

Combining high-order metric interpolation and geometry implicitization for curved r -adaption

Guillermo Aparicio-Estrems, Abel Gargallo-Peiró, Xevi Roca*

Barcelona Supercomputing Center, 08034 Barcelona, Spain

Abstract

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.

Keywords: r -adaption, mesh optimization, curved high-order meshes

*Corresponding author

Email addresses: `guillermo.aparicio@bsc.es` (Guillermo Aparicio-Estrems), `abel.gargallo@bsc.es` (Abel Gargallo-Peiró), `xevi.roca@bsc.es` (Xevi Roca)

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 [1, 2, 3]. 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) [4, 5] or a previously re-meshed cavity (local) [6, 7, 8].

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 [3, 4, 6, 9, 10]. 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 [11].

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 pointwise varying fashion through anisotropic procedures [12], geodesic approaches for curved edges [13, 14], shock-tracking methods [15, 16, 17], and deformation analogies [18, 19]. Alternatively, the curved r -adaption can be driven, as for straight-edged elements [4, 5], by distortion measures. These measures are defined point-wise and are aware of either a target deformation matrix [20] or a target metric [21].

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.

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 work extends our previous work [22]. 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 [23, 24]. Second, we represent the metric field as a log-Euclidean high-order metric interpolation [25] on a curved high-order mesh. Third, we locate physical points in the curved background mesh similar to the approach in [26]. We also need to extend to discrete metric fields a distortion-based curved r -adaption framework [21].

To match the curved boundaries, we also need three existing ingredients. First, a non-interpolative approach to match the target curved geometry [27, 28]. Second, an implicit CAD geometry representation method for 2D NURBS curves and a 3D NURBS surfaces [29] or for embedded NURBS entities [30] such as 3D curves. Third, a series of conjunction and trimming operations to assemble the implicit representations of the individual entities [29, 31].

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 [29, 30, 31]. Second, we also provide the first and second-order derivatives of the implicit representation of the model. As in [22], 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 paper 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 paper is organized as follows. In Section 2, we overview the related work. In Section 3, we introduce the preliminaries on metric-aware measures for high-order elements. In Section 4, we detail the high-order metric interpolation and its derivatives. In Section 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 6, we show Newton’s method results for different geometries, meshes, and metrics. Finally, we present the concluding remarks.

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 [20]. The method is really well-suited for simulation-driven r -adaption [26, 32]. 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 [21, 23, 24]. The formulation for an analytic metric is derived in [21], while a specific-purpose globalization and a pre-conditioned Netwon-CG method are proposed in [23, 24] 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 [33] high-order metric interpolation [25]. 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 [27, 28]. 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 [34]. 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 [29, 30, 31]. Specifically, one first computes the implicit representation of each NURBS entity. This is the case of a 2D NURBS curve and 3D NURBS surface [29] or a generally embedded NURBS entity [30]. Then, one applies convex-hull conjunction and normalization, convex-hull trimming, and NURBS conjunction to assemble the representations of the individual entities [29, 31]. 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.

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 [21]. To define and compute a Jacobian-based measure for simplices [5], three elements are required: the master, the ideal, and the physical, see Figure 1 for the linear triangle case. The master (E^M) is the element from which the isoparametric mapping is defined. The equilateral element (E^Δ) represents the target configuration in the isotropic case. The physical (E^P) is the element to be measured.

To summarize the results in [21], we present the expression of the metric distortion measure in terms of the equilateral element E^Δ . First, we need to compute a mapping from the master to the equilateral and physical elements, denoted as ϕ_Δ and ϕ_P , respectively. By means of these mappings,

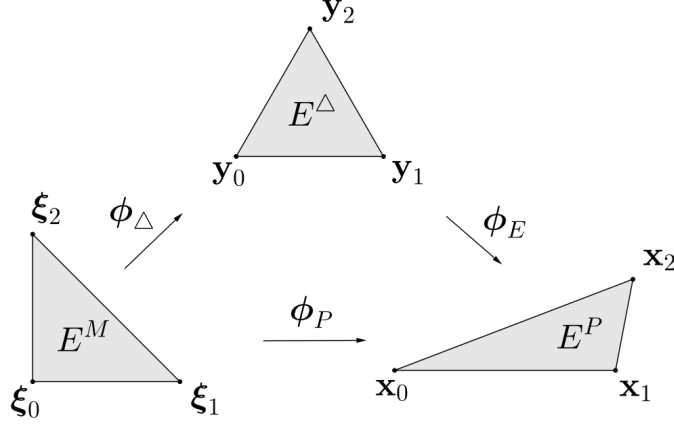


Figure 1: Mappings between the master, the ideal, and the physical elements in the linear case.

we determine a mapping between the equilateral and physical elements by the composition

$$\phi_E : E^\Delta \xrightarrow{\phi_\Delta^{-1}} E^M \xrightarrow{\phi_P} E^P.$$

As detailed in [21], 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 $\mathbf{y} \in E^\Delta$ as

$$\mathcal{N}\phi_E(\mathbf{y}) = \frac{\text{tr} \left(\mathbf{D}\phi_E(\mathbf{y})^\text{T} \cdot \mathbf{M}(\phi_E(\mathbf{y})) \cdot \mathbf{D}\phi_E(\mathbf{y}) \right)}{d \left(\det \left(\mathbf{D}\phi_E(\mathbf{y})^\text{T} \cdot \mathbf{M}(\phi_E(\mathbf{y})) \cdot \mathbf{D}\phi_E(\mathbf{y}) \right) \right)^{1/d}}, \quad (1)$$

where the Jacobian of the map ϕ_E is given by

$$\mathbf{D}\phi_E(\mathbf{y}) := \mathbf{D}\phi_P(\phi_\Delta^{-1}(\mathbf{y})) \cdot \mathbf{D}\phi_\Delta^{-1}(\mathbf{y}).$$

Herein, $\mathbf{D}\phi_P$ and $\mathbf{D}\phi_\Delta$ 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

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

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

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

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 $\mathbf{M}(\phi_E(\mathbf{y}))$, either using an analytical or a discretized representation.

We regularize the determinant in the denominator of Equation (1) in order to detect inverted elements [35, 36, 37, 38]. In particular, we define

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

where

$$\sigma = \det(\mathbf{D}\phi_E(\mathbf{y})) \sqrt{\det(\mathbf{M}(\phi_E(\mathbf{y})))}.$$

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

$$\mathcal{N}_0\phi_E(\mathbf{y}) := \frac{\text{tr}(\mathbf{D}\phi_E(\mathbf{y})^\text{T} \cdot \mathbf{M}(\phi_E(\mathbf{y})) \cdot \mathbf{D}\phi_E(\mathbf{y}))}{d\sigma_0^{2/d}}, \quad (2)$$

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

$$\mathcal{Q}\phi_E(\mathbf{y}) = \frac{1}{\mathcal{N}_0\phi_E(\mathbf{y})}. \quad (3)$$

Finally, we define the regularized elemental distortion by

$$\eta_{(E^P, \mathbf{M})} := \frac{\int_{E^\Delta} \mathcal{N}_0\phi_E(\mathbf{y}) \, d\mathbf{y}}{\int_{E^\Delta} 1 \, d\mathbf{y}},$$

and its corresponding quality

$$\mathbf{q}_{(E^P, \mathbf{M})} = \frac{1}{\eta_{(E^P, \mathbf{M})}}. \quad (4)$$

We can improve the mesh configuration by means of relocating the nodes of the mesh according to a given distortion measure [21, 23, 24, 39]. In [21] 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

$$\mathcal{F}(\mathcal{M}) := \sum_{E^P \in \mathcal{M}} \int_{E^\Delta} (\mathcal{N}_0 \phi_E(\mathbf{y}))^2 d\mathbf{y}, \quad (5)$$

which allows to pose the following global minimization problem

$$\mathcal{M}^* := \operatorname{argmin}_{\mathcal{M}} \mathcal{F}(\mathcal{M}), \quad (6)$$

to improve the mesh configuration according to \mathcal{F} . In particular, herein, the degrees of freedom of the minimization problem in Equation (6) correspond to the spatial coordinates of the mesh nodes.

To evaluate the distortion minimization formulation presented in Equation (6), an input metric is required. The reviewed r -adaption procedure has been applied for analytic metrics in [21]. In the following section, we detail the interpolation process that is required to extend the presented framework to discrete metrics.

4. Log-Euclidean metric interpolation

In this section, we formulate a metric interpolation process that allows both the distortion evaluation, Equation (2), and its optimization, Equation (6). In Section 4.1 we detail the log-Euclidean metric interpolation for linear and high-order elements first presented in [33] and [25, 40], respectively. Then, in Section 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.

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

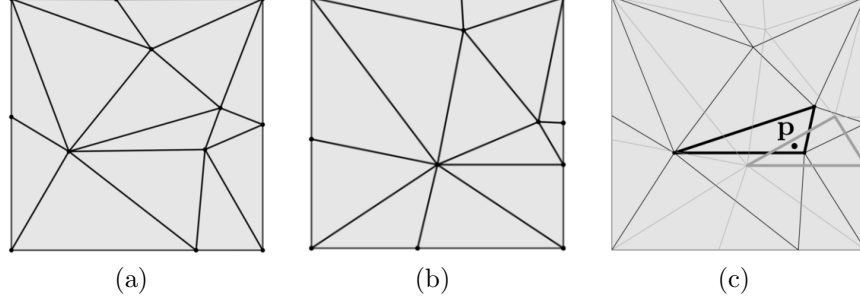


Figure 2: Point localization: (a) physical mesh, (b) background mesh, and (c) a point \mathbf{p} in the corresponding physical and background element (bold edges).

To evaluate the metric-aware distortion measure in Equation (2) featuring discrete metrics, two meshes are required. On the one hand, the *physical* mesh \mathcal{M} , Figure 2(a), is the domain where the elements are deformed in order to solve the problem presented in Equation (6). On the other hand, the background mesh $\hat{\mathcal{M}}$, Figure 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 [26, 41, 42]. 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 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 (2) over the equilateral element via the master element E^M . In particular, for the metric evaluation, we map via ϕ_P , each integration point $\boldsymbol{\xi} \in E^M$ to a point \mathbf{p} of the physical element E^P , see Figure 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 3. In addition, Figure 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 $\mathbf{p} = \phi_P(\boldsymbol{\xi}) \in E^P$, 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 $\hat{\boldsymbol{\xi}} = \phi_{\hat{P}}^{-1}(\hat{\mathbf{p}}) \in E^{\hat{M}}$.

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

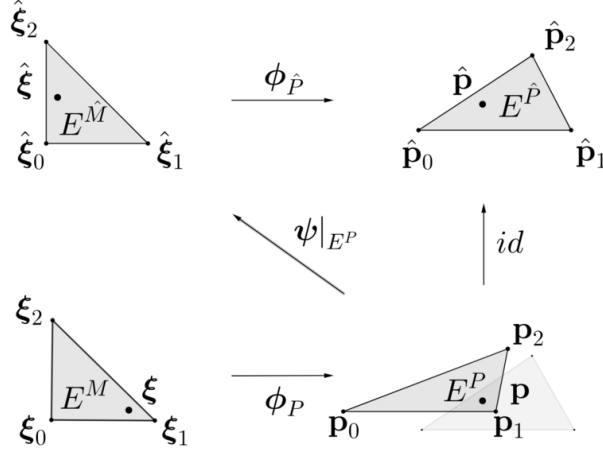


Figure 3: Mappings between the master and the physical elements (below) and their background analogs (above).

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

$$\begin{aligned} \psi|_{E^P} : E^P \cap E^{\hat{P}} &\rightarrow E^{\hat{M}} \\ \mathbf{p} &\mapsto \boldsymbol{\phi}_{\hat{P}}^{-1}(\mathbf{p}). \end{aligned} \quad (7)$$

Specifically, we solve the non-linear inverse expression in the image term, Equation (7), by applying Newton's minimization to the squared distance. That is, as in Section 2.3 of [41], we solve

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

The result is a numerical approximation of the point coordinates in the background master element. An alternative approach [26] is to seek the zeros of the vector equation

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

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 [33, 25]:

$$\mathbf{M}(\hat{\mathbf{N}}) := \exp(\mathbf{L}(\hat{\mathbf{N}})), \quad \mathbf{L}(\hat{\mathbf{N}}) := \sum_{j=1}^{\hat{n}} \hat{N}_j \log \hat{\mathbf{M}}_j, \quad (8)$$

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

$$\mathbf{M}(\hat{\mathbf{N}}) = \mathbf{U} \cdot \exp \mathbf{D} \cdot \mathbf{U}^T, \quad (9)$$

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

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 [43].

To compute the derivatives of the metric \mathbf{M} we first differentiate the eigenvalue-based exponential matrix function presented in Equation (9) and then we differentiate the \mathbf{L} function presented in Equation (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 (8). In particular, the first-order derivatives are given by

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

and the second-order derivatives are given by

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

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

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

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 appended in Section Appendix A.

In addition, the derivatives of the eigenvalues and eigenvectors depend on the derivatives of the \mathbf{L} function presented in Equation (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 $\mathbf{M}(\hat{\mathbf{N}}(\psi(\mathbf{p})))$ at a physical point \mathbf{p} , the derivatives of the map ψ presented in Equation (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 $\phi_{\hat{P}}^{-1}$ 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

$$\nabla \hat{\mathbf{N}} = \nabla_{\hat{\boldsymbol{\xi}}} \hat{\mathbf{N}} \cdot \nabla \phi_{\hat{P}}^{-1}, \quad (10)$$

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

$$\nabla^2 \hat{N}_j = \left(\nabla \phi_{\hat{P}}^{-1} \right)^T \cdot \nabla_{\hat{\boldsymbol{\xi}}}^2 \hat{N}_j \cdot \nabla \phi_{\hat{P}}^{-1} + \nabla_{\hat{\boldsymbol{\xi}}} \hat{N}_j \cdot \nabla^2 \phi_{\hat{P}}^{-1}, \quad (11)$$

where

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

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 5.1, we present a model implicitization for the mesh curving process. Then, in Section 5.2, we detail the first and second-order derivatives for the implicit representation. Finally, in Section 5.3, we consider the penalty method to solve the corresponding constrained second-order minimization process for the curving problem.

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 [30], and the Boolean algebraic operations for implicit representations [29, 31]. 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 Γ_i , $i = 1, \dots, n$. 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

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

In addition, the implicit representation of Γ can be obtained as in [30]

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

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 [31].

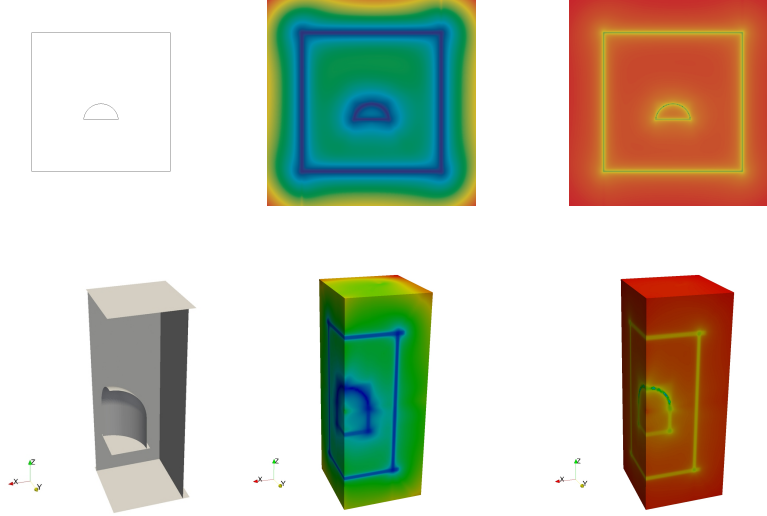


Figure 4: Implicit representation of (first row) a 2D CAD geometry, and (second row) a 3D CAD geometry. CAD model, and implicit representation in linear, and logarithmic scale in columns.

In Figure 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 [29]. 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 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 $\mathbb{P}(\mathbb{R}^D)$. 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 $\mathbf{x} \in \mathbb{P}(\mathbb{R}^D)$ as in [30]

$$\gamma(\mathbf{x}) := \det(\mathbb{M}(\mathbf{x}) \cdot \mathbb{M}(\mathbf{x})^T), \quad \mathbb{M}(\mathbf{x}) := \mathbb{M} \mathbf{x}. \quad (12)$$

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 $\gamma(\mathbf{x})$ to ensure that they match during the assembly procedure [29]. Specifically, we define the normalized function $\hat{\gamma}$ by

$$\hat{\gamma} := \frac{\gamma}{\|\nabla \gamma\|}. \quad (13)$$

The implicit function of a Bézier patch described in Equation (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 [29, 31]. Specifically, we first compute $\text{CH}(\Gamma)$, the implicit representation of the convex hull of the Bézier patch Γ . Then, to obtain an implicit representation of Γ trimmed by $\text{CH}(\Gamma)$, we use a trimming function, denoted by trim , proposed in [31]

$$\gamma_{\overline{\Gamma}} := \hat{\gamma}_{\text{CH}(\Gamma)} \text{ trim } \hat{\gamma}_{\Gamma} = \sqrt{\hat{\gamma}_{\Gamma}^2 + \left(\frac{\sqrt{\hat{\gamma}_{\Gamma}^4 + \hat{\gamma}_{\text{CH}(\Gamma)}^2} - \hat{\gamma}_{\text{CH}(\Gamma)}}{2} \right)^2}, \quad (14)$$

where γ_{Γ} denotes the representation of the Bézier patch Γ , see Equation (12). The trimming operation of Equation (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 $\text{Dom } \Gamma$ determined by the NURBS convex-hull $\text{CH}(\Gamma)$.

For a given model $\Lambda = \{\Gamma_1, \Gamma_2, \dots, \Gamma_n\}$, its implicitization γ_{Λ} is obtained via the r -conjunction \wedge of the implicitizations $\gamma_{\overline{\Gamma}_i}$ of the Bézier patches Γ_i [29]. In particular, for each Bézier patch Γ_i , we recursively update the model

Algorithm 1 Implicitization

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

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

representation as follows

$$\gamma_\Lambda \leftarrow \gamma_\Lambda \wedge \gamma_{\bar{\Gamma}_i} := \gamma_\Lambda + \gamma_{\bar{\Gamma}_i} - \sqrt{\gamma_\Lambda^2 + \gamma_{\bar{\Gamma}_i}^2}. \quad (15)$$

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

$$\gamma_H(\mathbf{x}) := \mathbf{n} \cdot \mathbf{x} + b. \quad (16)$$

In our case, the sign of the representation γ_H is chosen such that $\gamma_H < 0$ outside the convex region enclosed by $\text{CH}(\Gamma)$ and $\gamma_H \geq 0$ otherwise. Following, we apply the r -conjunction operation for each hyperplane H to obtain the convex-hull representation $\gamma_{\text{CH}(\Gamma)}$, see Equation (15). Finally, we obtain its normalized version $\hat{\gamma}_{\text{CH}(\Gamma)}$ by applying Equation (13).

In Algorithm 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 (12) and (13). Then, in Line 3, we consider the convex hull property of the Bézier control points for trimming [29]. We first obtain an implicit representation of the convex hull $\gamma_{\text{CH}(\Gamma_i)}$ by applying a pair-wise r -conjunction to the hyperplane functions, see Equation (16). Then, we compute its normalized representation $\hat{\gamma}_{\text{CH}(\Gamma_i)}$,

see Equation (13). In Line 4, we trim the Bézier patch representation $\hat{\gamma}_{\Gamma_i}$ in terms of $\hat{\gamma}_{\text{CH}(\Gamma_i)}$, see Equation (14). The obtained representation is denoted by $\gamma_{\bar{\Gamma}_i}$. Finally, in Lines 5-9, we obtain the implicit representation of the model Λ by pair-wise r -conjunction of $\gamma_{\bar{\Gamma}_i}$, see Equation (15).

5.2. Gradient and Hessian

Next, we present the gradient and Hessian of the geometry implicitization. In Section 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 the derivatives of the convex-hull and Bézier patch normalized representations.

As detailed in Section 5.1, we perform an r -conjunction operation to obtain the model representation. We compute the derivatives in a straightforward manner. Let's denote by $\nabla f * \nabla g$ the matrix with coefficients $\partial_j f \partial_k g$ for $j, k = 1, \dots, d$. Then, the derivatives of the r -conjunction, presented in Equation (15), are given by

$$\nabla f \wedge g = \nabla f + \nabla g - \nabla \sqrt{f^2 + g^2}, \quad (17)$$

and

$$\nabla^2 f \wedge g = \nabla^2 f + \nabla^2 g - \nabla^2 \sqrt{f^2 + g^2}, \quad (18)$$

where

$$\nabla \sqrt{f^2 + g^2} = \frac{f \nabla f + g \nabla g}{\sqrt{f^2 + g^2}}, \quad (19)$$

and

$$\begin{aligned} \nabla^2 \sqrt{f^2 + g^2} = & \frac{\nabla f * \nabla f + f \nabla^2 f + \nabla g * \nabla g + g \nabla^2 g}{\sqrt{f^2 + g^2}} - \\ & \frac{\nabla \sqrt{f^2 + g^2} * \nabla \sqrt{f^2 + g^2}}{\sqrt{f^2 + g^2}}. \end{aligned} \quad (20)$$

Following Equation (15), we consider that $f := \gamma_\Lambda$ and $g := \gamma_{\bar{\Gamma}}$.

Similarly to the r -conjunction, we compute the derivatives of the trimming operation, presented in Equation (14). We simplify the computations by noticing that

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

where, following Equation (14), we consider $f := \hat{\gamma}_{\text{CH}(\Gamma)}$, $h := \hat{\gamma}_\Gamma$, and $\tilde{h} := \gamma_{\bar{\Gamma}}$. Then, to obtain the derivatives of the trimming operation, we differentiate the term $\sqrt{f^2 + g^2}$, see Equations (19) and (20). In this case, the derivatives of g can be computed as follows

$$\nabla g = \frac{1}{2} \left(\frac{2h^3 \nabla h + f \nabla f}{\sqrt{h^4 + f^2}} - \nabla f \right), \quad (21)$$

and

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

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

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 [27, 44]. 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 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

$$\mathcal{G}(\partial\mathcal{M}(\Lambda_k)) := \int_{\partial\mathcal{M}(\Lambda_k)} \gamma^2. \quad (22)$$

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 3. Second, we consider Equation (22) to take into account the geometric deviation. Finally, we define the functional for the mesh quality and the geometric deviation

$$\mathcal{H}(\mathcal{M}; \lambda) := \mathcal{F}(\mathcal{M}) + \lambda \mathcal{G}(\partial\mathcal{M}), \quad (23)$$

where

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

and where λ corresponds to the penalty parameter. This parameter λ can be chosen heuristically or with an automatic procedure [44].

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

Algorithm 2 Distortion minimization

Input: $\Lambda, \mathcal{M}, \hat{\mathcal{M}}, \hat{\mathbf{M}}, \varepsilon, \lambda$ **Output:** \mathcal{M}^*

- 1: $\mathbf{X} \leftarrow \text{coordinates}(\mathcal{M})$
 - 2: $\partial\mathbf{X} \leftarrow \text{coordinates}(\partial\mathcal{M})$
 - 3: $\mathbf{M} := \mathbf{M}(\hat{\mathcal{M}}, \hat{\mathbf{M}}, \mathbf{X})$ ▷ Section 4.1
 - 4: $\nabla\mathbf{M} := \nabla\mathbf{M}(\hat{\mathcal{M}}, \hat{\mathbf{M}}, \mathbf{X}); \nabla^2\mathbf{M} := \nabla^2\mathbf{M}(\hat{\mathcal{M}}, \hat{\mathbf{M}}, \mathbf{X})$ ▷ Section 4.2
 - 5: $\gamma := \gamma(\Lambda, \partial\mathbf{X})$ ▷ Section 5.1
 - 6: $\nabla\gamma := \nabla\gamma(\Lambda, \partial\mathbf{X}); \nabla^2\gamma := \nabla^2\gamma(\Lambda, \partial\mathbf{X})$ ▷ Section 5.2
 - 7: $\mathcal{F} := \mathcal{F}(\mathbf{X}, \mathbf{M});$ ▷ Section 3, Equation (5)
 - 8: $\nabla\mathcal{F} := \nabla\mathcal{F}(\mathbf{X}, \mathbf{M}, \nabla\mathbf{M}); \nabla^2\mathcal{F} := \nabla^2\mathcal{F}(\mathbf{X}, \mathbf{M}, \nabla\mathbf{M}, \nabla^2\mathbf{M})$
 - 9: $\mathcal{G} := \mathcal{G}(\partial\mathbf{X}, \gamma);$ ▷ Section 5, Equation (22)
 - 10: $\nabla\mathcal{G} := \nabla\mathcal{G}(\partial\mathbf{X}, \gamma, \nabla\gamma); \nabla^2\mathcal{G} := \nabla^2\mathcal{G}(\partial\mathbf{X}, \gamma, \nabla\gamma, \nabla^2\gamma)$
 - 11: $\mathcal{H} \leftarrow \mathcal{F} + \lambda\mathcal{G}$ ▷ Section 5, Equation (23)
 - 12: $\mathbf{X}^* \leftarrow \text{Non-linearSolver}(\mathcal{H}, \nabla\mathcal{H}, \nabla^2\mathcal{H}, \mathbf{X}, \varepsilon)$ ▷ Section 3, Equation (6)
 - 13: $\mathcal{M}^* \leftarrow \text{update coordinates of } \mathcal{M} \text{ with } \mathbf{X}^*$
-

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 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 and Section 5, respectively. Note that the derivatives of \mathcal{F} and \mathcal{G} depend on the corresponding derivatives of

\mathbf{M} and γ , respectively.

Algorithm 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 $\nabla^2\hat{\mathbf{M}}$, see Section 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 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 implicitization 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 (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} .

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 (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), 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 (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 (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 (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 [45]. 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 [46] to generate the initial isotropic linear unstructured 2D and 3D meshes (the structured meshes are generated by subdivision), and the MMG algorithm [47] to generate the initial anisotropic linear unstructured 2D and 3D meshes. To construct the geometric models, we use the FreeCAD software [48]. Finally, we use the Quickhull (Qhull) algorithm [49] for the convex-hull computations required in the geometric model’s implicitization, see Section 5.

The Julia prototyping code is sequential, it corresponds to the implementation of the method presented in this work and the one presented in [21, 23, 24]. 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 6.1. In Section 6.2 we present the optimization results for a quadrilateral and a hexahedral domain. In Section 6.3 we compare the proposed discrete based-interpolation procedure with the analytical one from [21, 23, 24]. Finally, in Sections 6.4 and 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 6.5, we illustrate that our mesh adaption method based in the metric interpolation approach is compatible with curved boundaries.

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

$$\mathbf{M} = \nabla\varphi^T \cdot \mathbf{D} \cdot \nabla\varphi. \quad (24)$$

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:

$$\mathbf{D} := \begin{pmatrix} 1 & 0 \\ 0 & 1/h(y)^2 \end{pmatrix} \quad (25)$$

where the function h is defined by

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

Similarly, the metric for the 3D case is

$$\mathbf{D} := \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1/h(z)^2 \end{pmatrix}. \quad (26)$$

The deformation map φ in Equation (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 5 shows the anisotropic quotient [50] of the metric presented in Equations (25) and (26). Specifically, the anisotropic quotient of a metric tensor $\mathbf{M} \in \mathbb{R}^{d \times d}$ is given by

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

where λ_i , $i = 1, \dots, d$ 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.

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 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 same resolution over the domain. In particular, in 2D the

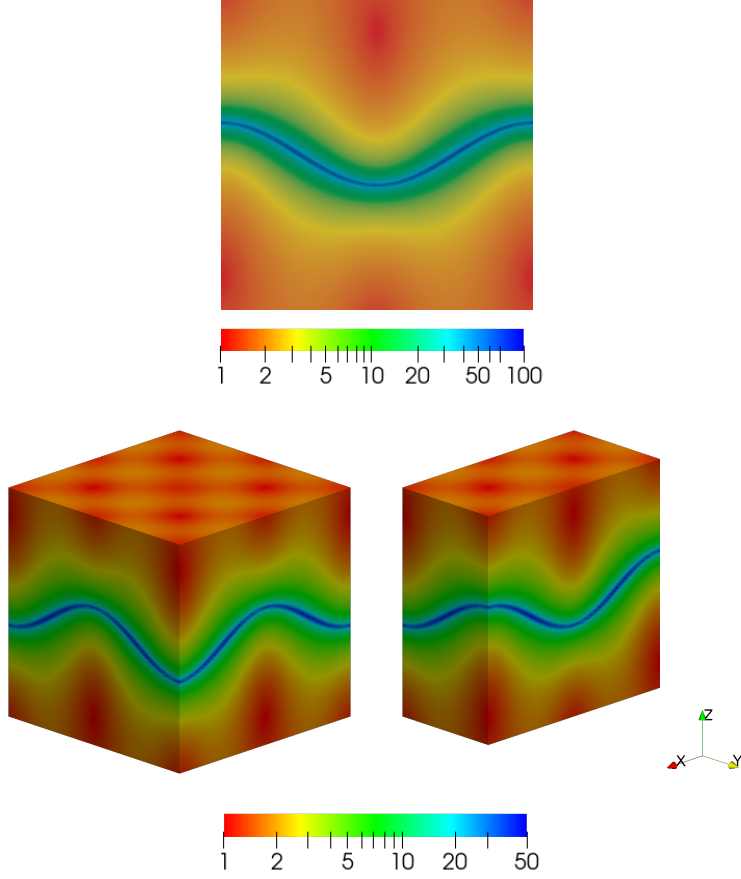


Figure 5: Anisotropic quotient values in logarithmic scale of the target metrics: (top) 2D case; (bottom left) boundaries of the 3D case; and (bottom right) solid slice of the 3D case.

three initial meshes are respectively composed of 312, 321, and 337 nodes and 558, 144, and 38 triangles, see Figures 6(a), 6(b), and 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 7(a), 7(b), 7(c), and 8(a), 8(b), 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 (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.

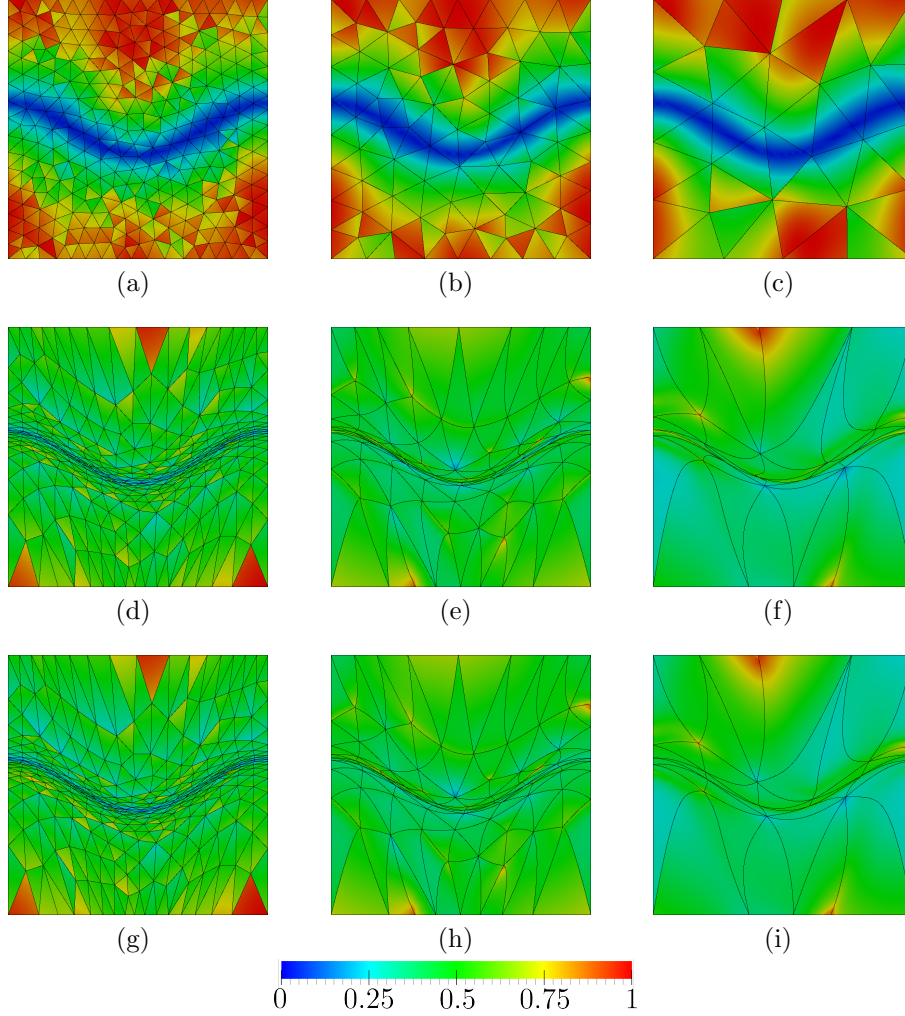


Figure 6: Point-wise distortion for triangular meshes of polynomial degree 1, 2, and 4 in columns. Initial straight-sided isotropic meshes, optimized meshes with discrete metric, and optimized meshes with analytic metric in rows.

We equip each mesh with the metric presented in Equation (24). We obtain the metric values from the log-Euclidean interpolation method presented in Section 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

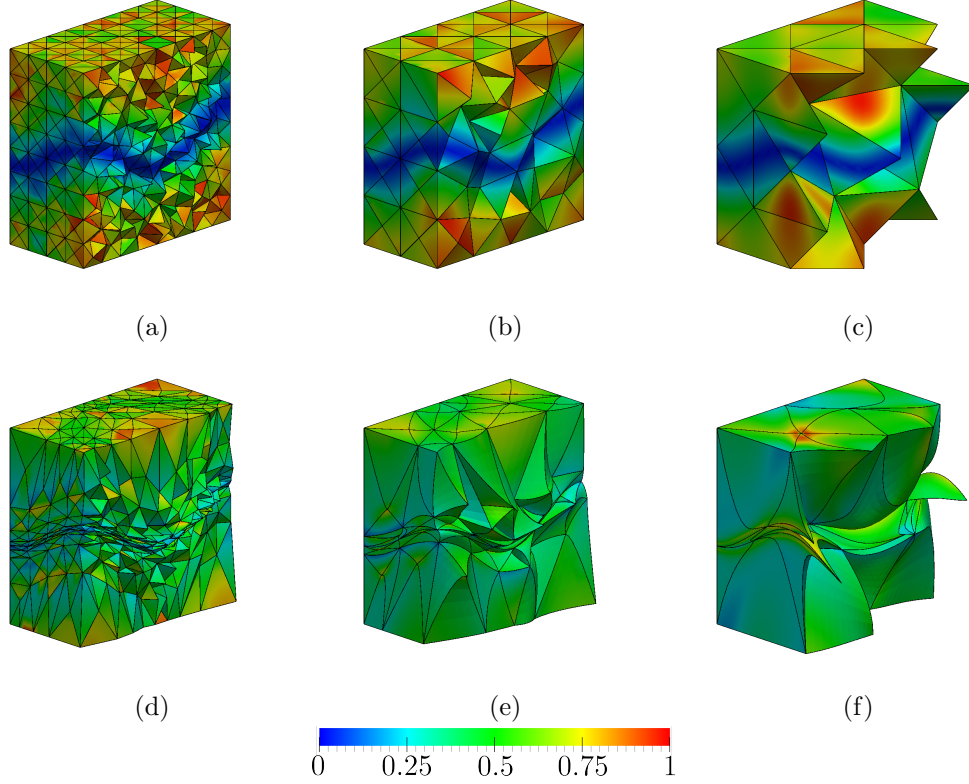


Figure 7: Clipped tetrahedral meshes of polynomial degree 1, 2, and 4 in columns. Initial straight-sided isotropic meshes and optimized meshes from initial meshes in rows.

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 relocating the mesh nodes while preserving their connectivity, as detailed in Section 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 6(d), 6(e), 6(f) we illustrate the optimized 2D meshes. In the 3D case, Figure 7(d),

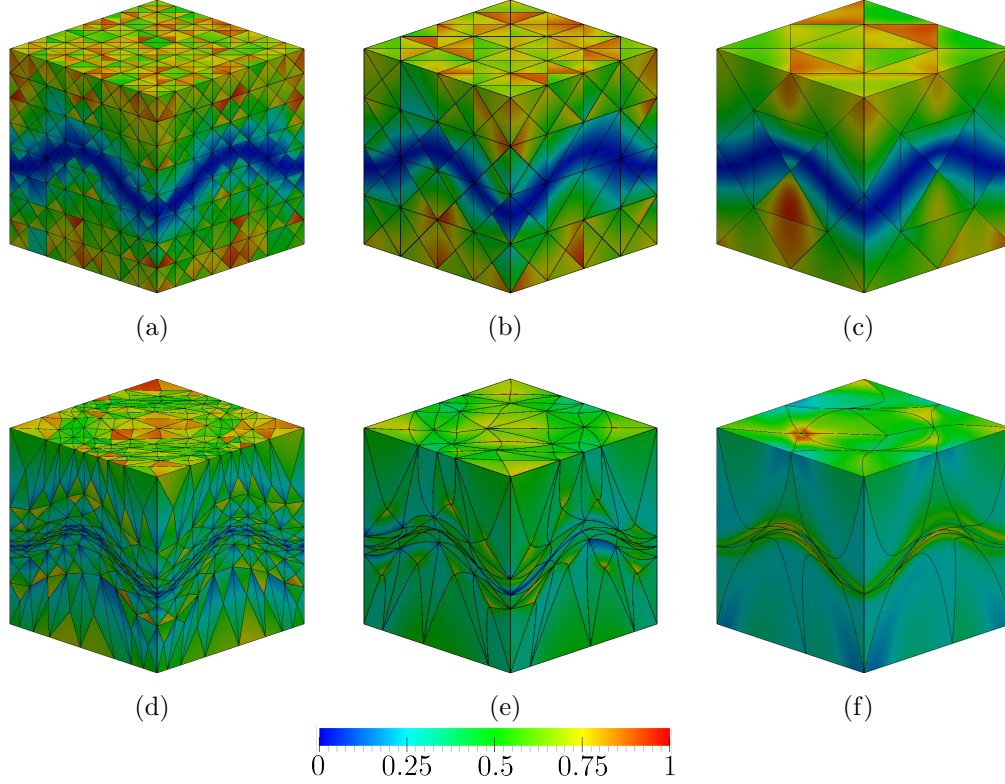


Figure 8: Boundary of tetrahedral meshes of polynomial degree 1, 2, and 4 in columns. Initial straight-sided isotropic meshes and optimized meshes from initial meshes in rows.

7(e), 7(f), and 8(d), 8(e), 8(f) show the clipped 3D meshes and the mesh boundary, respectively. We align the axes according to the ones of Figure 5. We observe that the elements lying in the anisotropic region are compressed to attain the stretching and alignment prescribed by the metric.

Tables 1 and 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 minimum 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.

Table 1: Quality statistics for the initial and optimized meshes with interpolated 2D metric.

Mesh deg.	Minimum		Maximum		Mean		Std dev.	
	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 2: Quality statistics for the initial and optimized meshes with interpolated 3D metric.

Mesh deg.	Minimum		Maximum		Mean		Std dev.	
	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

Table 3: Quality statistics for the initial and optimized meshes with analytic 2D metric.

Mesh deg.	Minimum		Maximum		Mean		Std dev.	
	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

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 6 we show the initial and optimized meshes. They are colored according to the point-wise quality measure of Equation (3) using the analytical metric expression.

To compare quantitatively both results, we compute the maximum dis-

tance 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 Figures 6(d), 6(e), and 6(f) with Figures 6(g), 6(h), and 6(i), respectively. In Table 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.

6.4. Distortion minimization: initial anisotropic straight-edged meshes

The results presented in Section 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 [6]. 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 [47], 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 (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, 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 9(a), 9(b), and 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.

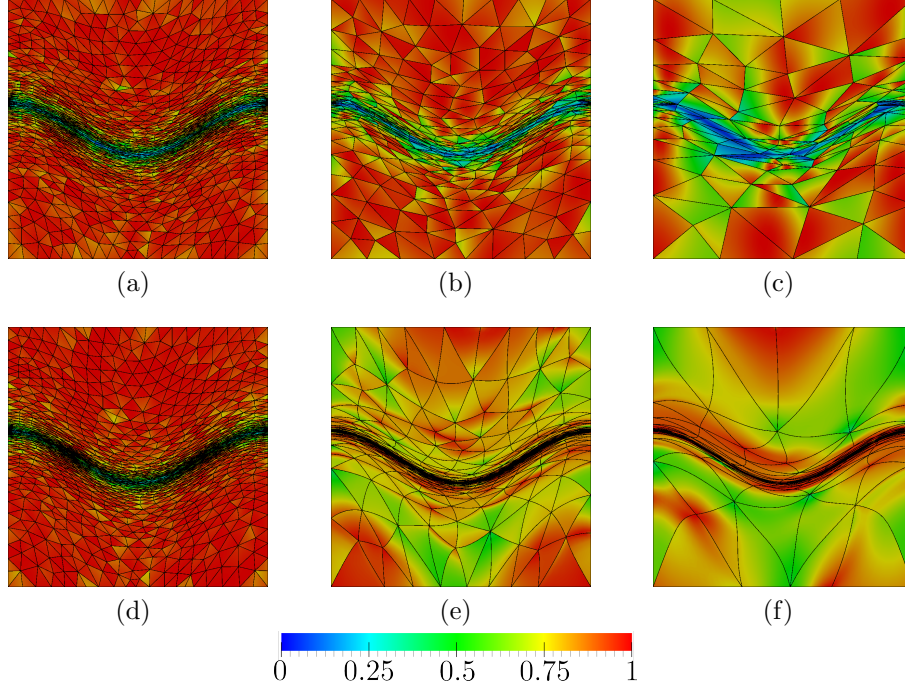


Figure 9: Point-wise distortion for triangular meshes of polynomial degree 1, 2, and 4 in columns. Initial straight-sided anisotropic meshes and optimized meshes from initial meshes in rows.

Table 4: Quality statistics for the initial MMG and optimized meshes with interpolated 2D metric.

Mesh deg.	Minimum		Maximum		Mean		Std dev.	
	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

The physical meshes \mathcal{M} are then optimized using the metric interpolation approach presented in this work. In Figures 9(d), 9(e), and 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, 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 configuration. 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, are better than for the isotropic meshes, Table 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 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.

6.5. Distortion minimization: curved boundaries

We following illustrate that our approach is compatible with curved boundaries. We consider a 2D example, in Section 6.5.1, and a 3D example, in Section 6.5.2. To this end, we first construct the geometric model with FreeCAD [48]. Next, we consider their implicit representation, see Section 5. Then, we generate the background and initial physical meshes coupled with a discrete metric, see Section 4. Finally, we apply our r -adaption method, presented in Section 3, by taking into account both the discrete metric and the implicit representation of the 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 5.3. Specifically, we set the penalty parameter $\lambda := 10^4$ in all examples, see Equation (23). In addition, to

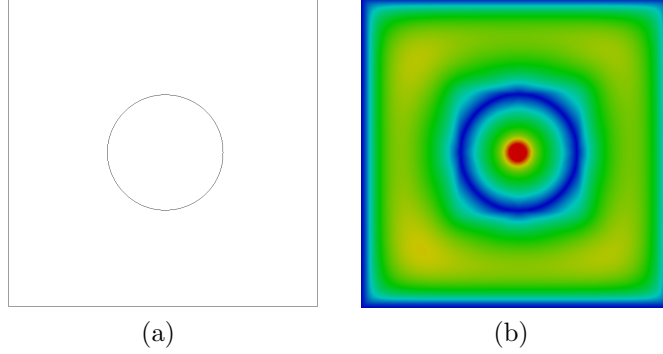


Figure 10: Parametric CAD and global implicit representation for the 2D model of a square with a circular hole.

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.

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 $\Omega_1 = K_1 \setminus C_1$, 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 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 10(b) a global implicit representation of the boundary $\Lambda_1 := \partial\Omega_1$, using the method presented in Section 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 (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 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 11(a), and 11(e). The quadratic and quartic physical

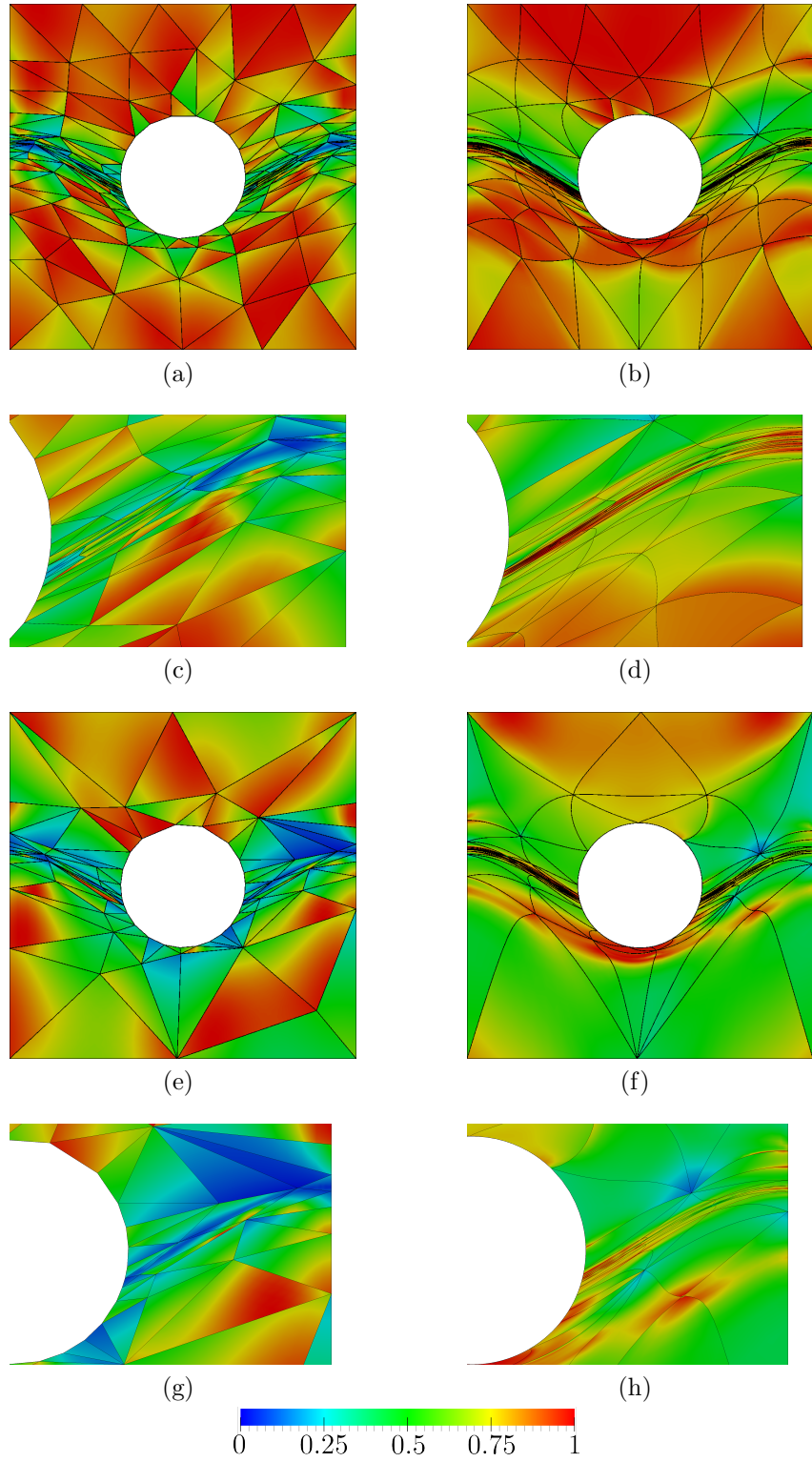


Figure 11: Point-wise distortion for triangular meshes of polynomial degree 2 in first and second (zoom) rows, and 4 in third and fourth (zoom) rows. Initial straight-sided anisotropic mesh and optimized mesh in columns.

Table 5: Quality statistics for the initial MMG and optimized mesh with interpolated 2D metric at the square with a circular hole.

Mesh deg.	Minimum		Maximum		Mean		Std dev.	
	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

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 11(b), and 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 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 11(c) and 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 11(d) and 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 implication. For the metric interpolation, the percentage of the total wall-clock time computing the derivatives is 45

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 $\Omega_2 = K_2 \setminus C_2$ 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

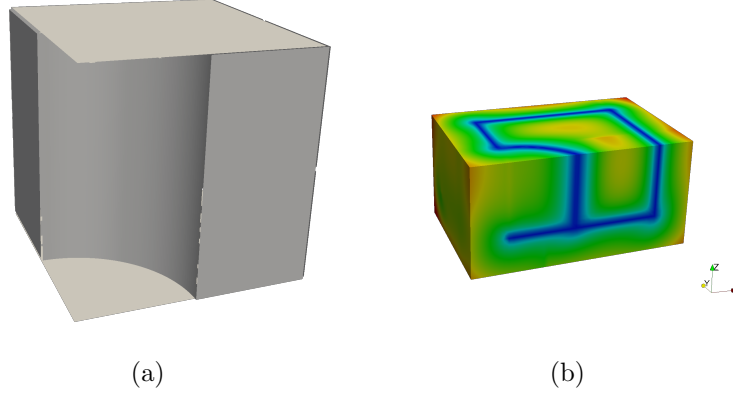


Figure 12: Parametric CAD and sliced global implicit representation for the 3D model of a cube trimmed by a cylinder.

0.25, height equal to $1/2$, and centered at the origin, see Figure 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 12(b) a global implicit representation of the boundary $\Lambda_2 := \partial\Omega_2$, using the method presented in Section 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 (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-sided anisotropic physical mesh \mathcal{M} by applying the MMG algorithm, see Figures 13(a), 13(c), and 13(e). The physical mesh is composed by 1261 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 13(b), 13(d), and 13(f), we illustrate the optimized meshes

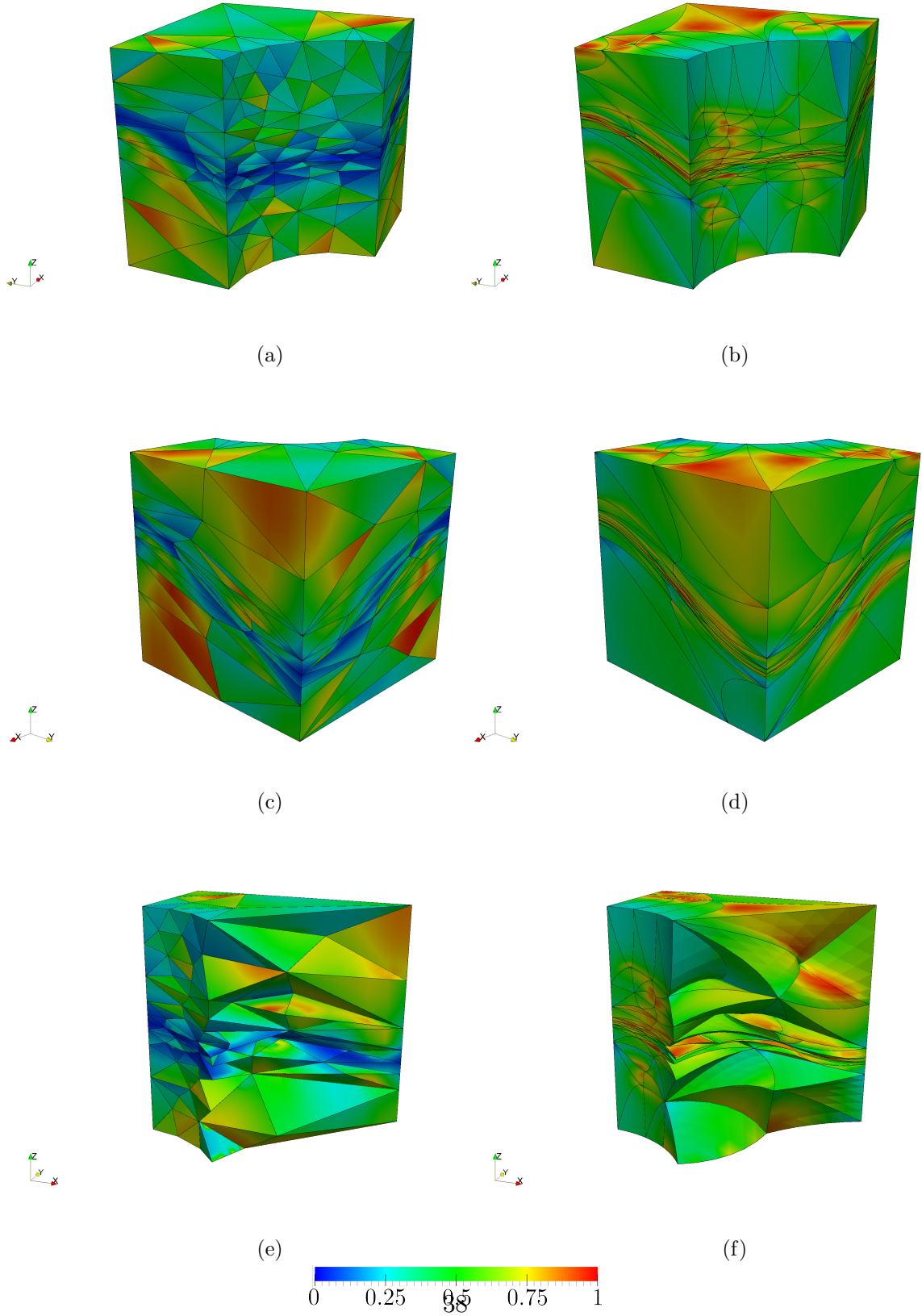


Figure 13: Point-wise distortion for quadratic tetrahedral meshes. Initial straight-sided anisotropic mesh and optimized mesh in columns.

Table 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

\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 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 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 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.

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 interpolation. 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 [1, 51]. 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 [14, 16, 25, 52] 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.

In perspective, this capability to match metric and geometry features might be an attractive ingredient for curved high-order adaption. Specifically, in goal-oriented or indicator-based adaptive processes, one would have a target high-order metric field in a current mesh approximating a target geometry. The combined approach would drive the curved r -adaption to globally (locally) relocate the current curved mesh (re-meshed cavity) according to the curved features of the solution and the geometry (cavity) boundary.

8. Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme under grant agreement No 715546. This work has also received

funding from the Generalitat de Catalunya under grant number 2017 SGR 1731. The work of X. Roca has been partially supported by the Spanish Ministerio de Economía y Competitividad under the personal grant agreement RYC-2015-01633.

Appendix A. 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 [43] and rewritten herein using our notation.

Let us consider, for $\ell = 1, \dots, d$, the eigenvalue equation for the eigenvector \mathbf{u}_ℓ 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

$$0 = \partial_j (\mathbf{L}_\ell \mathbf{u}_\ell) = (\partial_j \mathbf{L}_\ell) \cdot \mathbf{u}_\ell + \mathbf{L}_\ell \cdot \partial_j \mathbf{u}_\ell, \quad (\text{A.1})$$

$$0 = \partial_{jk} (\mathbf{L}_\ell \mathbf{u}_\ell) = (\partial_{jk} \mathbf{L}_\ell) \cdot \mathbf{u}_\ell + \mathbf{L}_\ell \cdot \partial_{jk} \mathbf{u}_\ell + (\partial_j \mathbf{L}_\ell) \cdot \partial_k \mathbf{u}_\ell + (\partial_k \mathbf{L}_\ell) \cdot \partial_j \mathbf{u}_\ell. \quad (\text{A.2})$$

For each ℓ one first computes the first-order derivative of the eigenvalue λ_ℓ by left-multiplying by \mathbf{u}_ℓ to Equation (A.1). Then, by solving the remaining unknown term of Equation (A.1) one obtains the first-order derivatives of the eigenvector \mathbf{u}_ℓ . In particular, the first-order derivatives of the eigenvalues and the eigenvectors are given by

$$\partial_j \lambda_\ell = \mathbf{u}_\ell^T \cdot \partial_j \mathbf{L} \cdot \mathbf{u}_\ell, \quad \partial_j \mathbf{u}_\ell = -\mathbf{L}_\ell^+ \cdot \partial_j \mathbf{L}_\ell \cdot \mathbf{u}_\ell,$$

where the operation \mathbf{L}_ℓ^+ is the Moore-Penrose pseudo-inverse matrix for the matrix \mathbf{L}_ℓ . We use the Moore-Penrose pseudo-inverse matrix instead of the inverse matrix because the matrix \mathbf{L}_ℓ 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}_ℓ to Equation (A.2). Then, by solving the remaining unknown term of Equation (A.2) one obtains the second-order

derivatives of the eigenvector \mathbf{u}_ℓ . In particular, the second-order derivatives of the eigenvalues are given by

$$\partial_{jk}\lambda_\ell = \mathbf{u}_\ell^T \cdot (\partial_k \mathbf{L}_\ell \cdot \partial_j \mathbf{u}_\ell + \partial_j \mathbf{L}_\ell \cdot \partial_k \mathbf{u}_\ell + \partial_{jk} \mathbf{L} \cdot \mathbf{u}_\ell),$$

$$\partial_{jk}\mathbf{u}_\ell = -\mathbf{L}_\ell^+ \cdot (\partial_k \mathbf{L}_\ell \cdot \partial_j \mathbf{u}_\ell + \partial_j \mathbf{L}_\ell \cdot \partial_k \mathbf{u}_\ell + \partial_{jk} \mathbf{L}_\ell \cdot \mathbf{u}_\ell) - (\partial_j \mathbf{u}_\ell \cdot \partial_k \mathbf{u}_\ell) \mathbf{u}_\ell,$$

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 $\mathbf{u}_\ell^T \cdot \mathbf{u}_\ell = 1$

$$0 = \partial_{jk} (\mathbf{u}_\ell^T \cdot \mathbf{u}_\ell) = 2\partial_{jk}\mathbf{u}_\ell^T \cdot \mathbf{u}_\ell + 2\partial_j \mathbf{u}_\ell^T \cdot \partial_k \mathbf{u}_\ell.$$

Appendix B. 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 5.2.

Herein, we consider the gradient and Hessian of the normalized representation $\hat{\gamma}$, presented in Equation (13). As before, we denote by $\nabla f * \nabla g$ the matrix with coefficients $\partial_j f \partial_k g$ for $j, k = 1, \dots, d$. 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 $j, k = 1, \dots, d$. Then, the derivatives of the normalized representation are given by

$$\nabla \hat{\gamma} = \frac{\nabla \gamma - \hat{\gamma} \nabla \|\nabla \gamma\|}{\|\nabla \gamma\|}, \quad (\text{B.1})$$

and

$$\hat{\gamma} \nabla^2 \hat{\gamma} = \frac{\hat{\gamma} \nabla^2 \gamma - \hat{\gamma}^2 \nabla^2 \|\nabla \gamma\| - \nabla \hat{\gamma} \otimes \hat{\gamma} \nabla \|\nabla \gamma\|}{\|\nabla \gamma\|}, \quad (\text{B.2})$$

where

$$\begin{aligned} \hat{\gamma} \nabla \|\nabla \gamma\| &= \frac{\hat{\gamma} \nabla^2 \gamma \cdot \nabla \gamma}{\|\nabla \gamma\|}, \\ \hat{\gamma}^2 \nabla^2 \|\nabla \gamma\| &= \frac{\hat{\gamma}^2 \nabla^3 \gamma \cdot \nabla \gamma + \hat{\gamma} \nabla^2 \gamma \cdot \hat{\gamma} \nabla^2 \gamma - \hat{\gamma} \nabla \|\nabla \gamma\| * \hat{\gamma} \nabla \|\nabla \gamma\|}{\|\nabla \gamma\|}. \end{aligned}$$

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 (21) and (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 $\hat{\gamma}_{\text{CH}(\Gamma)}$. 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 $\gamma_{\text{CH}(\Gamma)}$, see Equations (17) and (18). Finally, we differentiate the normalization of the convex-hull representation $\hat{\gamma}_{\text{CH}(\Gamma)}$, see Equations (B.1) and (B.2).

Now, we compute the derivatives for the determinant γ of Equation (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

$$\nabla\gamma(\mathbf{x}) = \text{tr}(\text{adj}(\mathbb{N}(\mathbf{x})) \cdot \nabla\mathbb{N}(\mathbf{x})), \quad (\text{B.3})$$

where $\mathbb{N}(\mathbf{x}) := \mathbb{M}(\mathbf{x}) \cdot \mathbb{M}(\mathbf{x})^T$. We consider the adjugate matrix $\text{adj}(\mathbb{N}(\mathbf{x}))$, instead of the inverse matrix, to avoid the singularity issues at the patch Γ . In particular, the adjugate matrix of $\mathbb{N}(\mathbf{x})$ is defined by the transposed cofactor matrix, and satisfying the relation $\text{adj}(\mathbb{N}(\mathbf{x})) = \gamma(\mathbf{x})\mathbb{N}(\mathbf{x})^{-1}$ [29]. 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.3). In particular, using the same notation as in Section 4.2, we compute the second derivatives for each j and k as

$$\gamma(\mathbf{x})\partial_{jk}\gamma(\mathbf{x}) = \text{tr}(\gamma(\mathbf{x})\partial_k\text{adj}(\mathbb{N}(\mathbf{x})) \cdot \partial_j\mathbb{N}(\mathbf{x}) + \gamma(\mathbf{x})\text{adj}(\mathbb{N}(\mathbf{x})) \cdot \partial_{jk}\mathbb{N}(\mathbf{x})).$$

In addition, the third derivatives are given by

$$\begin{aligned} \gamma(\mathbf{x})^2\partial_{jkl}\gamma(\mathbf{x}) = & \text{tr}(\gamma(\mathbf{x})^2\partial_{kl}\text{adj}(\mathbb{N}(\mathbf{x})) \cdot \partial_j\mathbb{N}(\mathbf{x}) + \\ & (\gamma(\mathbf{x})^2\partial_k\text{adj}(\mathbb{N}(\mathbf{x})) \cdot \partial_{jl}\mathbb{N}(\mathbf{x}) + \gamma(\mathbf{x})^2\partial_\ell\text{adj}(\mathbb{N}(\mathbf{x})) \cdot \partial_{jk}\mathbb{N}(\mathbf{x})), \end{aligned}$$

for each j , k , and ℓ . Note that, there is no third order term $\partial_{jkl}\mathbb{N}(\mathbf{x})$ because $\mathbb{N}(\mathbf{x})$ is a quadratic function on \mathbf{x} , see Equation (12).

Finally, we provide the derivatives of the adjugate matrix $\text{adj}(\mathbb{N}(\mathbf{x}))$. 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

$$\begin{aligned}\gamma(\mathbf{x}) \nabla \text{adj}(\mathbb{N}(\mathbf{x})) &= \gamma(\mathbf{x}) \nabla (\gamma(\mathbf{x}) \mathbb{N}(\mathbf{x})^{-1}) = \\ &\text{adj}(\mathbb{N}(\mathbf{x})) \nabla \gamma(\mathbf{x}) - \mathbb{N}(\mathbf{x}) \cdot \text{adj}(\mathbb{N}(\mathbf{x})) \cdot \mathbb{N}(\mathbf{x}).\end{aligned}$$

We apply the same reasoning for the Hessian by computing

$$\gamma(\mathbf{x})^2 \nabla^2 \text{adj}(\mathbb{N}(\mathbf{x})) = \gamma(\mathbf{x})^2 \nabla \left(\frac{1}{\gamma(\mathbf{x})} \gamma(\mathbf{x}) \nabla \text{adj}(\mathbb{N}(\mathbf{x})) \right).$$

In particular, using the same notation as in Section 4.2, for each j and k we have

$$\begin{aligned}\gamma^2 \partial_{jk} \text{adj}(\mathbb{N}) &= (\gamma \partial_{jk} \gamma) \text{adj}(\mathbb{N}) + (\partial_j \gamma) \gamma \partial_k \text{adj}(\mathbb{N}) - (\partial_k \gamma) \gamma \partial_j \text{adj}(\mathbb{N}) - \\ &\gamma \text{adj}(\mathbb{N}) \cdot \partial_{jk} \mathbb{N} \cdot \text{adj}(\mathbb{N}) - \gamma \partial_k \text{adj}(\mathbb{N}) \cdot \partial_j \mathbb{N} \cdot \text{adj}(\mathbb{N}) - \text{adj}(\mathbb{N}) \cdot \partial_j \mathbb{N} \cdot \gamma \partial_k \text{adj}(\mathbb{N}),\end{aligned}$$

where, for the sake of brevity, we omit the dependence on the \mathbf{x} variable of the functions γ and \mathbb{N} .

References

- [1] M. Yano, D. L. Darmofal, An optimization-based framework for anisotropic simplex mesh adaptation, *Journal of Computational Physics* 231 (22) (2012) 7626–7649.
- [2] A. Loseille, F. Alauzet, Continuous mesh framework part i: well-posed continuous interpolation error, *SIAM Journal on Numerical Analysis* 49 (1) (2011) 38–60. doi:doi.org/10.1137/090754078.
- [3] T. Coupez, L. Silva, E. Hachem, Implicit boundary and adaptive anisotropic meshing, in: *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, Springer, 2015, pp. 1–18.
- [4] W. Huang, R. D. Russell, *Adaptive Moving Mesh Methods*, Vol. 174 of *Applied Mathematical Sciences*, Springer, 2011.
- [5] P. M. Knupp, Algebraic mesh quality metrics, *SIAM J. Numer. Anal.* 23 (1) (2001) 193–218.

- [6] F. Alauzet, A. Loseille, A decade of progress on anisotropic mesh adaptation for computational fluid dynamics, *Computer-Aided Design*, Elsevier 72 (1) (2016) 13–39.
- [7] C. Gruau, T. Coupez, 3d tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric, *Computer Methods in Applied Mechanics and Engineering* 194 (48-49) (2005) 4951–4976.
- [8] P. Frey, F. Alauzet, Anisotropic mesh adaptation for cfd computations, *Computer methods in applied mechanics and engineering* 194 (48-49) (2005) 5068–5082.
- [9] F. Hecht, Bamg: bidimensional anisotropic mesh generator, User Guide. INRIA, Rocquencourt.
- [10] T. Coupez, Metric construction by length distribution tensor and edge based error for anisotropic adaptive meshing, *Journal of computational physics* 230 (7) (2011) 2391–2405.
- [11] K. J. Fidkowski, D. L. Darmofal, Review of output-based error estimation and mesh adaptation in computational fluid dynamics, *AIAA journal* 49 (4) (2011) 673–694.
- [12] T. Coupez, On a basis framework for high order anisotropic mesh adaptation 203 (2017) 141–153, research Note 26th International Meshing Roundtable.
- [13] A. Johnen, C. Geuzaine, T. Toulorge, J.-F. Remacle, Quality measures for curvilinear finite elements, *TILDA: Towards Industrial LES/DNS in Aeronautics* (2021) 221.
- [14] R. Zhang, A. Johnen, J.-F. Remacle, Curvilinear mesh adaptation, in: *International Meshing Roundtable*, Springer, 2018, pp. 57–69.
- [15] M. J. Zahr, P.-O. Persson, An optimization based discontinuous galerkin approach for high-order accurate shock tracking, in: *2018 AIAA Aerospace Sciences Meeting*, 2018, p. 0063.
- [16] M. J. Zahr, A. Shi, P.-O. Persson, Implicit shock tracking using an optimization-based high-order discontinuous galerkin method, *Journal of Computational Physics* 410 (2020) 109385.

- [17] M. J. Zahr, P.-O. Persson, An r-adaptive, high-order discontinuous galerkin method for flows with attached shocks, in: AIAA Scitech 2020 Forum, 2020, p. 0537.
- [18] J. Marcon, M. Turner, D. Moxey, S. J. Sherwin, J. Peiró, A variational approach to high-order r-adaptation, IMR26.
- [19] J. Marcon, G. Castiglioni, D. Moxey, S. J. Sherwin, J. Peiró, rp-adaptation for compressible flows, *International Journal for Numerical Methods in Engineering* 121 (23) (2020) 5405–5425.
- [20] V. Dobrev, P. Knupp, T. Kolev, K. Mittal, V. Tomov, The target-matrix optimization paradigm for high-order meshes, *SIAM Journal on Scientific Computing* 41 (1) (2019) B50–B68.
- [21] G. Aparicio-Estrems, A. Gargallo-Peiró, X. Roca, Defining a stretching and alignment aware quality measure for linear and curved 2d meshes, in: *International Meshing Roundtable*, Springer, 2018, pp. 37–55.
- [22] G. Aparicio-Estrems, A. Gargallo-Peiró, X. Roca, High-order metric interpolation for curved r-adaption by distortion minimization, in: *Proceedings of the 2022 SIAM International Meshing Roundtable*, Zenodo, 2022, pp. 1–12.
- [23] G. Aparicio-Estrems, A. Gargallo-Peiró, X. Roca, Anisotropic optimization of curved meshes: specific-purpose line-search and trust-region globalizations for newton’s method, in: *International Meshing Roundtable*, 2019.
- [24] G. Aparicio-Estrems, A. Gargallo-Peiró, X. Roca, Stretching and aligning piece-wise polynomial meshes to match curved anisotropic features, in: *International Conference on Spectral and High-Order Methods*, 2021.
- [25] L. Rochery, A. Loseille, P2 cavity operator and riemannian curved edge length optimization: a path to high-order mesh adaptation, in: *AIAA Scitech 2021 Forum*, 2021, p. 1781.
- [26] V. Dobrev, P. Knupp, T. Kolev, V. Tomov, Towards simulation-driven optimization of high-order meshes by the target-matrix optimization paradigm, in: *International Meshing Roundtable*, Springer, 2018, pp. 285–302.

- [27] E. Ruiz-Gironés, J. Sarrate, X. Roca, Generation of curved high-order meshes with optimal quality and geometric accuracy, *Procedia engineering* 163 (2016) 315–327.
- [28] E. Ruiz-Gironés, A. Gargallo-Peiró, J. Sarrate, X. Roca, An augmented lagrangian formulation to impose boundary conditions for distortion based mesh moving and curving, *Procedia engineering* 203 (2017) 362–374.
- [29] K. Upreti, T. Song, A. Tambat, G. Subbarayan, Algebraic distance estimations for enriched isogeometric analysis, *Computer Methods in Applied Mechanics and Engineering* 280 (2014) 28–56.
- [30] B. Laurent, Implicit matrix representations of rational bézier curves and surfaces, *Computer-Aided Design* 46 (2014) 14–24.
- [31] A. Biswas, V. Shapiro, Approximate distance fields with non-vanishing gradients, *Graphical Models* 66 (3) (2004) 133–159.
- [32] V. Dobrev, P. Knupp, T. Kolev, K. Mittal, R. Rieben, V. Tomov, Simulation-driven optimization of high-order meshes in ale hydrodynamics, *Computers & Fluids* 208 (2020) 104602.
- [33] V. Arsigny, P. Fillard, X. Pennec, N. Ayache, Log-euclidean metrics for fast and simple calculus on diffusion tensors, *Magnetic Resonance in Medicine* 56 (2006) 411–421.
- [34] P. Knupp, T. Kolev, K. Mittal, V. Z. Tomov, Adaptive surface fitting and tangential relaxation for high-order mesh optimization, *arXiv e-prints* (2021) arXiv–2105.
- [35] L. V. Branets, V. A. Garanzha, Distortion measure of trilinear mapping. application to 3-d grid generation, *Numerical linear algebra with applications* 9 (6-7) (2002) 511–526.
- [36] E. J. López, N. M. Nigro, M. A. Storti, Simultaneous untangling and smoothing of moving grids, *Int. J. Numer. Meth. Eng.* 76 (7) (2008) 994–1019.
- [37] J. M. Escobar, E. Rodríguez, R. Montenegro, G. Montero, J. M. González-Yuste, Simultaneous untangling and smoothing of tetrahedral meshes, *Comput. Meth. Appl. Mech. Eng.* 192 (25) (2003) 2775–2787.

- [38] A. Gargallo-Peiró, X. Roca, J. Peraire, J. Sarrate, Optimization of a regularized distortion measure to generate curved high-order unstructured tetrahedral meshes, *Int. J. Numer. Meth. Eng.* 103 (2015) 342–363.
- [39] A. Gargallo-Peiró, Validation and generation of curved meshes for high-order unstructured methods, Ph.D. thesis, Universitat Politècnica de Catalunya (2014).
- [40] D. Ekelschot, M. Ceze, S. M. Murman, A. Garai, Parallel high-order anisotropic meshing using discrete metric tensors, in: *AIAA Scitech 2019 Forum*, 2019, p. 1993.
- [41] K. Mittal, S. Dutta, P. Fischer, Nonconforming schwarz-spectral element methods for incompressible flow, *Computers & Fluids* 191 (2019) 104237.
- [42] J. Sitaraman, M. Floros, A. Wissink, M. Potsdam, Parallel domain connectivity algorithm for unsteady flow computations using overlapping and adaptive grids, *Journal of Computational Physics* 229 (12) (2010) 4703–4723.
- [43] A. L. Andrew, K.-W. E. Chu, P. Lancaster, Derivatives of eigenvalues and eigenvectors of matrix functions, *SIAM journal on matrix analysis and applications* 14 (4) (1993) 903–926.
- [44] E. Ruiz-Gironés, X. Roca, Automatic penalty and degree continuation for parallel pre-conditioned mesh curving on virtual geometry, *Computer-Aided Design* (2022) 103208.
- [45] J. Bezanson, A. Edelman, S. Karpinski, V. B. Shah, Julia: A fresh approach to numerical computing, *SIAM review* 59 (1) (2017) 65–98.
- [46] MATLAB, version 9.3.0.713579 (R2017b), The MathWorks Inc., Natick, Massachusetts, 2017.
- [47] C. Dobrzynski, Mmg3d: User guide, Ph.D. thesis, INRIA (2012).
- [48] J. Riegel, W. Mayer, Y. van Havre, Freecad (2016).
- [49] C. B. Barber, D. P. Dobkin, H. Huhdanpaa, The quickhull algorithm for convex hulls, *ACM Transactions on Mathematical Software (TOMS)* 22 (4) (1996) 469–483.

- [50] A. Loseille, R. Löhner, Anisotropic adaptive simulations in aerodynamics, AIAA 2010-16949th AIAA Aerospace Sciences Meeting, Fairfax, Va, USA.
- [51] O. Coulaud, A. Loseille, Very high order anisotropic metric-based mesh adaptation in 3d, Procedia Engineering 163, 25th International Meshing Roundtable.
- [52] R. Feuillet, A. Loseille, F. Alauzet, Optimization of p2 meshes and applications, Computer-Aided Design 124.