M}}$ and the subdivided background $\hat{\mathcal{M}}'$ meshes, represent faithfully the metric, even if their elemental node locations differ. Finally, we relocate the nodes of the initial physical mesh $\mathcal{M}$ by minimizing the size-shape distortion measure, see Section 2.4. In this case, to obtain the point-wise varying metric $\mathbf{M}$, we consider the high-order Log-Euclidean metric interpolation of the discrete target metric $\hat{\mathbf{M}}$ at the high-order background mesh $\hat{\mathcal{M}}$, see Rochery and Loseille (2021); Arsigny et al. (2006) for the details.

In Figures 2.10 and 2.11, we illustrate the triangular and tetrahedral physical meshes, respectively. That is, the initial, $\mathcal{M}$, and optimized, $\mathcal{M}^*$, meshes equipped with the metric $\mathbf{M}$. On the one hand, we consider the function u of Equation (2.24) with $\gamma = 100$, in 2D, and $\gamma = 10$, in 3D. Then, we obtain the metric $\mathbf{M}$ by interpolating the discrete metric $\hat{\mathbf{M}}$ at the background mesh $\hat{\mathcal{M}}$. Note that, the metric scaling is imposed by regularizing the discrete metric $\hat{\mathbf{M}}$ according to the $L^2(\Omega)$-norm, see Section 2.5.2.1. On the other hand, the 2D physical meshes are of polynomial degree $q = 1$, 2, and 4, and of size $h = 0.05$. Each mesh is composed of 327, 491, and 523

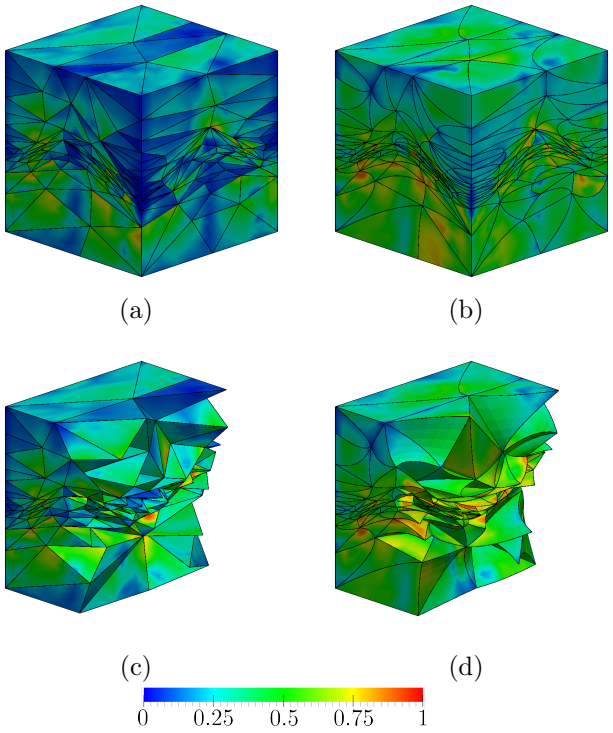


Figure 2.11: Point-wise size-shape quality measure for (columns) initial and optimized of (rows) full and clipped quadratic tetrahedral meshes.

nodes and of 611, 230, and 61 triangles, respectively. In the 3D case, the physical meshes are of polynomial degree $q = 2$ and of size $h = 0.1$. They are composed of 3754 nodes and 2425 tetrahedra, respectively. From the initial mesh $\mathcal{M}$, we observe that, at the sharp transition region, the elements are stretched, aligned, and scaled according to the metric. However, the straight-edged elements cannot align with the curved transition region. In contrast, for the optimized meshes $\mathcal{M}^*$, we observe that the elements are curved according to the point-wise stretching, alignment, and sizing of the metric.

2.5.2.3 Verifying results: distributions for Riemannian measures of distortion and mesh entities

Next, we illustrate how the size-shape distortion minimization enables a mesh that approximates more faithfully the target metric. For this, we measure the Riemannian length, area, and volume distributions of the mesh entities, see Equation (2.21). The results show that the size-shape distortion minimization enables an optimized mesh

Table 2.2: Size-shape quality, geometry, and error statistics of the initial meshes and the corresponding optimized meshes.

Measure	Mesh degree	Minimum		Maximum		Mean		Standard deviation	
		Initial	Optimized	Initial	Optimized	Initial	Optimized	Initial	Optimized
Quality	1	0.1019	0.2574	0.9779	0.9737	0.6710	0.7330	0.1898	0.1391
	2	0.0986	0.5229	0.9161	0.9812	0.6021	0.8538	0.1604	0.0883
	4	0.0249	0.6881	0.7565	0.9275	0.3756	0.8307	0.1761	0.0523
Length	1	0.3641	0.4293	5.1197	3.2040	1.2916	1.2520	0.5376	0.3503
	2	0.4711	0.5726	4.6246	2.3879	1.1160	1.0269	0.5172	0.2619
	4	0.3109	0.3295	4.0158	1.7523	1.1334	0.9856	0.7727	0.2653
Area	1	0.1998	0.3300	5.2956	3.5361	1.3135	1.3137	0.7967	0.6158
	2	0.2838	0.4376	5.1946	2.3034	0.8695	0.8696	0.6145	0.3219
	4	0.0911	0.4923	2.9104	1.5600	0.8195	0.8196	0.6847	0.2312

Table 2.3: Size-shape quality, geometry, and error statistics of the initial and optimized quadratic tetrahedral meshes.

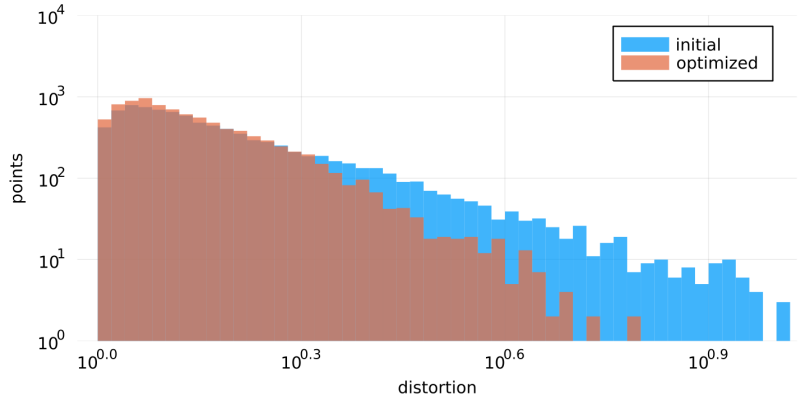
Measure	Minimum		Maximum		Mean		Standard deviation	
	Initial	Optimized	Initial	Optimized	Initial	Optimized	Initial	Optimized
Quality	0.0021	0.2117	0.8351	0.8545	0.3414	0.5315	0.1580	0.1082
Length	0.1760	0.2388	3.4170	3.4307	0.9340	0.8728	0.3686	0.2560
Area	0.0687	0.1673	3.6299	2.0579	0.5906	0.5557	0.3148	0.1797
Volume	0.0229	0.1168	2.3427	0.9282	0.3097	0.3097	0.2179	0.0941

featuring an improved approximation of the target metric.

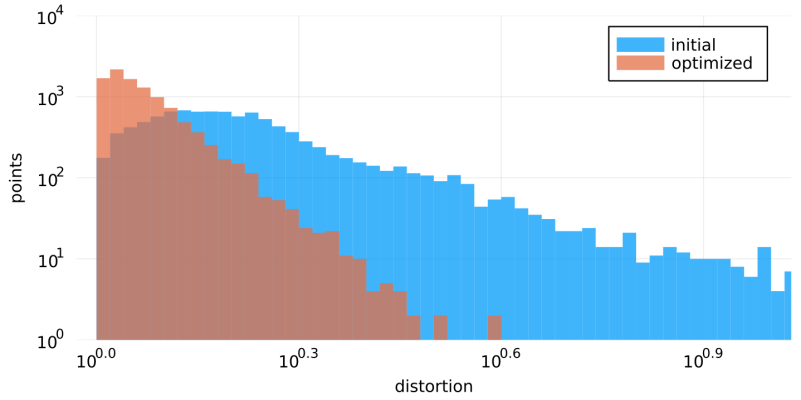
In Tables 2.2 and 2.3, we show the corresponding triangular and tetrahedral mesh statistics for the logarithmic distributions of elemental qualities (Equation (2.16)) and Riemannian measures. That is, lengths and areas in 2D and lengths, areas, and volumes in 3D. They allow us to compare the geometric quantities between the initial and optimized physical meshes in terms of the target metric. We observe that the minimum and standard deviation become closer to unit values in all cases.

In Figures 2.12, 2.13, and 2.14, we respectively show the point-wise distortion, length, and area of the initial and optimized triangular meshes. Similarly, in Figures 2.15(a), 2.15(b), and 2.15(c), we respectively show the point-wise length, area, and volume of the initial and optimized quadratic tetrahedral meshes. Note that, the geometric quantities are typically compared in terms of ratios that is, in a multiplicative form. Accordingly, we use a logarithmic scale to illustrate the different scales of the corresponding ratios. The logarithmic representation illustrates the behavior near the minimum, maximum, and geometric mean of the distribution.

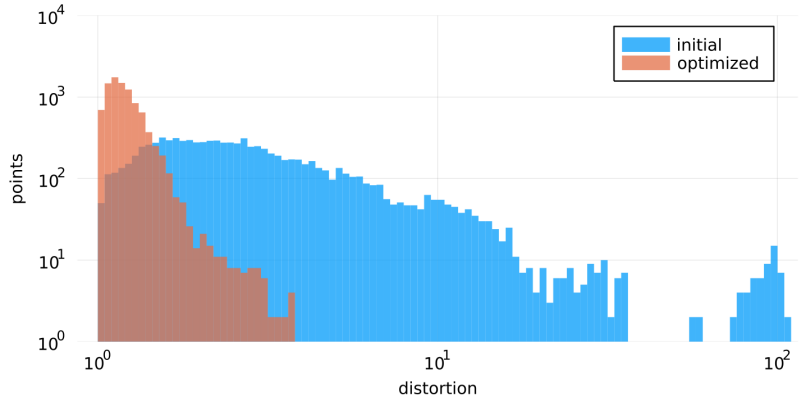
From the reasoning presented in Section 2.4.2, we observe that almost all measure statistics are improved for the optimized meshes. On the one hand, for the geometric



(a)

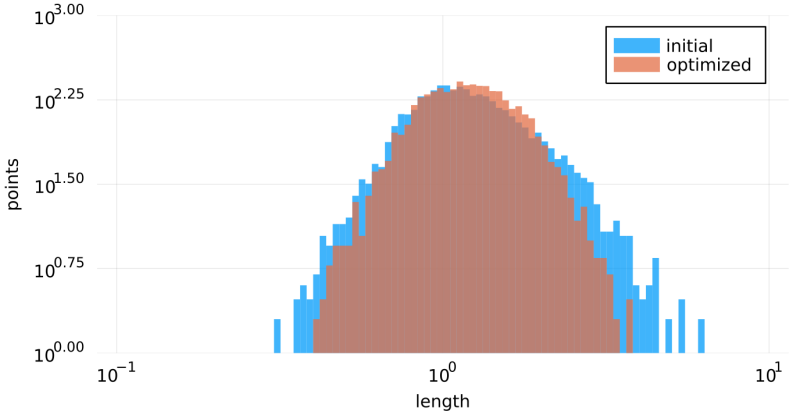


(b)

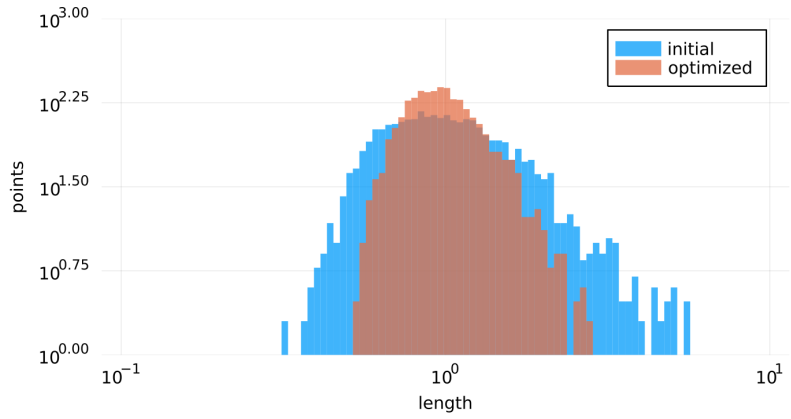


(c)

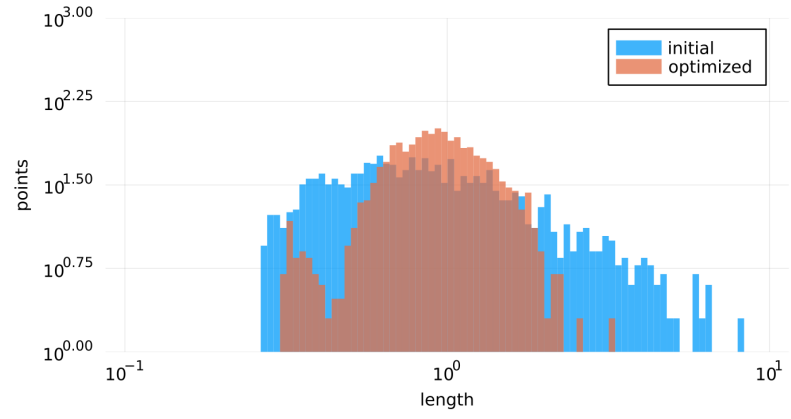
Figure 2.12: Logarithmic point-wise size-shape distortion histograms for (blue) initial and (orange) optimized meshes of polynomial degree 1, 2, and 4, respectively.



(a)

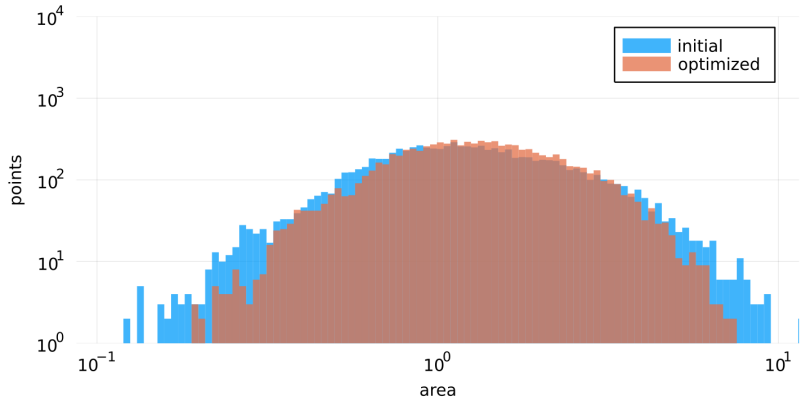


(b)

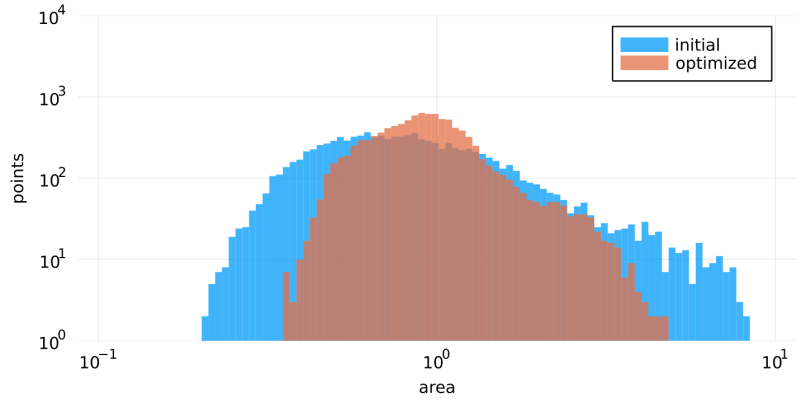


(c)

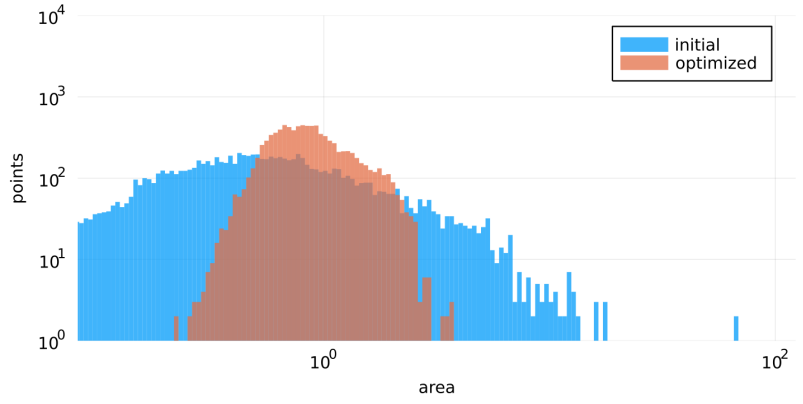
Figure 2.13: Logarithmic point-wise Riemannian length histograms for (blue) initial and (orange) optimized meshes of polynomial degree 1, 2, and 4, respectively.



(a)

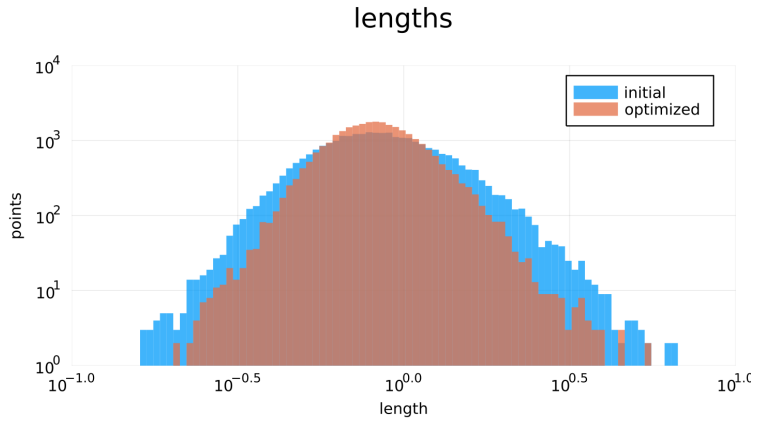


(b)

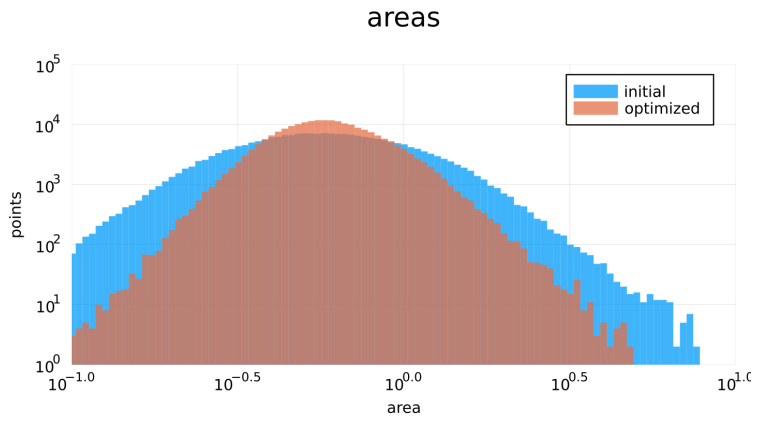


(c)

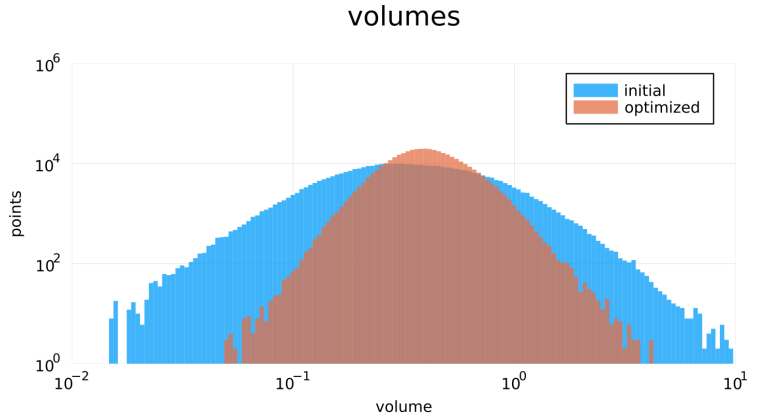
Figure 2.14: Logarithmic point-wise Riemannian area histograms for (blue) initial and (orange) optimized meshes of polynomial degree 1, 2, and 4, respectively.



(a)



(b)



(c)

Figure 2.15: Logarithmic point-wise Riemannian length, area, and volume histograms for (blue) initial and (orange) optimized quadratic tetrahedral meshes.

Table 2.4: Interpolation and approximation L^2-error of the initial meshes and the corresponding optimized meshes.

Dimension	Mesh degree	Nodes	Interpolation error		Approximation error	
			Initial	Optimized	Initial	Optimized
2	1	327	0.0494	0.0382	0.0382	0.0302
2	2	491	0.0404	0.0235	0.0314	0.0199
2	4	523	0.0980	0.0336	0.0688	0.0251
3	2	3754	0.0253	0.0121	0.0179	0.0089

measures, the tails are reduced in measure (horizontal axis) and magnitude (vertical axis). This reduction is because the quality measure is sensitive to points with volume far from the unit. Hence, these regions gain priority during the optimization process. On the other hand, the head range is increased. This increase is so because the global optimization of the squared quality measure tends to homogenize the points near a mean. Meanwhile, the measure and magnitude are almost preserved.

2.5.2.4 Interpolation and approximation error: curved high-order mesh matching the metric

To measure how a mesh $\mathcal{M}$ supports the approximation of the function u, we consider two error indicators (Brenner et al., 2008): the interpolation and the approximation errors. For briefness, we restrict to the $L^2(\Omega)$-norm error for a given domain Ω. On the one hand, the interpolation error e_I is defined by

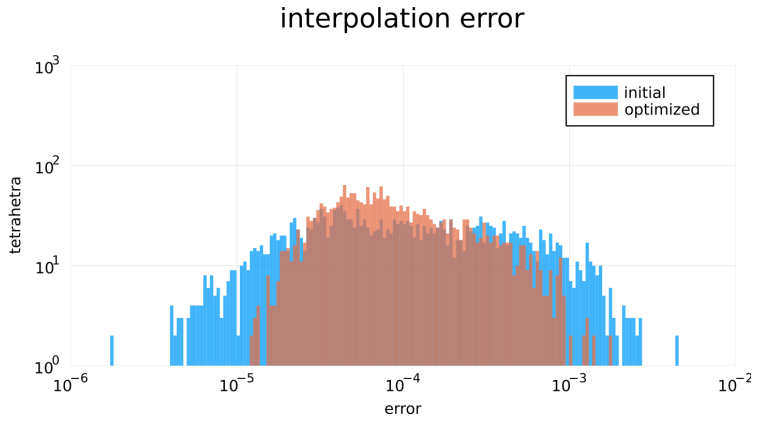
$$e_I = \|u - \Pi_{\mathcal{M}} u\|_{L^2(\Omega)},$$

where $\Pi_{\mathcal{M}}$ is the continuous mesh interpolation operator. It projects a function u to an interpolative basis with the nodal distribution detailed in Warburton (2006). On the other hand, we consider the approximation error e_A in the continuous Galerkin finite element space $V_{\mathcal{M}}$ defined by

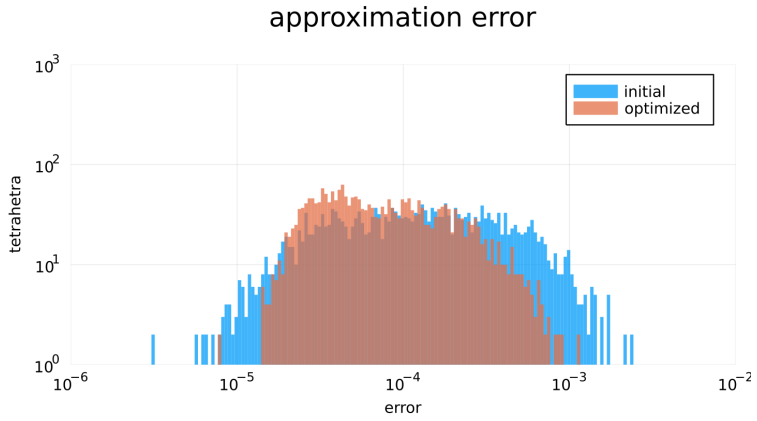
$$e_A = \min_{v \in V_{\mathcal{M}}} \|u - v\|_{L^2(\Omega)}.$$

Note that, since the interpolated function belongs to the finite element space, that is $\Pi_{\mathcal{M}} u$ in $V_{\mathcal{M}}$, the approximation error is less or equal than the interpolation error, *i.e.*, $e_A \leq e_I$.

In Table 2.4, we show the global interpolation and approximation error of the initial and optimized triangular and tetrahedral meshes of Section 2.5.2.2. We observe



(a)



(b)

Figure 2.16: Logarithmic distribution for the elemental interpolation and approximation error histograms for (blue) initial and (orange) optimized quadratic tetrahedral meshes.

that all quantities are improved. Similarly to the quality and geometry measures, a greater improvement is achieved for the high-order cases. In particular, we observe that the quartic triangular mesh is the one featuring the worst interpolation and approximation error. This is because the initial quartic mesh has low quality elements. Accordingly, the approximation of the function for the optimized mesh is limited by the initial mesh quality. In addition, the approximation error is less than the interpolation one. This is so because the best approximation approximates better the analytic function than the interpolated one.

In Figure 2.16, we illustrate the distribution of the elemental interpolation and approximation error for the initial and optimized quadratic tetrahedral meshes. On the one hand, the tails are reduced in measure (horizontal axis) and magnitude (vertical axis). This reduction shows that the maximum and minimum elemental error become closer in the optimized mesh than in the initial one. This also illustrates a reduced standard deviation for the optimized mesh. On the other hand, the head range is increased. Moreover, this head range is slightly translated to the left. This illustrates that the optimized mesh enables a more concentrated and reduced mean error than the initial mesh. From these observations, we conclude that the optimized mesh enables improved error statistics when compared to the initial one.

2.5.3 Size-shape distortion minimization for quartic interpolation: isotropic and anisotropic initial straight-edged meshes

In the following example, we apply the size-shape distortion minimization for quartic interpolation to isotropic and anisotropic initial straight-edged meshes. For this, we consider the function u of Equation (2.24) with $\gamma = 100$. We generate a background mesh $\hat{\mathcal{M}}$ and two initial physical meshes $\mathcal{M}$ of polynomial degree $q = 4$, and size $h = 0.1$. Specifically, the isotropic and anisotropic physical meshes are composed of 1923 and 1917 nodes and of 231 and 257 elements, respectively. We show the physical meshes in Figure 2.17, where they are colored according to the point-wise size-shape quality measure of Equation (2.15). On the one hand, we generate an initial isotropic mesh, see Figure 2.17(a). In this case, the initial physical mesh $\mathcal{M}$ and the background mesh $\hat{\mathcal{M}}$ coincide. We observe that almost all elements are of low quality. This is because the element stretching, alignment, or sizing does not match with the metric. As expected, the lowest quality elements lie in the sharp transition region. On the other hand, we generate an initial anisotropic mesh according to the discrete metric $\hat{\mathbf{M}}$ of the input function u, see Figure 2.17(b). We observe that almost all elements are of medium quality. In addition, the straight-edged elements approximate the curved transition region. Finally, the corresponding optimized meshes $\mathcal{M}^*$ are shown in Figures 2.17(c) and 2.17(d). We observe that, in both cases, the elements are accumulated and match the metric stretching, alignment, and sizing at the sharp transition region.

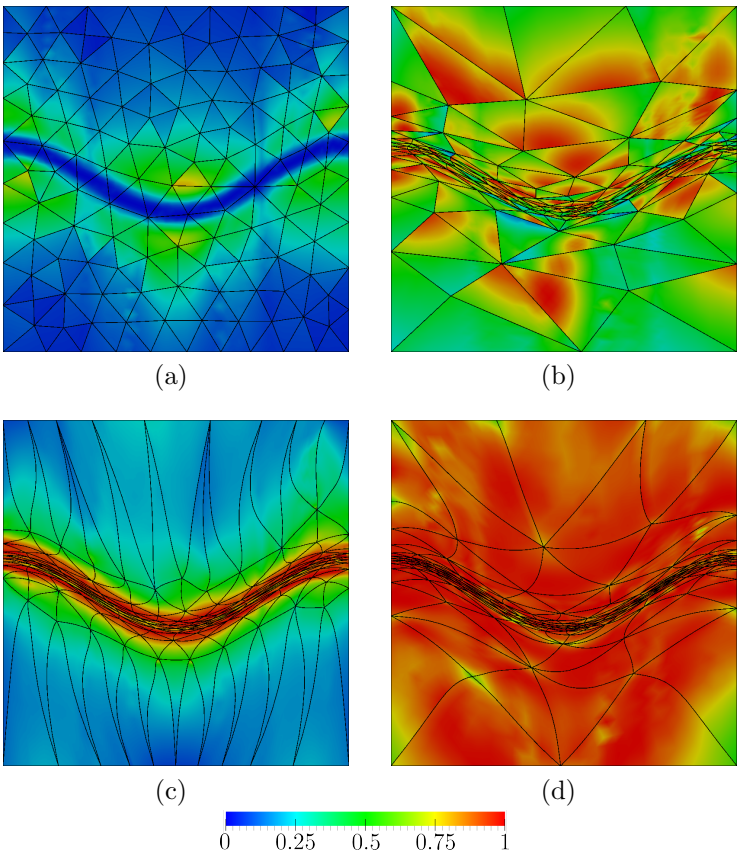


Figure 2.17: Point-wise size-shape quality for (columns) initial isotropic and anisotropic straight-edged meshes, and (rows) initial and optimized quartic meshes.

Table 2.5: Size-shape quality and geometry statistics of the initial isotropic and optimized quartic meshes.

Measure	Minimum		Maximum		Mean		Standard deviation	
	Initial	Optimized	Initial	Optimized	Initial	Optimized	Initial	Optimized
Quality	0.0189	0.1765	0.6413	0.9819	0.1613	0.7328	0.1149	0.2504
Length	0.1232	0.1462	9.1069	2.9094	0.9304	1.0989	1.5793	0.4197
Area	0.0219	0.3318	15.7890	1.8669	0.8658	0.8657	2.3650	0.3310

Table 2.6: Size-shape quality and geometry statistics of the initial anisotropic and optimized quartic meshes.

Measure	Minimum		Maximum		Mean		Standard deviation	
	Initial	Optimized	Initial	Optimized	Initial	Optimized	Initial	Optimized
Quality	0.1959	0.8063	0.9270	0.9955	0.6454	0.9288	0.1578	0.0384
Length	0.4171	0.5999	2.5246	1.6854	1.0921	1.0101	0.3823	0.1914
Area	0.2493	0.5571	2.7515	1.4499	0.8620	0.8620	0.4149	0.1775

Table 2.7: Interpolation and approximation L^2 error of the initial isotropic and anisotropic quartic meshes and the corresponding optimized meshes.

Initial Mesh	Nodes	Interpolation error		Approximation error	
		Initial	Optimized	Initial	Optimized
Isotropic	1923	0.1573	0.0029	0.1111	0.0022
Anisotropic	1917	0.0138	0.0031	0.0095	0.0024

In Tables 2.5 and 2.6, we show the statistics for elemental qualities (Equation (2.16)) and Riemannian lengths and areas. They allow us to compare between the initial and optimized meshes in terms of the target metric. We observe that the maximum, minimum, mean, and standard deviation become closer to unit values in almost all cases. That is, in general, all statistics are improved. As expected, we observe a greater improvement for the initial isotropic mesh.

The presented example shows how our method can be used to improve the error of a straight-edged mesh. In Table 2.7, we present the global interpolation and approximation error of the initial and optimized meshes. First, we observe that the approximation error is less than the interpolation one. This is because the approximation error compares the analytic function with its best approximation in the continuous finite element space, see Section 2.5.2.4. Since the best approximation approximates better the analytic function than the interpolated one, the approximation error is less than the interpolation one. Second, we observe that all quantities are improved for the optimized meshes. In particular, they are improved by almost two orders of magnitude for the initial isotropic mesh and by almost one order of magnitude for the initial anisotropic mesh. This is because the initial anisotropic mesh approximates the metric better than the initial isotropic one. Finally, the errors of the optimized meshes corresponding to the initial isotropic mesh and the initial anisotropic one are of the same order of magnitude. This phenomenon illustrates the potential of curved r-adaption.

In addition, the presented example shows the capability of curved elements to capture sharp curved transition regions. For the initial anisotropic mesh, we observe that, even if the straight-edged elements approximate the curved transition region, this is not sufficient. Only when we curve them, we gain one order of magnitude for the interpolation and approximation error.

In Figure 2.18, we show the point-wise L^2 approximation error. For the initial isotropic mesh, we observe that the error increases as we approximate to the sharp

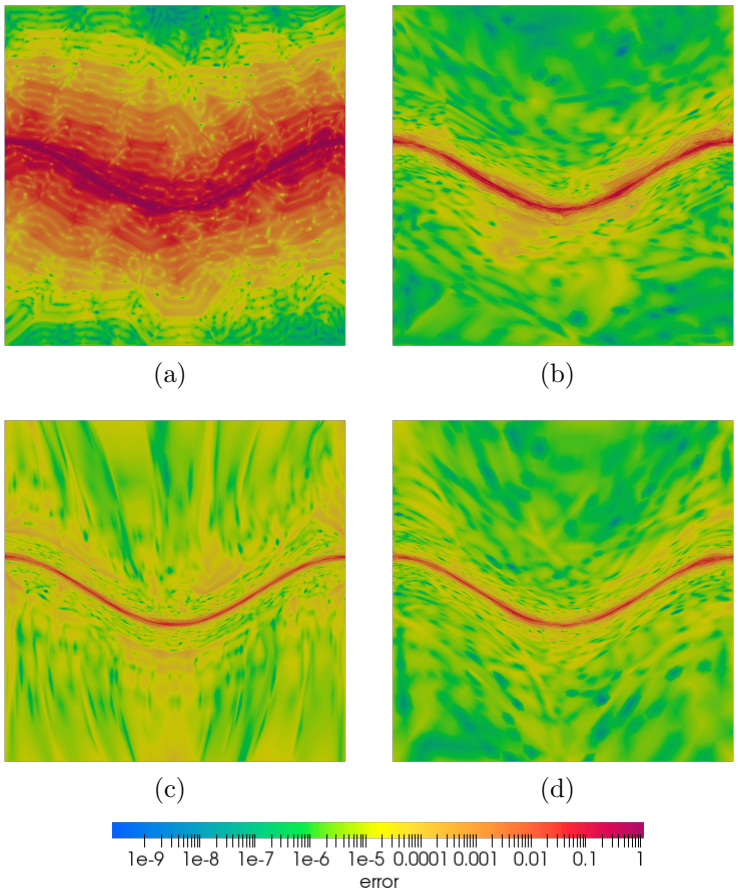


Figure 2.18: Point-wise error between the function u and its best $L^2(\Omega)$ approximation $u_{\mathcal{M}}$ for (columns) initial isotropic and anisotropic straight-edged, and (rows) initial and optimized quartic meshes.

transition region. This is because the isotropic elements cannot represent the sharp transition of the function. Then, in the optimized mesh, we observe that the error is localized at the sharp transition region in a smaller magnitude compared to the initial mesh. This is because the elements are stretched and aligned to match the sharp curved transition region. For the initial anisotropic mesh, we observe that the error is localized at the sharp transition region only. This is because the mesh has been previously adapted to match, with straight-edged elements, the sharp transition region. In the optimized mesh, we observe that this error fits the curved sharp transition region in a slightly smaller magnitude.

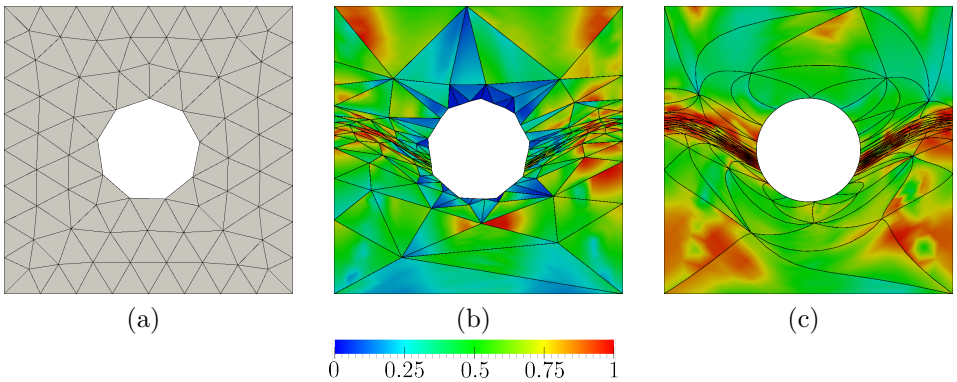


Figure 2.19: Background, initial adapted straight-edged, and optimized cubic meshes. Initial and optimized meshes are colored with the point-wise size-shape quality measure.

2.5.4 Size-shape distortion minimization with curved boundary for cubic interpolation: anisotropic initial straight-edged mesh

In the following example, we apply the size-shape distortion minimization with curved boundary for cubic interpolation to an anisotropic initial straight-edged mesh. For this, we consider the function u of Equation (2.24) with $\gamma = 100$ over the square domain with a circular hole Ω. Specifically, we denote the domain by $\Omega = K \backslash C$, where $K = [-0.5, 0.5]^2$ is a square, and where C is the circle with radius equal to 0.18 and centered at the origin. The domain Ω has two boundaries, the one of the square K and the one of the circle C. Although the inner boundary is smooth, the outer boundary contains sharp features such as corners.

In Figure 2.19, we show the background $\left(\hat{\mathcal{M}}\right)$ and physical ($\mathcal{M}$, $\mathcal{M}^*$ meshes, where the physical meshes are colored according to the point-wise size-shape quality measure of Equation (2.15). The background mesh $\hat{\mathcal{M}}$ is of polynomial degree $q = 3$, and size $h \approx 0.042$, see Figure 2.19(a). In particular, it is composed of 87 vertices, 660 nodes, and 133 triangles. From this background mesh $\hat{\mathcal{M}}$, we generate a physical cubic mesh $\mathcal{M}$ according to the input discrete metric $\hat{\mathbf{M}}$ and preserving the background mesh boundary $\partial\hat{\mathcal{M}}$. Specifically, in order to obtain an output MMG mesh, we uniformly subdivide the background mesh and we evaluate the fourth derivatives of u, $\nabla^4 u$, at the subdivided background mesh vertices. In this case, the input MMG linear mesh $\hat{\mathcal{M}}'$, which is different from the high-order background mesh $\hat{\mathcal{M}}$, is composed of

Table 2.8: Size-shape quality and geometry statistics of the initial adapted straight-edged and optimized cubic meshes.

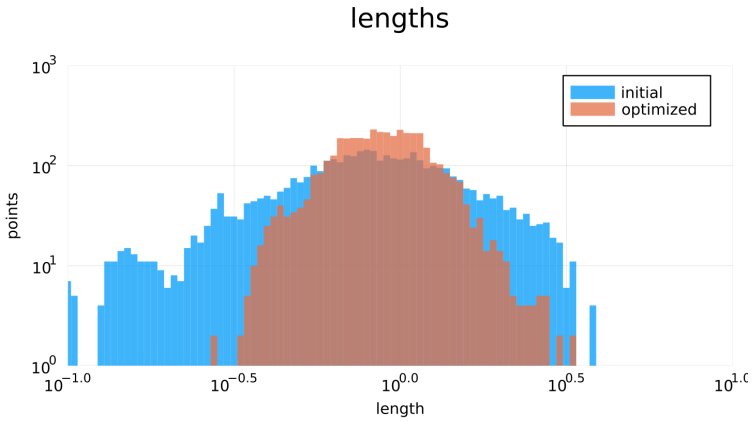
Measure	Minimum		Maximum		Mean		Standard deviation	
	Initial	Optimized	Initial	Optimized	Initial	Optimized	Initial	Optimized
Quality	0.0161	0.3549	0.9292	0.9864	0.4667	0.7804	0.2312	0.1554
Length	0.1022	0.3714	2.7706	2.1434	0.9635	0.9310	0.5344	0.2760
Area	0.0148	0.2235	3.0501	1.2100	0.6696	0.6428	0.5215	0.1758

660 vertices-nodes and 1197 triangles. Then, we obtain the discrete metric $\hat{\mathbf{M}}$ from the derivatives $\nabla^4 u$ by applying the log-simplex algorithm (Coulaud and Loseille, 2016a). The output MMG mesh is an adapted straight-edged physical mesh $\mathcal{M}$ composed of 951 nodes and 191 triangles, see Figure 2.19(b). We observe that almost all elements are of medium quality. In addition, the straight-edged elements approximate the curved transition region. Finally, we show the corresponding optimized mesh $\mathcal{M}^*$ in Figure 2.19(c). We observe that the elements are accumulated and match the metric stretching, alignment, and sizing at the sharp transition region.

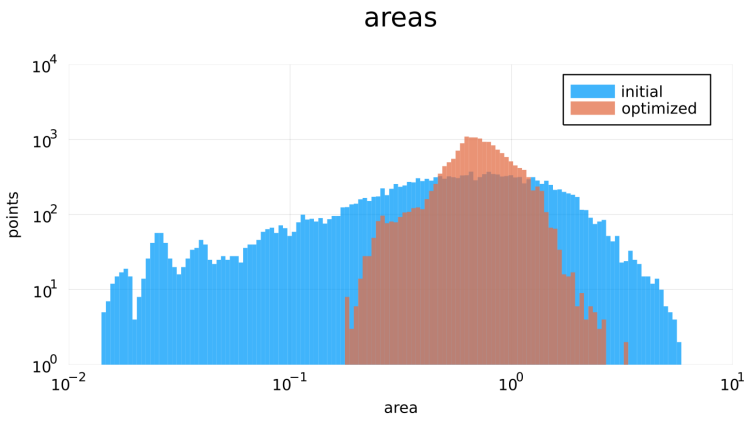
In Table 2.8, we show the statistics for elemental qualities (Equation (2.16)) and Riemannian lengths and areas. The table allow us to compare between the initial and optimized meshes in terms of the target metric. We observe that the maximum, minimum, mean, and standard deviation become closer to unit values in almost all cases. That is, in general, all statistics are improved.

In Figure 2.20, we show the point-wise Riemannian length and area of the initial and optimized triangular cubic meshes. As in Section 2.5.2.3, we use a logarithmic scale to illustrate the different scales of the corresponding ratios. The logarithmic representation illustrates the behavior near the minimum, maximum, and geometric mean of the distribution.

From the results, we observe that, when compared with straight-edged elements, curved elements approximate more faithfully the metric while preserving the curved features of the boundary. In this case, the stretching direction is almost aligned according to the tangent of the geometry. When considering straight-edged elements, in Figure 2.19(b), accumulating more degrees of freedom in the stretched regions may worsen the boundary representation at non-stretched regions. Moreover, this accumulation leads to triangles with small area near the boundary, see the area histogram in Figure 2.20(b). In contrast, when considering curved elements, in Figure 2.19(c), we observe that a single curved element represents the boundary more faithfully than several straight-edged elements. This flexibility of curved elements allows



(a)



(b)

Figure 2.20: Logarithmic point-wise length, and area histograms for (blue) initial and (orange) optimized cubic triangular meshes.

the degrees of freedom to slide and accumulate, from non-stretched regions to the stretched regions, featuring high-quality elements. In addition, those small elements initially generated near the boundary are enlarged according to the metric size and to the domain boundary, see the area histogram in Figure 2.20(b). For that reason, we observe how the elements are stretched, aligned, sized, and curved according to the stretching, alignment, and sizing of the metric. Hence, curved elements allow an improved representation of the metric while preserving the curved features of the boundary.

From the reasoning presented in Section 2.4.2, we observe that almost all measure

Table 2.9: Interpolation and approximation L^2 error of the initial adapted straight-edged and optimized cubic meshes.

Mesh	Interpolation error	Approximation error
Initial	0.0197	0.0144
Optimized	0.0058	0.0046

statistics are improved for the optimized meshes. On the one hand, for the geometric measures, the tails are reduced in measure (horizontal axis) and magnitude (vertical axis). This reduction is because the quality measure is sensitive to points with volume far from the unit. Hence, these regions gain priority during the optimization process. On the other hand, the head range is increased. This increase is so because the global optimization of the squared quality measure tends to homogenize the points near a mean. Meanwhile, the measure and magnitude are almost preserved.

The presented example shows how our method can be used to improve the error of a straight-edged mesh according to a curved boundary $\partial\Omega$. In Table 2.9, we present the global interpolation and approximation error of the initial and optimized mesh. As before, we observe that the approximation error is less than the interpolation one. This is because the approximation error compares the analytic function with its best approximation in the continuous finite element space, see Section 2.5.2.4. Since the best approximation approximates better the analytic function than the interpolated one, the approximation error is less than the interpolation one. On the other hand, we observe that the errors are improved three times for the optimized mesh. This is because the optimized mesh approximates better the metric of the function, reducing the function numerical error.

In addition, the presented example shows the capability of curved elements to capture sharp curved transition regions with curved boundaries. We observe that, even if the straight-edged elements approximate the curved transition region, this is not sufficient. Only when we curve them, we reduce the interpolation and approximation error.

2.6 Conclusions

The defined distortion measure is applied to curve straight-edged meshes to improve the node configuration according to the desired metric. To perform the distortion

minimization we use the framework for high-order optimization presented in Aparicio-Estrems et al. (2019). The numerical examples show optimized meshes with an improved stretching, alignment, and sizing according to the metric. This improvement leads in all cases to an increase of the minimum element mesh quality and a reduction of the standard deviation between the different element qualities.

To independently measure whether the optimized mesh matches the input metric, we propose point-wise Riemannian measures of the mesh entities equipped with the metric. These are the Riemannian edge length, surface area, and cell volume. The results show that the optimized meshes improve the length, area, and volume distributions in the metric sense. This illustrates that the distortion minimization enables meshes that effectively match the input metric.

To illustrate the potential applications of the method, we also measure the numerical error for an input function. These are the interpolation and approximation errors of the function matched by the mesh. The results show that the optimized meshes reduce both the interpolation and approximation errors. Moreover, our particular example illustrates that the distortion minimization reduces the numerical errors by one order of magnitude for an initial adapted mesh and by two orders of magnitude for an initial isotropic mesh. In addition, we apply the distortion minimization for domains with curved boundaries. The results show that the mesh approximates the stretching, alignment, and sizing of the discrete metric while preserving the curved features of the boundary model.

Chapter 3

A globalized and preconditioned Newton-CG solver

3.1 Introduction

To enhance global convergence of Newton's direction, globalization strategies modify Newton's direction to a feasible step. Standard globalization methods are divided into those using either trust-region (TR) (Nocedal and Wright, 2006; Conn et al., 2000; Bulteau and Vial, 1985) or line-search (LS) globalization (Nocedal and Wright, 2006). On the one hand, trust-region methods consistently deal with negative-curvature steps and direction candidates on subspaces. To this end, standard TR methods enable a step-length continuation by only evaluating the objective function, and they do it by promoting not only a sufficient decrease but also a sufficient progress criterion. For this, standard TR methods consider a predictor model, comparing the non-linear behavior of the objective function with a quadratic model in terms of the step size (Conn et al., 2000). However, it is unclear how to choose the initial trust-region radius in terms of the current mesh size. On the other hand, we prefer the simplicity of a backtracking line-search (BLS) strategy for a first implementation trial. Specifically, a standard BLS globalization considers the Newton direction reduced by a step-length factor using a sufficient decrease criterion (Nocedal and Wright, 2006).

To compute Newton's direction in large second-order optimization problems, it is standard to use an inexact Newton method with a conjugate gradient (CG) method (Diachin et al., 2006; Sastry and Shontz, 2009, 2012), using constant residual toler-

ance, and Jacobi preconditioning (Bertaccini and Durastante, 2018).

The preference for the CG method is based on three factors. First, the CG method is specific for symmetric and positive-definite matrices. This design is relevant near a minimum, where the symmetric Hessian of the objective function is also positive-definite. Second, its short-recurrence property allows computing a solution without requiring additional memory. Third, its negative-curvature termination condition is helpful in line-search strategies (Nocedal and Wright, 2006).

In iterative linear solvers, it is standard to set a constant tolerance threshold for the residual norm as a stopping criterion to control the accuracy. Furthermore, specifically for the CG method, one can consider a curvature tolerance threshold as a stopping criterion. The choice of these tolerance parameters impacts the accuracy and number of iterations of the iterative method and hence, on the evolution and computational cost of the nonlinear solver.

For a given constant tolerance, preconditioning techniques reduce the total number of matrix-vector products while preserving a comparable number of non-linear iterations. The total number of sparse matrix-vector products indicates the computational cost of inexact Newton solvers (Bertaccini and Durastante, 2018). In Newton-CG methods, this number corresponds to the total number of CG iterations.

3.1.1 Challenges

Unfortunately, for metric-aware curved high-order mesh optimization, standard globalized and preconditioned Newton-CG solvers have robustness and efficiency issues. In curved high-order metric-aware mesh optimization, we observe that these issues are triggered by non-uniform sizing, stretching ratios, and curved alignment. When more remarkable these characteristics are, more difficult the convergence with a general-purpose optimization solver. First, in each non-linear step, highly non-uniform mesh gradation stiffens the validity of the mesh deformations and the corresponding linear systems. Second, for high stretching ratios, the deformations in some directions are locally stiffer than in other directions. Third, curved alignment requires curved high-order elements. For these elements, when higher is the order, stiffer is the corresponding linear system.

The previous mesh characteristics challenge the global convergence of the non-linear solver and the solution of the corresponding linear systems. Specifically, progressing towards convergence, standard solvers might present three main issues:

- They might need additional backtracking line-search iterations because they do not continuously ensure sufficient decrease and progress. A standard BLS globalization reduces the Newton direction by a step-length factor using a sufficient decrease criterion (Nocedal and Wright, 2006). However, the step length is restarted at each non-linear iteration, impeding its continuous evolution during the optimization. Moreover, BLS does not promote sufficient progress.

- They might accumulate additional iterations of the linear solver because they use constant linear solver tolerance. Constant tolerances do not correctly predict the accuracy of a descent direction for a highly non-linear and non-convex objective function. Thus, they might require excessive precision far from the optima or feature insufficient accuracy to promote quadratic convergence near an optimum (Eisenstat and Walker, 1996; Dembo and Steihaug, 1983).

- They might obtain inaccurate steps because the preconditioner is inaccurate or numerically singular. Jacobi preconditioners favor a low computational cost instead of an accurate approximation of the Hessian matrix. This loss of accuracy in solving Newton's equation compromises the computational cost of the solver near an optimum, where quadratic convergence must be prioritized. Incomplete Cholesky factorization favors accurate Hessian approximation instead of low computational cost. However, it might lead to singular preconditioning when the Hessian is numerically singular.

It is critical to devise a solver to alleviate these issues because, without such a solver, it might be impossible to demonstrate the potential advantages of curved high-order optimization for high polynomial degrees, especially when the target metric features non-uniform size, high stretching, and curved alignment. The implementation of the resulting high-order mesh optimization solver can be later accelerated using fast GPU implementations (Camier et al., 2023).

3.1.2 Aim and contribution

We aim to alleviate the issues of standard solvers for metric-aware curved high-order mesh optimization. To this end, we propose a specific-purpose globalized and preconditioned Newton-CG solver. To devise the solver, we propose three main contributions:

- To continuously ensure sufficient decrease and progress for the LS globalization, we uniquely combine a step-length predictor featuring not only reduction but also amplification of the step length. This line search features memory and continuation of the step length while favoring the quadratic convergence of the Newton method.

- To reduce the number of iterations of the linear solver, we propose new dynamic forcing sequences that control the residual tolerance and sufficient positive curvature. Specifically, we propose a new forcing sequence for the residual. This sequence is suited to limit the number of CG iterations at the beginning of the optimization process and allow the necessary CG iterations to obtain a quadratic convergence rate near an optimum. To emulate steps with sufficient positive curvature, we also propose to define the normalized curvature of a given direction and a new dynamic forcing sequence for the curvature of the CG step. We define this sequence to limit CG-iterations when the Hessian is near to positive semidefinite without breaking the quadratic convergence rate near an optimum.

- To avoid numerically singular linear pre-conditioning, we propose three ingredients for our pre-conditioner. The first ingredient is a novel predictor that switches between the Jacobi pre-conditioner and the *root-free incomplete Cholesky factorization* (iLDL$^{\mathrm{T}}$(0)) with zero levels of fill-in (Bertaccini and Durastante, 2018). This switch uses a parameter indicating the acceptable numerical ill-conditioning of the factorization. The second is a new inequality accounting for the curvature of the resulting direction computed from the CG method. If the direction violates the curvature inequality, we consider that the computation of the used pre-conditioner is numerically unstable. The third ingredient reorders the unknowns used to compute the factorization. Several results presented in the literature indicate that the ordering of a matrix impacts the numerical instability of its factorization (Bertaccini and Durastante, 2018). To control this instability, we propose to use an ordering that tries to minimize the discarded fill of the incomplete factorization (D'Azevedo et al., 1992; Persson and Peraire, 2008). We also propose to reorder the mesh nodes according to the first nonzero eigenvalue of a metric-aware Laplacian spectral problem with Neumann boundary conditions.

Finally, to measure the performance of our specific-purpose solver in metric-aware curved high-order mesh optimization, we compare it with a standard solver. For the solver ingredients, we also compare the standard and specific-purpose approaches. To perform these comparisons, we measure the number of iterations for the non-linear loop and line-search globalization. In addition, we compare the total number of matrix-vector products. The results allow us to describe the influence of each ingredient on the proposed specific-purpose non-linear solver.

The remainder of this chapter is organized as follows. In Section 3.2, we introduce the r-adaption problem, the distortion minimization formulation, and an optimization overview. In Section 3.3, we present the standard and specific-purpose line-search globalizations for Newton's method. In Section 3.4, we present the standard and specific-purpose linear solvers for the inexact Newton method. In Section 3.5, we present a set of examples to compare both the standard and specific-purpose implementations. Finally, in Section 3.6, we present the main conclusions.

3.2 The problem: r-adaption, formulation, and optimization overview

We aim to propose a robust specific-purpose solver for the piece-wise polynomial mesh r-adaption problem. In this adaption problem, the input is a domain, equipped with a metric, and meshed with a piece-wise polynomial mesh, see Section 3.2.1 for a model case. We want to relocate the node coordinates of the input mesh, without modifying the topology, to obtain an output mesh that matches the stretching and alignment prescribed by the given metric. To this end, we can minimize the mesh distortion measure proposed in Aparicio-Estrems et al. (2018), see also Chapter 2, with the corresponding free node coordinates as design variables, see Section 3.2.2. Unfortunately, we have observed that the existent optimization solvers equipped with standard globalization strategies, see an overview in Section 3.2.3, might fail to drive an initial mesh to a local distortion minimum when the initial mesh highly mismatches the stretching and alignment of the given metric, especially when higher are the polynomial degrees, stretching ratios, and curvature of the alignment features. We seek a new robust and globalized minimization solver that overcomes these issues.

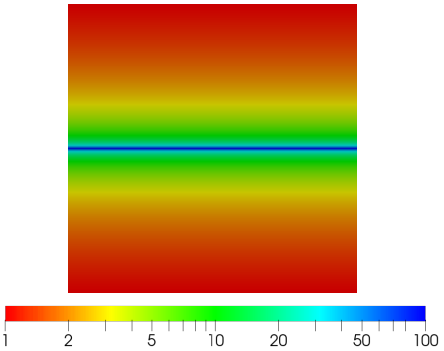


Figure 3.1: Unit square equipped with a metric matching a boundary layer: stretching ratio in logarithmic scale.

3.2.1 Curved high-order r-adaption: model case

To illustrate the r-adaption problem and to test the globalized minimization solvers considered through this work, we use a model case. In this model case, we consider the quadrilateral domain $\Omega = [-0.5, 0.5]^2$, equipped with a metric matching a boundary layer, and meshed with isotropic straight-sided triangular meshes of different polynomial degree but with the same resolution.

The boundary layer aligns with the x-axis, requires a constant unit element size along the x-direction, and a non-constant element size along the y-direction. This vertical element size grows linearly with the distance to the x-axis, with a factor $\gamma = 2$, and starts with the minimal value $h_{\min} = 10^{-2}$. Thus, as illustrated in Figure 3.1, between $y = -0.5$ and $y = 0.5$ the stretching ratio blends from $1 : 100$ to $1 : 1$. To match the boundary layer, we define the metric as:

$$\mathbf{D} = \begin{pmatrix} 1 & 0 \\ 0 & 1/h(y)^2 \end{pmatrix},$$

where $h(x) = h_{\min} + \gamma|x|$.

The meshes are of polynomial degree 1, 2, 4, and 8, and since they have the same resolution, they are composed of the same number of nodes, 481 nodes, but a different number of elements, 896, 224, 56, and 14 elements, respectively. In Figures 3.2(a), 3.2(b), 3.2(c), and 3.2(d) we show these meshes colored according to the pointwise stretching and alignment quality measure, proposed in Aparicio-Estrems et al. (2018) which will be detailed in Section 3.2.2. Points in blue color have low quality and points with red color have high quality. As we observe, the elements lying in the region of highest stretching ratio have less quality than the elements lying in

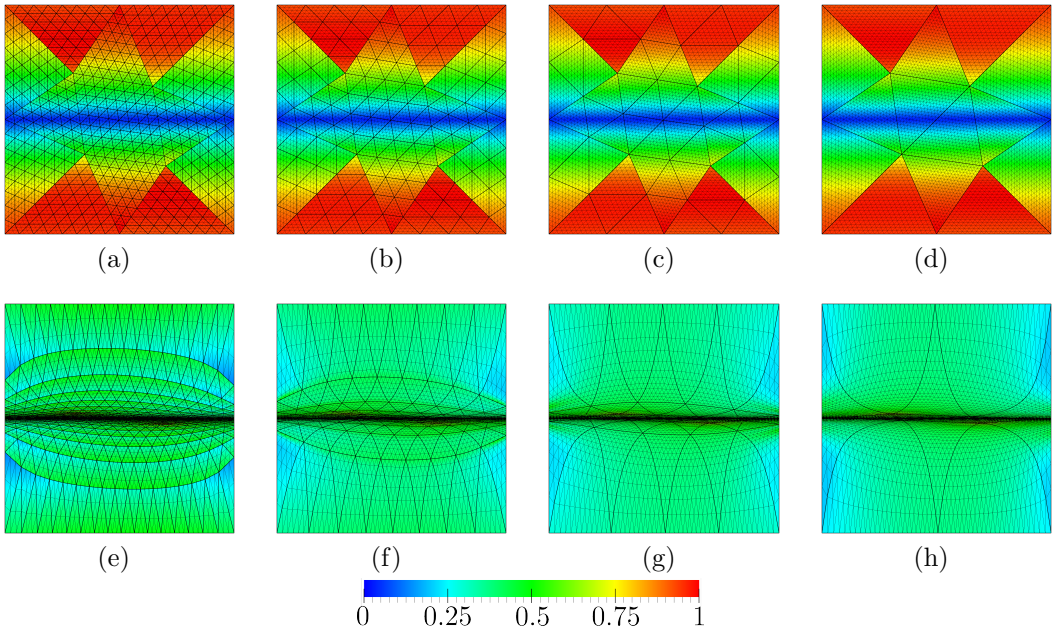


Figure 3.2: Triangular meshes of polynomial degree 1, 2, 4, and 8 in columns. Initial straight-sided isotropic meshes and optimized meshes from initial meshes in rows. These element vertices are for a visualization purpose, they are not the high-order degrees of freedom.

the isotropic region. This is because the generated meshes are almost isotropic and, when we equip them with the metric $\mathbf{D}$, the mesh quality measures a high deviation between the point-wise stretching and alignment of the mesh and the one of the metric near the region $y = 0$.

The node coordinates of the isotropic mesh may be far from the configuration satisfying the stretching and alignment of the metric. Furthermore, the stretching and alignment of the metric might be impossible to be fulfilled depending on the initial generated mesh. In our case, we look for an optimal configuration, which may not be unique, that approximates the stretching and alignment of the metric.

To obtain an optimal configuration, we minimize the distortion measure proposed by changing the coordinates of all the mesh nodes and preserving their connectivity. This can be done by considering all mesh node coordinates targeting a representation of the boundary, in Chapter 4, or by restricting the boundary mesh nodes to slide over the geometric boundary (Aparicio-Estrems et al., 2018). Herein, we consider that the coordinates of the inner nodes, and the one-dimensional coordinates of the inner nodes of the boundary segments, are the design variables. Thus, the inner nodes

are free to move, the vertex nodes are fixed, while the rest of boundary nodes are enforced to slide along the boundary segments.

The optimized meshes are illustrated in Figures 3.2(e), 3.2(f), 3.2(g), and 3.2(h). We observe that the elements away from the anisotropic region are enlarged vertically whereas the elements lying in the anisotropic region are compressed. In the optimized mesh, the minimum quality is improved and the standard deviation of the element qualities is reduced when compared with the initial configuration.

3.2.2 The minimization formulation: metric-aware distortion measure and free nodes

To match the stretching and alignment of a given metric, we relocate the nodes by minimizing the mesh distortion proposed in Aparicio-Estrems et al. (2018) with the corresponding free node coordinates as design variables. Following we summarize the definitions of the metric-aware point-wise, element, and mesh distortion, and we then state the minimization problem.

To define and compute the distortion of a piece-wise polynomial mesh $\mathcal{M}$ that approximates a domain $\Omega \subset \mathbb{R}^d$ equipped with an input metric $\mathbf{M}$, we need mappings between three elements: the master, the equilateral, and the physical. The master $E^M \subset \mathbb{R}^d$ is the element from which the iso-parametric mapping is defined. The equilateral (regular) element $E^\triangle$ is characterized by the element having unitary edge lengths. The physical element $E^P \in \mathcal{M}$ is the element to be measured. The respective mappings $\boldsymbol{\phi}_\triangle : E^M \to E^\triangle$, $\boldsymbol{\phi}_P : E^M \to E^P$ between the equilateral and the physical elements through the master element are obtained. The mapping $\boldsymbol{\phi}_\triangle(\boldsymbol{\xi})$ between the master element and the equilateral element depends only on a parameter $\boldsymbol{\xi} \in E^M$ while the mapping $\boldsymbol{\phi}_P(\boldsymbol{\xi}; \mathbf{x}_e)$ between the master and the physical element $e \in \mathcal{M}$ depends both on the parameter $\boldsymbol{\xi}$ and the corresponding physical element nodes $\mathbf{x}_e$.

Then, we define the point-wise distortion measure η of the physical element $e \in \mathcal{M}$ at a point $\mathbf{u}$ as

$$\eta(\mathbf{u}; \mathbf{x}_e) := \frac{\operatorname{tr}\left(\mathbf{A}(\mathbf{u}; \mathbf{x}_e)^{\mathrm{T}}\, \mathbf{M}(\boldsymbol{\phi}_P(\boldsymbol{\phi}_\triangle^{-1}(\mathbf{u}); \mathbf{x}_e))\, \mathbf{A}(\mathbf{u}; \mathbf{x}_e)\right)}{d\, \sigma_0^{2/d}}, \tag{3.1}$$

where $\mathbf{A}(\mathbf{u}; \mathbf{x}_e) := \mathbf{D}\boldsymbol{\phi}_P(\boldsymbol{\phi}_\triangle^{-1}(\mathbf{u}); \mathbf{x}_e)\ \left(\mathbf{D}\boldsymbol{\phi}_\triangle(\mathbf{u})\right)^{-1}$, for $\mathbf{u} \in E^\triangle$, and where

$$\sigma_0 := \frac{1}{2}(\sigma + |\sigma|), \quad \sigma := \det\left(\mathbf{A}(\mathbf{u}; \mathbf{x}_e)\right) \sqrt{\det \mathbf{M}(\boldsymbol{\phi}_P(\boldsymbol{\phi}_\triangle^{-1}(\mathbf{u}); \mathbf{x}_e))}.$$

We define the elemental distortion measure η_e of the physical element $e \in \mathcal{M}$ as the L^1 mean over the equilateral element $E^{\triangle}$

$$\eta_e := \frac{\left\| \eta\left(\cdot \, ; \mathbf{x}_e \right) \right\|_{L^1(E^{\triangle})}}{\|1\|_{L^1(E^{\triangle})}}. \tag{3.2}$$

Then, for the mesh $\mathcal{M}$ with nodes $\mathbf{x}_i \in \mathbb{R}^d,\ i = 1, ..., k$, and equipped with an input metric $\mathbf{M}$, we define the functional that measures the distortion by

$$F(\mathbf{x}_1, ..., \mathbf{x}_k) := \sum_{e \in \mathcal{M}} \left\| \eta\left(\cdot \, ; \mathbf{x}_{e,1}, ..., \mathbf{x}_{e,n_p} \right) \right\|^2_{L^2(E^{\triangle})}, \tag{3.3}$$

where we denote the coordinates of the n_p element nodes by $\mathbf{x}_e = \left(\mathbf{x}_{e,1}, ..., \mathbf{x}_{e,n_p} \right)$, and each pair (e, j) in $\mathbf{x}_{e,j}$ identifies the local j-th node of element e with their global mesh number i. That is, for nodal high-order elements is equivalent to determining the configuration of the nodes of the high-order mesh. Moreover, the element contribution to the objective function only depends on the nodes of that element.

For the optimization of the function F, each interior node is able to move in $\mathbb{R}^d$ and only the normal components of the mesh nodes of the boundary are fixed. Hence, the variables are composed of all the components of the interior nodes and the tangential components of the boundary nodes. We denote the vector containing all the n variable components by $x \in \mathbb{R}^n$, and since the other components are fixed we can define $f(x) := F(\mathbf{x}_1, ..., \mathbf{x}_k)$ with $f : \mathbb{R}^n \to \mathbb{R}$. Then, the optimization of the mesh distortion leads to an optimal mesh $\mathcal{M}^*$, where the nodes set $(\mathbf{x}_1^*, ..., \mathbf{x}_k^*)$ is determined by including the fixed node components to the optimal solution x^*. Note that this problem corresponds to an unconstrained minimization problem, and thus, we can solve it using the standard minimization and globalization techniques over-viewed in the following section.

3.2.3 Optimization overview

We have casted our adaption problem to an unconstrained minimization problem. To solve the problem, we first recall essential unconstrained optimization concepts, conditions, and notation, to finally detail an optimization algorithm.

Let us consider the unconstrained minimization of a non-linear smooth function $f : \mathbb{R}^n \to \mathbb{R}$:

$$x^* = \text{argmin}_{x \in \mathbb{R}^n} f(x),$$

with gradient and Hessian denoted by ∇f and $\mathrm{H}f$, respectively.

To decide which points are candidates or local minimizers, we can derive first and second-order conditions from f (Nocedal and Wright, 2006). To derive these conditions, we consider a point $x \in \mathbb{R}^n$ and a sufficiently small step $s \in \mathbb{R}^n$ to obtain two local approximations of $f(x+s)$ from Taylor's theorem. These two approximations lead to the first and second-order conditions of the minimization problem, respectively. On the one hand, an approximation of first order in s, *linear model*, can be computed as

$$f(x+s) \approx f(x) + s^{\mathrm{T}} \nabla f(x). \tag{3.4}$$

The linear model leads to the first-order necessary conditions. That is, if x^* is a local minimizer of f then

$$\nabla f(x^*) = 0.$$

We refer to x^* as a *stationary point* if it fulfills the latter condition. On the other hand, a second-order approximation in s, *quadratic model*, can be computed as

$$f(x+s) \approx f(x) + s^{\mathrm{T}} \nabla f(x) + \frac{1}{2} s^{\mathrm{T}} \mathrm{H}f(x) s. \tag{3.5}$$

The quadratic model leads to the second-order sufficient conditions. That is, if

$$\nabla f(x^*) = 0 \quad \text{and} \quad \mathrm{H}f(x^*) \quad \text{is positive definite,}$$

then x^* is a strict local minimizer of f, see a proof in Nocedal and Wright (2006). Note that second-order sufficient conditions are not necessary. For instance, there are functions with strict local minimizers where the Hessian matrix vanishes.

Accordingly, to minimize f, given an initial point x_0, we seek a sequence of nonlinear iterates $\{x_k\}$ that has to converge to a stationary point x^*,

$$\lim_{x_k \to x^*} \|\nabla f(x_k)\| = 0,$$

expecting to find a local minimizer. We terminate the sequence when either no more progress can be made or when it seems that a solution point has been approximated with sufficient accuracy, *e.g.*, when the residual $\|\nabla f(x_k)\|$ is less to a fixed tolerance. In practice, the sequence is obtained by iteratively computing, from the current point x_k, a step s_k that determines a next point $x_{k+1} = x_k + s_k$ with a lower value of f, that is, $f(x_{k+1}) < f(x_k)$. To ensure a sufficient decrease of the objective function and the convergence to either a stationary point or even to a local minimizer, it is standard to compute the step s_k using a *globalization* strategy, see Section 3.3.

We consider globalization strategies that start from a given search direction p_k. This search direction can be obtained either from the linear or the quadratic model, and ideally it should lead to a decrease of the objective function. To this end, the direction is required to be a *descent direction*, that is,

$$p_k^{\mathrm{T}} \nabla f(x_k) < 0. \tag{3.6}$$

The search direction that locally produces the greatest decrease in the linear model, Equation (3.4), is the *steepest descent* direction given by

$$p_k = -\nabla f(x_k).$$

However, for non-linear functions, the steepest-descent direction might not provide a sufficient decrease of f. For instance, this is the case near those minimum where the function is locally quadratic.

In this region, we can derive from the quadratic model, Equation (3.5), a direction with a quadratic rate of local convergence, the *Newton direction* (Nocedal and Wright, 2006). This direction satisfies the *Newton Equation* given by the linear system of equations

$$\mathrm{H}f(x_k) p_k = -\nabla f(x_k). \tag{3.7}$$

The corresponding Newton direction is a descent direction, see Equation (3.6), whenever the Hessian is positive definite.

To enforce the search direction fulfills the descent property, we might need to switch to the opposite of the Newton direction. This is so since when the Hessian is non-singular but non-positive definite, the Newton direction is defined but it might violate the descent condition in Equation (3.6). In this case, a practical choice for p_k is the Newton direction times the sign of its scalar product with the steepest-descent direction. We call this direction the *signed Newton direction*.

A general second-order optimization solver, incorporating the previous concepts, obtains a step s_k by applying a globalization strategy to a numerical approximation of the Newton direction p_k, see Algorithm 3.1. This algorithm corresponds to the general scheme for the standard solvers we aim to modify to obtain our specific-purpose solver. The inputs are the objective function, its gradient, its Hessian, an initial guess, and the linear solver choice: direct or iterative. Note that, for the iterative solver, we need to detail the choice of a preconditioner. The output is a configuration expected to be at least locally optimal up to an input tolerance. First,

Algorithm 3.1 Second-order optimization

Input: $f,\ \nabla f,\ \mathrm{H}f,\ x,$ solver, preconditioner
Output: x^*

```
1:  procedure OptimizeFunction
2:      stop ← false
3:      k ← 0
4:      α ← 1
5:      s ← 0
6:      g ← ∇f(x),  H ← Hf(x)
7:      g_0 ← g,  H_0 ← H
8:      η ← 0.5,  τ ← 0.01
9:      while stop is false do
10:         p ← NewtonDirection(g, H, x, s, solver, preconditioner, η, τ)
11:         s,  α ← GlobalizationLS(x, p, α, f, g, H)
12:         if k = 0 then
13:             s_0 ← s
14:         end if
15:         x ← x + s
16:         g ← ∇f(x),  H ← Hf(x)
17:         η,  τ ← ForcingSequences (x, s_0, g_0, H_0, s, g, H)
18:         stop ← StoppingCondition(∇f, x, s, k)
19:         k ← k + 1
20:     end while
21:     x* ← x
22: end procedure
```

we setup the parameters, Lines 2-8. In particular, we set the stopping criterion, Line 2, the initial non-linear iteration, Line 3, the initial step length, Line 4, and the initial step, Line 5. In addition, we set the initial values for the dynamic estimators, Line 8, considered in the specific-purpose optimization solvers. Second, we perform the non-linear iteration, Line 9. Specifically, we compute the exact/inexact approximation of the Newton direction, Line 10. Then, we obtain the corresponding step via a line-search globalization, Line 11. Following, we obtain the new point by applying the step, Line 15. Next, we evaluate the gradient and the Hessian of the objective function, Line 16. They are used in the next non-linear iteration and, for the specific-purpose case, to update the dynamic estimators, Line 17. Then, we check the stopping criterion, Line 18. Finally, we upgrade the current non-linear iteration, Line 19. Once the loop stopped we set the output point as the obtained one, Line 21.

3.3 Line-search globalization: standard and specific-purpose strategies

To propose a robust specific-purpose optimization solver for the r-adaption problem, Section 3.2.1, a specific-purpose globalization strategy is critical. To obtain such a strategy, we propose to improve the standard line-search globalization. To this end, in Section 3.3.1, we first review the standard backtracking line-search strategy (Nocedal and Wright, 2006). Then, in Section 3.3.2, we detail the proposed modification. Our contribution is to propose a linear predictor model and a new procedure for the computation of the step length.

3.3.1 Standard backtracking line search: sufficient decrease

The *backtracking* line-search (BLS) strategy is a systematic approach to promote global convergence in a non-linear solver. It is a specific line-search globalization strategy. These strategies minimize a function over a sequence of search paths reducing a multi-variable problem into a one-dimensional problem. A basic line-search strategy consists in computing a suitable *step length* α_k for a given descent direction p_k, see Equation (3.6), and determining the step s_k as:

$$s_k = \alpha_k p_k.$$

In order to obtain a sufficient decrease of f, such step length should satisfy the *Armijo condition* (Nocedal and Wright, 2006)

$$f(x_k + s_k) < f(x_k) + c s_k^{\mathrm{T}} \nabla f(x_k), \tag{3.8}$$

where c is a constant in $(0, 1)$. In its most basic form, a backtracking line-search strategy proceeds by reducing the step length until the Armijo condition is satisfied.

To enforce that successive reduction of the step length leads to a sufficient decrease, Equation (3.8), it is preferred to use a small constant like $c = 10^{-4}$, see Nocedal and Wright (2006). The constant c of the Armijo condition controls the balance between the decrease of the objective function and the step-length condition. A large constant admits only those step lengths providing a large decrease of the objective function. Accordingly, the desired decrease might not be achieved by reducing the step length monotonically, and hence, one needs an advanced search

Algorithm 3.2 Standard BLS

Input: $x_k,\ p_k,\ \alpha_k,\ f,\ g_k$
Output: $s_k,\ \alpha_{k+1}$
Set: $c = 10^{-4}$, $\gamma = 2$, $\alpha_{\min} = 2^{-20}$

1: **procedure** `StandardGlobalizationBLS`
2: $s_k \leftarrow \alpha_k p_k$
3: **while** $f(x_k + s_k) > f(x_k) + cs_k^T g_k$ and $\alpha_k > \alpha_{\min}$ **do**
4: $\alpha_k \leftarrow \alpha_k / \gamma$
5: $s_k \leftarrow \alpha_k p_k$
6: **end while**
7: $\alpha_{k+1} \leftarrow 1$
8: **end procedure**

strategy to set a valid step length. On the contrary, a small constant admits the step lengths providing a small decrease, and thus, we can use the simple successive reduction strategy to set the step length.

In Algorithm 3.2, we detail a standard BLS strategy with constants $c = 10^{-4}$, $\gamma = 2$, and $\alpha_{\min} = 2^{-20}$ such as presented in Nocedal and Wright (2006). The algorithm inputs are: the point x_k, the descent direction p_k, the step length α_k, the objective function f, and the value of the gradient of f at x_k, $g_k := \nabla f(x_k)$. The algorithm outputs are: the new point x_{k+1}, the step s_k, and the next initial value of the step length α_{k+1}. The step length α_k is divided by a factor $\gamma > 1$ iteratively until it satisfies the Armijo condition, Equation (3.8), and while the factor $\alpha > \alpha_{\min}$, Line 3. Finally, the standard BLS strategy restarts the next initial value of the step length to one, Line 7.

3.3.2 Specific-purpose line search: prediction and continuation of the step length

To propose a line-search strategy that promotes sufficient decrease and progress, we detail two main ingredients. First, we consider a predictor that indicates if a step length is either large or small. Second, taking into account the predictor, we propose to promote sufficient decrease and progress by either reducing or amplifying the step length. Finally, we combine these ingredients to propose a line-search algorithm featuring memory and continuation of the step length while favoring quadratic convergence of Newton method.

3.3.2.1 Step-length predictor: indicating large and small step length

As in the standard strategy presented in Algorithm 3.2, we consider a step length determined by the Armijo condition. However, instead of using the standard inequality presented in Equation (3.8), we propose to use the linear model of the objective function

$$\phi(s;x) := f(x) + s^{\mathrm{T}}\nabla f(x).$$

Note that the step s is a descent direction, see Equation (3.6), if and only if $\phi(s;x) < \phi(0;x)$.

Analogously to the standard trust-region formulation presented in Conn et al. (2000), for each step s and for the model ϕ, we can define a predictor given by

$$\rho(s;x) := \frac{f(x) - f(x+s)}{\phi(0;x) - \phi(s;x)},$$

where the model ϕ is linear in our line-search strategy, while it is quadratic in trust-region strategies.

The predictor serves as an indicator of the quality of the step length of a descent direction. For a given descent direction, $\phi(s;x) < \phi(0;x)$, the predictor can be either non-positive or positive. When $\rho(s;x) \leq 0$ it indicates that the step does not provide a decrease of the objective function, that is $f(x+s) \geq f(x)$. When $\rho(s;x) > 0$, there is a decrease in the objective function, and thus, ρ indicates the quality of the step length. On the one hand, a low value of the predictor, $\rho(s;x) \approx 0$, indicates a step length far away from those neighborhoods where the function behaves as the linear model. This negligible decrease indicates a large step length. On the other hand, a high value of the predictor, $\rho(s;x) \approx 1$, means that the objective function behaves as the linear model. This linear behavior indicates a small step length.

3.3.2.2 Promoting sufficient decrease and progress: reducing and amplifying step length

We propose to control the step length according to the value of the predictor. We aim to promote a step length that provides a sufficient decrease of the objective function and that is sufficiently large so that the objective function is not in the linear regime. Heuristically, if we reduce the step length we expect to increase the value of the predictor. On the contrary, if we amplify the step length we expect to decrease the value of the predictor.

We can control the sufficient decrease of the objective function in terms of the predictor. This is so since the Armijo condition of Equation (3.8) is equivalent to the bound $\rho(s;x) > c_{\min}$. In particular, for a descent step s, the condition $f(x+s) < f(x) + c_{\min} s^T \nabla f(x)$ is equivalent to

$$\rho(s;x) = \frac{f(x) - f(x+s)}{-s^T \nabla f(x)} > c_{\min}. \tag{3.9}$$

Even if the decrease of the objective function is reasonable, the step might be too short. The successive reduction of the step length might not ensure reasonable progress. To address this issue, it is standard to use line-search globalizations accounting for the Wolfe conditions (Nocedal and Wright, 2006). Herein, we propose an alternative adequate to our problem. In contrast to the existent line-search strategies for the Wolfe conditions, our methodology does not require additional evaluations of the gradient of the objective function at the line-search iterations.

To promote sufficient progress, we propose to amplify the step length iteratively that is, $\alpha_k \leftarrow \gamma \alpha_k$. The amplification of the step length leads to a greater decrease of the objective function. However, it might also reduce the value of the predictor. To avoid an excessive reduction of the predictor value, which might violate the Armijo condition of Equation (3.9), we propose a stopping criterion for the amplifying iterations.

Our criterion stops the amplifying iterations whenever the predictor indicates that the step-length quality exceeds a threshold. Specifically, we stop when $\rho(\gamma s_k; x_k) < c_{\max}$ for a given constant $c_{\max}$. By choosing $c_{\max} \geq c_{\min}$ we ensure that the Armijo condition is satisfied for the step s_k at each amplifying iteration. It may happen that amplifying the step length does not decrease the objective function monotonically, not fulfilling the goal of the line-search iteration. To address this issue in the amplifying iteration, we propose to add the condition $f(x_k + \gamma s_k) < f(x_k + s_k)$. This condition enforces to decrease the objective function.

3.3.2.3 Specific-purpose line-search algorithm

The main objective of the specific-purpose LS, Algorithm 3.3, is to perform a continuation of the step length. This continuation is expected to generate a smooth sequence of non-linear iterations. The inputs and the outputs of Algorithm 3.3 are the same as the ones of Algorithm 3.2. The constants $c_{\min} = 10^{-4}$, $\gamma = 2$, and $\alpha_{\min} = 2^{-20}$ correspond to standard values (Nocedal and Wright, 2006). In addition, we propose to set

Algorithm 3.3 Specific-purpose LS

Input: $x_k,\ p_k,\ \alpha_k,\ f,\ g_k$
Output: $s_k,\ \alpha_{k+1}$
Set: $c_{\min} = 10^{-4}\ ,\ c_{\max} = 0.25,\ \gamma = 2,\ \alpha_{\min} = 2^{-20}$

```
1:  procedure Specific-purposeGlobalizationLS
2:      s_k ← α_k p_k
3:      φ(0; x_k) ← f(x_k)
4:      φ(s_k; x_k) ← f(x_k) + s_k^T g_k
5:      ρ(s_k; x_k) ← (f(x_k) − f(x_k + s_k)) / (φ(0; x_k) − φ(s_k; x_k))
6:      if ρ(s_k; x_k) < c_min then
7:          while ρ(s_k; x_k) < c_min and α_k > α_min do
8:              α_k ← α_k/γ
9:              s_k ← α_k p_k
10:         end while
11:     else
12:         while ρ(γ s_k; x_k) > c_max and f(x_k + γ s_k) < f(x_k + s_k) do
13:             α_k ← γ α_k
14:             s_k ← α_k p_k
15:         end while
16:     end if
17:     if ρ(s_k; x_k) < c_max then
18:         α_{k+1} ← α_k/γ
19:     else
20:         α_{k+1} ← α_k
21:     end if
22: end procedure
```

the new constant $c_{\max} = 0.25$ to favor quadratic convergence of Newton method near the optimum without additional line-search iterations, see the reasoning in Appendix A.2. The algorithm starts, Lines 2-5, setting up the main variables and functions: step, current model, model for the step, and predictor.

The algorithm continues by deciding to either reduce or amplify the step length, Lines 6-16. If the sufficient-decrease condition is violated, we decide to reduce the step length, Line 6. Otherwise, we decide to amplify the step length, Line 11. Then, we proceed to the line-search iteration. The reduction iterations are the ones of the standard BLS, Lines 6-10. In contrast, we improve the standard BLS, Lines 11-21, by enlarging and updating the step length α_k and the step s_k, respectively.

First, we decide to amplify the step length while the sufficient-progress condition is violated and the additional decrease of the objective function is fulfilled, Line 12.

We remark that if no amplifying iterations are performed, the input step length for the current direction is preserved. Finally, we update the step length, Lines 17-21. These instructions provide a step-length memory to the specific-purpose strategy instead of restarting with a step length equal to one in the standard strategy, Line 7 of Algorithm 3.2. In particular, we only update the step length by reducing it whenever it has not sufficient quality, Line 17. We consider this update to prevent an additional reduction iteration at the next non-linear iteration, as it is proposed for trust-region methods (Conn et al., 2000).

3.4 Newton-CG solvers: standard and specific-purpose methods

After proposing the specific-purpose globalization in Section 3.3, we aim to improve the performance of the non-linear optimization method. For this, in this section, we present the standard inexact Newton method, with the standard residual and curvature tolerances and the standard preconditioner. Then, we present the specific-purpose inexact Newton method, with specific-purpose residual and curvature tolerances and a specific-purpose preconditioner.

3.4.1 Standard Newton-CG method

Next, we present the standard features of the inexact Newton method. These are the residual and curvature forcing sequences and the preconditioner. Then, we combine them to obtain a numerical approximation of the Newton direction.

3.4.1.1 Existing residual and curvature forcing sequences

In an inexact Newton optimization process, the residual and curvature tolerances of the CG method are given by the so-called forcing sequences and forcing terms (Eisenstat and Walker, 1996; Dembo and Steihaug, 1983). In this section, we present an existing choice of these two estimators. On the one hand, residual forcing terms are presented in Eisenstat and Walker (1996); Dembo and Steihaug (1983); Nash and Sofer (1990). They are proposed to avoid *oversolving* the linear system of Newton Equation (3.7). On the other hand, a standard constant curvature forcing term is

Algorithm 3.4 Standard Forcing Sequences
Input: $x,\ s_0,\ g_0,\ H_0,\ s,\ g,\ H$
Output: $\eta,\ \tau$
1: **procedure** `ForcingSequences`
2: $\eta = 10^{-9},\ \tau = 0$
3: **end procedure**

presented for the CG method in Dembo and Steihaug (1983). It is proposed to limit the total amount of CG iterations.

The first estimator is the forcing sequence for the residual r_k of the iterative method. Specifically, it is denoted by η and it is used as a stopping criterion for the iterative method through the following expression

$$\|r_k\| < \eta \|\nabla f(x_k)\|.$$

In practice, it is standard to set $\eta = 10^{-9}$, in order to achieve a desirable accuracy, see Algorithm 3.4.

In contrast, dynamic forcing sequences $\{\eta_k\}$ for the residual have been proposed in the literature (Eisenstat and Walker, 1996; Dembo and Steihaug, 1983; Nash and Sofer, 1990). Specifically, the stopping criterion for the iterative method is now given by

$$\|r_k\| < \eta_k \|\nabla f(x_k)\|. \tag{3.10}$$

The choice of η_k have been reported to be critical to the efficiency of the inexact Newton method (Eisenstat and Walker, 1996).

Referring to curvature forcing sequences, a constant estimator for the sufficient positive curvature of the CG-step d_k is presented in the literature (Dembo and Steihaug, 1983). Specifically, it is denoted by ϵ and it is used as a stopping criterion for the iterative method by the following expression

$$d_k^{\mathrm{T}} \mathrm{H} f(x_k) d_k < \epsilon\ d_k^{\mathrm{T}} d_k. \tag{3.11}$$

It is standard to set $\epsilon = 0$ to avoid negative curvature directions in the next CG iterations, see Algorithm 3.4.

3.4.1.2 Standard Preconditioner

In addition to the forcing sequences, the use of a preconditioner constitutes an important ingredient to improve the efficiency and accuracy of the CG method. Specifically,

Algorithm 3.5 Standard Numerical Approximation of Newton Direction

Input: g, H, σ, x, solver, preconditioner, η, τ
Output: p

```
1: procedure NewtonDirection
2:     switch solver do
3:         case direct
4:             p ← −H\g
5:         case iterative
6:             preconfun(r) ← diag(H)\r
7:             ϵ ← τ
8:             p ← CG(H, −g, 0, preconfun, n, η, ϵ)
9:     p ← sign(−g^T p) p
10: end procedure
```

when the initial guess is far from a minimizer, the diagonal preconditioner is a cheap but sufficient approximation of the Hessian to obtain a desirable inexact approximation of the Newton direction.

3.4.1.3 Standard Numerical Approximation of Newton Direction

The standard inexact Newton method is summarized in terms of the standard forcing sequences and the standard preconditioner presented in Sections 3.4.1.1 and 3.4.1.2, respectively. This procedure is used to determine the descent direction, Line 10, for the optimization method, Algorithm 3.1.

In Algorithm 3.5, we present the numerical approximation of the Newton direction. The inputs are the gradient $g = \nabla f(x)$, the Hessian $H = \mathrm{H}f(x)$, the MDF ordering of the n unknowns for the initial Hessian $\sigma = \mathrm{MDF}\left(\mathrm{H}f(x_0)\right)$, the current point x, the solver type (iterative), and the preconditioner function. It is standard to set the parameters $\eta = 10^{-9}$ and $\tau = 0$, see Algorithm 3.4. The output is a descent direction. First, we decide which solver is used to compute the Newton approximation: direct for an exact Newton approximation, Line 3, and iterative for an inexact Newton approximation, Line 5. The exact Newton approximation is computed using a sparse LU factorization, Line 4. To compute the inexact Newton approximation, we consider the diagonal of the Hessian as a preconditioner, Line 6. Then we apply the preconditioned CG algorithm with null initial guess, Line 8. Finally, we obtain a descent direction by correcting its sign according to the steepest-descent direction, Line 9.

Algorithm 3.6 Specific-purpose Forcing Sequences

Input: $x,\ s_0,\ g_0,\ H_0,\ s,\ g,\ H$
Output: $\eta,\ \tau$
Set: $\eta_{\max} = 0.5,\ \tau_{\max} = 0.01$

procedure `ForcingSequences`
  $Z_0 \leftarrow \text{GramSchmidt}(-g_0, s_0)$
  $\tilde{g}_0 \leftarrow Z_0^{\mathrm{T}} g_0,\ \tilde{H}_0 \leftarrow Z_0^{\mathrm{T}} H_0 Z_0$
  $\tilde{q}_0 \leftarrow -\tilde{H}_0 \backslash \tilde{g}_0$
  $\tilde{\kappa}_0 \leftarrow \tilde{q}_0^{\mathrm{T}} \tilde{H}_0 \tilde{q}_0 / \tilde{q}_0^{\mathrm{T}} \tilde{q}_0$
  $Z \leftarrow \text{GramSchmidt}(-g, s)$
  $\tilde{g} \leftarrow Z^{\mathrm{T}} g, \quad \tilde{H} \leftarrow Z^{\mathrm{T}} H Z$
  $\tilde{q} \leftarrow -\tilde{H} \backslash \tilde{g}$
  $\tilde{\kappa} \leftarrow \tilde{q}^{\mathrm{T}} \tilde{H} \tilde{q} / \tilde{q}^{\mathrm{T}} \tilde{q}$
  $\eta \leftarrow \frac{\|\tilde{q}\|}{\|s_0\|}, \quad \tau \leftarrow \frac{|\tilde{\kappa}|}{|\tilde{\kappa}_0|}$
  $\eta \leftarrow \min(\eta, \eta_{\max})$
  $\tau \leftarrow \min(\tau, \tau_{\max})$
  $\tau \leftarrow \min(\tau, \eta)$
end procedure

3.4.2 Specific-purpose Newton-CG method

In what follows we present the specific-purpose Newton-CG method. For this, we first detail the specific-purpose residual and curvature forcing sequences and then specific-purpose preconditioner. Finally, we combine them to obtain a specific-purpose numerical approximation of the Newton direction.

3.4.2.1 Specific-purpose residual and curvature forcing sequences

The main disadvantage of the standard forcing sequences is the failure of accuracy prediction for inexact Newton approximations. On the one hand, constant forcing sequences keep the accuracy fixed. This is unpractical because the additional accuracy required near an optimum may require, at the same time, an unnecessary computational cost at the first iterations, far from that optimum. On the other hand, the dynamic forcing sequences for the residual presented in Section 3.4.1.1 predict the accuracy in terms of a scaled variation between the objective function and the linear model (Eisenstat and Walker, 1996). We have observed that, even if they predict a better accuracy than the fixed sequences, they do not predict a desirable accuracy in our specific problem.

Next, we present the specific-purpose dynamic forcing sequences for the CG method. For this, we use two additional inexact approximations of the Newton direction: the *restricted Newton direction* (based in a subspace restriction concept presented in Bulteau and Vial (1985)) and the *incomplete Newton direction*. Finally, the residual and curvature forcing terms are obtained in terms of these approximations and the corresponding forcing sequences.

Our restricted Newton direction is given by the Newton Equation

$$H_k q_k = -g_k, \tag{3.12}$$

restricted to the subspace $W_k := \text{span}\{-g_k, s_{k-1}\}$ generated by the steepest-descent direction $-g_k$ and the last step $s_{k-1} = x_k - x_{k-1}$, and where $g_k := \nabla f(x_k)$ and $H_k := \text{H}f(x_k)$. In particular, we consider the Gram-Schmidt orthonormalization procedure to the ordered basis $\{-g_k, s_{k-1}\}$. This results in an orthonormal basis Z_k of the subspace W_k, where the columns of Z_k are the vectors forming the basis. From this basis, we define the projection of the gradient and the Hessian onto the subspace W_k as

$$\tilde{g}_k := Z_k^{\text{T}} g_k, \quad \tilde{H}_k := Z_k^{\text{T}} H_k Z_k.$$

Then, in the restricted form, Equation (3.12) reduces to the two-dimensional linear system

$$\tilde{H}_k \tilde{q}_k = -\tilde{g}_k,$$

and the restricted Newton direction is given by the pre-projected direction $Z_k \tilde{q}_k$. Then, we define the forcing sequences by

$$\eta_k := \frac{\|\tilde{q}_k\|}{\|s_0\|}, \quad \tau_k := \frac{|\tilde{\kappa}_k|}{|\tilde{\kappa}_0|}, \tag{3.13}$$

where $\tilde{\kappa}_k := \kappa\left(Z_k \tilde{q}_k; x_k\right)$ is the normalized curvature of the restricted Newton direction, see Appendix A.3. The sequence η_k is used for the stopping criterion presented in Equation (3.10). In addition, the sequence τ_k is used for the stopping criterion presented in Equation (3.11). Specifically, we set the curvature forcing term $\epsilon_k = \tau_k |\kappa_k|$, where κ_k is the curvature of the Newton direction. To compute κ_k, we observe that, from Equation (3.12), we have

$$\kappa_k := \frac{q_k^{\text{T}} H_k q_k}{q_k^{\text{T}} q_k} = \frac{q_k^{\text{T}} \left(-g_k\right)}{q_k^{\text{T}} q_k}. \tag{3.14}$$

Instead of computing a solution q_k of Equation (3.12), we compute an incomplete approximation $\hat{q}_k$ of q_k using the chosen preconditioner, denoted as M_k, and defining $\hat{q}_k$ by

$$M_k \hat{q}_k = -g_k.$$

We call $\hat{q}_k$ the incomplete Newton direction. Then, using Equation (3.14) and the equation presented above, we approximate the κ_k as follows

$$\kappa_k \approx \mu := \frac{\hat{q}_k^{\mathrm{T}} (-g_k)}{\hat{q}_k^{\mathrm{T}} \hat{q}_k}. \tag{3.15}$$

Finally, we approximate the curvature forcing term as follows $\epsilon_k \approx \tau_k |\mu|$.

It is standard to apply safeguards to the forcing sequences (Dembo and Steihaug, 1983; Eisenstat and Walker, 1996). Similarly, we observe that by choosing a safeguard for the forcing sequences presented in Equation (3.13), we can improve the inexact Newton method. One for the residual forcing sequence η_k given by $\eta_k \leftarrow \min(\eta_k, \eta_{\max})$, we set $\eta_{\max} = 0.5$. The other for the curvature forcing sequence τ_k given by $\tau_k \leftarrow \min(\tau_k, \tau_{\max}, \eta_k)$ with $\tau_{\max} = 0.01$. This value is set to avoid an excessive influence of the curvature forcing sequence at the initial non-linear iterations.

For our optimization problem, we propose a new forcing sequence for the residual which is suited to limit the number CG-iterations at the beginning of the optimization process and allowing the necessary CG-iterations to obtain a quadratic convergence rate near an optimum. On the other hand, we propose to define the normalized curvature of a given direction and a new dynamic forcing sequence for the curvature of the CG-step to emulate CG-steps with sufficient positive curvature. We define this sequence to limit CG-iterations when the Hessian is near to positive semi-definite without breaking the quadratic convergence rate near an optimum.

In our problem, the main advantage of the specific-purpose forcing sequences is to efficiently predict a desirable accuracy of the inexact Newton approximation at each stage of the optimization process that is, far and near an optimum. This is because they are based in a cheap but faithful approximation of the Newton direction. Specifically, this approximation is obtained by restricting the Newton equation in a subspace spanned by the steepest-descent direction and the step of the last non-linear iteration. Consequently, the forcing sequences predict a decrease of accuracy at the first iterations, obtaining steps approximating the steepest-descent direction.

In addition, they predict an increase of accuracy near an optimum, obtaining steps approximating the Newton direction, in order to preserve second-order convergence.

3.4.2.2 Specific-purpose preconditioner

In addition to the forcing sequences presented before, the choice of the preconditioner impacts on the efficiency of the iterative method. We remark that a more accurate preconditioner, sensitive to the magnitude of the entries of the Hessian matrix, can be numerically unstable for an ill-conditioned matrix. For this reason, we propose three procedures to reduce both the numerical instabilities and its potential impact in the non-linear optimization process. In this section, we define the preconditioner and then, we present its numerical instability issues together with the three procedures to mitigate them. Then, we present the linear solver obtained from the modifications presented in this section and in Section 3.4.1.

In what follows, we present the specific-purpose preconditioner for the CG method. In addition, we control the numerical instability issues by applying three different procedures: a switch criterion between two preconditioners, a curvature inequality limitation, and an ordering that minimizes the discarded fill of the factorization.

The first procedure consists in switching between the Jacobi preconditioner and the *root-free incomplete Cholesky factorization* (iLDL$^{\mathrm{T}}$(0)) with zero levels of fill-in (Bertaccini and Durastante, 2018). This switch uses a parameter indicating the numerical instability of the factorization.

The second one is based on an inequality of the curvature of the resulting direction computed from the CG method. If the direction violates the curvature inequality, we consider that the computation of the used preconditioner is numerically unstable.

Finally, the third condition consists in the ordering of the unknowns used to compute the factorization. Several results presented in the literature indicate that the ordering of a matrix has an impact on the numerical instability of its factorization (Bertaccini and Durastante, 2018). To control this instability we propose to use an ordering that tries to minimize the discarded fill of the incomplete factorization.

When the initial guess is far from a minimizer, the minimization meets different configurations of the objective function. These configurations can be determined in terms of the Hessian. Roughly speaking, the Hessian starts at a highly indefinite configuration where the positive and negative eigenvalues have large magnitudes. Then, the magnitude of the negative eigenvalues become smaller and the Hessian tends to

be nearly singular. After this, the Hessian is positive definite and nearly singular, with small positive eigenvalues. Finally, in the convergence region, the Hessian is positive definite with no small positive eigenvalues. Between these configurations some oscillations may occur, exceptionally switching between an indefinite configuration to a positive definite one. These Hessian configurations are approximately represented in the preconditioner.

Accordingly, we propose to use the preconditioner to detect the Hessian configurations. Specifically, we expect that the diagonal matrix D of a Hessian decomposition indicates when the factorization is indefinite, positive definite, and numerically singular. To this end, in addition to the Jacobi preconditioner, Section 3.4.1.2, we consider the *root-free incomplete Cholesky factorization* (iLDL$^{\mathrm{T}}$(0)) with zero level of fill-in (Bertaccini and Durastante, 2018).

When applied to the CG method, the iLDL$^{\mathrm{T}}$(0) preconditioner provides an accurate approximation of the Newton direction. This is especially useful for points near a minimizer, where the Newton direction needs to be solved with a high level of accuracy to preserve the quadratic convergence. However, when the initial guess is far from a minimizer, the iLDL$^{\mathrm{T}}$(0) preconditioner may provide low-quality directions interfering with the evolution of the optimization process. Finally, we have observed that when the negative values of the matrix D tend to cluster, the factorization tends to be more numerically stable.

We propose to use the Jacobi preconditioner whenever the iLDL$^{\mathrm{T}}$(0) factorization is supposed to provide low quality directions. Specifically, we first obtain the iLDL$^{\mathrm{T}}$(0) preconditioner from an incomplete LU factorization with zero levels of fill-in (iLU(0)), as

$$M_k := \tilde{L} D \tilde{L}^{\mathrm{T}},$$

where the $\tilde{L}$ and D factors are given by

$$\tilde{L} = \frac{1}{2}\left(L + D \backslash U^{\mathrm{T}}\right), \quad D = \mathrm{diag}(U).$$

To guess the factorization quality, we consider the negative value with smallest magnitude, $d_{\min}$, and the negative value with largest magnitude, $d_{\max}$, of the diagonal matrix D. Then, we use the iLDL$^{\mathrm{T}}$(0) factorization whenever the quantities $d_{\min}$ and $d_{\max}$ are similar. This condition corresponds to check if their ratio is smaller than some fixed quantity. In particular, we consider that the quantities $d_{\min}$ and $d_{\max}$ are

similar when the following condition is satisfied

$$\frac{d_{\max}}{d_{\min}} < \delta, \tag{3.16}$$

for $\delta := 10$. We assume that we are near an optimum when the matrix D has no negative values and, in such case, we use the iLDL$^{\mathrm{T}}(0)$ factorization. On the contrary, when $d_{\max}/d_{\min} \geq \delta$, we will use the Jacobi preconditioner. The larger the parameter δ, more iLDL$^{\mathrm{T}}(0)$ factorizations are used instead of the Jacobi preconditioner. For ill-conditioned problems, this may cause some instability issues breaking the continuity of the optimization process by choosing consecutive steps with nearly opposite directions.

In addition to the numerical instabilities described before, we have observed that the iLDL$^{\mathrm{T}}$ preconditioner can provide low quality directions p_k. That is, directions with a low value of the predictor $\rho(\alpha_k p_k; x_k)$ and requiring too many reductions of the length step α_k. To avoid such directions, we use the Jacobi preconditioner whenever the CG method with the iLDL$^{\mathrm{T}}$ preconditioner stopped because a CG-step of negative curvature is encountered and the CG solution p_k violates the limited curvature inequality

$$\kappa\left(p_k; x_k\right) 10^{-2} \leq \tau_k |\kappa_k|, \tag{3.17}$$

where κ_k is approximated as in Equation (3.15) and τ_k is presented in Equation (3.13). In both cases, these quantities are computed using a diagonal preconditioner, $M_k = \mathrm{diag}(H_k)$.

We have observed that when an iLU type preconditioner is used (including iCHOL and iLDL$^{\mathrm{T}}$ preconditioners) the ordering of the unknowns has a major effect on the convergence of the conjugate gradients iterative method. In our case, where at a given non-linear iteration the mesh may contain highly stretched and curved elements, it is crucial to compute a high-quality preconditioner to ensure convergence of the conjugate gradients method. Furthermore, we are interested in orderings that can take into account in an automatic way both the principal directions of the anisotropy and the ordering of the elements instead of the individual unknowns, especially for high-order elements.

For anisotropic problems (D'Azevedo et al., 1992) and high-order elements (Persson and Peraire, 2008), the minimum discarded fill (MDF) method provides good convergence results. We only compute the MDF ordering at the beginning of the optimization process that is, for the initial Hessian $H_0 = \mathrm{H}f(x_0)$. We use the computed

Algorithm 3.7 Preconditioner

Input: H, σ, preconditioner
Output: preconfun
Set: $\delta = 10$

1: **procedure** `Factorize`
2: **switch** preconditioner **do**
3: **case** Jacobi
4: $M \leftarrow \mathrm{diag}(H)$
5: **case** iLDL
6: $H^\sigma \leftarrow H(\sigma, \sigma)$
7: $[L, U] \leftarrow \mathrm{iLU}(H^\sigma, 0)$
8: $D \leftarrow \mathrm{diag}(U)$
9: $\tilde{L} \leftarrow 0.5\,(L + D \backslash U)$
10: $d_{\max} \leftarrow \max_{1 \le i \le n,\ D_{ii} < 0} |D_{ii}|$
11: $d_{\min} \leftarrow \min_{1 \le i \le n,\ D_{ii} < 0} |D_{ii}|$
12: **if** $\frac{d_{\max}}{d_{\min}} \ge \delta$ **then**
13: $M \leftarrow \mathrm{diag}(H)$
14: preconditioner $\leftarrow$ Jacobi
15: **else**
16: $P = \mathrm{Id}(:, \sigma)$
17: $M \leftarrow P^T \tilde{L} D \tilde{L}^{\mathrm{T}} P$
18: preconditioner $\leftarrow$ iLDL
19: **end if**
20: preconfun$(r) = M \backslash r$
21: **end procedure**

permutation when the iLDL$^{\mathrm{T}}$ factorization of the Hessian $H_k = \mathrm{H}f(x_k)$ is computed at the non-linear iteration k and when the corresponding linear system of equations in Equation (3.12) is solved, in Lines 3 and 14 of Algorithm A.1, see Appendix A.1. We remark that this ordering is not used for the matrix-vector products.

In Algorithm 3.7, we detail the factorization of the Hessian. The inputs are the evaluated Hessian H, the MDF permutation σ, and the preconditioner choice. First, in Line 2, we switch between the Jacobi and the iLDL$^{\mathrm{T}}(0)$ preconditioner. When the iLDL$^{\mathrm{T}}(0)$ preconditioner is chosen, we first apply the permutation to the Hessian, Line 6. Then, we compute the iLDL$^{\mathrm{T}}(0)$ in terms of the iLU(0) preconditioner, Lines 7-9. Note that, it is standard to describe the iLDL$^{\mathrm{T}}(0)$ factorization of the Hessian H, Line 17, in terms of the matrix representation P of the permutation σ, Line 16, where $\mathrm{Id}(:, \sigma)$ denotes the identity matrix Id with columns arranged according to σ

(Bertaccini and Durastante, 2018). Finally, we apply the switching criterion in Lines 10-19. The output is the preconditioner function.

We propose to use as a preconditioner an incomplete, symmetric, and root-free factorization. Firstly, we have chosen an incomplete factorization as a matter of performance and storage. It is well known that computing a complete factorization of a sparse matrix produces, in general, almost dense triangular factors (Bertaccini and Durastante, 2018), leading to a more expensive matrix-vector products (if required in the factorization) and requiring more memory to store the matrix. Secondly, since the CG method requires symmetric matrices, the Cholesky factorization is more appropriate than other factorizations, such as LU. Finally, contrary to the standard Cholesky factorization, its root-free version can be computed at each non-linear iteration of the optimization process. This is because the existence of the root-free factorization does not depend on the matrix H_k being positive definite or indefinite (Bertaccini and Durastante, 2018; Kershaw, 1978).

3.4.2.3 Specific-purpose Numerical Approximation of Newton Direction

The specific-purpose inexact Newton method is summarized in terms of the specific-purpose forcing sequences and the specific-purpose preconditioner presented in Sections 3.4.2.1 and 3.4.2.2, respectively. This procedure is used to determine the descent direction, Line 10, for the optimization method, Algorithm 3.1.

In Algorithm 3.8, we summarize the updates of the inexact Newton method, presented in this section and in Section 3.4.1. The inputs are the gradient $g = \nabla f(x)$, the Hessian $H = \mathrm{H}f(x)$, the MDF ordering of the n unknowns for the initial Hessian $\sigma = \mathrm{MDF}\left(\mathrm{H}f(x_0)\right)$, the current point x, the solver type (iterative), the preconditioner function, and the value of the residual and curvature forcing sequences at the current non-linear iteration η and τ respectively, see Equation (3.13). First, in Line 6, we compute the preconditioner of the permuted matrix H_k^σ, which is the $\mathrm{iLDL}^{\mathrm{T}}(0)$ factorization or the Jacobi preconditioner depending on the criterion presented in Equation (3.16). Then, in Lines 7-9, we compute the curvature forcing term from the forcing sequence and, next, in Line 10, we compute the CG direction. The output of the algorithm is the descent direction p.

In addition, in Algorithm 3.9, we incorporate the curvature safeguard, see Equation (3.17). Specifically, in Lines 11-20, we apply the curvature limitation criterion. We first check, in Line 11, if the CG direction has negative curvature. In such case,

Algorithm 3.8 Specific-purpose Numerical Approximation of Newton Direction with standard preconditioner

Input: g, H, σ, x, solver, preconditioner, η, τ
Output: p

```
1:  procedure NewtonDirection
2:      switch solver do
3:          case direct
4:              p ← −H\g
5:          case iterative
6:              preconfun ← Factorize(H, σ, preconditioner)
7:              q̂ ← preconfun(−g)
8:              κ ← (q̂ᵀ(−g)) / (q̂ᵀq̂)
9:              ϵ ← τ|κ|
10:             p ← CG(H, −g, 0, preconfun, n, η, ϵ)
11:     p ← sign(−gᵀp) p
12: end procedure
```

we update the curvature forcing term in terms of the Jacobi preconditioner. Finally, in Line 17, if the direction violates the limited curvature inequality, we compute the CG point using the diagonal preconditioner.

3.5 Results

In this section, we compare both optimization solvers: specific-purpose versus standard. To do it so, we first present the implementation details, in Section 3.5.1. Then, in Section 3.5.2, we propose a set of r-adaption tests where the initial guess is far from a minimizer. Following, in Sections 3.5.3 and 3.5.4, we compare the specific-purpose versus the standard globalizations and linear solvers for the model case presented in Section 3.2.1. Finally, in Section 3.5.5, we compare the optimization solvers for all the r-adaption tests. They are compared in terms of the non-linear iterations, line-search iterations, and matrix-vector products. In addition, we compare both optimization solvers for an initial guess near to an optimal configuration, Section 3.5.6. This is the case of a previously h-adapted mesh according to the test metric.

Because our goal is to optimize the mesh distortion, instead of including mathematical proofs of mesh validity, we detail how we numerically enforce the positiveness of the element Jacobians. Specifically, we use a numerical valid-to-valid approach that

Algorithm 3.9 Specific-purpose Numerical Approximation of Newton Direction with iLDL$^{\mathrm{T}}(0)$ preconditioner

Input: $g,\ H,\ \sigma,\ x,$ solver, preconditioner, $\eta,\ \tau$
Output: p

procedure `NewtonDirection`
  switch solver **do**
    case direct
      $p \leftarrow -H \backslash g$
    case iterative
      $\text{preconfun}, \text{preconditioner} \leftarrow \text{Factorize}(H, \sigma, \text{preconditioner})$
      $\hat{q} \leftarrow \text{preconfun}(-g)$
      $\kappa \leftarrow \frac{\hat{q}^{\mathrm{T}}(-g)}{\hat{q}^{\mathrm{T}}\hat{q}}$
      $\epsilon \leftarrow \tau|\kappa|$
      $p \leftarrow \text{CG}(H, -g, \mathbf{0}, \text{preconfun}, n, \eta, \epsilon)$
      if $p^{\mathrm{T}}Hp < 0$ and preconditioner = iLDL **then**
        $M \leftarrow \text{diag}(H)$
        $\text{preconfun}(r) = M \backslash r$
        $\hat{q} \leftarrow \text{preconfun}(-g)$
        $\kappa \leftarrow \frac{\hat{q}^{\mathrm{T}}(-g)}{\hat{q}^{\mathrm{T}}\hat{q}}$
        $\epsilon \leftarrow \tau|\kappa|$
        if $|p^{\mathrm{T}}Hp| > 10^2\epsilon\ p^{\mathrm{T}}p$ **then**
          $p \leftarrow \text{CG}(H, -g, \mathbf{0}, \text{preconfun}, n, \eta, \epsilon)$
        end if
      end if
  $p \leftarrow \text{sign}(-g^{\mathrm{T}}p)\ p$
end procedure

uses four ingredients. First, because we want numerically valid results, we enforce mesh validity on the integration points. Second, to initialize the optimization, we start from a numerically valid mesh. Third, to penalize inverted elements, we modify the point-wise distortion to be infinity for non-positive Jacobians. Specifically, we regularize the element Jacobians to be zero for non-positive Jacobians, so their reciprocals are infinite. Note that these reciprocals appear in the distortion expression, and thus, they determine the infinite distortion value. Fourth, to enforce numerically valid mesh displacements, we equip Newton's method with a line-search, see Section 3.3. Specifically, if the mesh optimization update is invalid in any integration point, the objective function, Equation (3.3), is infinite. In that case, the step is divided by two until it leads to a valid mesh update.

3.5.1 Implementation

As a proof of concept, a mesh optimizer is developed in Julia 1.4.2 (Bezanson et al., 2017). For this, we use the following external packages: Arpack v0.5.0, Einsum v0.4.1, ILUZero v0.1.0, and TensorOperations v3.1.0. In addition, we use specific functions to solve sparse linear systems. First, we use the Julia internal CHOLMOD package from SuiteSparse as a direct solver, see Line 4 of Algorithms 3.5, 3.8, and 3.9. Specifically, we solve the linear system by computing a sparse LDLt factorization. Second, we use the preconditioned Conjugate Gradients (CG) algorithm (Saad, 2003) as an iterative method, see Line 8 of Algorithm 3.5, Line 10 of Algorithm 3.8, and Lines 10 and 18 of Algorithm 3.9. Third, we compute the iLU(0) factorization with the ILUZero.jl package, see Line 7 of Algorithm 3.7.

The Julia prototyping code is sequential, it corresponds to the implementation of the method presented in this chapter and to the method presented in Aparicio-Estrems et al. (2018). In all the examples, the optimization is reduced to find a minimum of a non-linear unconstrained multi-variable function. The ordering of the mesh nodes and of the degrees of freedom is detailed in Appendix A.4. The stopping condition is set to reach an absolute root mean square residual, that is $\|\nabla f(x)\|_{\ell^2}/\sqrt{n}$ for $x \in \mathbb{R}^n$, smaller than 10^{-4}. Each optimization process has been performed in a node featuring two Intel Xeon Platinum 8160 CPU with 24 cores, each at 2.10 GHz, and 96 GB of RAM memory.

3.5.2 Examples setup: domains and metrics

We consider the quadrilateral domain $\Omega = [-0.5, 0.5]^2$ for the two-dimensional examples and the hexahedral domain $\Omega = [-0.5, 0.5]^3$ for the three-dimensional ones. Each domain is equipped with a metric matching a boundary layer. In particular, our target metric $\mathbf{M}$ is characterized by a boundary layer metric with a diagonal matrix $\mathbf{D}$ and a deformation map φ by the following expression

$$\mathbf{M} = \nabla\varphi^{\mathrm{T}}\ \mathbf{D}\ \nabla\varphi, \tag{3.18}$$

where $\mathbf{D}$ is a boundary layer metric, and φ is a deformation map used to align the stretching with a given manifold. The constructions of both $\mathbf{D}$ and φ are detailed in Appendix A.5.

The anisotropy of the metric $\mathbf{M}$ can be described by two quantities: the anisotropic ratio and the anisotropic quotient (Loseille and Löhner, 2010). On the one hand, the

Table 3.1: Metric examples classified in terms of id, name, parameters, anisotropic quantities, and figure.

Id	Name	Parameters				Anisotropic		Figure
		$\mathbf{D}$	φ	γ	$1/h_{\min}$	ratio	quotient	
1st	Line	$\mathbf{D}$	(x, y)	2	100	100	100	3.3(a)
2nd	Curve	$\mathbf{D}$	$\left(x, \frac{1}{\sqrt{100+4\pi^2}} g(y, x, 1)\right)$	2	100	120	120	3.3(c)
3rd	Curves	$\mathbf{D}_{\text{cross}}$	$\frac{1}{\sqrt{100+4\pi^2}} (g(x, y, 1), g(y, x, 1))$	2	100	120	120	3.3(e)
4th	Plane	$\mathbf{D}$	(x, y, z)	2	50	50	50	3.3(b)
5th	Surface	$\mathbf{D}$	$\left(x, y, \frac{1}{\sqrt{100+8\pi^2}} g(z, y, x)\right)$	2	50	60	60	3.3(d)
6th	Surfaces	$\mathbf{D}_{\text{cross}}$	$\frac{1}{\sqrt{100+8\pi^2}} (g(x, y, z), g(y, z, x), g(z, y, x))$	2	50	60	3600	3.3(f)

anisotropic ratio is defined by the maximum local elongation. Specifically, at a physical point $\mathbf{p} \in \Omega \subset \mathbb{R}^d$ it is given by

$$\text{ratio}\,(\mathbf{p}) := \sqrt{\frac{\max\limits_{i=1,\dots,d} \lambda_i\,(\mathbf{p})}{\min\limits_{i=1,\dots,d} \lambda_i\,(\mathbf{p})}} > 1, \tag{3.19}$$

where $\lambda_i\,(\mathbf{p}) > 0,\ i = 1, ..., d$ are the eigenvalues of $\mathbf{M}\,(\mathbf{p}) \in \mathbb{R}^{d\times d}$. The maximum anisotropic ratio attained in Ω is denoted by $\text{ratio}_{\max} = \max\limits_{\mathbf{p}\in\Omega} \text{ratio}\,(\mathbf{p})$.

On the other hand, the anisotropic quotient represents the overall anisotropic ratio. Specifically, at a physical point $\mathbf{p} \in \Omega \subset \mathbb{R}^d$, the anisotropic quotient is given by

$$\text{quo}\,(\mathbf{p}) := \frac{\sqrt{\det\,(\mathbf{M}\,(\mathbf{p}))}}{\left(\min\limits_{i=1,\dots,d} \lambda_i\,(\mathbf{p})\right)^{d/2}} > 1. \tag{3.20}$$

The maximum anisotropic quotient attained in Ω is denoted by $\text{quo}_{\max} = \max\limits_{\mathbf{p}\in\Omega} \text{quo}\,(\mathbf{p})$.

In Table 3.1, we present six examples of metrics. In the first column, we show the numbering. Then, in the second column we show a descriptive name. Specifically, Line and Plane correspond to the boundary layer metrics over a line and a plane, respectively. In contrast, Curve and Surface correspond to the boundary layer metrics over a deformed line and a deformed plane, respectively. We use a singular noun for a layer over one entity and a plural noun for an intersection of two layers in 2D and three layers in 3D. In the third column, we present the parameters that characterize the metric, see Appendix A.5: the boundary layer metric $\mathbf{D}$, the deformation map φ in terms of the function $g(x, y, z) := 10x - \cos(2\pi y)\cos(2\pi z)$, the

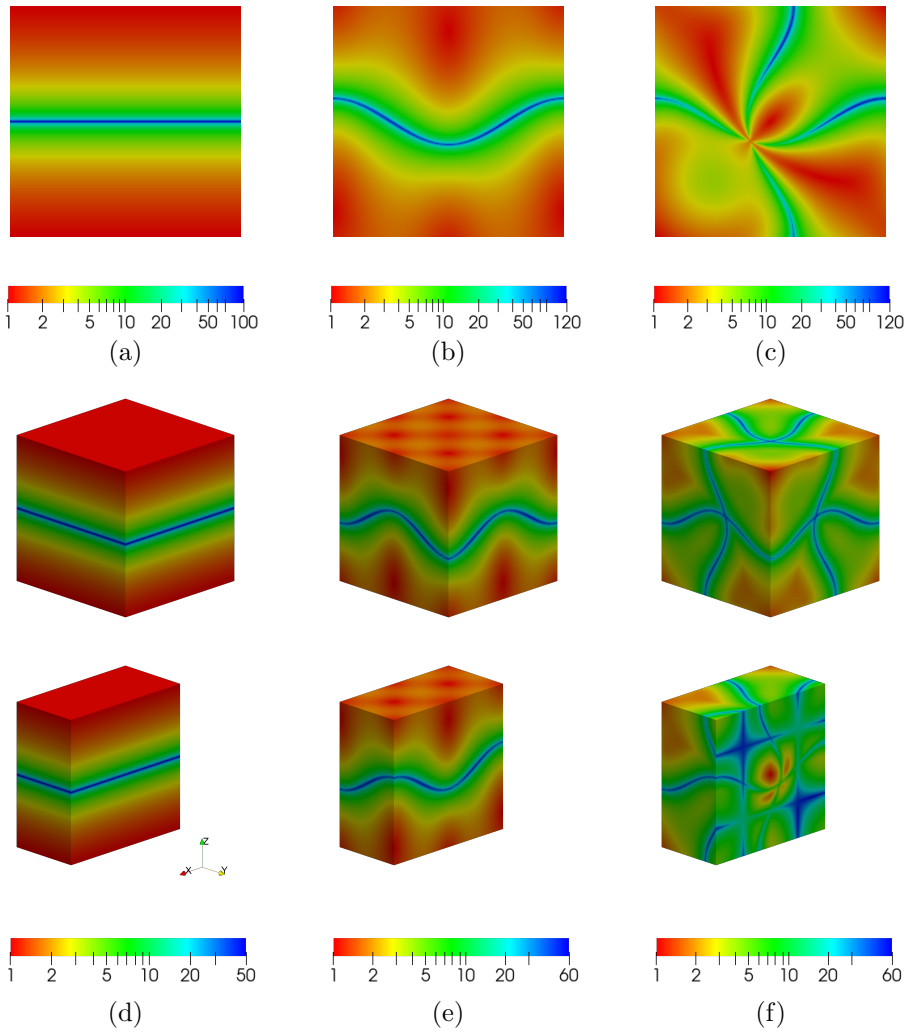


Figure 3.3: Anisotropic ratio in logarithmic scale for the different (columns) metric examples and (rows) domain dimensions.

growth factor γ, and the inverse of the imposed stretching $h_{\min}$, $1/h_{\min}$. Then, in the fourth column we present the approximate anisotropic ratio and quotient defined in Equations (3.19) and (3.20), respectively. Finally, in the last column we include the figure corresponding to the metric.

In Figure 3.3, we show the anisotropic ratio of the test metrics. We can observe that, it blends between 1 and $1/h_{\min}$ together with a contribution of the deformation φ. In addition, the maximum anisotropic ratio is attained at the zero-level sets of the last or each component of the deformation map φ depending on which boundary layer metric is used. That is, according to the ordering presented in Table 3.1 at:

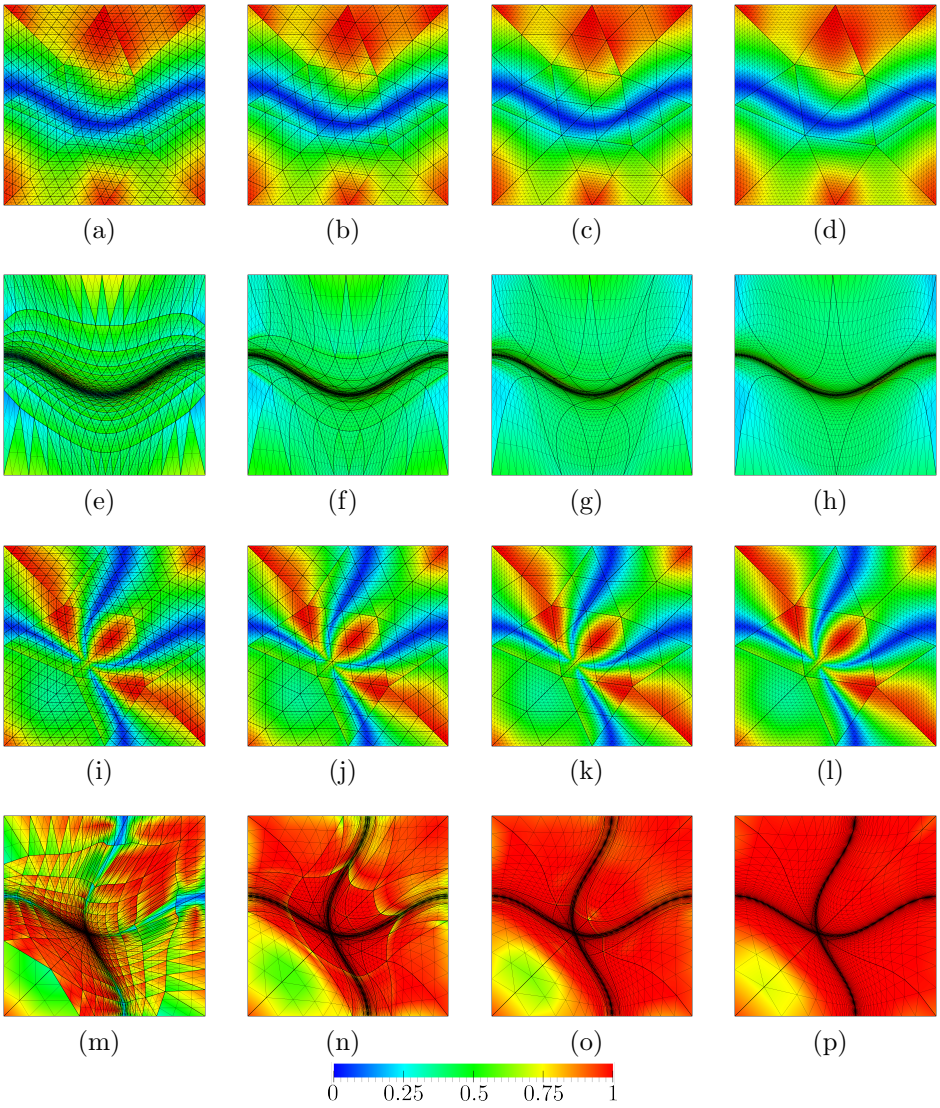


Figure 3.4: point-wise quality measure for meshes of (columns) polynomial degree 1, 2, 4, and 8 equipped with the (a-h) second metric and (i-p) third target metric: (a-d, i-l) initial straight-sided isotropic meshes, and (e-h,m-p) optimized meshes.

line $y = 0$; curve $g(y, x, 1) = 0$; the curves $g(y, x, 1) = 0$, $g(x, y, 1) = 0$; plane $z = 0$; surface $g(z, y, x) = 0$, and the surfaces $g(z, y, x) = 0$, $g(y, x, z) = 0$, $g(x, y, z) = 0$, respectively. Finally, note that at the intersection of two entities in 2D and three entities in 3D, the anisotropic ratio attains its minimum value, equal to one. This is because the stretching alignments span all the space, producing a sizing effect without stretching on a particular direction.

As initial guess of the mesh optimizer we generate isotropic meshes with the

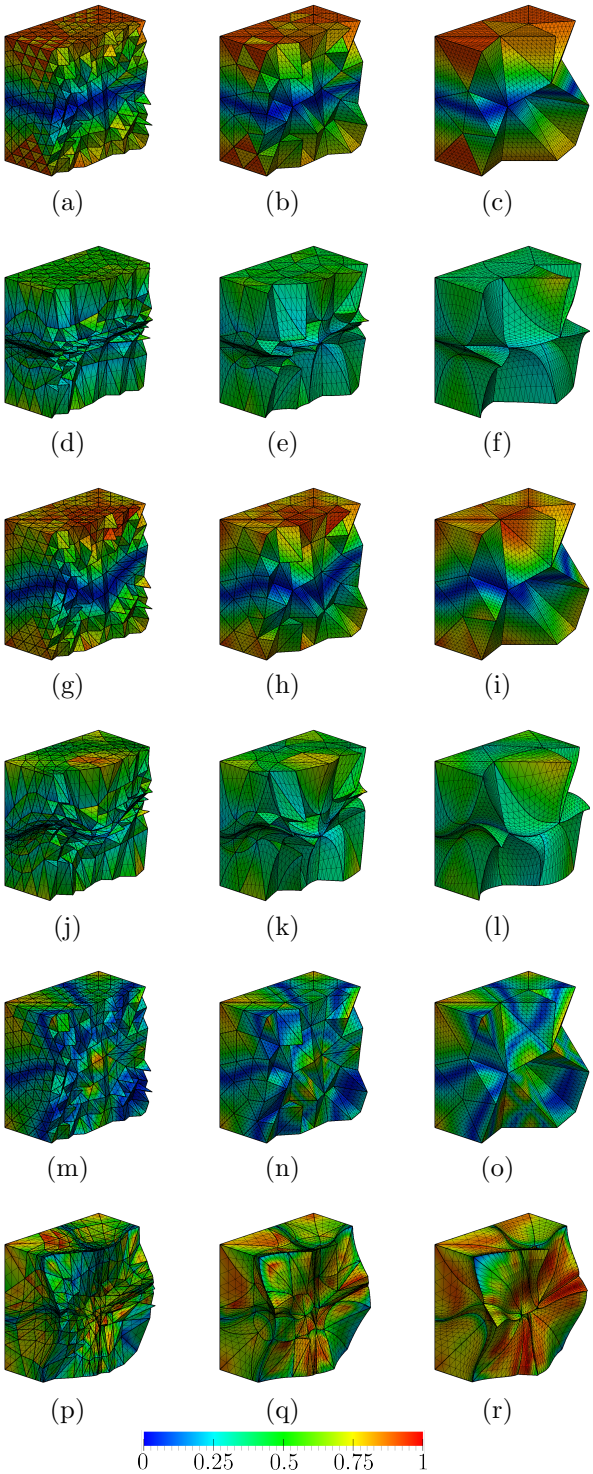


Figure 3.5: point-wise quality measure for meshes of (columns) polynomial degree 1, 2, and 4 equipped with the (a-f) first metric, (g-l) second metric, and (m-r) third target metric. (a-c, g-i, m-o) initial straight-sided isotropic meshes, and (d-f, j-l, p-r) optimized meshes.

MATLAB PDE Toolbox (MATLAB, 2017). The initial isotropic linear unstructured 2D and 3D meshes are presented in Figures 3.2(d) and 3.5(c), respectively. The structured meshes of lower polynomial degree are generated by subdivision.

In 2D, for each considered metric, we generate four meshes of polynomial degree 1, 2, 4, and 8. The meshes feature the same resolution and hence have the same number of nodes, 481 nodes, but a different number of elements, 896, 224, 56, and 14 elements, respectively. The meshes from Figures 3.2, 3.4(a)-3.4(h), and 3.4(i)-3.4(p) correspond to the metrics 1, 2, and 3, see Table 3.1.

In 3D, for each considered metric, we consider three meshes of polynomial degree 1, 2, and 4. The meshes feature the same resolution and hence, the same number of nodes, 1577 nodes, but a different number of elements, 7296, 912, and 114 elements, respectively. The meshes from Figures 3.5(a)-3.5(f), 3.5(g)-3.5(l), and 3.5(m)-3.5(r) correspond to the metrics 4, 5, and 6, see Table 3.1.

The meshes are colored according to the point-wise stretching and alignment quality measure, proposed in Aparicio-Estrems et al. (2018) and detailed in Equation (3.1) of Section 3.2.2. As we observe, the elements lying in the region of highest stretching ratio have less quality than the elements lying in the isotropic region.

To obtain an optimal configuration, we minimize the distortion measure by relocating the mesh nodes while preserving their connectivity, see Section 3.2.2. The coordinates of the inner nodes, and the coordinates tangent to the boundary, are the design variables. Thus, the inner nodes are free to move, the vertex nodes are fixed, while the rest of boundary nodes are enforced to slide along the boundary facets of the domain Ω. The total amount of degrees of freedom for the 2D and 3D meshes is 894 and 3957, respectively. The optimized meshes are illustrated in Figures 3.2 and 3.4 for the 2D cases and in Figure 3.5 for the 3D cases. We observe that the elements away from the anisotropic region are enlarged vertically whereas the elements lying in the anisotropic region are compressed. Moreover, the minimum quality is improved, and the standard deviation of the element qualities is reduced.

3.5.3 Line-search globalization: standard versus specific-purpose

Following, we compare the line-search globalization strategies, presented in Section 3.3, and their effect in the non-linear optimization method. For this, we apply the

Table 3.2: Non-linear and line-search iterations for the exact Newton method composed of standard and specific-purpose LS. Both globalizations are coupled with the direct solver.

Mesh	Non-linear iterations		Line-search iterations	
degree	Standard	Specific-purpose	Standard	Specific-purpose
1	54	37	159	31
2	82	91	328	155
4	88	78	304	92
8	203	125	771	171

Newton method presented in Section 3.2.3, with the corresponding globalization strategy and linear solver, to the first test metric presented in Table 3.1.

The standard and the specific-purpose globalization strategies are presented in Sections 3.3.1 and 3.3.2, respectively. To compare them we use an exact Newton method. Specifically, we consider the optimization method presented in Algorithm 3.1 with a globalization strategy, Line 11, and a direct solver, Line 10. In particular, the direct solver computes, Line 3 of Algorithm 3.5, the exact approximation of the Newton equation presented in Equation (3.7) using the complete sparse LDL^t preconditioner of the CHOLMOD package (Chen et al., 2008).

The results of our numerical experiments allow comparing the standard, Algorithm 3.2, and specific-purpose, Algorithm 3.3, globalization strategies in terms of the required line-search iterations, see Table 3.2. For meshes of polynomial degree 1, 2, 4, and 8, we report the number of non-linear and line-search iterations required to optimize the model case. We report these numbers for the exact Newton method equipped with the standard and specific-purpose globalizations.

We conclude that the specific-purpose strategy improves the standard one. The results show that the number of line-search iterations is reduced. Meanwhile, the number of non-linear iterations remain in the same order of magnitude, yet tending to be smaller. We can explain these improvements by highlighting two factors. First, the specific-purpose strategy can enlarge the step length with line-search iterations, a larger advance that promotes to reduce the number of non-linear iterations. Second, for the specific-purpose strategy, by reusing the last step length we promote to reduce the total amount of line-search iterations. In contrast, for the standard line-search strategy each direction has step length at most one, limiting the length of the step.

Table 3.3: Non-linear iterations and matrix-vector products for the inexact Newton methods with Jacobi preconditioner and Jacobi/iLDL$^{\mathrm{T}}$ preconditioners with specific-purpose LS globalization. Inexact Newton methods are distinguished by standard and specific-purpose forcing terms.

Mesh degree	Non-linear iterations		Matrix-vector products		Preconditioner
	Standard	Specific-purpose	Standard	Specific-purpose	
1	24	20	1856	677	Jacobi
	24	17	333	113	Jacobi/iLDL$^{\mathrm{T}}$
2	29	29	1824	624	Jacobi
	30	23	579	155	Jacobi/iLDL$^{\mathrm{T}}$
4	48	40	4858	1521	Jacobi
	48	31	1162	223	Jacobi/iLDL$^{\mathrm{T}}$
8	109	73	10031	2864	Jacobi
	109	83	3538	1452	Jacobi/iLDL$^{\mathrm{T}}$

3.5.4 Inexact Newton method: standard versus specific-purpose

Next, we compare the inexact Newton methods presented in Section 3.4. Specifically, we compare the influence of the forcing sequences and of the preconditioner in the non-linear optimization method. For this, we equip the meshes with the first metric presented in Table 3.1. Moreover, we globalize the non-linear solver, Section 3.2.3, with the specific-purpose LS strategy, Section 3.3. Finally, we optimize the meshes using the different approaches to compute the inexact Newton direction.

The standard and the specific-purpose inexact Newton methods are presented in Sections 3.4.1 and 3.4.2, respectively. We compare them in two steps. In both cases, we compare the standard inexact Newton method, that uses the standard forcing terms, Section 3.4.1.1, with the specific-purpose inexact Newton method, that uses the specific-purpose forcing terms, Section 3.4.2.1. In the first case, we use the Jacobi preconditioner presented in Section 3.4.1.2, see Line 6 of Algorithms 3.5 and 3.8. In the second case, we use the Jacobi/iLDL$^{\mathrm{T}}$ preconditioner switch presented in Section 3.4.2.2, see Line 6 of Algorithms 3.5 and 3.9.

The results of our numerical experiments allow comparing between the standard, Line 5 of Algorithm 3.5, and the specific-purpose, Line 5 of Algorithm 3.9, inexact Newton methods in terms of the number of required matrix-vector products, see Table 3.3. The model case is optimized with the specific-purpose LS strategy for meshes of polynomial degree 1, 2, 4, and 8. For these meshes, we report the number of non-linear

iterations and matrix-vector products required by the standard and specific-purpose inexact Newton methods.

We conclude that the specific-purpose preconditioned inexact Newton method significantly improves the standard one. Specifically, the matrix-vector products are reduced by one order of magnitude. On the one hand, the specific-purpose forcing terms feature a total number of matrix-vector products smaller than with the standard one. The number of products is reduced because the forcing terms stop the linear iterations when sufficient accuracy and positive curvature is reached. Accordingly, these conditions ensure that the reduction does not hamper the quality of the inexact Newton direction. Meanwhile, the number of non-linear iterations remain in the same order of magnitude, yet being smaller or equal. On the other hand, the specific-purpose pre-conditioner features a total number of matrix-vector products smaller than with the standard one. This number of products is reduced because the specific-purpose pre-conditioner automatically switches to a more accurate iLDL$^{\mathrm{T}}$ decomposition. Specifically, it only improves the accuracy when the Hessian is predicted to be numerically positive. Thus, the solver obtains a highly accurate Newton direction with fewer matrix-vector products. Meanwhile, the number of non-linear iterations stills almost unchanged. Finally, the combination of specific-purpose forcing terms and pre-conditioner features a number of matrix-vector products one order of magnitude smaller than for the standard one. This reduction is because we combine the advantages of the specific-purpose forcing terms and the specific-purpose preconditioner. Moreover, the number of non-linear iterations is reduced. In addition, we observe that when augmenting the mesh polynomial degree, the total amount of matrix-vector products and the number of non-linear iterations are increased. This can be explained by highlighting that when the mesh polynomial degree is increased the Hessian $\mathbf{H}f$ becomes more ill-conditioned. Hence, the CG method needs more iterations to converge.

3.5.5 Newton-CG solver: standard versus specific-purpose

In what follows, we compare both optimization solvers: standard and specific-purpose. To this end, we consider the r-adaption problem for the domains, metrics, and meshes presented in Section 3.5.2. Finally, we present the results obtained from the optimization processes.

Each optimization solver is composed of a globalization and a linear solver, see Al-

Table 3.4: Non-linear iterations, line-search iterations, and matrix-vector products for standard and specific-purpose optimization methods.

Example	Mesh degree	Non-linear iterations		Line-search iterations		Matrix-vector products		
		Standard	Specific-purpose	Standard	Specific-purpose	Standard	Specific-purpose	Speedup
Line	1	24	17	48	29	1646	113	14.57
	2	32	23	81	33	2309	155	14.90
	4	47	31	197	53	4284	223	19.21
	8	74	83	358	196	14710	1452	10.13
Curve	1	30	23	85	29	6689	145	46.13
	2	91	37	575	74	29626	339	87.39
	4	102	54	582	85	24497	597	41.03
	8	374	78	1721	190	30725	2150	14.29
Curves	1	43	28	174	38	4196	177	23.71
	2	211	57	1473	157	35378	835	42.37
	4	140	81	768	184	20535	1279	16.06
	8	858	102	3169	295	43317	4056	10.68
Plane	1	45	30	80	27	3943	378	10.43
	2	219	89	1298	91	10118	1194	08.47
	4	229	167	1534	267	22437	2985	07.52
Surface	1	69	51	87	117	16859	635	26.55
	2	287	131	1968	140	44547	2641	16.87
	4	290	118	2040	165	67997	1980	34.34
Surfaces	1	252	58	2322	64	51698	554	93.32
	2	361	152	2646	278	72078	2250	32.03
	4	288	164	2203	272	73593	2897	25.40

gorithm 3.1. On the one hand, the standard optimization method is composed of the standard BLS, see Section 3.3.1, and the standard linear solver with the Jacobi preconditioner, see Section 3.4.1. On the other hand, the specific-purpose optimization method is composed of the specific-purpose LS, see Section 3.3.2, and the specific-purpose linear solver with the Jacobi/iLDL$^{\mathrm{t}}$(0) preconditioner switch, see Section 3.4.2.

The results of our numerical experiments allow comparing the specific-purpose optimization method with the standard one, see Table 3.4. The non-linear and line-search iterations and the matrix-vector products are reported for 2D meshes of polynomial degree 1, 2, 4, and 8 and for 3D meshes of polynomial degree 1, 2, and 4.

We conclude that the specific-purpose optimization method significantly improves the standard one. In particular, the total amount of matrix-vector products is reduced one order of magnitude. Meanwhile, the number of non-linear and line-search iterations is reduced. As detailed in Sections 3.5.3 and 3.5.4, these reductions in the number of linear and non-linear iterations arise from combining the specific-purpose

inexact Newton solver and the specific-purpose line-search globalization. Moreover, as detailed in Section 3.5.4, we observe again that, when augmenting the polynomial degree for each tested case, the total number of matrix-vector products, non-linear iterations, and line-search iterations increases.

3.5.6 Application: metric-aware curved high-order optimization of an h-adapted mesh

To compare the standard and specific-purpose solvers in an adaptive application, we optimize the distortion of an anisotropic mesh previously adapted to match a metric. In practice, adapted meshes are obtained by combining local topological operations that modify the mesh connectivity and local r-adaption operations that modify the mesh coordinates (Alauzet and Loseille, 2016). Herein, we optimize the adapted mesh with the standard and specific-purpose optimization methods.

Although we generate meshes adapted to a target metric with MMG (Dobrzynski, 2012), our goal is not to compare the distortion minimization with the MMG package. Actually, we acknowledge MMG because it generates an initial straight-sided mesh that matches the stretching and alignment of the target metric.

We consider the hexahedral domain $\Omega = [-0.5, 0.5]^3$ with the Plane metric presented in Ibanez et al. (2017)

$$\mathbf{M} = \frac{1}{h_{\mathrm{m}}^2} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1/h(z)^2 \end{pmatrix},$$

where the function h is presented in Equation (A.2) with stretching $h_{\mathrm{min}} = 0.01$, growth factor $\gamma = 2(1 - h_{\mathrm{min}})$, and size $h_{\mathrm{m}} = 0.1$. Note that the stretching ratio of this metric is similar to the one presented in Figure 3.3(d).

First we generate an initial adapted mesh, see Figure 3.6(a). Specifically, we consider an isotropic linear tetrahedral mesh of size 0.01 with MATLAB (2017). We equip such mesh with the target metric evaluated at the mesh vertices and we use the MMG algorithm (Dobrzynski, 2012) to obtain an anisotropic mesh. This mesh is composed of 11092 nodes and 57448 tetrahedra. We observe that the elements lying in the anisotropy region are stretched and aligned matching the target metric. In addition, we observe that far from the anisotropic region the elements are almost isotropic and with an approximate size of $h_{\mathrm{m}} = 0.1$.

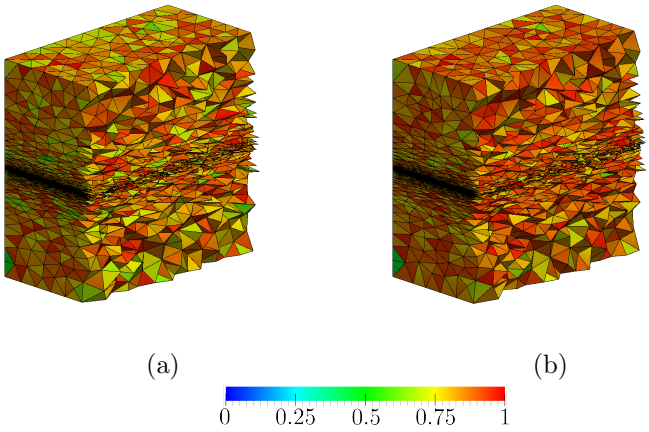


Figure 3.6: Linear tetrahedral meshes coupled with the Plane metric. (a) Initial adapted mesh; and (b) the corresponding optimized mesh.

Table 3.5: Quality and Geometry Statistics of the initial adapted mesh and the corresponding optimized mesh.

Measure	Minimum		Maximum		Mean		Standard deviation	
	Initial	Optimized	Initial	Optimized	Initial	Optimized	Initial	Optimized
Shape	0.3565	0.3804	0.9970	0.9972	0.8380	0.8690	0.3079	0.2790
Length	0.2340	0.4211	1.875	1.6481	0.9544	0.9425	0.1814	0.1641
Area	0.1528	0.1806	1.8377	1.7901	0.7265	0.7165	0.1776	0.1591
Volume	0.0583	0.0819	1.7251	1.3557	0.4865	0.4865	0.1418	0.1317

Table 3.6: Non-linear iterations, line-search iterations, and matrix-vector products for standard and specific-purpose optimization methods.

Non-linear iterations		Line-search iterations		Matrix-vector products	
Standard	Specific-purpose	Standard	Specific-purpose	Standard	Specific-purpose
7	7	0	0	1040	126

We obtain an optimal configuration, Figure 3.6(b), by minimizing the distortion measure presented in Section 3.2.2. The total amount of degrees of freedom is 30072. To compare both meshes we measure the element quality and geometry statistics, see Table 3.5. The measures are: the elemental shape quality presented in Equation (3.2), the lengths of the edges, the areas of the triangular faces, and the volumes of the tetrahedra. For each measure the minimum is improved, and the maximum and standard deviation are reduced.

We compare the specific-purpose optimization method with the standard one in terms of the non-linear and line-search iterations and the matrix-vector products,

see Table 3.6. From the results, we conclude that the specific-purpose optimization method significantly improves the standard one. In particular, the total amount of matrix-vector products is reduced almost one order of magnitude. Meanwhile, the total amount of non-linear and line-search iterations stays unchanged. This is because the initial mesh is near to the optimal one and hence, the descent direction of both optimization methods is a faithful approximation of the Newton direction. That is, since the direction of both optimization methods are similar, the same number of non-linear iterations to converge are required. In addition, since the Newton direction has step length equal to one, no line-search iterations are required, see Section 3.3.2.

3.6 Concluding remarks

We have presented a specific-purpose non-linear solver for curved high-order metric-aware mesh optimization. To this end, the solver combines a specific-purpose line search with a specific-purpose preconditioned Newton-CG solver with dynamic forcing terms.

The proposed specific-purpose line-search globalization continues the step length by ensuring sufficient decrease and progress. Compared with a standard backtracking line search, it reduces the number of line-search iterations because it reuses the step length. Meanwhile, the number of non-linear iterations remains in the same order of magnitude, yet it tends to be smaller because the line search can enlarge the step length.

Regarding the proposed specific-purpose preconditioned Newton-CG solver with dynamic forcing terms, it reduces the total number of matrix-vector products by one order of magnitude. It also reduces the number of non-linear iterations. Compared with standard solvers, the proposed solver significantly improves the performance indicators because it combines the advantages of the proposed forcing terms and pre-conditioner.

When we combine both ingredients, we also conclude that the specific-purpose non-linear solver reduces the total number of matrix-vector products by one order of magnitude. Moreover, it also reduces the number of non-linear and line-search iterations. Compared with standard solvers, all these iteration numbers are reduced because it combines the advantages of the specific-purpose inexact Newton solver and the specific-purpose line-search globalization.

For the standard and the specific-purpose solvers, we observe that higher polynomial degrees and stretching ratios lead to higher total numbers of matrix-vector products, non-linear iterations, and line-search iterations. Still, these iteration numbers are smaller for the specific-purpose solver.

Chapter 4

Combining high-order metric interpolation and geometry implicitization

4.1 Introduction

The capability to relocate mesh nodes without changing the mesh topology, referred to as r-adaptivity, is a key ingredient in many adaptive PDE-based applications (Yano and Darmofal, 2012; Loseille and Alauzet, 2011; Coupez et al., 2015). In these applications, to improve the solution accuracy, an error indicator or estimator determines the target stretching and alignment of the mesh. Then, to match these target features, an r-adaption procedure modifies the whole mesh (global) (Huang and Russell, 2011; Knupp, 2001) or a previously re-meshed cavity (local) (Alauzet and Loseille, 2016; Gruau and Coupez, 2005; Frey and Alauzet, 2005).

In either case, r-adaptivity contributes to increasing the solution accuracy for a fixed number of degrees of freedom supported on a straight-edged mesh (Coupez et al., 2015; Huang and Russell, 2011; Alauzet and Loseille, 2016; Hecht, 1998; Coupez, 2011). However, straight-edged meshes might not be an efficient support in many applications. Especially in applications where additional straight-edged mesh elements are artificially required to match highly curved solution features (Fidkowski and Darmofal, 2011).

To efficiently match curved solution features, many practitioners have recently started to exploit curved high-order meshes. These meshes can be stretched and aligned in a point-wise varying fashion through anisotropic procedures (Coupez, 2017), geodesic approaches for curved edges (Johnen et al., 2021; Zhang et al., 2018), shock-tracking methods (Zahr and Persson, 2018; Zahr et al., 2020; Zahr and Persson, 2020), and deformation analogies (Marcon et al., 2017, 2020). Alternatively, the curved r-adaption can be driven, as for straight-edged elements (Huang and Russell, 2011; Knupp, 2001), by distortion measures. These measures are defined point-wise and are aware of either a target deformation matrix (Dobrev et al., 2019) or a target metric (Aparicio-Estrems et al., 2018).

In adaptivity applications, the target deformations and metrics are not known a priori. These target fields are reconstructed a posteriori from the solution on the last mesh. Specifically, this mesh supports the resulting discrete representation of the target field. This discrete representation is key to interpolate the required field values in the adaptive procedure. However, to also preserve the geometric accuracy, the mesh adaption procedures have to be devised to simultaneously match the target curved boundaries. Hence, to enable high-order adaptivity, we need the capability to interpolate target fields on a high-order mesh while matching a target boundary.

4.1.1 Aim and contribution

Considering the previous issues, we aim to use Newton's optimization for distortion-based curved r-adaption to a discrete high-order metric field and a geometry model. This chapter extends our previous work Aparicio-Estrems et al. (2022). In this extension, we also detail how to compatibly combine an optimization based r-adaption with a valid-to-valid mesh curving approach. To this end, our contribution is to propose an implicit model representation that measures the deviations of the mesh to the target geometry.

For the optimization based r-adaption, we need three existent ingredients. First, to minimize the distortion, we use the specific-purpose solver in Chapter 3. Second, we represent the metric field as a log-Euclidean high-order metric interpolation (Rochery and Loseille, 2021) on a curved high-order mesh. Third, we locate physical points in the curved background mesh similar to the approach in Dobrev et al. (2018). We also need to extend to discrete metric fields a distortion-based curved r-adaption framework (Aparicio-Estrems et al., 2018).

To match the curved boundaries, we also need three existing ingredients. First, a non-interpolative approach to match the target curved geometry (Ruiz-Gironés et al., 2016, 2017). Second, an implicit CAD geometry representation method for 2D NURBS curves and a 3D NURBS surfaces (Upreti et al., 2014) or for embedded NURBS entities (Laurent, 2014) such as 3D curves. Third, a series of conjuction and trimming operations to assemble the implicit representations of the individual entities (Upreti et al., 2014; Biswas and Shapiro, 2004).

To compatibly combine the optimization based r-adaption with the mesh curving, the main novelty is twofold. First, for the non-interpolative mesh curving approach, we propose a model implicitization (Upreti et al., 2014; Laurent, 2014; Biswas and Shapiro, 2004). Second, we also provide the first and second-order derivatives of the implicit representation of the model. As in Aparicio-Estrems et al. (2022), we also provide the first and second derivatives in physical coordinates for the log-Euclidean high-order metric interpolation. The model implicitation derivatives and the metric interpolation derivatives are critical to use Newton's method for distortion minimization while targeting a curved geometry. This minimization leads to unprecedented second-order optimization results for curved r-adaption for a discrete high-order metric representation on a curved (straight-edged) mesh while targeting a curved (straight-edged) geometry.

This chapter focuses on enabling Newton's method for r-adaption, but it is focused neither on r-adaption nor h-adaption cycles. Specifically, we detail how to optimize the high-order mesh coordinates to match a target metric and a curved boundary. Then, to verify the methodology and the corresponding derivatives, we optimize initial isotropic and anisotropic straight-edged meshes. These results do not consider any adaptivity cycles because we want to demonstrate if Newton's method can be used.

The rest of the chapter is organized as follows. In Section 4.2, we overview the related work. In Section 4.3, we introduce the preliminaries on metric-aware measures for high-order elements. In Section 4.4, we detail the high-order metric interpolation and its derivatives. In Section 4.5, we propose an implicit representation for NURBS models, and we obtain the first and second derivatives of this representation. Moreover, we detail the objective function that accounts for the metric and geometry deviations. In Section 4.6, we show Newton's method results for different geometries, meshes, and metrics. Finally, we present the concluding remarks.

4.2 Related work

Next, we overview the work related to matching a target discrete field and a target geometry model. Regarding matching discrete fields, we overview works on target deformations, target metrics, and discrete field representations. For matching geometry models, the related work is about non-interpolative mesh curving, surface fitting methods, and implicit geometry representations.

To match a deformation matrix, distortion optimization for curved r-adaption to a discrete target field is detailed in Dobrev et al. (2019). The method is really well-suited for simulation-driven r-adaption (Dobrev et al., 2018, 2020). It evaluates the distortion in a physical point by interpolating the target matrix on a discrete field. Although the derivatives of the target matrices are not zero, the method assumes they are zero. Moreover, the second derivatives are also assumed to be zero. Since non-null derivatives are assumed to be zero although the approach implements Newton's method, the curved r-adaption minimization corresponds to a quasi-Newton method.

To match a metric, distortion-based curved r-adaption to an analytic field can be performed with Newton's minimization (Chapter 3) The formulation for an analytic metric is derived in Aparicio-Estrems et al. (2018), while a specific-purpose globalization and a pre-conditioned Netwon-CG method are proposed in Chapter 3 to minimize the mesh distortion. Since the method deals with an analytic metric, it does not specify the derivatives for a metric represented by a discrete high-order field.

Regarding a discrete field representation, a convenient approach is to use a log-Euclidean (Arsigny et al., 2006) high-order metric interpolation (Rochery and Loseille, 2021). This metric interpolation drives a cavity-based adaption approach, where the remeshed cavities are improved by locally smoothing the curved quadratic edges. To smooth these edges, the method optimizes the mid-node position. The optimization only uses the first derivatives of the log-Euclidean metric interpolation in terms of the curved edge coordinates. Accordingly, the method does not provide the first and second derivatives of the discrete metric field in physical coordinates.

High-order mesh curving methods that approximate the target geometry in a non-interpolative manner are presented in Ruiz-Gironés et al. (2016, 2017). Specifically, a new methodology to optimize a curved high-order mesh in terms of both element quality and a distance-based geometric approximation is developed. For this, a penalty method is proposed to solve the constrained minimization problem.

Previous surface fitting methods based in field interpolation are presented in Knupp et al. (2021). They are specially designed for dynamically changing geometry according to a solution. For this, a background mesh is required to interpolate the solution. Moreover, the resolution of the background mesh determines the precision of the dynamic geometry. Hence, for CAD models, the background mesh resolution controls the geometry accuracy.

In contrast to previous methods, implicit CAD geometry representation methods provide a field for geometric approximation without using a background mesh (Upreti et al., 2014; Laurent, 2014; Biswas and Shapiro, 2004). Specifically, one first computes the implicit representation of each NURBS entity. This is the case of a 2D NURBS curve and 3D NURBS surface (Upreti et al., 2014) or a generally embedded NURBS entity (Laurent, 2014). Then, one applies convex-hull conjunction and normalization, convex-hull trimming, and NURBS conjunction to assemble the representations of the individual entities (Upreti et al., 2014; Biswas and Shapiro, 2004). Even if they are not a full representation of the model they provide a useful tool for representing the model in a entity-wise fashion.

4.3 Preliminaries: metric-aware measures for high-order elements

In this section, we review the definition of the Jacobian-based quality measure for high-order elements equipped with a metric, presented in Aparicio-Estrems et al. (2018). To define and compute a Jacobian-based measure for simplices (Knupp, 2001), three elements are required: the master, the ideal, and the physical, see Figure 4.1 for the linear triangle case. The master (E^M) is the element from which the isoparametric mapping is defined. The equilateral element $\left(E^{\triangle}\right)$ represents the target configuration in the isotropic case. The physical (E^P) is the element to be measured.

To summarize the results in Aparicio-Estrems et al. (2018), we present the expression of the metric distortion measure in terms of the equilateral element $E^{\triangle}$. First, we need to compute a mapping from the master to the equilateral and physical elements, denoted as $\boldsymbol{\phi}_{\triangle}$ and $\boldsymbol{\phi}_P$, respectively. By means of these mappings, we determine a mapping between the equilateral and physical elements by the composition

$$\boldsymbol{\phi}_E : E^{\triangle} \xrightarrow{\boldsymbol{\phi}_{\triangle}^{-1}} E^M \xrightarrow{\boldsymbol{\phi}_P} E^P.$$

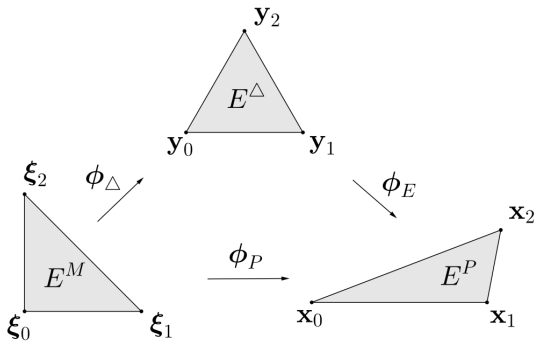


Figure 4.1: Mappings between the master, the ideal, and the physical elements in the linear case.

As detailed in Aparicio-Estrems et al. (2018), we define the point-wise distortion measure for a high-order element E^P equipped with a point-wise metric $\mathbf{M}$, at a point $\mathbf{y} \in E^{\triangle}$ as

$$\mathcal{N}\boldsymbol{\phi}_E(\mathbf{y}) = \frac{\operatorname{tr}\left(\mathbf{D}\boldsymbol{\phi}_E(\mathbf{y})^{\mathrm{T}}\ \mathbf{M}\left(\boldsymbol{\phi}_E(\mathbf{y})\right)\ \mathbf{D}\boldsymbol{\phi}_E(\mathbf{y})\right)}{d\left(\det\left(\mathbf{D}\boldsymbol{\phi}_E(\mathbf{y})^{\mathrm{T}}\ \mathbf{M}\left(\boldsymbol{\phi}_E(\mathbf{y})\right)\ \mathbf{D}\boldsymbol{\phi}_E(\mathbf{y})\right)\right)^{1/d}}, \tag{4.1}$$

where the Jacobian of the map $\boldsymbol{\phi}_E$ is given by

$$\mathbf{D}\boldsymbol{\phi}_E(\mathbf{y}) := \mathbf{D}\boldsymbol{\phi}_P(\boldsymbol{\phi}_{\triangle}^{-1}(\mathbf{y}))\ \mathbf{D}\boldsymbol{\phi}_{\triangle}^{-1}(\mathbf{y}).$$

Herein, $\mathbf{D}\boldsymbol{\phi}_P$ and $\mathbf{D}\boldsymbol{\phi}_{\triangle}$ denote the Jacobian of the physical and equilateral transformation, respectively. Specifically, the physical mapping can be expressed in terms of the d-simplex shape functions N_i, that is

$$\boldsymbol{\phi}_P(\boldsymbol{\xi}) = \sum_{i=1}^{n} N_i(\boldsymbol{\xi})\mathbf{x}_i,$$

where $n = \binom{d+p}{p}$ is the number of nodes, $\boldsymbol{\xi}$ are the master coordinates, and $\mathbf{x}_i$ denotes the physical coordinates of the high-order nodes. In addition, the equilateral mapping can be expressed in terms of the linear shape functions N_i, that is

$$\boldsymbol{\phi}_{\triangle}(\boldsymbol{\xi}) = \sum_{i=1}^{d+1} N_i(\boldsymbol{\xi})\mathbf{y}_i,$$

where $\mathbf{y}_i$ are the coordinates of an equilateral d-simplex.

Note that $\mathcal{N}$ is a non-linear operator that transforms a mapping between the equilateral and physical elements to a mapping from an point to a scalar. In this work, for operators, we use the standard notation without parentheses.

Note that the distortion measure is independent of the computation of the metric $\mathbf{M}\left(\boldsymbol{\phi}_E(\mathbf{y})\right)$, either using an analytical or a discretized representation.

We regularize the determinant in the denominator of Equation (4.1) in order to detect inverted elements (Branets and Garanzha, 2002; López et al., 2008; Escobar et al., 2003; Gargallo-Peiró et al., 2015c). In particular, we define

$$\sigma_0 = \frac{1}{2}(\sigma + |\sigma|),$$

where

$$\sigma = \det\left(\mathbf{D}\boldsymbol{\phi}_E(\mathbf{y})\right)\sqrt{\det\left(\mathbf{M}\left(\boldsymbol{\phi}_E(\mathbf{y})\right)\right)}.$$

Then, we define the point-wise regularized distortion measure of a physical element E^P at a point $\mathbf{y} \in E^\triangle$ as

$$\mathcal{N}_0\boldsymbol{\phi}_E(\mathbf{y}) := \frac{\mathrm{tr}(\mathbf{D}\boldsymbol{\phi}_E(\mathbf{y})^{\mathrm{T}}\ \mathbf{M}\left(\boldsymbol{\phi}_E(\mathbf{y})\right)\ \mathbf{D}\boldsymbol{\phi}_E(\mathbf{y}))}{d\ \sigma_0^{2/d}}, \tag{4.2}$$

where we introduce the sub-script 0 to distinguish the regularized operator from the non-regularized one. In addition, we define the corresponding point-wise quality measure

$$\mathcal{Q}\boldsymbol{\phi}_E(\mathbf{y}) = \frac{1}{\mathcal{N}_0\boldsymbol{\phi}_E(\mathbf{y})}. \tag{4.3}$$

Finally, we define the regularized elemental distortion by

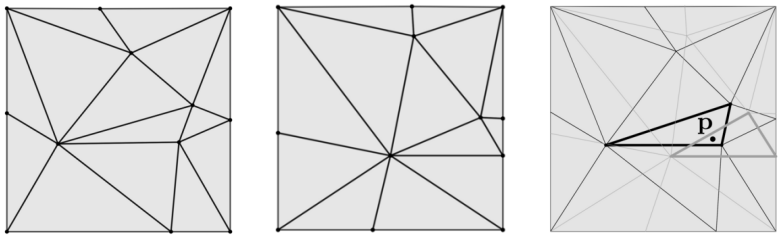
$$\eta_{(E^P,\mathbf{M})} := \frac{\int_{E^\triangle}\mathcal{N}_0\boldsymbol{\phi}_E(\mathbf{y})\ d\mathbf{y}}{\int_{E^\triangle} 1\ d\mathbf{y}},$$

and its corresponding quality

$$\mathrm{q}_{(E^P,\mathbf{M})} = \frac{1}{\eta_{(E^P,\mathbf{M})}}. \tag{4.4}$$

We can improve the mesh configuration by means of relocating the nodes of the mesh according to a given distortion measure (Chapter 3). In Aparicio-Estrems et al. (2018) it is proposed an optimization of the distortion (quality) of a mesh $\mathcal{M}$ equipped with a target metric $\mathbf{M}$ that describes the desired alignment and stretching of the mesh elements. To optimize a given mesh $\mathcal{M}$, first it is defined the mesh distortion by

$$\mathcal{F}\left(\mathcal{M}\right) := \sum_{E^P\in\mathcal{M}}\int_{E^\triangle}\left(\mathcal{N}_0\boldsymbol{\phi}_E(\mathbf{y})\right)^2\ d\mathbf{y}, \tag{4.5}$$



(a) (b) (c)

Figure 4.2: Point localization: (a) physical mesh, (b) background mesh, and (c) a point $\mathbf{p}$ in the corresponding physical and background element (bold edges).

which allows to pose the following global minimization problem

$$\mathcal{M}^* := \text{argmin}_{\mathcal{M}} \mathcal{F}(\mathcal{M}), \tag{4.6}$$

to improve the mesh configuration according to $\mathcal{F}$. In particular, herein, the degrees of freedom of the minimization problem in Equation (4.6) correspond to the spatial coordinates of the mesh nodes.

To evaluate the distortion minimization formulation presented in Equation (4.6), an input metric is required. The reviewed r-adaption procedure has been applied for analytic metrics in Aparicio-Estrems et al. (2018). In the following section, we detail the interpolation process that is required to extend the presented framework to dicrete metrics.

4.4 Log-Euclidean metric interpolation

In this section, we formulate a metric interpolation process that allows both the distortion evaluation, Equation (4.2), and its optimization, Equation (4.6). In Section 4.4.1 we detail the log-Euclidean metric interpolation for linear and high-order elements first presented in Arsigny et al. (2006) and Rochery and Loseille (2021); Ekelschot et al. (2019), respectively. Then, in Section 4.4.2 we present, as a contribution of this work, the gradient and the Hessian of the log-Euclidean interpolation. Their computation will be used for the distortion minimization problem.

4.4.1 Metric Interpolation

In this section, we introduce the definition of the log-Euclidean metric interpolation at the background mesh. First, we introduce the required notation of the mappings

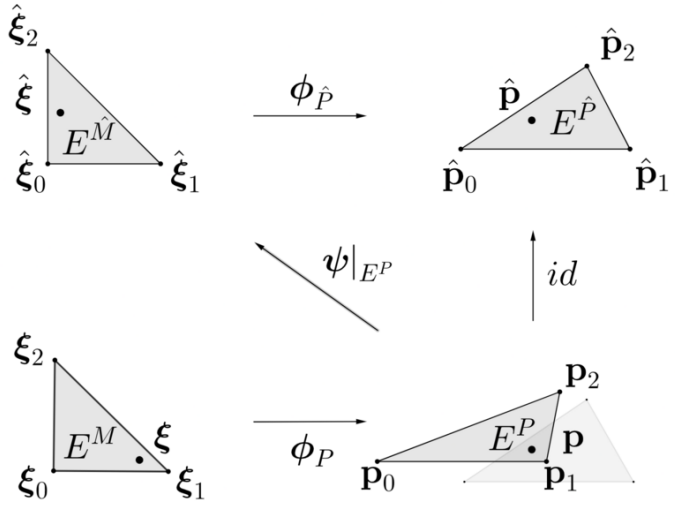


Figure 4.3: Mappings between the master and the physical elements (below) and their background analogs (above).

and their parameters with the corresponding diagram. Secondly, we detail the interpolation procedure.

To evaluate the metric-aware distortion measure in Equation (4.2) featuring discrete metrics, two meshes are required. On the one hand, the *physical* mesh $\mathcal{M}$, Figure 4.2(a), is the domain where the elements are deformed in order to solve the problem presented in Equation (4.6). On the other hand, the background mesh $\hat{\mathcal{M}}$, Figure 4.2(b), is a mesh that stores discrete metric values as a nodal field.

To evaluate the point-wise metric-aware distortion measure, we need to compute the interpolation of the point-wise metric values. For this, the localization between both meshes is required (Dobrev et al., 2018; Mittal et al., 2019; Sitaraman et al., 2010). In particular, a physical point $\mathbf{p} \in \mathcal{M}$ is located at the background mesh $\hat{\mathcal{M}}$ where the metric is interpolated, see Figure 4.2(c). In what follows, we introduce the elements and the mappings required for this localization procedure.

We integrate the distortion measure presented in Equation (4.2) over the equilateral element via the master element E^M. In particular, for the metric evaluation, we map via $\boldsymbol{\phi}_P$, each integration point $\boldsymbol{\xi} \in E^M$ to a point $\mathbf{p}$ of the physical element E^P, see Figure 4.3. To compute the metric at $\mathbf{p}$ we need to locate $\mathbf{p}$ in the background mesh, where the values of the metric are stored, see the intersection between E^P and the background element $E^{\hat{P}}$ in Figure 4.3. In addition, Figure 4.3 shows the procedure to obtain the coordinate to interpolate the metric from the quadrature points.

In particular, we map a reference point $\boldsymbol{\xi} \in E^M$ to a physical point $\mathbf{p} = \boldsymbol{\phi}_P(\boldsymbol{\xi}) \in E^P$, which we identify it with a point $\hat{\mathbf{p}} \in E^{\hat{P}}$ of the background mesh and its preimage is the background reference point $\hat{\boldsymbol{\xi}} = \boldsymbol{\phi}_{\hat{P}}{}^{-1}(\hat{\mathbf{p}}) \in E^{\hat{M}}$.

Given a physical point $\mathbf{p}$, we find it convenient to denote by ψ any mapping from a background element containing $\mathbf{p}$ that provides the coordinates in the background master element $E^{\hat{M}}$. Using this notation, we understand that any projection of a physical point $\mathbf{p}$ onto a point $\hat{\boldsymbol{\xi}}$ of the background master element $E^{\hat{M}}$ corresponds to the evaluation of the non-linear function $\hat{\boldsymbol{\xi}} = \psi(\mathbf{p})$.

To evaluate this non-linear function, we exploit that the expression of $\psi|_{E^P}$, defined in the intersection of a physical element E^P and a fixed background element $E^{\hat{P}}$, is given by

$$\begin{array}{rccc} \psi|_{E^P}: & E^P \cap E^{\hat{P}} & \rightarrow & E^{\hat{M}} \\ & \mathbf{p} & \mapsto & \boldsymbol{\phi}_{\hat{P}}^{-1}(\mathbf{p}). \end{array} \tag{4.7}$$

Specifically, we solve the non-linear inverse expression in the image term, Equation (4.7), by applying Newton's minimization to the squared distance. That is, as in Section 2.3 of Mittal et al. (2019), we solve

$$\hat{\boldsymbol{\xi}} = \text{argmin}_{\hat{\zeta}} \left|\left| \boldsymbol{\phi}_{\hat{P}}\left(\hat{\zeta}\right) - \mathbf{p} \right|\right|^2.$$

The result is a numerical approximation of the point coordinates in the background master element. An alternative approach (Dobrev et al., 2018) is to seek the zeros of the vector equation

$$\boldsymbol{\phi}_{\hat{P}}\left(\hat{\boldsymbol{\xi}}\right) - \mathbf{p} = \mathbf{0}.$$

Once the background master coordinates associated to a given physical point have been computed, it is necessary to interpolate the metric supported by the background mesh at the corresponding master coordinate. To do so, we use the log-Euclidean interpolation proposed in Arsigny et al. (2006); Rochery and Loseille (2021):

$$\mathbf{M}\left(\hat{\mathbf{N}}\right) := \exp\left(\mathbf{L}(\hat{\mathbf{N}})\right), \quad \mathbf{L}(\hat{\mathbf{N}}) := \sum_{j=1}^{\hat{n}} \hat{N}_j \log \hat{\mathbf{M}}_j, \tag{4.8}$$

where for the j-th node of the master element $E^{\hat{M}}$, $\hat{\mathbf{M}}_j$, and $\hat{N}_j$ are the corresponding metric value and shape function, respectively. In addition, $\hat{\mathbf{N}}$ denotes all the shape functions, $\hat{n} = \binom{d+\hat{p}}{\hat{p}}$ is the number of nodes, and where $\hat{p}$ is the interpolation degree

which corresponds to the polynomial degree of the master element $E^{\hat{M}}$. Finally, $\mathbf{M}(\hat{\mathbf{N}})$ is characterized by the *eigenvalue-based* matrix exponential function

$$\mathbf{M}\left(\hat{\mathbf{N}}\right) = \mathbf{U}\ \exp\mathbf{D}\ \mathbf{U}^{\mathrm{T}}, \tag{4.9}$$

where $\mathbf{D}$, $\mathbf{U}$ are given from the eigenvalue decomposition of the matrix $\mathbf{L}(\hat{\mathbf{N}}) =: \mathbf{U}\ \mathbf{D}\ \mathbf{U}^{\mathrm{T}}$. Finally, for each physical point $\mathbf{p}$ the metric interpolation is given by $\mathbf{M}\left(\hat{\mathbf{N}}\left(\psi\left(\mathbf{p}\right)\right)\right)$.

4.4.2 Gradient and Hessian

This section provides the expressions for the gradient and Hessian of the metric interpolation over a background mesh in terms of the physical coordinates. For this, we detail first the case for the metric interpolation at a single element and then for the background mesh. In particular, our approach uses the gradient and Hessian of the eigenvalue decomposition presented in Andrew et al. (1993).

To compute the derivatives of the metric $\mathbf{M}$ we first differentiate the eigenvalue-based exponential matrix function presented in Equation (4.9) and then we differentiate the $\mathbf{L}$ function presented in Equation (4.8). By denoting x_j the coordinates of $\mathbf{p}$ and $\partial_j := \frac{\partial}{\partial x_j}$, $\partial_{jk} := \partial_j\partial_k = \frac{\partial}{\partial x_j}\frac{\partial}{\partial x_k}$ the partial derivatives in terms of the physical coordinates of $\mathbf{p}$, we can compute the spatial derivatives of the metric interpolation of Equation (4.8). In particular, the first-order derivatives are given by

$$\begin{aligned}\partial_j\mathbf{M}(\hat{\mathbf{N}}) = \partial_j \exp\mathbf{L}(\hat{\mathbf{N}}) = \partial_j\left(\mathbf{U}\ \exp\mathbf{D}\ \mathbf{U}^{\mathrm{T}}\right) = \\ \left(\partial_j\mathbf{U}\right)\ \exp\mathbf{D}\ \mathbf{U}^{\mathrm{T}} + \mathbf{U}\ \left(\partial_j\exp\mathbf{D}\right)\ \mathbf{U}^{\mathrm{T}} + \mathbf{U}\ \exp\mathbf{D}\ \left(\partial_j\mathbf{U}^{\mathrm{T}}\right),\end{aligned}$$

and the second-order derivatives are given by

$$\begin{aligned}\partial_{jk}\mathbf{M}(\hat{\mathbf{N}}) = \partial_{jk} \exp\mathbf{L}(\hat{\mathbf{N}}) = \partial_{jk}\left(\mathbf{U}\ \exp\mathbf{D}\ \mathbf{U}^{\mathrm{T}}\right) = \\ \left(\partial_{jk}\mathbf{U}\right)\ \exp\mathbf{D}\ \mathbf{U}^{\mathrm{T}} + \partial_k\mathbf{U}\ \left(\partial_j\exp\mathbf{D}\right)\ \mathbf{U}^{\mathrm{T}} + \partial_k\mathbf{U}\ \exp\mathbf{D}\ \left(\partial_j\mathbf{U}^{\mathrm{T}}\right) + \\ \left(\partial_j\mathbf{U}\right)\ \partial_k\exp\mathbf{D}\ \mathbf{U}^{\mathrm{T}} + \mathbf{U}\ \left(\partial_{jk}\exp\mathbf{D}\right)\ \mathbf{U}^{\mathrm{T}} + \mathbf{U}\ \partial_k\exp\mathbf{D}\ \left(\partial_j\mathbf{U}^{\mathrm{T}}\right) + \\ \left(\partial_j\mathbf{U}\right)\ \exp\mathbf{D}\ \partial_k\mathbf{U}^{\mathrm{T}} + \mathbf{U}\ \left(\partial_j\exp\mathbf{D}\right)\ \partial_k\mathbf{U}^{\mathrm{T}} + \mathbf{U}\ \exp\mathbf{D}\ \left(\partial_{jk}\mathbf{U}^{\mathrm{T}}\right).\end{aligned}$$

Note that, since the matrix $\mathbf{D}$ is diagonal, we have

$$\begin{aligned}\partial_j\exp\mathbf{D} &= \exp\left(\mathbf{D}\right)\ \partial_j\mathbf{D}, \\ \partial_{jk}\exp\mathbf{D} &= \exp\left(\mathbf{D}\right)\ \left(\partial_k\mathbf{D}\ \partial_j\mathbf{D} + \partial_{jk}\mathbf{D}\right).\end{aligned}$$

The presented first and second-order derivatives of the metric require the first and second-order spatial derivatives of the eigenvalue decomposition (eigenvalues and eigenvectors), respectively. Their computation is detailed in Appendix B.1.

In addition, the derivatives of the eigenvalues and eigenvectors depend on the derivatives of the $\mathbf{L}$ function presented in Equation (4.8). In particular, they are given by

$$\nabla \mathbf{L} = \sum_j \left(\log \hat{\mathbf{M}}_j \right) \nabla \hat{N}_j, \quad \nabla^2 \mathbf{L} = \sum_j \left(\log \hat{\mathbf{M}}_j \right) \nabla^2 \hat{N}_j,$$

where ∇ is the gradient with respect to physical coordinates. Therefore, to differentiate the metric interpolation $\mathbf{M}\left(\hat{\mathbf{N}}\left(\psi\left(\mathbf{p}\right)\right)\right)$ at a physical point $\mathbf{p}$, the derivatives of the map ψ presented in Equation (4.7) and of the shape functions $\hat{\mathbf{N}}$ are required.

The derivatives of $\psi|_{E^P}$ are given, at each patch $E^P \cap E^{\hat{P}}$, by the ones of the inverse of the physical map $\boldsymbol{\phi}_{\hat{P}}^{-1}$ corresponding to the background mesh. To obtain the derivatives of the shape functions $\hat{\mathbf{N}}$ in terms of the physical coordinates $\mathbf{p}$, we consider the chain rule for the composition $\hat{\mathbf{N}} \circ \psi|_{E^P}$ and the restriction of the map $\psi|_{E^P}$ at each patch $E^P \cap E^{\hat{P}}$. We finally obtain the gradient

$$\nabla \hat{\mathbf{N}} = \nabla_{\hat{\boldsymbol{\xi}}} \hat{\mathbf{N}} \; \nabla \boldsymbol{\phi}_{\hat{P}}^{-1}, \tag{4.10}$$

where $\nabla_{\hat{\boldsymbol{\xi}}}$ is the gradient with respect to $\hat{\boldsymbol{\xi}}$ coordinates, and the Hessian

$$\nabla^2 \hat{N}_j = \left(\nabla \boldsymbol{\phi}_{\hat{P}}^{-1} \right)^{\mathrm{T}} \; \nabla^2_{\hat{\boldsymbol{\xi}}} \hat{N}_j \; \nabla \boldsymbol{\phi}_{\hat{P}}^{-1} + \nabla_{\hat{\boldsymbol{\xi}}} \hat{N}_j \; \nabla^2 \boldsymbol{\phi}_{\hat{P}}^{-1}, \tag{4.11}$$

where

$$\begin{aligned} \nabla \boldsymbol{\phi}_{\hat{P}}^{-1} &= \left(\nabla_{\hat{\boldsymbol{\xi}}} \boldsymbol{\phi}_{\hat{P}} \right)^{-1}, \\ \nabla^2 \boldsymbol{\phi}_{\hat{P}}^{-1} = \nabla \left(\left(\nabla_{\hat{\boldsymbol{\xi}}} \boldsymbol{\phi}_{\hat{P}} \right)^{-1} \right) &= -\nabla \boldsymbol{\phi}_{\hat{P}}^{-1} \; \nabla^2_{\hat{\boldsymbol{\xi}}} \boldsymbol{\phi}_{\hat{P}} \; \nabla \boldsymbol{\phi}_{\hat{P}}^{-1}. \end{aligned}$$

4.5 Implicit CAD representation: metric and geometry aware optimization

Herein, we propose a high-order mesh curving method by an implicitization that measures the geometric deviation. First, in Section 4.5.1, we present a model implicitization for the mesh curving process. Then, in Section 4.5.2, we detail the first and second-order derivatives for the implicit representation. Finally, in Section 4.5.3, we consider the penalty method to solve the corresponding constrained second-order minimization process for the curving problem.

4.5.1 Implicit CAD representation

In this section, we present an entity-wise CAD representation for curves in 2D, and for curves, and surfaces in 3D. For this, we consider the implicit representation of embedded NURBS (Laurent, 2014), and the Boolean algebraic operations for implicit representations (Upreti et al., 2014; Biswas and Shapiro, 2004). Then, we assemble these representations to obtain an implicit representation of a CAD model. Finally, we detail the algorithm of the considered methodology.

We consider a CAD model Λ composed of a sequence of NURBS entities. These NURBS entities can be decomposed into a sequence of Bézier patches $\Gamma_i,\ i = 1, ..., n$. In particular, we describe a d-dimensional Bézier patch $\Gamma \subset \mathbb{R}^D$ embedded in a D-dimensional space in terms of a parameterization

$$\varphi_\Gamma : [0,1]^d \to \mathbb{R}^D, \quad \varphi_\Gamma(u) \in \Gamma,\ u \in [0,1]^d.$$

In addition, the implicit representation of Γ can be obtained as in Laurent (2014)

$$\gamma_\Gamma : \mathbb{R}^D \to \mathbb{R}, \quad \gamma_\Gamma(x) = 0 \text{ if and only if } x \in \Gamma.$$

Our objective is to obtain a representation γ_Λ of the model Λ that is expressed in terms of the representations γ_{Γ_i} of the patches Γ_i. To combine these implicit representations we use algebraic Boolean operations between real-valued functions (Biswas and Shapiro, 2004).

In Figure 4.4, we show a 2D and a 3D model. They are mapped via the Bézier parameterizations φ_{Γ_i} and their level-sets are represented via the implicit function γ_Λ. The level-sets are illustrated in linear and logarithmic scaling. As we observe, the functions are numerically zero at the model. In addition, they smoothly increase far from the model region.

The implicit representation of a CAD model requires a knot preprocessing of the NURBS entities. Specifically, two knot insertion procedures are required (Upreti et al., 2014). The first knot insertion, is used to decompose the NURBS entity into Bézier patches. The second one, is used to avoid auto-intersections for curves of degree $p \geq 3$. In this case, we perform an auto-intersection detection process. Note that the auto-intersection points are given by the equation $\|\nabla\gamma_\Gamma\| = 0$. Then, we detect the auto-intersections by minimizing the quantity $\|\nabla\gamma_\Gamma\|^2$ via a one-dimensional search bisection over the parametric line.

To trim the implicit representation in its corresponding domain, we consider the convex-hull of the Bézier patch control points. Specifically, for degenerate cases we

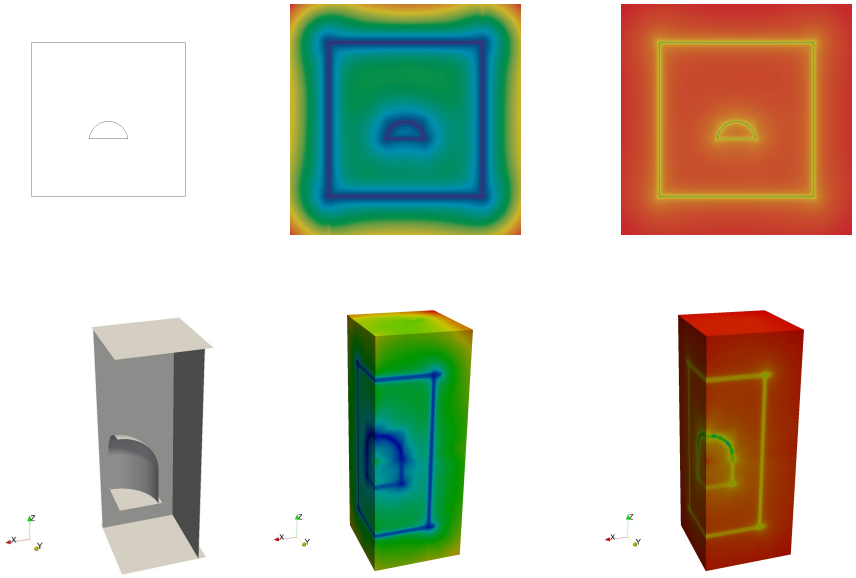


Figure 4.4: Implicit representation of (first row) a 2D CAD geometry, and (second row) a 3D CAD geometry. CAD model, and implicit representation in linear, and logarithmic scale in columns.

extrude the set of control points. To compute the extrusion directions we perform a null space computation via the singular value decomposition. This defines a valid convex-hull. We apply this procedure to the degenerate cases given by 2D segments, 3D planes, 3D curves, and 3D cylinders. Furthermore, we also apply this procedure to approximately degenerate cases such as almost flat curves and surfaces.

For each Bézier patch Γ, we consider its implicit representation γ defined in the projective space $\mathbb{P}\left(\mathbb{R}^D\right)$. In particular, the patch control points determine a vector of matrices that is, $\mathbb{M} = (\mathbb{M}_x, \mathbb{M}_y, \mathbb{M}_z, \mathbb{M}_w)$ in 3D, corresponding to the projective coordinates x, y, z, and w. Then, we define the implicit representation at a point $\mathbf{x} \in \mathbb{P}\left(\mathbb{R}^D\right)$ as in Laurent (2014)

$$\gamma\left(\mathbf{x}\right) := \det\left(\mathbb{M}\left(\mathbf{x}\right) \ \mathbb{M}\left(\mathbf{x}\right)^{\mathrm{T}}\right), \quad \mathbb{M}\left(\mathbf{x}\right) := \mathbb{M}\ \mathbf{x}. \tag{4.12}$$

For example, in 3D we set $\mathbf{x} = (x, y, z, 1)$ and hence, $\mathbb{M}\ \mathbf{x} = \mathbb{M}_x x + \mathbb{M}_y y + \mathbb{M}_z z + \mathbb{M}_w$. Finally, we normalize the functions $\gamma\left(\mathbf{x}\right)$ to ensure that they match during the assembly procedure (Upreti et al., 2014). Specifically, we define the normalized function $\hat{\gamma}$ by

$$\hat{\gamma} := \frac{\gamma}{\|\nabla\gamma\|}. \tag{4.13}$$

The implicit function of a Bézier patch described in Equation (4.13) extends over an infinite parametric space. For this reason, it is standard to trim the patch via a convex hull operation to ensure that the function does not extend beyond the patch limits (Upreti et al., 2014; Biswas and Shapiro, 2004). Specifically, we first compute $\mathrm{CH}(\Gamma)$, the implicit representation of the convex hull of the Bézier patch Γ. Then, to obtain an implicit representation of Γ trimmed by $\mathrm{CH}(\Gamma)$, we use a trimming function, denoted by trim, proposed in Biswas and Shapiro (2004)

$$\gamma_{\overline{\Gamma}} := \hat{\gamma}_{\mathrm{CH}(\Gamma)} \text{ trim } \hat{\gamma}_{\Gamma} = \sqrt{{\hat{\gamma}_{\Gamma}}^2 + \left(\frac{\sqrt{{\hat{\gamma}_{\Gamma}}^4 + {\hat{\gamma}_{\mathrm{CH}(\Gamma)}}^2} - \hat{\gamma}_{\mathrm{CH}(\Gamma)}}{2} \right)^2}, \tag{4.14}$$

where γ_{Γ} denotes the representation of the Bézier patch Γ, see Equation (4.12). The trimming operation of Equation (4.14) is twice differentiable at all points where $\hat{\gamma}_{\Gamma} \neq 0$. Here, the function $\gamma_{\overline{\Gamma}}$ is an implicit representation of the Bézier patch Γ in its parametric domain $\mathrm{Dom}\,\Gamma$ determined by the NURBS convex-hull $\mathrm{CH}(\Gamma)$.

For a given model $\Lambda = \{\Gamma_1, \Gamma_2, ..., \Gamma_n\}$, its implicitization γ_{Λ} is obtained via the r-conjunction $\wedge$ of the implicitizations $\gamma_{\overline{\Gamma_i}}$ of the Bézier patches Γ_i (Upreti et al., 2014). In particular, for each Bézier patch Γ_i, we recursively update the model representation as follows

$$\gamma_{\Lambda} \leftarrow \gamma_{\Lambda} \wedge \gamma_{\overline{\Gamma}_i} := \gamma_{\Lambda} + \gamma_{\overline{\Gamma}} - \sqrt{{\gamma_{\Lambda}}^2 + {\gamma_{\overline{\Gamma}}}^2}. \tag{4.15}$$

To obtain the convex-hull representation of a Bézier patch Γ, $\mathrm{CH}(\Gamma)$, we apply r-conjunction to the hyperplane functions of the convex hull entities. Specifically, for each hyperplane entity H of the convex hull $\mathrm{CH}(\Gamma)$ we consider its unit normal component $\mathbf{n}$ and its affine term b. Then, the implicit representation of H is given by

$$\gamma_{\mathrm{H}}(\mathbf{x}) := \mathbf{n} \cdot \mathbf{x} + b. \tag{4.16}$$

In our case, the sign of the representation γ_{H} is chosen such that $\gamma_{\mathrm{H}} < 0$ outside the convex region enclosed by $\mathrm{CH}(\Gamma)$ and $\gamma_{\mathrm{H}} \geq 0$ otherwise. Following, we apply the r-conjunction operation for each hyperplane H to obtain the convex-hull representation $\gamma_{\mathrm{CH}(\Gamma)}$, see Equation (4.15). Finally, we obtain its normalized version $\hat{\gamma}_{\mathrm{CH}(\Gamma)}$ by applying Equation (4.13).

In Algorithm 4.1, we describe how to obtain the implicit representation γ_{Λ} of a model Λ. In Line 2, we compute for each Bézier patch Γ_i of Λ its implicit function γ_{Γ_i} and we normalize it, see Equations (4.12) and (4.13). Then, in Line 3, we consider the

Algorithm 4.1 Implicitization

Input: $\Lambda := \{\Gamma_1,\ \Gamma_2,\ ...,\ \Gamma_n\}$
Output: γ_Λ

1: **for** $i = 1, ..., n$ **do**
2: $\quad\hat{\gamma}_{\Gamma_i}$ = normalized implicitization of Γ_i
3: $\quad\hat{\gamma}_{\mathrm{CH}(\Gamma_i)}$ = normalized implicitization of the convex hull of Γ_i, $\mathrm{CH}(\Gamma_i)$
4: $\quad\gamma_{\overline{\Gamma}_i}$ = trimming of $\hat{\gamma}_{\Gamma_i}$ with $\hat{\gamma}_{\mathrm{CH}(\Gamma_i)}$
5: $\quad$**if** $i = 1$ **then**
6: $\quad\quad\gamma_\Lambda \leftarrow \gamma_{\overline{\Gamma}_1}$
7: $\quad$**else**
8: $\quad\quad\gamma_\Lambda \leftarrow \gamma_\Lambda \wedge \gamma_{\overline{\Gamma}_i}$ r-conjunction
9: $\quad$**end if**
10: **end for**

convex hull property of the Bézier control points for trimming (Upreti et al., 2014). We first obtain an implicit representation of the convex hull $\gamma_{\mathrm{CH}(\Gamma_i)}$ by applying a pair-wise r-conjunction to the hyperplane functions, see Equation (4.16). Then, we compute its normalized representation $\hat{\gamma}_{\mathrm{CH}(\Gamma_i)}$, see Equation (4.13). In Line 4, we trim the Bézier patch representation $\hat{\gamma}_{\Gamma_i}$ in terms of $\hat{\gamma}_{\mathrm{CH}(\Gamma_i)}$, see Equation (4.14). The obtained representation is denoted by $\gamma_{\overline{\Gamma}_i}$. Finally, in Lines 5-9, we obtain the implicit representation of the model Λ by pair-wise r-conjunction of $\gamma_{\overline{\Gamma}_i}$, see Equation (4.15).

4.5.2 Gradient and Hessian

Next, we present the gradient and Hessian of the geometry implicitization. In Section 4.5.1, we describe the geometry implicitization in terms of the trimming and r-conjunction operations of the convex-hull and Bézier patch normalized representations. Accordingly, we describe in this section the derivatives of the trimming and r-conjunction operations. For completeness, we detail in Appendix B.2 the derivatives of the convex-hull and Bézier patch normalized representations.

As detailed in Section 4.5.1, we perform an r-conjunction operation to obtain the model representation. We compute the derivatives in a straight-forward manner. Lets denote by $\nabla f * \nabla g$ the matrix with coefficients $\partial_j f \partial_k g$ for $j,\ k = 1, ..., d$. Then, the derivatives of the r-conjunction, presented in Equation (4.15), are given by

$$\nabla\ f \wedge g = \nabla f + \nabla g - \nabla\ \sqrt{f^2 + g^2}, \tag{4.17}$$

and

$$\nabla^2 \ f \wedge g = \nabla^2 f + \nabla^2 g - \nabla^2 \ \sqrt{f^2 + g^2}, \tag{4.18}$$

where

$$\nabla \ \sqrt{f^2 + g^2} = \frac{f\nabla f + g\nabla g}{\sqrt{f^2 + g^2}}, \tag{4.19}$$

and

$$\nabla^2 \ \sqrt{f^2 + g^2} = \frac{\nabla f * \nabla f + f\nabla^2 f + \nabla g * \nabla g + g\nabla^2 g}{\sqrt{f^2 + g^2}} - \frac{\nabla\sqrt{f^2 + g^2} * \nabla\sqrt{f^2 + g^2}}{\sqrt{f^2 + g^2}}. \tag{4.20}$$

Following Equation (4.15), we consider that $f := \gamma_\Lambda$ and $g := \gamma_{\overline{\Gamma}}$.

Similarly to the r-conjunction, we compute the derivatives of the trimming operation, presented in Equation (4.14). We simplify the computations by noticing that

$$\tilde{h} := f \text{ trim } h = \sqrt{f^2 + g^2} \quad \text{for} \quad g := \frac{\sqrt{h^4 + f^2} - f}{2},$$

where, following Equation (4.14), we consider $f := \hat{\gamma}_{\mathrm{CH}(\Gamma)}$, $h := \hat{\gamma}_\Gamma$, and $\tilde{h} := \gamma_{\overline{\Gamma}}$. Then, to obtain the derivatives of the trimming operation, we differentiate the term $\sqrt{f^2 + g^2}$, see Equations (4.19) and (4.20). In this case, the derivatives of g can be computed as follows

$$\nabla g = \frac{1}{2}\left(\frac{2h^3\nabla h + f\nabla f}{\sqrt{h^4 + f^2}} - \nabla f\right), \tag{4.21}$$

and

$$\nabla^2 g = \frac{1}{2}\left(\frac{2h^2\left(h\nabla^2 h + 3\nabla h * \nabla h\right) + f\nabla^2 f + \nabla f * \nabla f}{\sqrt{h^4 + f^2}} - \frac{\nabla\sqrt{h^4 + f^2} * \nabla\sqrt{h^4 + f^2}}{\sqrt{h^4 + f^2}} - \nabla^2 f\right),$$

where the term $\nabla\sqrt{h^4 + f^2}$ can be computed from Equation (4.19) for the functions f and h^2.

As we observe, the derivatives of both the r-conjunction and the trimming operation require the derivatives of the convex-hull and Bézier patch normalized representations. For completeness, we detail these last derivatives in Appendix B.2.

4.5.3 Minimizing metric and geometry deviations

In this section, we consider a modification of the methodology to generate curved high-order meshes featuring optimal mesh quality and geometric accuracy presented in Ruiz-Gironés et al. (2016); Ruiz-Gironés and Roca (2022). This technique combines a distortion measure and a geometric L^2-disparity measure into a single objective function. While the element distortion term takes into account the mesh quality, the L^2-disparity term takes into account the geometric error introduced by the mesh approximation to the target geometry. Herein, the target geometry is an implicit representation.

Our input data is a CAD model, Λ, composed of several geometric entities in such manner that

$$\Lambda = \bigcup_{k=1}^{n} \Lambda_k,$$

where each geometric entity is composed of sub-entities. These sub-entities are curves in 2D, and curves and surfaces in 3D. In 3D, we consider that the curves are embedded directly in the containing space.

In our representation, we consider that the curves are the image of a segment. Moreover, we consider that the surfaces are the image of a rectangular region. For the 3D cases, we consider the implicitization of the curves and surfaces. In this manner, we can allow the inner curve (surface) nodes to target the implicitization of the corresponding curve (surface).

In what follows, we propose an entity-wise implicit representation of the CAD model Λ. We use it to measure the geometric deviation between the mesh and the model. In particular, for each geometric entity Λ_k we consider the implicit representation, see Section 4.5. This geometric entity is approximated by a set of boundary mesh entities, denoted by $\partial\mathcal{M}(\Lambda_k)$. Instead of measuring the geometric error, herein we account from the geometric deviation through the average of the square of the level set value. This term is zero when on top of the target CAD entity, and the square ensures deriviability at the zero-level set. Specifically, this deviation measure is integrated over the candidate boundary mesh entities as follows

$$\mathcal{G}\left(\partial\mathcal{M}\left(\Lambda_k\right)\right) := \int_{\partial\mathcal{M}(\Lambda_k)} \gamma^2. \tag{4.22}$$

Note that, the model representation γ_{Λ_k} is not differentiable at the zero level-set. By considering the squared function ${\gamma_{\Lambda_k}}^2$ we avoid the derivative singularity.

Our objective is to determine an optimal physical mesh, $\mathcal{M}$, in terms of mesh quality and geometric deviation. First, the mesh quality deviation term, distortion, is presented in Section 4.3. Second, we consider Equation (4.22) to take into account the geometric deviation. Finally, we define the functional for the mesh quality and the geometric deviation

$$\mathcal{H}\left(\mathcal{M};\lambda\right) := \mathcal{F}\left(\mathcal{M}\right) + \lambda\mathcal{G}\left(\partial\mathcal{M}\right), \tag{4.23}$$

where

$$\mathcal{G}\left(\partial\mathcal{M}\right) := \sum_{k=1}^{n} \mathcal{G}\left(\partial\mathcal{M}\left(\Lambda_k\right)\right),$$

and where λ corresponds to the penalty parameter. This parameter λ can be chosen heuristically or with an automatic procedure (Ruiz-Gironés and Roca, 2022).

To deal with corners and geometric edges, we distinguish between nodes targeting points or curves of the geometry. For points, we associate the corresponding node with the incident curves. Moreover, for this node, the objective function accounts for the measure of the distance to all the incident curves. Thus, the optimal node is close to the target point because it is close to all the incident curves. For curves, we associate the corresponding nodes with the curve and the incident surfaces. Moreover, for these nodes, the objective function accounts for the measure of the distance to the curve and the two incident surfaces. Thus, the optimal nodes are close to the target curve and the two incident surfaces.

In Algorithm 4.2, we outline the structure of the distortion minimization. The algorithm inputs are: a CAD model Λ, a physical mesh $\mathcal{M}$, a background mesh $\hat{\mathcal{M}}$ equipped with a discrete metric $\hat{\mathbf{M}}$, a residual tolerance ε, and a penalty parameter λ. The output is an optimized physical mesh $\mathcal{M}^*$ with the same connectivity of $\mathcal{M}$ and matching the metric $\hat{\mathbf{M}}$ and the curved boundary Λ. To outline the algorithm, we assign variables, and we declare the corresponding functions and their derivatives in terms of previously defined functions and derivatives. We recall that, the implementation details of the values and derivatives of the log-Euclidean interpolation $\mathbf{M}$ and the implicitation γ are detailed in Section 4.4 and Section 4.5, respectively. Note that the derivatives of $\mathcal{F}$ and $\mathcal{G}$ depend on the corresponding derivatives of $\mathbf{M}$ and γ, respectively.

Algorithm 4.2 proceeds as follows. First, we assign the volume and boundary mesh coordinates to $\mathbf{X}$ and $d\mathbf{X}$, respectively. From these coordinates, we declare the Log-Euclidean interpolation of the discrete metric $\hat{\mathbf{M}}$ and its derivatives, $\nabla\hat{\mathbf{M}}$ and $\nabla^2\hat{\mathbf{M}}$,

Algorithm 4.2 Distortion minimization

Input: Λ, $\mathcal{M}$, $\hat{\mathcal{M}}$, $\hat{\mathbf{M}}$, ε, λ
Output: $\mathcal{M}^*$

1: $\mathbf{X} \leftarrow \text{coordinates}(\mathcal{M})$
2: $\partial\mathbf{X} \leftarrow \text{coordinates}(\partial\mathcal{M})$
3: $\mathbf{M} := \mathbf{M}\left(\hat{\mathcal{M}}, \hat{\mathbf{M}}, \mathbf{X}\right)$ ▷ Section 4.4.1
4: $\nabla\mathbf{M} := \nabla\mathbf{M}\left(\hat{\mathcal{M}}, \hat{\mathbf{M}}, \mathbf{X}\right)$; $\nabla^2\mathbf{M} := \nabla^2\mathbf{M}\left(\hat{\mathcal{M}}, \hat{\mathbf{M}}, \mathbf{X}\right)$ ▷ Section 4.4.2
5: $\gamma := \gamma\left(\Lambda, \partial\mathbf{X}\right)$ ▷ Section 4.5.1
6: $\nabla\gamma := \nabla\gamma\left(\Lambda, \partial\mathbf{X}\right)$; $\nabla^2\gamma := \nabla^2\gamma\left(\Lambda, \partial\mathbf{X}\right)$ ▷ Section 4.5.2
7: $\mathcal{F} := \mathcal{F}\left(\mathbf{X}, \mathbf{M}\right)$; ▷ Section 4.3, Equation (4.5)
8: $\nabla\mathcal{F} := \nabla\mathcal{F}\left(\mathbf{X}, \mathbf{M}, \nabla\mathbf{M}\right)$; $\nabla^2\mathcal{F} := \nabla^2\mathcal{F}\left(\mathbf{X}, \mathbf{M}, \nabla\mathbf{M}, \nabla^2\mathbf{M}\right)$
9: $\mathcal{G} := \mathcal{G}\left(\partial\mathbf{X}, \gamma\right)$; ▷ Section 4.5, Equation (4.22)
10: $\nabla\mathcal{G} := \nabla\mathcal{G}\left(\partial\mathbf{X}, \gamma, \nabla\gamma\right)$; $\nabla^2\mathcal{G} := \nabla^2\mathcal{G}\left(\partial\mathbf{X}, \gamma, \nabla\gamma, \nabla^2\gamma\right)$
11: $\mathcal{H} \leftarrow \mathcal{F} + \lambda\mathcal{G}$ ▷ Section 4.5, Equation (4.23)
12: $\mathbf{X}^* \leftarrow \text{Non-linearSolver}\left(\mathcal{H}, \nabla\mathcal{H}, \nabla^2\mathcal{H}, \mathbf{X}, \varepsilon\right)$ ▷ Section 4.3, Equation (4.6)
13: $\mathcal{M}^* \leftarrow$ update coordinates of $\mathcal{M}$ with $\mathbf{X}^*$

see Section 4.4. In addition, from the CAD model Λ, we declare the implicitization γ and its derivatives, $\nabla\gamma$ and $\nabla^2\gamma$, in terms of $d\mathbf{X}$, see Section 4.5. Then, we declare the distortion functional $\mathcal{F}$ and the boundary functional $\mathcal{G}$. For these functionals, we also declare the dependency of their derivatives in terms of the values and derivatives of the metric $\mathbf{M}$ interpolation and the geometry implicitation γ. These declarations allow assigning the objective function $\mathcal{H}$ according to the functionals, $\mathcal{F}$ and $\mathcal{G}$, and the penalty parameter λ, see Equation (4.23). Finally, we call a second-order non-linear solver to minimize the objective function up to a residual tolerance ε. This results in an adapted mesh $\mathcal{M}^*$ with coordinates $\mathbf{X}^*$ and with the same connectivity as $\mathcal{M}$.

4.6 Results

In this section, we present a 2D and a 3D example to illustrate the applicability of our distortion minimization framework for curved r-adaption to a high-order metric interpolation while preserving the implicit representation of the boundary. First, we generate a background mesh $\hat{\mathcal{M}}$ and we evaluate the analytical metric $\mathbf{M}$ at the background mesh nodes. Second, we generate an initial physical mesh $\mathcal{M}$ and we measure its distortion (quality) by interpolating the metric. Then, by relocating the

nodes, we minimize the mesh distortion problem presented in Equation (4.6) using the framework presented in this work. Moreover, in the last examples, we consider a boundary term that takes into account the geometric deviation. We relocate the nodes to minimize the distortion measure while preserving the curved features of the boundary.

To summarize the results, we present a statistics table for the element quality of Equation (4.4), and the figures for the initial and optimized meshes. Specifically, we show the minimum quality, the maximum quality, the mean quality, and the standard deviation of the initial and optimized meshes. We highlight that in all cases, the optimized mesh increases the minimum element quality and it does not include any inverted element. In addition, the meshes resulting after the optimization are composed of elements aligned and stretched to match the target metric tensor. In all figures, the meshes are colored according to the point-wise quality presented in Equation (4.3).

Because our goal is to optimize the mesh distortion using the detailed derivatives, instead of including mathematical proofs of mesh validity, we detail how we numerically enforce the positiveness of the element Jacobians. Specifically, we use a numerical valid-to-valid approach that uses four ingredients. First, because we want numerically valid results, we enforce mesh validity on the integration points. Second, to initialize the optimization, we start from a numerically valid mesh. Third, to penalize inverted elements, we modify the point-wise distortion, Equation (4.3), to be infinity for non-positive Jacobians. Specifically, we regularize the element Jacobians to be zero for non-positive Jacobians, so their reciprocals are infinite. Note that these reciprocals appear in the distortion expression, and thus, they determine the infinite distortion value. Fourth, to enforce numerically valid mesh displacements, we equip Newton's method with a backtracking line-search. Specifically, if the mesh optimization update is invalid in any integration point, the objective function, Equation (4.6), is infinite. In that case, the step is divided by two until it leads to a valid mesh update.

As a proof of concept, a mesh optimizer has been developed in Julia 1.6.2 (Bezanson et al., 2017). For this, we use the following external packages: Arpack v0.5.0, ILUZero v0.1.0, and TensorOperations v3.1.0. In addition, we use the MATLAB PDE Toolbox (MATLAB, 2017) to generate the initial isotropic linear unstructured 2D and 3D meshes (the structured meshes are generated by subdivision), and the MMG al-

gorithm (Dobrzynski, 2012) to generate the initial anisotropic linear unstructured 2D and 3D meshes. To construct the geometric models, we use the FreeCAD software (Riegel et al., 2016). Finally, we use the Quickhull (Qhull) algorithm (Barber et al., 1996) for the convex-hull computations required in the geometric model's implicitization, see Section 4.5.

The Julia prototyping code is sequential, it corresponds to the implementation of the method presented in this chapter and the one presented in Chapter 3. In all the examples, the optimization corresponds to finding a minimum of a nonlinear unconstrained multi-variable function. In particular, the mesh optimizer uses an unconstrained line-search globalization with an iterative preconditioned conjugate gradients linear solver. The stopping condition is set to reach an absolute root mean square residual, defined as $\frac{\|\nabla f(x)\|_{\ell^2}}{\sqrt{n}}$ for $x \in \mathbb{R}^n$, smaller than 10^{-4} or a length-step smaller than 10^{-4}. Each optimization process has been performed in a node featuring two Intel Xeon Platinum 8160 CPU with 24 cores, each at 2.10 GHz, and 96 GB of RAM memory.

Following, we first present the target domains to be meshed, and the considered metrics on the domain, Section 4.6.1. In Section 4.6.2, we present the optimization results for a quadrilateral and a hexahedral domain. In Section 4.6.3, we compare the proposed discrete based-interpolation procedure with the analytical one from Aparicio-Estrems et al. (2018, 2019, 2021). Finally, in Sections 4.6.4 and 4.6.5, we show the application of the discrete metric approach to optimize an anisotropic mesh adapted to a given metric generated by the MMG algorithm. In particular, in Section 4.6.5, we illustrate that our mesh adaption method based in the metric interpolation approach is compatible with curved boundaries.

4.6.1 Domains and metrics

We consider the quadrilateral domain $\Omega = [-0.5, 0.5]^2$ for the two-dimensional examples and the hexahedral domain $\Omega = [-0.5, 0.5]^3$ for the three-dimensional ones. Each domain is equipped with a metric matching a boundary layer. In particular, our target metric $\mathbf{M}$ is characterized by a boundary layer metric with a diagonal matrix $\mathbf{D}$ and a deformation map φ by the following expression

$$\mathbf{M} = \nabla\varphi^{\mathrm{T}}\ \mathbf{D}\ \nabla\varphi. \tag{4.24}$$

In what follows, we first detail the boundary layer metric $\mathbf{D}$ and then the deformation map φ.

The boundary layer aligns with the x-axis (xy-plane) in the 2D case (3D case). It determines a constant unit element size along the x-direction (xy-directions), and a non-constant element size along the y-direction (z-direction). This vertical element size grows linearly with the distance to the x-axis (xy-plane), with a factor $\alpha = 2$, and starts with the minimal value $h_{\min} = 0.01$ ($h_{\min} = 0.02$). Thus, the stretching ratio blends from $1:100$ to $1:1$ (from $1:50$ to $1:1$) between $y = -0.5$ and $y = 0.5$ (between $z = -0.5$ and $z = 0.5$). We define the metric for the 2D case as:

$$\mathbf{D} := \begin{pmatrix} 1 & 0 \\ 0 & 1/h(y)^2 \end{pmatrix} \tag{4.25}$$

where the function h is defined by

$$h(x) := h_{\min} + \alpha|x|.$$

Similarly, the metric for the 3D case is

$$\mathbf{D} := \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1/h(z)^2 \end{pmatrix}. \tag{4.26}$$

The deformation map φ in Equation (4.24) aligns the stretching of $\mathbf{D}$ according to a given curve in the 2D examples and at a given surface in the 3D examples. In the 2D case, we define the map φ by

$$\varphi(x, y) = \left(x, \frac{10y - \cos(2\pi x)}{\sqrt{100 + 4\pi^2}} \right),$$

and, in the 3D case by

$$\varphi(x, y, z) = \left(x, y, \frac{10z - \cos(2\pi x)\cos(2\pi y)}{\sqrt{100 + 8\pi^2}} \right).$$

Figure 4.5 shows the anisotropic quotient (Loseille and Löhner, 2010) of the metric presented in Equations (4.25) and (4.26). Specifically, the anisotropic quotient of a metric tensor $\mathbf{M} \in \mathbb{R}^{d \times d}$ is given by

$$\text{quo} = \max_{i=1,\dots,d} \sqrt{\frac{\det(\mathbf{M})}{\lambda_i^d}}$$

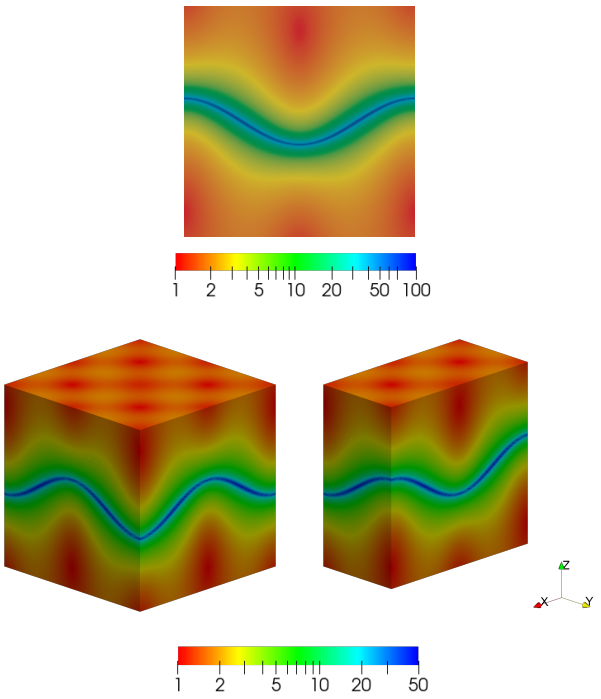


Figure 4.5: Anisotropic quotient values in logarithmic scale of the target metrics: (top) 2D case; (bottom left) boundaries of the 3D case; and (bottom right) solid slice of the 3D case.

where $\lambda_i,\ i = 1, ..., d$ are the eigenvalues of $\mathbf{M}$. The considered metric $\mathbf{M}$ attains the highest level of anisotropy, close to the curve described by the points $(x, y) \in \Omega$ such that $\varphi(x, y) = (x, 0)$ in 2D, and close the surface described by the points $(x, y, z) \in \Omega$ such that $\varphi(x, y, z) = (x, y, 0)$ in 3D.

4.6.2 Distortion minimization: initial isotropic straight-edged meshes

In this example, we present the optimization results for initially isotropic meshes on the domain equipped with the metrics presented in Section 4.6.1. We describe first the initial meshes $\mathcal{M}$ together with the background meshes $\hat{\mathcal{M}}$ where the metric is interpolated. Next, we present the optimized meshes $\mathcal{M}^*$ and to conclude, we present the results obtained from the optimization process. Herein, both the background and physical meshes are meshes of the same polynomial degree.

The initial meshes $\mathcal{M}$ are of polynomial degree 1, 2, and 4. The three meshes feature approximately the same number of nodes and they have approximately the

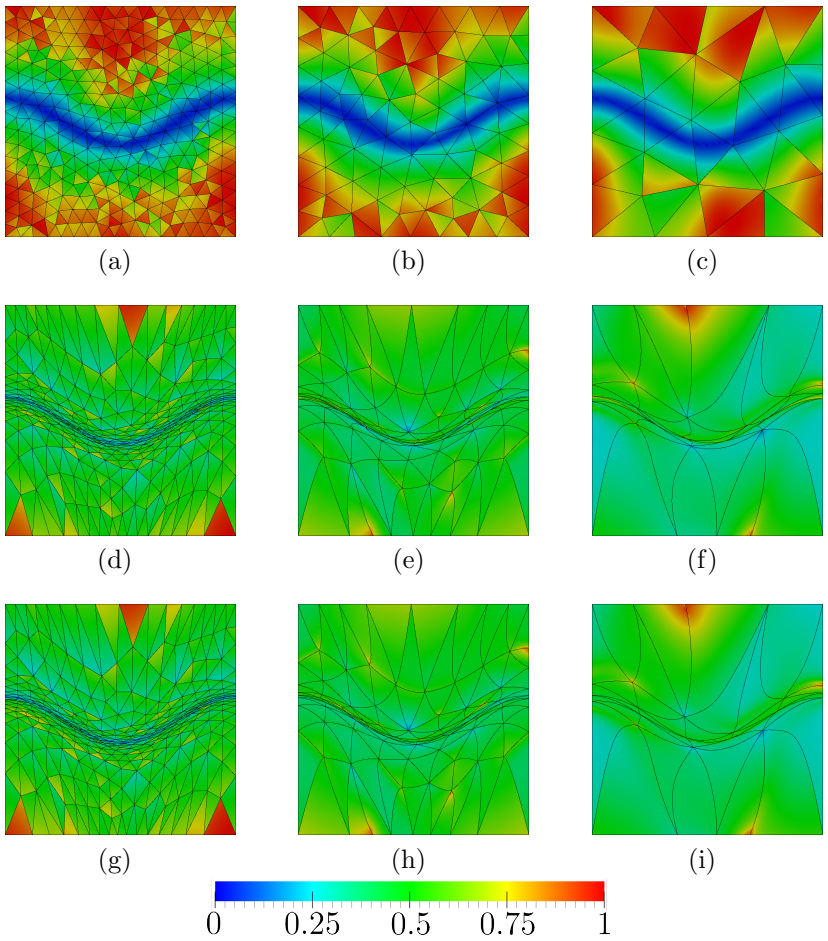


Figure 4.6: Point-wise distortion for triangular meshes of polynomial degree 1, 2, and 4 in columns. Initial straight-sided isotropic meshes, optimized meshes with discrete metric, and optimized meshes with analytic metric in rows.

same resolution over the domain. In particular, in 2D the three initial meshes are respectively composed of 312, 321, and 337 nodes and 558, 144, and 38 triangles, see Figures 4.6(a), 4.6(b), and 4.6(c). In 3D, they are respectively composed of 2 356, 2 362, and 2 373 nodes and 11 699, 1 464, and 184 tetrahedra. Figures 4.7(a), 4.7(b), 4.7(c), and 4.8(a), 4.8(b), 4.8(c) show the clipped 3D meshes and the mesh boundary, respectively. The meshes are colored according to the point-wise stretching and alignment quality measure, presented in Equation (4.3). Points in blue color have low quality and points with red color have high quality. As we observe, the elements lying in the region of highest stretching ratio have less quality than the elements lying in the isotropic region.

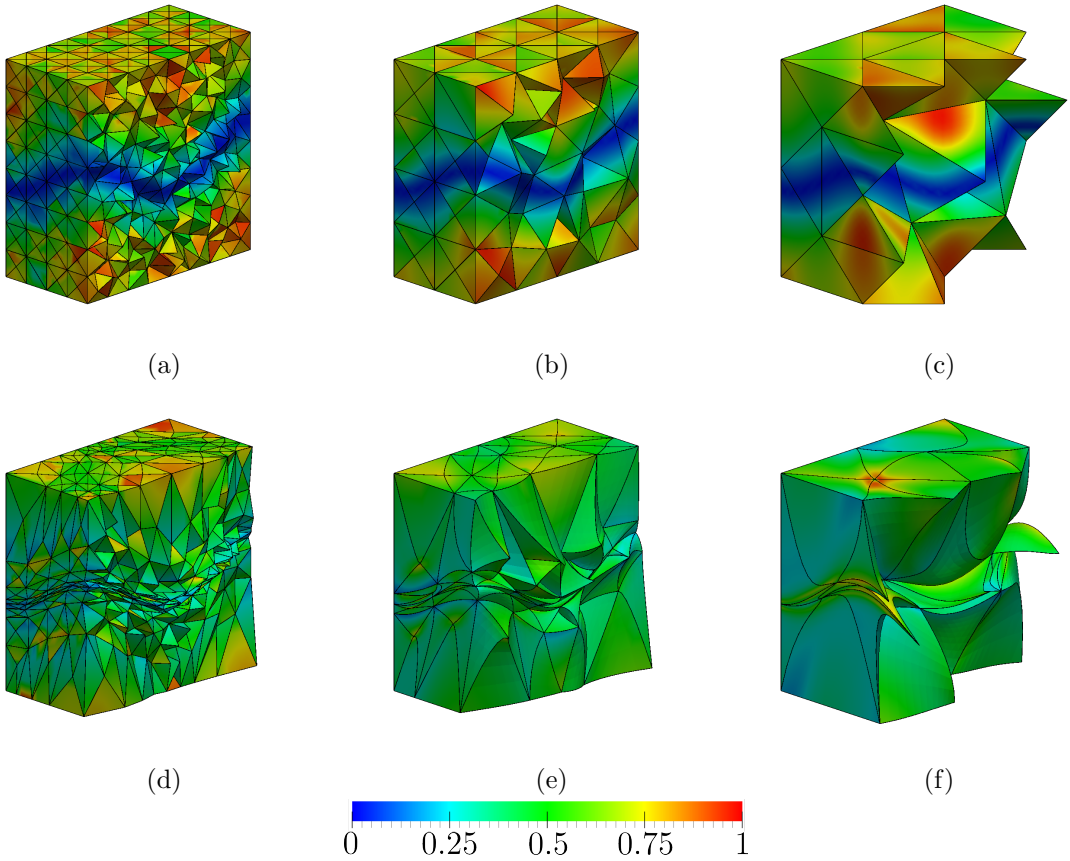


Figure 4.7: Clipped tetrahedral meshes of polynomial degree 1, 2, and 4 in columns. Initial straight-sided isotropic meshes and optimized meshes from initial meshes in rows.

We equip each mesh with the metric presented in Equation (4.24). We obtain the metric values from the log-Euclidean interpolation method presented in Section 4.4. In particular, we interpolate the metrics from a background mesh $\hat{\mathcal{M}}$. The background meshes are of polynomial degree 1, 2, and 4 according to the polynomial degree of the initial meshes $\mathcal{M}$. We impose the three background meshes to feature almost the same number of nodes and to have almost the same resolution over the domain, $h_{\min}/2$. In particular, the resolution of the 2D background meshes is $h_{\min}/2 = 0.005$. They are composed of 65 170, 64 329, and 62 761 nodes and 129 318, 31 910, and 7 782 triangles. The resolution of the 3D background meshes is $h_{\min}/2 = 0.01$. They are composed of 1 773 415, 1 798 531, and 1 837 851 nodes and 10 438 221, 1 319 008, and 168 441 tetrahedra.

To obtain an optimal configuration $\mathcal{M}^*$, we minimize the mesh distortion by

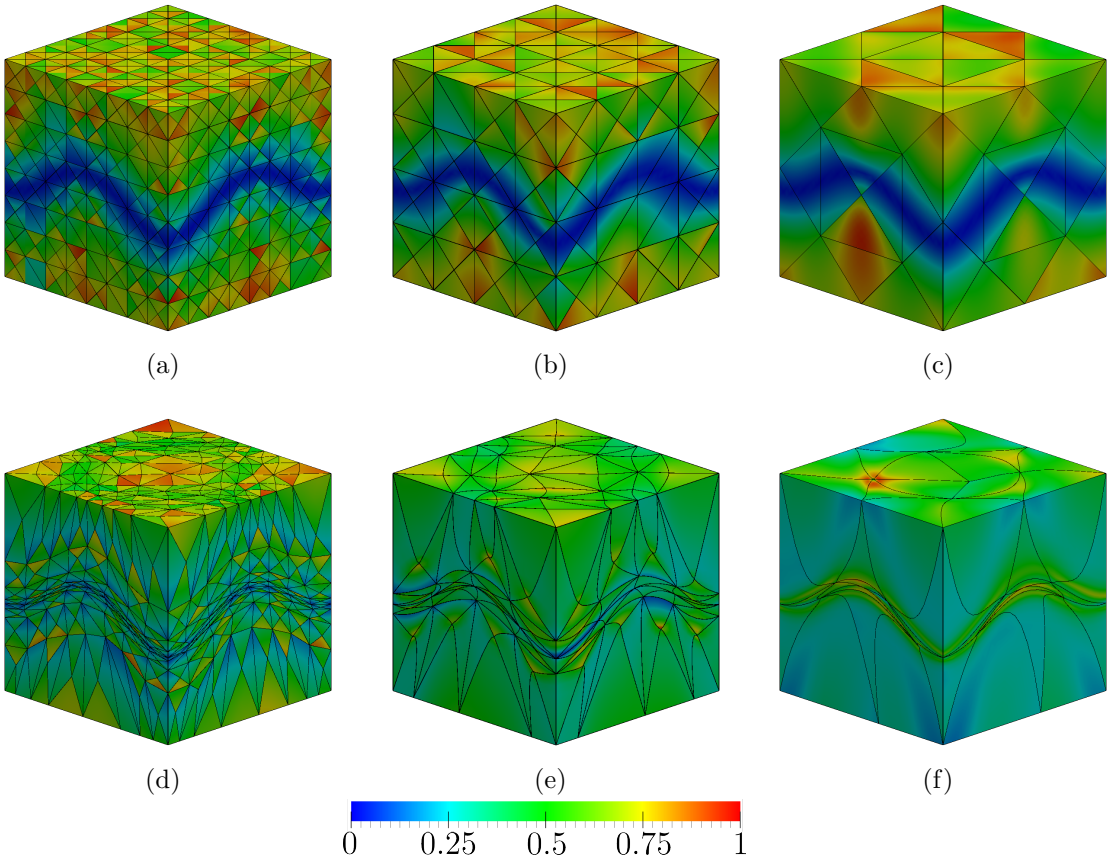


Figure 4.8: Boundary of tetrahedral meshes of polynomial degree 1, 2, and 4 in columns. Initial straight-sided isotropic meshes and optimized meshes from initial meshes in rows.

relocating the mesh nodes while preserving their connectivity, as detailed in Section 4.3. The coordinates of the inner nodes, and the coordinates tangent to the boundary, are the design variables. Thus, the inner nodes are free to move, the vertex nodes are fixed, while the rest of boundary nodes are enforced to slide along the boundary facets of the domain Ω. In Figures 4.6(d), 4.6(e), 4.6(f) we illustrate the optimized 2D meshes. In the 3D case, Figure 4.7(d), 4.7(e), 4.7(f), and 4.8(d), 4.8(e), 4.8(f) show the clipped 3D meshes and the mesh boundary, respectively. We align the axes according to the ones of Figure 4.5. We observe that the elements lying in the anisotropic region are compressed to attain the stretching and alignment prescribed by the metric.

Tables 4.1 and 4.2 show the quality statistics of both the initial and optimized meshes for the 2D and 3D cases, respectively. In all the optimized meshes the mini-

Table 4.1: Quality statistics for the initial and optimized meshes with interpolated 2D metric.

Mesh	Minimum		Maximum		Mean		Std dev.	
deg.	Initial	Final	Initial	Final	Initial	Final	Initial	Final
1	0.0299	0.1724	0.9957	0.9551	0.6100	0.4462	0.2769	0.1039
2	0.0554	0.2878	0.9921	0.6268	0.5918	0.4545	0.2835	0.0638
4	0.0803	0.3072	0.9835	0.5806	0.5339	0.4439	0.2922	0.0760

Table 4.2: Quality statistics for the initial and optimized meshes with interpolated 3D metric.

Mesh	Minimum		Maximum		Mean		Std dev.	
deg.	Initial	Final	Initial	Final	Initial	Final	Initial	Final
1	0.0175	0.1222	0.9905	0.9334	0.5550	0.4236	0.2660	0.1241
2	0.0320	0.2987	0.9695	0.7467	0.5194	0.4576	0.2735	0.0691
4	0.0409	0.3231	0.8931	0.6737	0.4490	0.4702	0.2711	0.0749

mum is improved and the standard deviation of the element qualities is reduced when compared with the initial configuration. In addition, when comparing the curved meshes with the straight-edged ones, we observe that the curved meshes are more flexible. That is, the curved meshes achieve a higher improvement of the minimum quality and the standard deviation. This is because the curved elements can approximate the curved stretching of the metric in the point-wise sense and hence, more accurately.

4.6.3 Validation: analytic versus discrete

To validate the proposed method, we compare 2D curved r-adaption results for the high-order metric interpolation with the results corresponding to an analytic metric evaluation. Considering the initial meshes presented in the previous section, we optimize the distortion measure by evaluating the analytical metric expression, instead of interpolating it in the background mesh. In Figure 4.6 we show the initial and optimized meshes. They are colored according to the point-wise quality measure of Equation (4.3) using the analytical metric expression.

To compare quantitatively both results, we compute the maximum distance of the node coordinates of the optimized configurations. The maximum distances are around $2.2 \cdot 10^{-2}$, $7.6 \cdot 10^{-2}$, and $8.2 \cdot 10^{-2}$ for the linear, quadratic, and quartic cases, obtaining comparable nodal configurations, as it can be observed when comparing

Table 4.3: Quality statistics for the initial and optimized meshes with analytic 2D metric.

Mesh	Minimum		Maximum		Mean		Std dev.	
deg.	Initial	Final	Initial	Final	Initial	Final	Initial	Final
1	0.0279	0.1684	0.9957	0.9581	0.6100	0.4484	0.2770	0.1088
2	0.0563	0.3358	0.9921	0.6432	0.5919	0.4569	0.2835	0.0623
4	0.0799	0.3096	0.9835	0.6318	0.5339	0.4473	0.2923	0.0634

Figures 4.6(d), 4.6(e), and 4.6(f) with Figures 4.6(g), 4.6(h), and 4.6(i), respectively. In Table 4.3, we present the quality statistics of the initial and optimized meshes using the analytical metric evaluation. To compare the quality improvement of both approaches, we compute the difference between the mean of the analyzed quality statistics, obtaining a value below 10^{-2}. Thus, the quality improvement driven by the optimization using the proposed metric interpolation procedure is analogous to the one given by the analytical metric, obtaining in all cases high-quality configurations with a minimum quality over 0.1.

4.6.4 Distortion minimization: initial anisotropic straight-edged meshes

The results presented in Section 4.6.2 show the application of the metric interpolation procedure to optimize isotropic meshes in a domain equipped with a metric. However, in practice, anisotropic meshes are generated combining topological mesh operations that modify the mesh connectivity and mesh r-adaption procedures (Alauzet and Loseille, 2016). To illustrate a practical example, we consider an initial anisotropic straight-sided mesh. Then, we apply the anisotropic r-adaption method presented in this work.

Although we generate meshes adapted to a target metric with MMG (Dobrzynski, 2012), our goal is not to compare the distortion minimization with the MMG package. Actually, we acknowledge MMG because it generates an initial straight-edged mesh that matches the stretching and alignment of the target metric.

First, we consider the target metric presented in Equation (4.24) with $h_{\min} = 0.01$. Second, we generate a linear isotropic triangular background mesh $\hat{\mathcal{M}}$ of input size $h_{\min}/2 = 0.005$ with MATLAB. We normalize the target metric according to the size of the physical meshes $\mathcal{M}$ namely, 0.0625, 0.125, and 0.25 for the linear, quadratic,

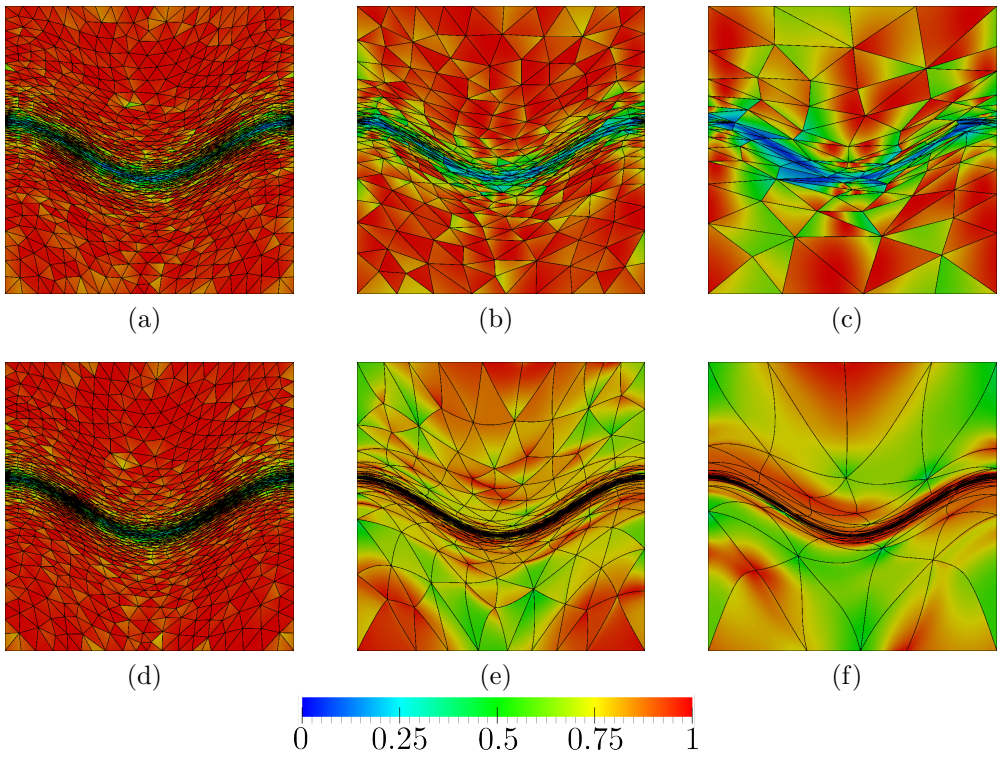


Figure 4.9: Point-wise distortion for triangular meshes of polynomial degree 1, 2, and 4 in columns. Initial straight-sided anisotropic meshes and optimized meshes from initial meshes in rows.

and quartic case, respectively. These sizes are chosen in order to obtain a comparable mesh resolution according to the mesh polynomial degree. Then, we couple each background mesh with the target metric evaluated at the background mesh vertices. We apply the MMG algorithm to obtain an initial straight-sided anisotropic physical mesh $\mathcal{M}$ of polynomial degree 1, 2, and 4, see Figures 4.9(a), 4.9(b), and 4.9(c). In particular, the physical meshes are composed by 1 161 nodes and 2 137 triangles, 1 333 nodes and 624 triangles and, 1 525 nodes and 180 triangles, respectively.

The physical meshes $\mathcal{M}$ are then optimized using the metric interpolation approach presented in this work. In Figures 4.9(d), 4.9(e), and 4.9(f), we illustrate the optimized meshes $\mathcal{M}^*$. We observe that the elements lying in the anisotropic region are compressed to attain the stretching and alignment prescribed by the metric.

In Table 4.4, we show the quality statistics of both the initial and optimized meshes. In all the optimized meshes the minimum is improved and the standard deviation of the element qualities is reduced when compared with the initial config-

Table 4.4: Quality statistics for the initial MMG and optimized meshes with interpolated 2D metric.

Mesh	Minimum		Maximum		Mean		Std dev.	
deg.	Initial	Final	Initial	Final	Initial	Final	Initial	Final
1	0.0365	0.1794	0.9988	0.9989	0.7806	0.7961	0.2273	0.2040
2	0.0624	0.6300	0.9982	0.9913	0.6966	0.8692	0.2558	0.0788
4	0.0424	0.6063	0.9774	0.9965	0.5677	0.9137	0.2681	0.0886

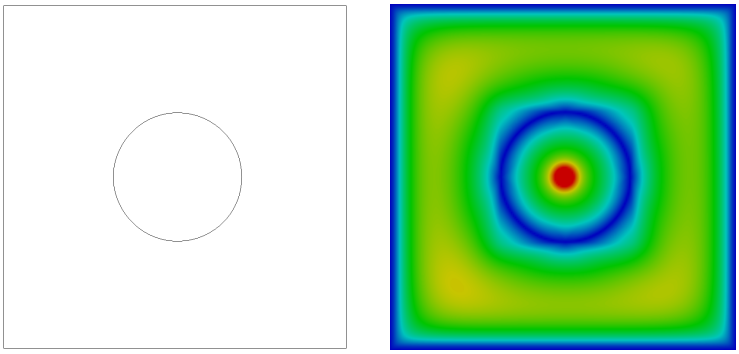
uration. We conclude that, with the same metric data and hence, the same inputs, the r-adaption mesh post-processing improves the quality of the meshes generated with the MMG algorithm. In addition, for the straight-edged case, we have presented a global method to improve the stretching and alignment prescribed by the metric after applying an h-adaption approach.

For a fixed metric, usually the better the initial straight-edged mesh is, the better the optimized mesh is. For instance, for different degrees, the mean quality statistics for the initial anisotropic meshes, Table 4.4, are better than for the isotropic meshes, Table 4.1. The anisotropic meshes have this advantage because their topology and geometry are adapted to match the corresponding scaling of the target metric. This prior metric matching facilitates that the curved optimization reaches a better final quality.

As in the examples presented in Section 4.6.2, when comparing the curved meshes with the straight-edged ones, we observe that the curved meshes are more flexible. That is, the curved meshes achieve a higher improvement of the minimum quality and the standard deviation. This is because the curved elements can approximate the curved stretching of the metric in the point-wise sense and hence, more accurately.

4.6.5 Distortion minimization: curved boundaries

We following illustrate that our approach is compatible with curved boundaries. We consider a 2D example, in Section 4.6.5.1, and a 3D example, in Section 4.6.5.2. To this end, we first construct the geometric model with FreeCAD (Riegel et al., 2016). Next, we consider their implicit representation, see Section 4.5. Then, we generate the background and initial physical meshes coupled with a discrete metric, see Section 4.4. Finally, we apply our r-adaption method, presented in Section 4.3, by taking into account both the discrete metric and the implicit representation of the



(a) (b)

Figure 4.10: Parametric CAD and global implicit representation for the 2D model of a square with a circular hole.

geometry. This enables an optimized physical mesh that approximates the stretching and alignment of the metric while preserving the curvature of the boundary.

To accommodate the curved boundaries we include, to the presented functional, a boundary term that takes into account the mesh deviation to the boundaries of the domain, see Section 4.5.3. Specifically, we set the penalty parameter $\lambda := 10^4$ in all examples, see Equation (4.23). In addition, to approximate the metric stretching, we optimize the mesh using the metric interpolation approach presented in this work. Finally, when optimizing the mesh functional all mesh nodes coordinates are free that is, each mesh node moves in $\mathbb{R}^2$, in the 2D case, and in $\mathbb{R}^3$, in the 3D case.

4.6.5.1 2D curved model: square with a circular hole

For the 2D model Λ_1, we consider a square with a circular hole. Specifically, the domain is denoted by $\Omega_1 = K_1 \backslash C_1$, where $K_1 = [-0.5, 0.5]^2$ is a square, and where C_1 is the circle with radius equal to 0.18 and centered at the origin, see Figure 4.10(a). The domain Ω_1 has two boundaries, the one of the square K_1 and the one of the circle C_1. We illustrate in Figure 4.10(b) a global implicit representation of the boundary $\Lambda_1 := \partial\Omega_1$, using the method presented in Section 4.5.1. Although the inner boundary is smooth, the outer boundary contains sharp features such as corners.

We equip the domain Ω_1 with the target metric presented in Equation (4.24) with $h_{\min} = 0.01$. Then, we generate with MATLAB two isotropic triangular background meshes $\hat{\mathcal{M}}$ of polynomial degree 2 and 4. They have an input resolution $h_{\min}/2 = 0.005$ over Ω_1 that is, of input size 0.01 and 0.02, respectively. We normalize the

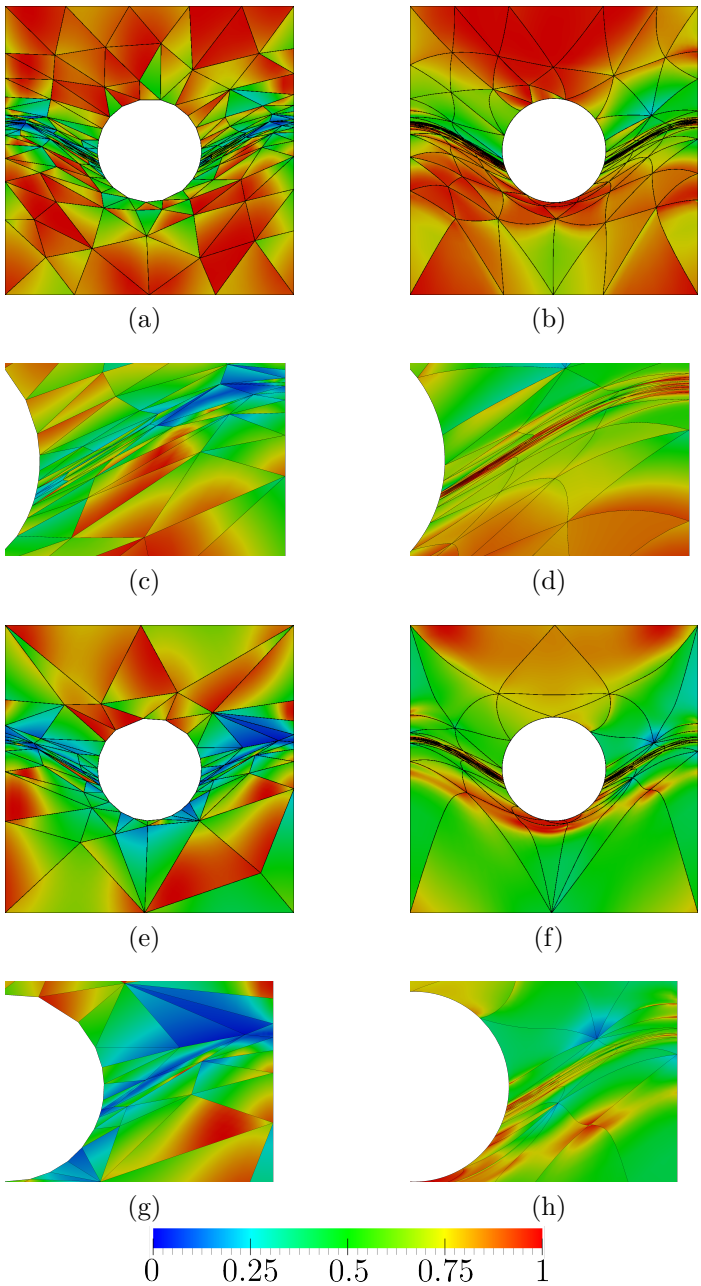


Figure 4.11: Point-wise distortion for triangular meshes of polynomial degree 2 in first and second (zoom) rows, and 4 in third and fourth (zoom) rows. Initial straight-sided anisotropic mesh and optimized mesh in columns.

Table 4.5: Quality statistics for the initial MMG and optimized mesh with interpolated 2D metric at the square with a circular hole.

Mesh	Minimum		Maximum		Mean		Std dev.	
deg.	Initial	Final	Initial	Final	Initial	Final	Initial	Final
2	0.0823	0.4140	0.9914	0.9943	0.5764	0.8224	0.2508	0.1281
4	0.0590	0.4045	0.9646	0.9850	0.4177	0.7321	0.2292	0.1461

target metric according to size $h = 0.25$ in the quadratic case, and according to size $h = 0.5$ in the quartic case. Then, we couple each background mesh with the target metric evaluated at the background mesh vertices. From each background mesh $\hat{\mathcal{M}}$, we obtain an initial straight-sided anisotropic physical mesh $\mathcal{M}$ by applying the MMG algorithm, see Figures 4.11(a), and 4.11(e). The quadratic and quartic physical meshes are respectively composed by 518 nodes and 220 triangles, and 944 nodes and 106 triangles. Note that, since the MMG algorithm requires a linear background mesh, we subdivide the background meshes in order to preserve their resolution. Specifically, our linear background meshes for the MMG algorithm are obtained by subdividing the quadratic background mesh once, and the quartic background mesh twice.

In Figures 4.11(b), and 4.11(f), we illustrate the optimized meshes $\mathcal{M}^*$. We observe that the elements lying in the anisotropic region are compressed to attain the stretching and alignment prescribed by the metric. Note that the boundary elements are curved to match both the metric and the curved domain boundaries. In Table 4.5, we show the quality statistics of both the initial and optimized mesh. In the optimized mesh the minimum, the mean, and the standard deviation of the element qualities are improved when compared with the initial configuration.

From the results, we observe that, when compared with straight-sided elements, curved elements approximate more faithfully the metric while preserving the curved features of the boundary. In this case, the stretching direction is almost aligned according to the tangent of the geometry. When considering straight-edged elements, in Figures 4.11(c) and 4.11(g), accumulating more degrees of freedom in the stretched regions may worsen the boundary representation at non-stretched regions. In contrast, when considering curved elements, in Figures 4.11(d) and 4.11(h), we observe that a single curved element represents the boundary more faithfully than several straight-sided elements. This flexibility of curved elements allows the degrees of freedom to slide and accumulate, from non-stretched regions to the stretched regions, featuring

high-quality elements. For that reason, we observe how the elements are stretched, aligned, and curved according to the stretching and alignment of the metric. Hence, curved elements allow an improved representation of the metric while preserving the curved features of the boundary.

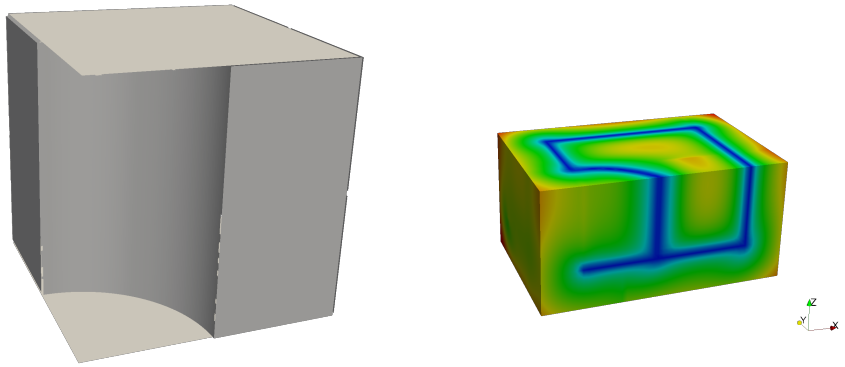
We use a non-optimized prototype to demonstrate that the detailed derivatives enable Newton's method. Nevertheless, to illustrate the computational cost, we next report the wall-clock time and the most expensive parts when matching a target metric and curved boundary. The report is an initial reference for future improvements because the prototype is unoptimized.

For this two-dimensional example, the total wall-clock time is 2 194 seconds for degree two and 17 911 seconds for degree four. The wall-clock time is higher for the second case because of two main reasons: the number of mesh points and the polynomial degree.

First, the mesh features more points for degree four (944 points) than for degree two (518 points). Note that both cases are initialized with a straight-edged mesh adapted to the corresponding scaling of the metric. This scaling accounts for the difference of points between an element of degree two and an element of degree four. Unfortunately, the resulting adapted straight-edged mesh features 220 and 106 elements for degrees two and four, respectively. Thus, the initial meshes do not feature a comparable number of points, a difference that computationally benefits the example of degree two.

Second, the higher the order, the higher the computational cost is. For higher orders, the Hessians of the objective function densify, and the initial approximations worsen. Regarding density, note that the elemental contributions to the Hessian have around six times more non-zero entries for degree four than for degree two. In this example, computing each elemental contribution to the Hessian needs 0.15 seconds for degree four and 0.03 seconds for degree two. Regarding initial approximations, they are worse because the initial straight-edged mesh is of degree one, and thus, the difference of degrees is higher for degree four. In this example, the non-linear problem needs 693 iterations for degree four and 229 iterations for degree two.

Finally, for both degrees, the most expensive part is to compute the elemental contributions to the gradient and the Hessian, a computation that needs the derivatives of the metric interpolation and the geometry implicitation. For the metric interpolation, the percentage of the total wall-clock time computing the derivatives



(a) (b)

Figure 4.12: Parametric CAD and sliced global implicit representation for the 3D model of a cube trimmed by a cylinder.

is 45

4.6.5.2 3D curved model: a cube trimmed by a cylinder

For the 3D model Λ_2, we consider a cube trimmed by a cylinder. Specifically, our domain is denoted by $\Omega_2 = K_2 \backslash C_2$ where $K_2 = [-0.5, 0]^2 \times [-0.25, 0.25]$ is a box, and where C_2 is the cylinder with radius equal to 0.25, height equal to $1/2$, and centered at the origin, see Figure 4.12(a). The boundary of the domain Ω_2 is composed of seven curves and seven surfaces. Six surfaces correspond to the cube K_2 and one correspond to the cylinder C_2. Six curves correspond to the boundary curves of each surface boundary of the cube, and one curve correspond to the intersection of the surface boundary of the cylinder C_2 with the cube. We illustrate in Figure 4.12(b) a global implicit representation of the boundary $\Lambda_2 := \partial\Omega_2$, using the method presented in Section 4.5.1. Although the inner boundary is smooth, the outer boundary contains sharp features such as corners and sharp edges.

We equip the domain Ω_2 with the target metric presented in Equation (4.24) with $h_{\min} = 0.02$. Then, we generate with MATLAB a quadratic isotropic tetrahedral background mesh $\hat{\mathcal{M}}$ of input resolution $h_{\min} = 0.02$ over Ω_2 that is, of input size 0.04. We normalize the target metric according to size $h = 0.5$. Then, we couple each background mesh with the target metric evaluated at the background mesh vertices. From this background mesh $\hat{\mathcal{M}}$, we obtain an initial quadratic straight-

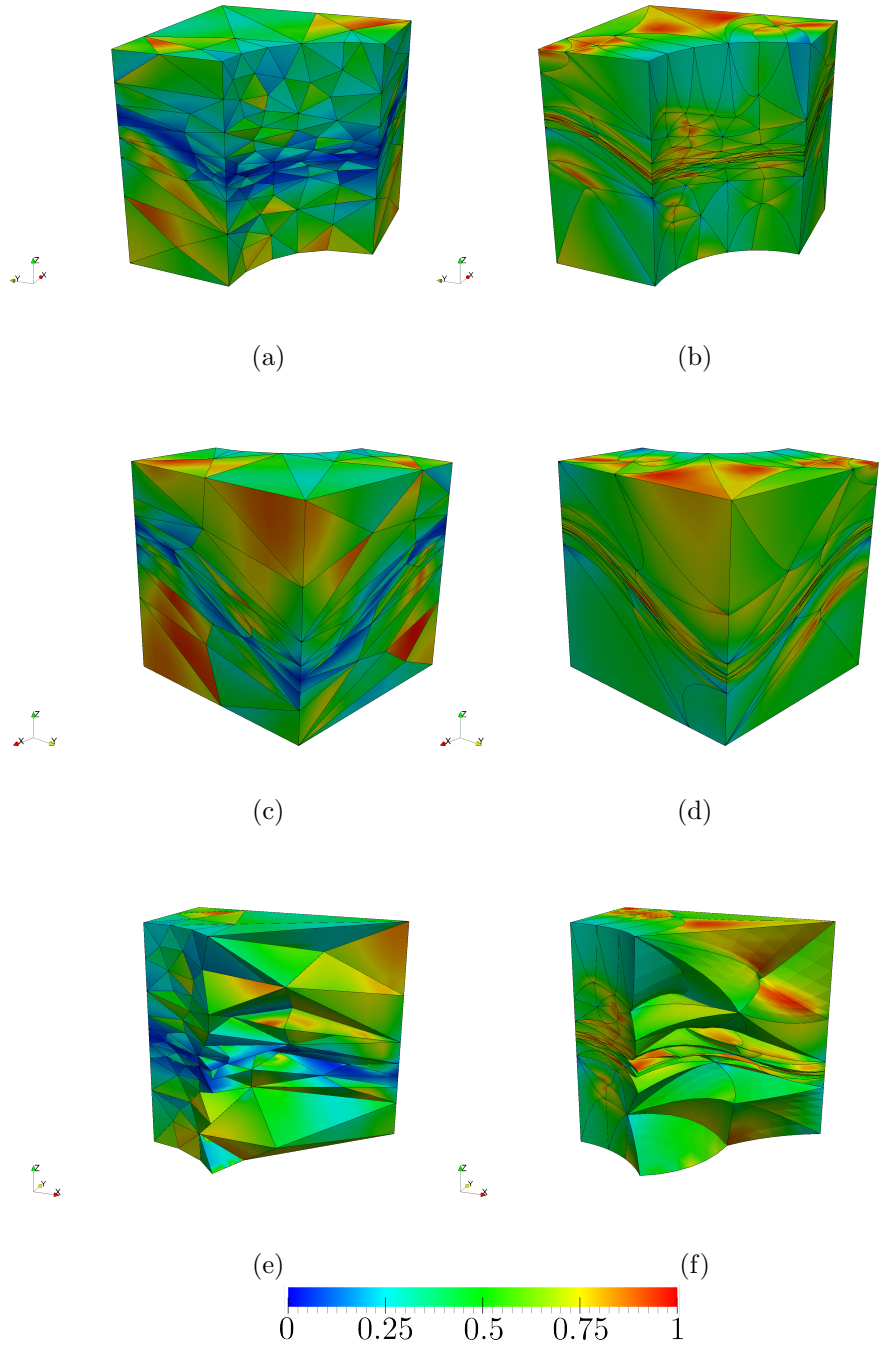


Figure 4.13: Point-wise distortion for quadratic tetrahedral meshes. Initial straight-sided anisotropic mesh and optimized mesh in columns.

Table 4.6: Quality statistics for the initial MMG and optimized mesh with interpolated 3D metric at the cube trimmed by a cylinder.

Mesh	Minimum	Maximum	Mean	Standard deviation
Initial	0.0506	0.9489	0.3519	0.1874
Optimized	0.3315	0.9198	0.6661	0.1144

sided anisotropic physical mesh $\mathcal{M}$ by applying the MMG algorithm, see Figures 4.13(a), 4.13(c), and 4.13(e). The physical mesh is composed by 1 261 nodes and 695 tetrahedra. Note that, since the MMG algorithm requires a linear background mesh, we subdivide once our quadratic background mesh in order to preserve its resolution.

In Figures 4.13(b), 4.13(d), and 4.13(f), we illustrate the optimized meshes $\mathcal{M}^*$. We observe that the elements lying in the anisotropic region are compressed to attain the stretching and alignment prescribed by the metric. Note that the boundary elements are curved to match both the metric and the curved domain boundaries. In Table 4.6, we show the quality statistics of both the initial and optimized mesh. In the optimized mesh the minimum, the mean, and the standard deviation of the element qualities are improved when compared with the initial configuration.

From the results, we observe that, when compared with straight-sided elements, curved elements approximate more faithfully the metric while preserving the curved features of the boundary. In this case, the stretching direction and the curvature of the geometry are independent. Accordingly, when considering straight-edged elements, in Figure 4.13(e), more stretched elements may enable a lower resolution of the boundary. That is, the achieved resolution of the boundary limits the achieved stretching, and vice-versa. In contrast, when considering curved elements, in Figure 4.13(f), we observe that more degrees of freedom can be accumulated at the stretched directions while preserving the curved features of the boundary. As before, we conclude that curved elements allow an improved representation of the metric while preserving the curved features of the boundary.

4.7 Concluding remarks

In conclusion, we have obtained unprecedented second-order optimization results in curved r-adaption to a metric and geometry targets. We have represented the discrete metric in a curved background mesh as a high-order log-Euclidean metric interpola-

tion. For this metric interpolation, we have detailed the first and second derivatives in terms of the physical coordinates. Moreover, we have considered the geometry model as an implicit representation of the NURBS entities. For this implicit representation, we have detailed the first and second derivatives.

The derivatives of the metric interpolation and the implicit representation have allowed minimizing the objective function with Newton's method, an objective function that accounts for the metric and geometry deviations. The discrete metric results compare well with the analytic metric results. In all the results, the method exploits the non-constant Jacobian of curved high-order elements. This mechanism allows the technique to simultaneously match curved features of the metric and the geometry.

To meet our goal, we have enabled Newton's method for curved r-adaption. Nevertheless, we have planned new directions and improvements for the near future. First, to demonstrate the applications of our method and the advantages of adapted curved meshes, we have planned to r-adapt the curved meshes to the steady state of inviscid flows. At this point, we cannot obtain the required discrete metrics because we need to implement existing goal-oriented error estimators for high-order methods (Yano and Darmofal, 2012; Coulaud and Loseille, 2016b). Second, we have demonstrated a key ingredient for curved r-adaption. Nevertheless, combining curved r-adaption with curved h-adaption might be more efficient. To illustrate this combination, we have used an external straight-edged adaptive mesher. However, to properly match the requirements of high-order methods in h-adaption, it is mandatory to use local cavity operators for curved meshes. Regarding these curved operators, we have planned to combine existing approaches (Zhang et al., 2018; Zahr et al., 2020; Rochery and Loseille, 2021; Feuillet et al., 2020) with our approaches. Specifically, our distortion minimization for high-order metric and curved boundaries can also optimize a local cavity. To this end, we will match the cavity interior to the target high-order metric while the old cavity boundaries represent the target geometry.

Chapter 5

Conclusions and future work

In this thesis we have demonstrated metric-aware optimization of high-order meshes on curved geometry with the mesh coordinates as design variables. To this end, we have fulfilled the following objectives: evaluating in an optimizable manner, between a curved high-order mesh and a target metric, not only the shape and orientation (Aparicio-Estrems et al., 2018), but also the size matching (Chapter 2); optimizing a curved high-order mesh to tightly match a non-uniform anisotropic target metric (Chapter 3); accounting and optimizing for a discrete metric (Aparicio-Estrems et al., 2022) while simultaneously targeting a curved geometry (Chapter 4).

To optimize and curve the high-order meshes according to a target metric and a target geometry, we have used mathematical formulations and derivations, design of computational methods, heuristics, computer implementations, run-time checks, and verification approaches.

There are two key central findings in this thesis. First, to enforce unitary Riemannian lengths for all the mesh entities on point-wise varying metrics, we have needed to define a point-wise metric-aware distortion measure accounting for the shape, orientation, and size. Using an entity-wise metric-aware measure we could only enforce unitary Riemannian measures for that type of entity. Second, to solve problems with non-uniform anisotropic point-wise metrics featuring curved sharp features, we have needed to propose a specific-purpose non-linear solver. Without this solver we could only demonstrate metric-aware optimization of curved high-order meshes for simpler metrics.

The work carried out in this thesis leaves open some research activities that should

be performed in the near future. First, the presented method could be applied for a goal-oriented error estimator, *e.g.*, corresponding to the numerical solution of a flow problem. In this case, a metric-based error estimator would be obtained from the reconstructed high-order derivatives of the numerical solution of a non-linear PDE. Then, the distortion minimization could enable an adapted mesh according to the numerical solution. Second, the method could be applied to adapt a curved high-order mesh according to a target surface. For this, the input metric corresponds to the Riemannian surface metric. Third, the geometry implicitization derivatives could be used to trace rays according to an input geometric model. This could be done by applying the Newton method to the presented geometry implicitization. Fourth, the r-adaptive mesh distortion minimization method could be coupled with h-adaptive techniques, such as local cavity operators or local bisection refinement. Fifth, the efficiency of the mesh distortion minimization could be further improved by considering a distributed implementation. In this case, the elemental contributions of the Hessian matrix would be computed in a parallel form, accelerating the computational runtime. Finally, the presented solver might be helpful in minimization problems where the non-linear objective is indefinite for initial approximations out of the positive definite region surrounding the local minima.

Nevertheless, we have contributed to error-driven curved high-order adaptivity. Specifically, we have proposed methods that feature the advantages of metric-driven curved adaptivity yet for high-polynomial degree, enforcing unitary Riemannian volumes, and specifically solving for curved sharp features.

In perspective, our methods for metric-aware optimization of curved high-order meshes of high-polynomial degree will be a key ingredient in error-driven curved high-order adaptivity, an adaptation that enhances the simulation accuracy in problems where the solution presents sharp curved features. In this case, the mesh topology will be modified by curved local cavity operators or curved local bisection refinements. Then, on these meshes, our methods will modify the coordinates of the whole mesh or the local mesh cavity to precisely match the sharp curved features.

Appendix A

Newton-CG solver: details and tests metrics

A.1 Standard CG

The CG method is an iterative procedure to solve linear systems of equations. As many other iterative solvers, it starts from an initial guess. Then, it generates a sequence of approximate solutions, derived from the previous ones, which in the limit are supposed to converge to an analytical solution. In this section, we present the algorithmic details of the standard preconditioned CG method (Saad, 2003; Dembo and Steihaug, 1983).

Consider the CG method, presented in Algorithm A.1, at the k-th non-linear iterate x_k, applied to the preconditioned version of the linear system of Newton Equation (3.7) with a preconditioner M_k of the Hessian matrix $\mathrm{H}f(x_k)$

$$M_k \mathrm{H}f(x_k) p_k = -M_k \nabla f(x_k).$$

We denote as p_k^i the direction corresponding to the i-th iteration of the CG method and similarly for the CG-residual r and the CG-step d. The input arguments are the Hessian matrix $H_k = \mathrm{H}f(x_k)$, the gradient vector $g_k = \nabla f(x_k)$, an initial guess p_k^0 which in this work is set to be equal to the zero vector $\mathbf{0}$, the maximum number of iterations $i_{\max}$, the residual forcing value η_k, the curvature forcing value τ_k and the preconditioner function $z = \mathrm{preconfun}(r)$ which solves the linear system $M_k z = r$. In Line 2, we setup the main variables: the CG-step d, the CG-step multiplier β, the

Algorithm A.1 Conjugate Gradients (Dembo and Steihaug, 1983)

Input: $H,\ g,\ p,$ preconfun, $i_{\max},\ \eta,\ \epsilon$
Output: $p^*,\ d^*$

```
1:  procedure CG
2:      d ← 0, β ← 0, r ← −g − Hp, i ← 1
3:      z ← preconfun(r)
4:      while i ≤ i_max do
5:          z̃ ← z, r̃ ← r
6:          d ← z + βd
7:          if d^T H d < ε d^T d then
8:              p* ← p, d* ← d
9:              return
10:         else
11:             α ← r^T z / d^T H d
12:             p ← p + αd, r ← r − αHd
13:         end if
14:         z ← preconfun(r)
15:         if ‖r‖ < η‖g‖ then
16:             p* ← p
17:             return
18:         end if
19:         β ← r^T z / r̃^T z̃
20:         i ← i + 1
21:     end while
22: end procedure
```

residual $r_k^0 := -\nabla f(x_k) - \mathrm{H}(x_k)p_k^0$ and, the current iteration value i. In Lines 3 and 14, the function preconfun is applied to a vector r. Then, in Line 4, we proceed to the main loop. We compute a step d_k^i at each CG-iteration $i \geq 1$, Line 6, providing a new direction $p_k^i = p_k^{i-1} + \alpha_k^i d_k^i$, Line 12, and a residual,

$$r_k^i := r\left(p_k^i; x_k\right) := -\nabla f(x_k) - \mathrm{H}(x_k)p_k^i.$$

The main loop iterates while d_k^i is a negative curvature direction that is, $d^{\mathrm{T}}Hd < 0$, Line 7, the imposed tolerance is achieved by the residual r_k^i, Line 15, or the iteration has exceeded the limit permitted, Line 4. Finally, the outputs of the algorithm are the CG-point p^* and the CG-step d^*.

The CG method is designed for positive definite systems $\mathrm{H}f(x)$, which usually appear at points x near an optimum x^*. However, for points far from an optimum the Hessian $\mathrm{H}f$ may not be positive definite. Then, as in the standard CG-algorithm,

we terminate the CG-iteration, in Line 7 with $\epsilon = 0$, whenever a CG-step d_k^i of negative curvature is encountered. In this case, we provide the last direction p_k^{i-1}, see Nocedal and Wright (2006).

It could happen that at the first CG-iteration the algorithm stops because a CG-step of negative curvature is encountered. In such case, the CG method returns the *scaled steepest-descent direction* presented in Bellavia and Berrone (2007) given by

$$M_k p_k = -\nabla f(x_k).$$

A.2 Setting c_{max}: quadratic convergence without line-search iterations

In what follows, we propose the value of the non-constant parameter $c_{\max}$. It is required for the sufficient-progress condition of the specific-purpose LS globalization strategy, see Section 3.3.2. For this reason, we choose a value that preserves the main features of a second-order Newton method.

We propose to set $c_{\max} = 0.25$ to obtain quadratic convergence near a minimizer x^* without additional line-search iterations. To this end, two conditions are required. First, it is required that the current point x is sufficiently near a minimizer x^* so the second order model presented in Equation (3.5) is a faithful approximation. Second, it is required that step is an exact approximation to the Newton direction so it satisfies the Newton Equation presented in Equation (3.7). Notice that, the Newton direction has step length equal to one in the region of quadratic convergence. Then, by applying the Newton Equation in the second order term of the quadratic model and, assuming that the step length of the Newton direction is one as desired, we obtain the equation

$$\begin{aligned} f(x+s) &\approx f(x) + s^{\mathrm{T}}\nabla f(x) + \frac{1}{2}s^{\mathrm{T}}\mathrm{H}f(x)s \\ &\overset{s=-\mathrm{H}f(x)^{-1}\nabla f(x)}{=} f(x) + s^{\mathrm{T}}\nabla f(x) - \frac{1}{2}s^{\mathrm{T}}\nabla f(x) = f(x) + \frac{1}{2}s^{\mathrm{T}}\nabla f(x). \end{aligned}$$

Now, in terms of the predictor this equation can be reduced to

$$\rho(s;x) = \frac{f(x) - f(x+s)}{-s^{\mathrm{T}}\nabla f(x)} \approx \frac{1}{2}.$$

This shows that, as the current point x tends to a minimizer x^* with the Newton direction, the predictor tends to the value $\frac{1}{2}$. Since $\rho(s;x) > c_{\min}$, no reducing iterations are performed to the step length. Moreover, even if $\rho(s;x) > c_{\max}$, no amplifying

iterations will be performed because the step length equal to one is optimal for the Newton direction near a minimizer. To prevent a modification of the step length we need to avoid, in Line 17, the reducing update which will require an additional amplifying iteration in the next non-linear iteration. Hence, it is sufficient to choose a constant $c_{\max}$ satisfying $c_{\min} < c_{\max} \leq 0.5$. The constant $c_{\max} = 0.25$ has been chosen since it is equally spaced from its limits 0 and 0.5. This permits to obtain quadratic convergence near a minimizer without line-search iterations.

A.3 Normalized curvature

In this section, we propose a criterion to check the curvature sign. On the one hand, it is standard to check the positiveness of curvature by means of the scalar product. On the other hand, a curvature constrain to limit the number of CG iterations, guarantee stability and sufficient positive curvature it is proposed (Dembo and Steihaug, 1983). For this reason, we propose to define the normalized curvature of a direction p at a point x as

$$\kappa(p; x) := \frac{p^{\mathrm{T}} \mathrm{H} f(x) p}{p^{\mathrm{T}} p}.$$

Thus, the constrain $\kappa_k^i > 0$ stills unchanged, while the constrain $\kappa_k^i > \epsilon$ becomes

$$\kappa_k^i := \kappa(d_k^i; x_k) = \frac{{d_k^i}^{\mathrm{T}} \mathrm{H} f(x_k) d_k^i}{{d_k^i}^{\mathrm{T}} d_k^i} > \epsilon.$$

This motivates us to propose a dynamic curvature forcing sequence $\{\tau_k\}$ detailed in Equation (3.13).

A.4 Ordering of the mesh nodes

Herein we fix an arrangement for the degrees of freedom to obtain results independent on the node ordering. This arrangement is performed in two steps. First, we perform a mesh node ordering. It is involved in the mesh distortion evaluation, see Section 3.2.2. Second, we perform an arrangement for the degrees of freedom over the mesh node ordering. It determines an arrangement for the optimization method, see Section 3.2.3. Note that, both arrangements may perturb the numerical conditioning of the Hessian and hence, the total number of matrix-vector products performed in the optimization problem.

We propose a node ordering that aims to concentrate the contributions of comparable magnitudes according to the stretching and alignment of the target metric. Our mesh node ordering relies in the class of *spectral* orderings (Clift et al., 1995; Paulino et al., 1994a,b). However, it slightly differs from the methods presented in the literature since it is focused to take into account information about the anisotropy of the target metric instead of the mesh connectivity only. We remark that the presented node ordering is used to couple the matrix elements according to their magnitude independently to the chosen preconditioner.

The node ordering is given by the partial ordering relation of an eigenfunction with lowest non-zero eigenvalue of the Laplace-Beltrami operator. That is, for a piece-wise polynomial mesh $\mathcal{M}$ of a bounded domain Ω equipped with a metric $\mathbf{M}$ and with Lipschitz boundary $\partial\Omega$, the ordering of the mesh nodes that we propose is computed from an eigenfunction with the lowest non-zero eigenvalue $\lambda_1 > 0$ of the Laplacian eigenproblem with Neumann boundary conditions (see Chavel (1984))

$$\begin{cases} -\Delta_{\mathbf{G}} u = \lambda_1 u & \text{in } \Omega \\ \frac{\partial u}{\partial \mathbf{n}} = 0 & \text{on } \partial\Omega \end{cases},$$

where, in our case, we set $\mathbf{G} = \mathbf{D}\boldsymbol{\phi}_P^{\mathrm{T}}\, \mathbf{M}\, \mathbf{D}\boldsymbol{\phi}_P$ which is the embedded or extrinsic mesh metric in the metric space $(\mathbb{R}^m, \mathbf{M})$, $\mathbf{D}\boldsymbol{\phi}_P$ is the Jacobian of the mapping $\boldsymbol{\phi}_P$ between the reference element and the physical element of the mesh and $\mathbf{n}$ is the outward normal of the boundary $\partial\Omega$. Then, the partial ordering relation $u(\mathbf{x}_i) < u(\mathbf{x}_j)$ (where $\mathbf{x}_i$ are the nodes of the mesh for any ordering) determines an ordering of the mesh nodes. In this work an eigenfunction u is computed using a continuous Galerkin finite element method over the mesh $\mathcal{M}$. Moreover, this reordering algorithm is used only one time, before the non-linear optimization.

Once we have defined the mesh node ordering, we prescribe an arrangement for the degrees of freedom. Note that, each node $\mathbf{x} \in \mathbb{R}^m$ contains m degrees of freedom. In the 2D case ($m = 2$), each node contains 2 degrees of freedom, the x-component and the y-component which we locate contiguous on the corresponding global mesh node $\mathbf{x}_i \in \mathbb{R}^m$. For example, if the mesh nodes are denoted by $\mathbf{x}_i = (x_i, y_i)\,,\ i = 1, ..., k$ then the corresponding variable representing the mesh is given by $(x_1, y_1, x_2, y_2, ..., x_k, y_k)$. The gradient is given by $\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, ..., \frac{\partial f}{\partial x_k}, \frac{\partial f}{\partial y_k}\right)$ and the components of the Hessian $\mathrm{H}f$ are then straightforward. Analogously, we apply this procedure for 3D meshes. In our case, the nodes lying at the boundary of the domain are permitted to slide on the boundary where they belong.

A.5 Test metrics

In this section, we detail the boundary layer metric $\mathbf{D}$ of the target metric $\mathbf{M}$, see Equation (3.18) and Table 3.1. Specifically, we propose two choices for the metric $\mathbf{D}$, a boundary layer over a curve (surface) $\mathbf{D}$ or a boundary layer over two intersecting curves (three intersecting surfaces) $\mathbf{D}_{\text{cross}}$.

On the one hand, the boundary layer $\mathbf{D}$ aligns with the x-axis (xy-plane) in the 2D case (3D case). It requires a constant unit element size along the x-direction (xy-directions), and a non-constant element size along the y-direction (z-direction). This vertical element size grows linearly with the distance to the x-axis (xy-plane), with a factor $\gamma = 2$, and starts with the minimal value $h_{\min} = 10^{-2}$ ($h_{\min} = 2 \cdot 10^{-2}$). Thus, for the 2D example illustrated in Figure 3.3(a), between $y = -0.5$ and $y = 0.5$ the stretching ratio blends from $1 : 100$ to $1 : 1$. For the 3D case, illustrated in Figure 3.3(b), between $z = -0.5$ and $z = 0.5$ the stretching ratio blends from $1 : 50$ to $1 : 1$. To match the boundary layer, we define the metric as:

$$\mathbf{D} := \begin{pmatrix} 1 & 0 \\ 0 & 1/h(y)^2 \end{pmatrix}, \quad \mathbf{D} := \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1/h(z)^2 \end{pmatrix}. \tag{A.1}$$

where the function h is defined by

$$h(x) := h_{\min} + \gamma |x|. \tag{A.2}$$

The metric of Equation (A.1) is the metric induced by the following deformation

$$\psi(x, y) = (x, H(y)), \quad \psi(x, y, z) = (x, y, H(z)),$$

that is $\mathbf{D} = \nabla\psi^{\mathrm{T}} \cdot \nabla\psi$ and being H the function given by

$$H(x) := \frac{1}{\gamma} \log\left(\frac{h(x)}{h_{\min}}\right). \tag{A.3}$$

On the other hand, we consider a metric $\mathbf{D}_{\text{cross}}$ consisting in the intersection of boundary layers with a stretching in each axis direction at the corresponding orthogonal hyperplane: in the x-direction at the line $x = 0$ and in the y-direction at the line $y = 0$ in 2D and with a stretching in the x-direction at the plane $x = 0$, in the

y-direction at the plane $y = 0$ and in the z-direction at the plane $z = 0$ in 3D, that is

$$\mathbf{D}_{\text{cross}} := \begin{pmatrix} 1/h(x)^2 & 0 \\ 0 & 1/h(y)^2 \end{pmatrix}, \quad \mathbf{D}_{\text{cross}} := \begin{pmatrix} 1/h(x)^2 & 0 & 0 \\ 0 & 1/h(y)^2 & 0 \\ 0 & 0 & 1/h(z)^2 \end{pmatrix}. \tag{A.4}$$

The metric of Equation (A.4) is the metric induced by the deformation

$$\psi_{\text{cross}}(x, y) = (H(x), H(y)), \quad \psi_{\text{cross}}(x, y, z) = (H(x), H(y), H(z)),$$

that is $\mathbf{D}_{\text{cross}} = \nabla\psi_{\text{cross}}^{\text{T}} \cdot \nabla\psi_{\text{cross}}$ and being H the function presented in Equation (A.3). As expected, the 2D intersection boundary layer metric presented in Equation (A.4) aligns with the x-axis at the line $y = 0$ and with the y-axis at the line $x = 0$, requires a constant unit element size along the diagonals of the square $x + y = 0$ and $x - y = 0$. Locally, the element size grows linearly along each axis with the distance to the orthogonal line, with a factor $\gamma = 2$, and starts with the minimal value $h_{\text{min}} = 10^{-2}$. Thus, between $y = -0.5$ and $y = 0.5$ the stretching ratio blends from $1 : 100$ to $1 : 1$.

Analogously, the 3D intersection boundary layer metric presented in Equation (A.4) aligns with the xy-axis at the plane $z = 0$, with the zx-axis at the plane $y = 0$ and with the yz-axis at the plane $x = 0$, requires a constant unit element size along the 4 diagonal lines of the cube. Locally, the element size grows linearly along each axis with the distance to the corresponding orthogonal plane, with a factor $\gamma = 2$, and starts with the minimal value $h_{\text{min}} = 2 \cdot 10^{-2}$. Thus, between $z = -0.5$ and $z = 0.5$ the stretching ratio blends from $1 : 2500$ to $1 : 1$. Note that in this case, the maximum stretching ratio is given by h_{min}^2 and it is attained at the intersection of each plane $x = 0$, $y = 0$ and $z = 0$ with the boundary of the hexahedron Ω.

For $\mathbf{D} = \mathbf{D}$ the metric $\mathbf{M}$ attains the highest level of stretching ratio, close to the curve described by the points $(x, y) \in \Omega$ such that $\varphi(x, y) = (x, 0)$ in 2D and close the surface described by the points $(x, y, z) \in \Omega$ such that $\varphi(x, y, z) = (x, y, 0)$ in 3D. Note that the metric $\mathbf{M}$ is induced by the map $\psi := \psi \circ \varphi$. Analogously, for $\mathbf{D} = \mathbf{D}_{\text{cross}}$ the metric $\mathbf{M}$ attains the highest level of stretching ratio, close the intersection of curves described by the points $(x, y) \in \Omega$ such that $\varphi(x, y) = (x, 0)$ or $\varphi(x, y) = (0, y)$ in 2D and close the surface described by the points $(x, y, z) \in \Omega$ such that $\varphi(x, y, z) = (x, y, 0)$ or $\varphi(x, y, z) = (x, 0, z)$ or $\varphi(x, y, z) = (0, y, z)$ in 3D. Note that the metric $\mathbf{M}$ is induced by the map $\psi := \psi_{\text{cross}} \circ \varphi$.

Appendix B

Derivatives of the eigenvalue decomposition and the implicit representation

---

B.1 Derivatives of the eigenvalue decomposition

In this Appendix, we detail the first and second-order spatial derivatives of the eigenvalue decomposition (eigenvalues and eigenvectors), first presented in Andrew et al. (1993) and rewritten herein using our notation.

Let us consider, for $\ell = 1, ..., d$, the eigenvalue equation for the eigenvector $\mathbf{u}_\ell$ with eigenvalue λ_ℓ

$$\mathbf{L}_\ell \mathbf{u}_\ell := (\mathbf{L} - \lambda_\ell \mathbf{I})\, \mathbf{u}_\ell = 0,$$

where $\mathbf{L}$ is a symmetric matrix and $\mathbf{I}$ is the identity matrix. Then, by taking its first-order and second-order derivatives we respectively obtain

$$0 = \partial_j \left(\mathbf{L}_\ell \mathbf{u}_\ell\right) = \left(\partial_j \mathbf{L}_\ell\right)\ \mathbf{u}_\ell + \mathbf{L}_\ell\ \partial_j \mathbf{u}_\ell, \tag{B.1}$$

$$\begin{aligned} 0 = \partial_{jk} \left(\mathbf{L}_\ell \mathbf{u}_\ell\right) = \left(\partial_{jk} \mathbf{L}_\ell\right)\ \mathbf{u}_\ell + \mathbf{L}_\ell\ \partial_{jk} \mathbf{u}_\ell + \\ \left(\partial_j \mathbf{L}_\ell\right)\ \partial_k \mathbf{u}_\ell + \left(\partial_k \mathbf{L}_\ell\right)\ \partial_j \mathbf{u}_\ell. \end{aligned} \tag{B.2}$$

For each ℓ one first computes the first-order derivative of the eigenvalue λ_ℓ by left-multiplying by $\mathbf{u}_\ell$ to Equation (B.1). Then, by solving the remaining unknown

term of Equation (B.1) one obtains the first-order derivatives of the eigenvector $\mathbf{u}_\ell$. In particular, the first-order derivatives of the eigenvalues and the eigenvectors are given by

$$\partial_j \lambda_\ell = \mathbf{u}_\ell^{\mathrm{T}}\ \partial_j \mathbf{L}\ \mathbf{u}_\ell, \quad \partial_j \mathbf{u}_\ell = -\mathbf{L}_\ell^{+}\ \partial_j \mathbf{L}_\ell\ \mathbf{u}_\ell,$$

where the operation $\mathbf{L}_\ell^{+}$ is the Moore-Penrose pseudo-inverse matrix for the matrix $\mathbf{L}_\ell$. We use the Moore-Penrose pseudo-inverse matrix instead of the inverse matrix because the matrix $\mathbf{L}_\ell$ is singular. In addition, the redundant equations are satisfied automatically.

The second-order derivatives are obtained by applying a similar procedure. For each ℓ one first computes the second-order derivative of the eigenvalue λ_ℓ by left-multiplying by $\mathbf{u}_\ell$ to Equation (B.2). Then, by solving the remaining unknown term of Equation (B.2) one obtains the second-order derivatives of the eigenvector $\mathbf{u}_\ell$. In particular, the second-order derivatives of the eigenvalues are given by

$$\partial_{jk} \lambda_\ell = \mathbf{u}_\ell^{\mathrm{T}}\ \left(\partial_k \mathbf{L}_\ell\ \partial_j \mathbf{u}_\ell + \partial_j \mathbf{L}_\ell\ \partial_k \mathbf{u}_\ell + \partial_{jk} \mathbf{L}\ \mathbf{u}_\ell\right),$$

$$\partial_{jk} \mathbf{u}_\ell = -\mathbf{L}_\ell^{+}\ \left(\partial_k \mathbf{L}_\ell\ \partial_j \mathbf{u}_\ell + \partial_j \mathbf{L}_\ell\ \partial_k \mathbf{u}_\ell + \partial_{jk} \mathbf{L}_\ell\ \mathbf{u}_\ell\right) - \left(\partial_j \mathbf{u}_\ell\ \partial_k \mathbf{u}_\ell\right) \mathbf{u}_\ell,$$

where the last term of the second-order derivative of the eigenvector is obtained by imposing the second-order derivative of the imposed normalization condition $\mathbf{u}_\ell^{\mathrm{T}}\ \mathbf{u}_\ell = 1$

$$0 = \partial_{jk} \left(\mathbf{u}_\ell^{\mathrm{T}}\ \mathbf{u}_\ell\right) = 2\partial_{jk} \mathbf{u}_\ell^{\mathrm{T}}\ \mathbf{u}_\ell + 2\partial_j \mathbf{u}_\ell^{\mathrm{T}}\ \partial_k \mathbf{u}_\ell.$$

B.2 Derivatives of the implicit representation

In this Appendix, we detail the first and second-order derivatives of the normalized representation, the convex-hull representation, and the implicit representation of a Bézier patch. They are used in the computation of the gradient and Hessian for the implicit representation, see Section 4.5.2.

Herein, we consider the gradient and Hessian of the normalized representation $\hat{\gamma}$, presented in Equation (4.13). As before, we denote by $\nabla f * \nabla g$ the matrix with coefficients $\partial_j f \partial_k g$ for $j,\ k = 1, ..., d$. In addition, we consider the symmetric term $\nabla f \otimes \nabla g := \nabla f * \nabla g + \nabla g * \nabla f$ given by the matrix with coefficients $\partial_j f \partial_k g + \partial_k f \partial_j g$ for $j,\ k = 1, ..., d$. Then, the derivatives of the normalized representation are given by

$$\nabla \hat{\gamma} = \frac{\nabla \gamma - \hat{\gamma} \nabla \|\nabla \gamma\|}{\|\nabla \gamma\|}, \tag{B.3}$$

and

$$\hat{\gamma}\nabla^2\hat{\gamma} = \frac{\hat{\gamma}\nabla^2\gamma - \hat{\gamma}^2\nabla^2\|\nabla\gamma\| - \nabla\hat{\gamma}\otimes\hat{\gamma}\nabla\|\nabla\gamma\|}{\|\nabla\gamma\|}, \tag{B.4}$$

where

$$\hat{\gamma}\nabla\|\nabla\gamma\| = \frac{\hat{\gamma}\nabla^2\gamma\ \nabla\gamma}{\|\nabla\gamma\|},$$

$$\hat{\gamma}^2\nabla^2\|\nabla\gamma\| = \frac{\hat{\gamma}^2\nabla^3\gamma\ \nabla\gamma + \hat{\gamma}\nabla^2\gamma\ \hat{\gamma}\nabla^2\gamma - \hat{\gamma}\nabla\|\nabla\gamma\| * \hat{\gamma}\nabla\|\nabla\gamma\|}{\|\nabla\gamma\|}.$$

We observe that they require the first, second, and third derivatives of γ. In addition, we consider these terms when differentiating the trimming operation, see Equations (4.21) and (4.22) for $h = \hat{\gamma}$. In particular, the chain rule involves the terms $\nabla\hat{\gamma}$ and $\hat{\gamma}\nabla^2\hat{\gamma}$, and the terms $\nabla\gamma$, $\gamma\nabla^2\gamma$, and $\gamma^2\nabla^3\gamma$. As we can see, this observation is advantageous because a straight-forward computation of the second and third derivatives, $\nabla^2\gamma$, and $\nabla^3\gamma$, involves a singularity at the corresponding zero level-set of γ. For this reason, instead of computing directly the derivatives we consider them multiplied by the representation γ.

Next, we compute the derivatives of the convex-hull representation $\hat{\gamma}_{\mathrm{CH}(\Gamma)}$. In particular, note that these derivatives are trivial since the representation of each hyperplane entity is linear. Then, we differentiate the r-conjunction between the hyperplane representations $\gamma_{\mathrm{CH}(\Gamma)}$, see Equations (4.17) and (4.18). Finally, we differentiate the normalization of the convex-hull representation $\hat{\gamma}_{\mathrm{CH}(\Gamma)}$, see Equations (B.3) and (B.4).

Now, we compute the derivatives for the determinant γ of Equation (4.12). That is, $\nabla\gamma$, $\gamma\nabla^2\gamma$, and $\gamma^2\nabla^3\gamma$. First, compute the gradient of the determinant by using the Jacobi's formula

$$\nabla\gamma\left(\mathbf{x}\right) = \mathrm{tr}\left(\mathrm{adj}\left(\mathbb{N}\left(\mathbf{x}\right)\right)\ \nabla\mathbb{N}\left(\mathbf{x}\right)\right), \tag{B.5}$$

where $\mathbb{N}\left(\mathbf{x}\right) := \mathbb{M}\left(\mathbf{x}\right)\ \mathbb{M}\left(\mathbf{x}\right)^{\mathrm{T}}$. We consider the adjugate matrix $\mathrm{adj}\left(\mathbb{N}\left(\mathbf{x}\right)\right)$, instead of the inverse matrix, to avoid the singularity issues at the patch Γ. In particular, the adjugate matrix of $\mathbb{N}\left(\mathbf{x}\right)$ is defined by the transposed cofactor matrix, and satisfying the relation $\mathrm{adj}\left(\mathbb{N}\left(\mathbf{x}\right)\right) = \gamma\left(\mathbf{x}\right)\mathbb{N}\left(\mathbf{x}\right)^{-1}$ (Upreti et al., 2014). Secondly, we compute the higher-order derivatives $\gamma\nabla^2\gamma$, and $\gamma^2\nabla^3\gamma$ by differentiating the terms inside the trace function $\nabla\gamma$, see Equation (B.5). In particular, using the same notation as in Section 4.4.2, we compute the second derivatives for each j and k as

$$\gamma\left(\mathbf{x}\right)\partial_{jk}\gamma\left(\mathbf{x}\right) = \mathrm{tr}\left(\gamma\left(\mathbf{x}\right)\partial_k\mathrm{adj}\left(\mathbb{N}\left(\mathbf{x}\right)\right)\ \partial_j\mathbb{N}\left(\mathbf{x}\right) + \gamma\left(\mathbf{x}\right)\mathrm{adj}\left(\mathbb{N}\left(\mathbf{x}\right)\right)\ \partial_{jk}\mathbb{N}\left(\mathbf{x}\right)\right).$$

In addition, the third derivatives are given by

$$\gamma(\mathbf{x})^2 \partial_{jk\ell}\gamma(\mathbf{x}) = \operatorname{tr}\left(\gamma(\mathbf{x})^2 \partial_{k\ell}\operatorname{adj}(\mathbb{N}(\mathbf{x}))\ \partial_j\mathbb{N}(\mathbf{x}) + \gamma(\mathbf{x})^2 \partial_k\operatorname{adj}(\mathbb{N}(\mathbf{x}))\ \partial_{j\ell}\mathbb{N}(\mathbf{x}) + \gamma(\mathbf{x})^2 \partial_\ell\operatorname{adj}(\mathbb{N}(\mathbf{x}))\ \partial_{jk}\mathbb{N}(\mathbf{x})\right),$$

for each j, k, and ℓ. Note that, there is no third order term $\partial_{jk\ell}\mathbb{N}(\mathbf{x})$ because $\mathbb{N}(\mathbf{x})$ is a quadratic function on $\mathbf{x}$, see Equation (4.12).

Finally, we provide the derivatives of the adjugate matrix $\operatorname{adj}(\mathbb{N}(\mathbf{x}))$. In particular, we present them in terms of the derivatives of the inverse matrix multiplied by the determinant. Then, to rewrite the obtained expression in terms of the adjugate matrix, we multiply both expressions by the determinant γ. Specifically, the gradient is given by

$$\gamma(\mathbf{x})\nabla\operatorname{adj}(\mathbb{N}(\mathbf{x})) = \gamma(\mathbf{x})\nabla\left(\gamma(\mathbf{x})\mathbb{N}(\mathbf{x})^{-1}\right) = \operatorname{adj}(\mathbb{N}(\mathbf{x}))\nabla\gamma(\mathbf{x}) - \mathbb{N}(\mathbf{x})\ \operatorname{adj}(\mathbb{N}(\mathbf{x}))\ \mathbb{N}(\mathbf{x}).$$

We apply the same reasoning for the Hessian by computing

$$\gamma(\mathbf{x})^2\nabla^2\operatorname{adj}(\mathbb{N}(\mathbf{x})) = \gamma(\mathbf{x})^2\nabla\left(\frac{1}{\gamma(\mathbf{x})}\gamma(\mathbf{x})\nabla\operatorname{adj}(\mathbb{N}(\mathbf{x}))\right).$$

In particular, using the same notation as in Section 4.4.2, for each j and k we have

$$\gamma^2\partial_{jk}\operatorname{adj}(\mathbb{N}) = (\gamma\partial_{jk}\gamma)\operatorname{adj}(\mathbb{N}) + (\partial_j\gamma)\gamma\partial_k\operatorname{adj}(\mathbb{N}) - (\partial_k\gamma)\gamma\partial_j\operatorname{adj}(\mathbb{N}) - \gamma\operatorname{adj}(\mathbb{N})\ \partial_{jk}\mathbb{N}\ \operatorname{adj}(\mathbb{N}) - \gamma\partial_k\operatorname{adj}(\mathbb{N})\ \partial_j\mathbb{N}\ \operatorname{adj}(\mathbb{N}) - \operatorname{adj}(\mathbb{N})\ \partial_j\mathbb{N}\ \gamma\partial_k\operatorname{adj}(\mathbb{N}),$$

where, for the sake of brevity, we omit the dependence on the $\mathbf{x}$ variable of the functions γ and $\mathbb{N}$.

Bibliography

Alauzet, F. and A. Loseille (2016). A decade of progress on anisotropic mesh adaptation for computational fluid dynamics. *Computer-Aided Design, Elsevier 72*(1), 13–39.

Andrew, A. L., K.-W. E. Chu, and P. Lancaster (1993). Derivatives of eigenvalues and eigenvectors of matrix functions. *SIAM journal on matrix analysis and applications 14*(4), 903–926.

Aparicio-Estrems, G., A. Gargallo-Peiró, and X. Roca (2018). Defining a stretching and alignment aware quality measure for linear and curved 2d meshes. In *International Meshing Roundtable*, pp. 37–55. Springer.

Aparicio-Estrems, G., A. Gargallo-Peiró, and X. Roca (2019). Anisotropic optimization of curved meshes: specific-purpose line-search and trust-region globalizations for newton's method. In *International Meshing Roundtable*.

Aparicio-Estrems, G., A. Gargallo-Peiró, and X. Roca (2021). Stretching and aligning piece-wise polynomial meshes to match curved anisotropic features. In *International Conference on Spectral and High-Order Methods*.

Aparicio-Estrems, G., A. Gargallo-Peiró, and X. Roca (2022). High-order metric interpolation for curved r-adaption by distortion minimization. In *Proceedings of the 2022 SIAM International Meshing Roundtable*, pp. 1–12. Zenodo.

Aparicio-Estrems, G., A. Gargallo-Peiró, and X. Roca (2023a). Combining high-order metric interpolation and geometry implicitization for curved r-adaption. *Computer-Aided Design*.

Aparicio-Estrems, G., A. Gargallo-Peiró, and X. Roca (2023b). Defining a size-shape quality measure for linear and curved meshes equipped with a metric. *In preparation*.

Aparicio-Estrems, G., A. Gargallo-Peiró, and X. Roca (2023c). A globalized and preconditioned newton-cg solver for metric-aware curved high-order mesh optimization. *In preparation*.

Arsigny, V., P. Fillard, X. Pennec, and N. Ayache (2006). Log-euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic Resonance in Medicine 56*, 411–421.

Barber, C. B., D. P. Dobkin, and H. Huhdanpaa (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS) 22*(4), 469–483.

Barrera, J.-L., T. Kolev, K. Mittal, and V. Tomov (2023). High-order mesh morphing for boundary and interface fitting to implicit geometries. *Computer-Aided Design*, 103499.

Bellavia, S. and S. Berrone (2007). Globalization strategies for newton–krylov methods for stabilized fem discretization of navier–stokes equations. *Journal of Computational Physics 226*(2), 2317–2340.

Bertaccini, D. and F. Durastante (2018). *Iterative methods and preconditioning for large and sparse linear systems with applications*. Chapman and Hall/CRC.

Bezanson, J., A. Edelman, S. Karpinski, and V. B. Shah (2017). Julia: A fresh approach to numerical computing. *SIAM review 59*(1), 65–98.

Biswas, A. and V. Shapiro (2004). Approximate distance fields with non-vanishing gradients. *Graphical Models 66*(3), 133–159.

Branets, L. V. and V. A. Garanzha (2002). Distortion measure of trilinear mapping. application to 3-d grid generation. *Numerical linear algebra with applications 9*(6-7), 511–526.

Brenner, S. C., L. R. Scott, and L. R. Scott (2008). *The mathematical theory of finite element methods*, Volume 3. Springer.

Bulteau, J. P. and J. P. Vial (1985, Dec). A restricted trust region algorithm for unconstrained optimization. *Journal of Optimization Theory and Applications 47*(4), 413–435.

Camier, J.-S., V. Dobrev, P. Knupp, T. Kolev, K. Mittal, R. Rieben, and V. Tomov (2023). Accelerating high-order mesh optimization using finite element partial assembly on gpus. *Journal of Computational Physics 474*, 111808.

Chavel, I. (1984). *Eigenvalues in Riemannian geometry*, Volume 115. Academic press.

Chen, Y., T. A. Davis, W. W. Hager, and S. Rajamanickam (2008). Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS) 35*(3), 1–14.

Clift, S. S., H. D. Simon, and W.-P. Tang (1995). Spectral ordering techniques for incomplete lu preconditoners for cg methods.

Conn, A. R., N. I. Gould, and P. L. Toint (2000). *Trust region methods*, Volume 1. Siam.

Coulaud, O. and A. Loseille (2016a). Very high order anisotropic metric-based mesh adaptation in 3d. *Procedia Engineering*, 353–365. Proceedings of the 25th International Meshing Roundtable.

Coulaud, O. and A. Loseille (2016b). Very high order anisotropic metric-based mesh adaptation in 3d. *Procedia Engineering 163*. 25th International Meshing Roundtable.

Coupez, T. (2011). Metric construction by length distribution tensor and edge based error for anisotropic adaptive meshing. *Journal of computational physics 230*(7), 2391–2405.

Coupez, T. (2017). On a basis framework for high order anisotropic mesh adaptation. *203*, 141–153. Research Note 26th International Meshing Roundtable.

Coupez, T., L. Silva, and E. Hachem (2015). Implicit boundary and adaptive anisotropic meshing. In *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, pp. 1–18. Springer.

D'Azevedo, E. F., P. A. Forsyth, and W.-P. Tang (1992). Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems. *SIAM Journal on Matrix Analysis and Applications 13*(3), 944–961.

Dembo, R. S. and T. Steihaug (1983, Jun). Truncated-newtono algorithms for large-scale unconstrained optimization. *Mathematical Programming 26*(2), 190–212.

Diachin, L. F., P. Knupp, T. Munson, and S. Shontz (2004). A comparison of inexact newton and coordinate descent meshoptimization technqiues. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States).

Diachin, L. F., P. Knupp, T. Munson, and S. Shontz (2006). A comparison of two optimization methods for mesh quality improvement. *Engineering with Computers 22*(2), 61–74.

Dobrev, V., P. Knupp, T. Kolev, K. Mittal, R. Rieben, and V. Tomov (2020). Simulation-driven optimization of high-order meshes in ale hydrodynamics. *Computers & Fluids 208*, 104602.

Dobrev, V., P. Knupp, T. Kolev, K. Mittal, and V. Tomov (2019). The target-matrix optimization paradigm for high-order meshes. *SIAM Journal on Scientific Computing 41*(1), B50–B68.

Dobrev, V., P. Knupp, T. Kolev, K. Mittal, and V. Tomov (2021). hr-adaptivity for nonconforming high-order meshes with the target matrix optimization paradigm. *Engineering with Computers*, 1–17.

Dobrev, V., P. Knupp, T. Kolev, and V. Tomov (2018). Towards simulation-driven optimization of high-order meshes by the target-matrix optimization paradigm. In *International Meshing Roundtable*, pp. 285–302. Springer.

Dobrzynski, C. (2012). *MMG3D: User guide*. Ph. D. thesis, INRIA.

Eisenstat, S. C. and H. F. Walker (1996). Choosing the forcing terms in an inexact newton method. *SIAM Journal on Scientific Computing 17*(1), 16–32.

Ekelschot, D., M. Ceze, S. M. Murman, and A. Garai (2019). Parallel high-order anisotropic meshing using discrete metric tensors. In *AIAA Scitech 2019 Forum*, pp. 1993.

Escobar, J. M., E. Rodríguez, R. Montenegro, G. Montero, and J. M. González-Yuste (2003). Simultaneous untangling and smoothing of tetrahedral meshes. *Comput. Meth. Appl. Mech. Eng. 192*(25), 2775–2787.

Feuillet, R. (2019). *Embedded and high-order meshes: two alternatives to linear body-fitted meshes*. Ph. D. thesis, Université Paris-Saclay (ComUE).

Feuillet, R., A. Loseille, and F. Alauzet (2019). P 2 mesh optimization operators. *27th International Meshing Roundtable 27*, 3–21.

Feuillet, R., A. Loseille, and F. Alauzet (2020). Optimization of p2 meshes and applications. *Computer-Aided Design 124*.

Fidkowski, K. J. and D. L. Darmofal (2011). Review of output-based error estimation and mesh adaptation in computational fluid dynamics. *AIAA journal 49*(4), 673–694.

Frey, P. and F. Alauzet (2005). Anisotropic mesh adaptation for cfd computations. *Computer methods in applied mechanics and engineering 194*(48-49), 5068–5082.

Gargallo-Peiró, A., X. Roca, J. Peraire, and J. Sarrate (2015a). Distortion and quality measures for validating and generating high-order tetrahedral meshes. *Eng. Comput. 31*, 423–437.

Gargallo-Peiró, A., X. Roca, J. Peraire, and J. Sarrate (2015b). A distortion measure to validate and generate curved high-order meshes on CAD surfaces with independence of parameterization. *Int. J. Numer. Meth. Eng. 106*(13), 1100–1130.

Gargallo-Peiró, A., X. Roca, J. Peraire, and J. Sarrate (2015c). Optimization of a regularized distortion measure to generate curved high-order unstructured tetrahedral meshes. *Int. J. Numer. Meth. Eng. 103*, 342–363.

Gargallo-Peiró, A., E. Ruiz-Gironés, X. Roca, and J. Sarrate (2015). On curving high-order hexahedral meshes. In *24th International Meshing Roundtable (IMR24), October 11-14, 2014, Austin, TX*, pp. 1–5. Elsevier.

Gruau, C. and T. Coupez (2005). 3d tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric. *Computer Methods in Applied Mechanics and Engineering 194*(48-49), 4951–4976.

Hecht, F. (1998). Bamg: bidimensional anisotropic mesh generator. *User Guide. INRIA, Rocquencourt*.

Huang, W. and R. D. Russell (2011). *Adaptive Moving Mesh Methods*, Volume 174 of *Applied Mathematical Sciences*. Springer.

Ibanez, D., N. Barral, J. Krakos, A. Loseille, T. Michal, and M. Park (2017). First benchmark of the unstructured grid adaptation working group. *Procedia engineering 203*, 154–166.

Johnen, A., C. Geuzaine, T. Toulorge, and J.-F. Remacle (2021). Quality measures for curvilinear finite elements. *TILDA: Towards Industrial LES/DNS in Aeronautics*, 221.

Kershaw, D. S. (1978). The incomplete cholesky-conjugate gradient method for the iterative solution of systems of linear equations. *Journal of computational physics 26*(1), 43–65.

Knupp, P., T. Kolev, K. Mittal, and V. Z. Tomov (2021). Adaptive surface fitting and tangential relaxation for high-order mesh optimization. *arXiv e-prints*, arXiv–2105.

Knupp, P. M. (2001). Algebraic mesh quality metrics. *SIAM J. Numer. Anal. 23*(1), 193–218.

Laurent, B. (2014). Implicit matrix representations of rational bézier curves and surfaces. *Computer-Aided Design 46*, 14–24.

López, E. J., N. M. Nigro, and M. A. Storti (2008). Simultaneous untangling and smoothing of moving grids. *Int. J. Numer. Meth. Eng. 76*(7), 994–1019.

Loseille, A. and F. Alauzet (2011). Continuous mesh framework part i: well-posed continuous interpolation error. *SIAM Journal on Numerical Analysis 49*(1), 38–60.

Loseille, A. and R. Löhner (2010). Anisotropic adaptive simulations in aerodynamics. *AIAA 2010-169*. 49th AIAA Aerospace Sciences Meeting, Fairfax, Va, USA.

Marcon, J. (2019). *Mesh adaptation for high-order flow simulations*. Ph. D. thesis, Imperial College London.

Marcon, J., G. Castiglioni, D. Moxey, S. J. Sherwin, and J. Peiró (2020). rp-adaptation for compressible flows. *International Journal for Numerical Methods in Engineering 121*(23), 5405–5425.

Marcon, J., M. Turner, D. Moxey, S. J. Sherwin, and J. Peiró (2017). A variational approach to high-order r-adaptation. *IMR26*.

MATLAB (2017). *version 9.3.0.713579 (R2017b)*. Natick, Massachusetts: The MathWorks Inc.

Mittal, K., S. Dutta, and P. Fischer (2019). Nonconforming schwarz-spectral element methods for incompressible flow. *Computers & Fluids 191*, 104237.

Nash, S. G. and A. Sofer (1990). Assessing a search direction within a truncated-newton method. *Operations Research Letters 9*(4), 219 – 221.

Nocedal, J. and S. Wright (2006). *Numerical optimization*. Springer Science & Business Media.

Paulino, G. H., I. F. Menezes, M. Gattass, and S. Mukherjee (1994a). Node and element resequencing using the laplacian of a finite element graph: part i-general concepts and algorithm. *International Journal for Numerical Methods in Engineering 37*(9), 1511–1530.

Paulino, G. H., I. F. Menezes, M. Gattass, and S. Mukherjee (1994b). Node and element resequencing using the laplacian of a finite element graph: part ii-implementation and numerical results. *International Journal for Numerical Methods in Engineering 37*(9), 1531–1555.

Persson, P.-O. and J. Peraire (2008). Newton-gmres preconditioning for discontinuous galerkin discretizations of the navier–stokes equations. *SIAM Journal on Scientific Computing 30*(6), 2709–2733.

Riegel, J., W. Mayer, and Y. van Havre (2016). Freecad.

Roca, X., A. Gargallo-Peiró, and J. Sarrate (2012). Defining quality measures for high-order planar triangles and curved mesh generation. In *Proc. 20th Int. Meshing Roundtable*, pp. 365–383. Springer International Publishing.

Rochery, L. and A. Loseille (2021). P2 cavity operator and riemannian curved edge length optimization: a path to high-order mesh adaptation. In *AIAA Scitech 2021 Forum*, pp. 1781.

Ruiz-Gironés, E., A. Gargallo-Peiró, J. Sarrate, and X. Roca (2017). An augmented lagrangian formulation to impose boundary conditions for distortion based mesh moving and curving. *Procedia engineering 203*, 362–374.

Ruiz-Gironés, E. and X. Roca (2022). Automatic penalty and degree continuation for parallel pre-conditioned mesh curving on virtual geometry. *Computer-Aided Design*, 103208.

Ruiz-Gironés, E., J. Sarrate, and X. Roca (2016). Generation of curved high-order meshes with optimal quality and geometric accuracy. *Procedia engineering 163*, 315–327.

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*. SIAM.

Sanjaya, D. (2019). *Towards Automated, Metric-Conforming, Mesh Optimization For High-Order, Finite-Element Methods*. Ph. D. thesis, University of Michigan.

Sanjaya, D. P. and K. J. Fidkowski (2016). Improving high-order finite element approximation through geometrical warping. *AIAA Journal 54*(12), 3994–4010.

Sastry, S. P. and S. M. Shontz (2009). A comparison of gradient-and hessian-based optimization methods for tetrahedral mesh quality improvement. In *Proceedings of the 18th International Meshing Roundtable*, pp. 631–648. Springer.

Sastry, S. P. and S. M. Shontz (2012). Performance characterization of nonlinear optimization methods for mesh quality improvement. *Engineering with Computers 28*(3), 269–286.

Sitaraman, J., M. Floros, A. Wissink, and M. Potsdam (2010). Parallel domain connectivity algorithm for unsteady flow computations using overlapping and adaptive grids. *Journal of Computational Physics 229*(12), 4703–4723.

Steihaug, T. (1983). The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis 20*(3), 626–637.

Upreti, K., T. Song, A. Tambat, and G. Subbarayan (2014). Algebraic distance estimations for enriched isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering 280*, 28–56.

Warburton, T. (2006). An explicit construction of interpolation nodes on the simplex. *Journal of engineering mathematics 56*(3), 247–262.

Yano, M. and D. L. Darmofal (2012). An optimization-based framework for anisotropic simplex mesh adaptation. *Journal of Computational Physics 231*(22), 7626–7649.

Zahr, M. J. and P.-O. Persson (2018). An optimization based discontinuous galerkin approach for high-order accurate shock tracking. In *2018 AIAA Aerospace Sciences Meeting*, pp. 0063.

Zahr, M. J. and P.-O. Persson (2020). An r-adaptive, high-order discontinuous galerkin method for flows with attached shocks. In *AIAA Scitech 2020 Forum*, pp. 0537.

Zahr, M. J., A. Shi, and P.-O. Persson (2020). Implicit shock tracking using an optimization-based high-order discontinuous galerkin method. *Journal of Computational Physics 410*, 109385.

Zhang, R. (2022, 03). *Metric-based curvilinear mesh generation and adaptation*. Ph. D. thesis, Université Catholique de Louvain.

Zhang, R., A. Johnen, and J.-F. Remacle (2018). Curvilinear mesh adaptation. In *International Meshing Roundtable*, pp. 57–69. Springer.