SUBDIVIDED LINEAR AND CURVED MESHES PRESERVING FEATURES OF A LINEAR MESH MODEL

Albert Jiménez-Ramos[1] Abel Gargallo-Peiró[1] Xevi Roca[1]

[1] *Barcelona Supercomputing Center, 08304, Barcelona, Spain. xevi.roca@bsc.es*

ABSTRACT

To provide straight-edged and curved piece-wise polynomial meshes that target a unique smooth geometry while preserving the sharp features and smooth regions of the model, we propose a new fast curving method based on hierarchical subdivision and blending. There is no need for underlying target geometry, it is only needed a straight-edged mesh with boundary entities marked to characterize the geometry features, and a list of features to recast. The method features a unique sharp-to-smooth modeling capability not fully available in standard CAD packages. The goal is to obtain a volume mesh that under successive refinement leads to smooth regions bounded by the corresponding sharp features. The examples show that it is possible to refine and obtain smooth curves and surfaces while preserving sharp features determined by vertices and polylines. We conclude that the method is well-suited to curve large quadratic and quartic meshes in low-memory configurations.

Keywords: mesh curving, surrogate geometry, geometry modeling, subdivision, blending

1. INTRODUCTION

In flow simulations for wind energy, transport of pollutants, and bio-engineering the boundary of the computational domain is usually represented by a straight-edged mesh obtained by sampling real data. This straight-edged mesh approximates the geometry, at different scales, corresponding to the viscous surfaces to analyze such as topography [1, 2], urban areas [3], and human organs. The mesh also presents a series of sharp features, vertices, polylines bounded by vertices, that the method should preserve, and that bound the smooth regions of the computational model.

The resolution to approximate the geometry could be insufficient for the required flow analysis, and thus, additional refinement of the boundary mesh would be required. However, a standard refinement approach, when no target geometry is available, could be inadequate for flow simulation in a twofold way. First, the refined mesh might reproduce precisely the geometry of the first straight-edged mesh and thus, introduce artificial flow artifacts close to initially non-smooth features that should be smooth. Second, the refined mesh might target a smooth surface geometry, implicitly determined by the initial straight-edged mesh, but without adequately respecting after successive refinement the sharp features, curves, and vertices, of the computational model. Ideally, vertices should remain fixed, and polylines should target a smooth limit curve.

Solving these issues is essential for those flow analyses that start from a mesh obtained by sampling real data where the computational model presents smooth regions bounded by sharp features. Even they can be useful in aeronautical applications where only legacy data, in a format of vertices, and polyline and surface meshes, is available. In some applications, practitioners might also need, a non-standard but flexible *sharp-to-smooth modeling* capability, to remove some sharp features ensuring that surrounding regions become smooth along with the removed feature.

Intending to provide piece-wise linear meshes or curved piece-wise polynomial meshes that target a unique smooth geometry while preserving the sharp features of the model, our contribution is to propose a new fast curving method based on hierarchical sub-

division and blending with sharp-to-smooth modeling capabilities. Our approach only needs an initial straight-edged mesh with boundary triangles marked with surface identifiers, and a list of features to recast. There is no need for underlying target geometry. The goal of the method is to obtain a volume mesh of the flow domain that under successive refinement leads to smooth regions bounded by the sharp features determined after recasting. The recasting operation is devised to implement a sharp-to-smooth modeling capability. We favored a fast and explicit curving method, based on subdivision and blending, to an implicit approach formulation that features validity guarantees or untangling capabilities, based on boundary curving and optimization, but slower and more memory demanding. This favoring is so since the appearance of invalid elements is small compared with the scale of the generated meshes, and fast local untangling can repair those invalid elements. This work details our mesh curving methodology and illustrates its application with the included numerical examples.

The proposed fast curving method based on hierarchical subdivision and blending with sharp-to-smooth modeling capabilities is novel in many aspects:

- In mesh curving it is standard to have explicit access to the boundary of the target geometry [4, 5, 6, 7, 8, 9, 10, 11, 12] however, fewer works have considered the case when the target geometry is not explicitly available. The work presented in [13] proposes two curving methods based in weighted least squares approximations and piece-wise polynomial fittings to generate curved meshes of the target surfaces. Both techniques guarantee $\mathcal{C}^0$-continuous curved surface meshes whereas, in this work we approximate with at least $\mathcal{C}^0$-continuous meshes a surrogate geometry composed of feature surfaces with an interior that is $\mathcal{C}^1$-continuous and $\mathcal{C}^2$-continuous almost everywhere. Furthermore, herein, the $\mathcal{C}^0$-continuity of the surface mesh is increased to $\mathcal{C}^2$-continuity when using regular configurations of quartic elements to interpolate the limit surface.

- The curved surface meshes provided by the method in [13] can be used to bound the mesh volumes and thus, to generate curved high-order meshes when the boundary surfaces are not explicitly available. Furthermore, it is possible to remove sharp features by selecting one-by-one the mesh entities defining it [14]. Besides the surface mesh curving method, the main differences with the latter work are that herein we use: a hierarchical blending approach to curve the mesh volume; and an all-in-one feature selection to perform sharp-to-smooth recasting.

- In mesh generation, Loop's subdivision surfaces have been used before to define surrogate geometry [15] and also to curve quadratic and quartic surface meshes [16] but not to obtain volume meshes with curved boundaries. Later, the butterfly subdivision scheme [17] has been used to relocate the boundary nodes when the geometry is unavailable [18], but the volume is not curved using hierarchical blending. Note that previous methods use blending for mesh curving but in the specific case of curving boundary layer elements when the target geometry is available [19].

- There are alternative subdivision methods to generate curved volume meshes, featuring parallel implementations, but they need as input a curved mesh to define the surrogate geometry [20].

The organization of the rest of the paper is as follows. First, in Sect. 2, we present the problem statement and the methodology used in this work to solve it. Second, in Sect. 3, we present some preliminary results on subdivision methods required to develop the main contributions of this paper. We detail these contributions in Sect. 4 and Sect. 5, where we present the method to generate curved high-order surface and volume meshes from a given linear mesh when the target geometry is unavailable. Following, in Sect. 6, we present several results to illustrate the capabilities and main features of the presented methods. To conclude, in Sect. 7, we present some concluding remarks, to sum up the main contributions of this work.

2. PROBLEM STATEMENT AND METHODOLOGY

2.1 Problem Statement

The input data is a tetrahedral mesh with its boundary entities marked to characterize the geometry features, and a list of the features to be recast. A geometry feature is characterized by a set of entities of the mesh with the same identifier. Specifically, a vertex feature describes a vertex point to preserve, and it is characterized by the global identifier of the point to be preserved. A curve feature describes a curve to preserve. Each curve feature is characterized by a list of edges with the same curve identifier. Finally, a surface feature describes a surface to preserve. A list of boundary triangles with the same identifier characterizes a surface feature.

Alternatively, we can obtain the geometry features by considering a tetrahedral mesh when only the surface features are described. That is, if only the boundary triangles are marked, we can retrieve the feature curves from the intersection of the boundary of two or more feature surfaces. Similarly, vertex features can

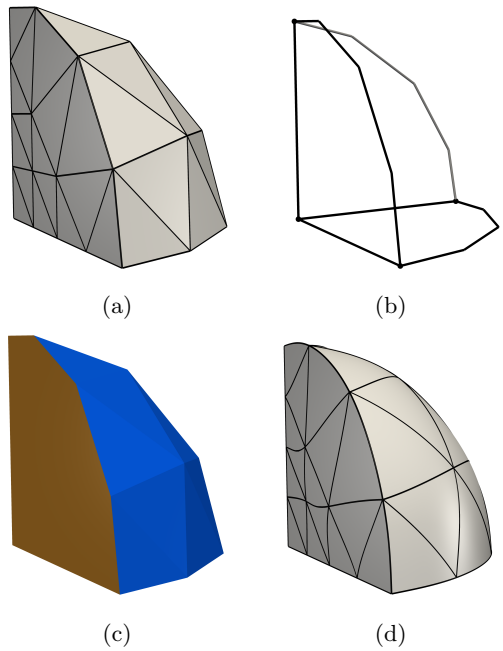


Figure 1: Method: (a) a linear tetrahedral mesh, with marked boundary (b) curves and (c) surfaces, is curved to obtain (d) a curved tetrahedral mesh of polynomial degree four.

be determined by the intersection of two or more feature curves.

In addition to the tetrahedral mesh and the geometry features, we have an optional input of which geometry features to recast. Recasting a geometry feature consists in removing it from the list of features to preserve and merging adjacent regions to obtain a smooth model along the removed feature. That is, if we recast a curve, we remove the curve feature and merge the two adjacent surfaces. While when we recast a vertex, we remove the vertex feature and merge the curves incident to the vertex. Since each geometry feature is associated with a unique identifier, the list of features to be recast is a sub-sequence of these unique identifiers.

Given the input modeled tetrahedral mesh, the output of this work is a high-order tetrahedral mesh of polynomial degree p with a boundary preserving the marked sharp features and satisfying three properties. First, high-order element vertices interpolate the initial linear mesh nodes. Second, the nodes of the high-order edges that belong to a feature curve and are not adjacent to a feature vertex (*inner curve edges*) interpolate a cubic $\mathcal{C}^2$-continuous curve. Third, the nodes of the high-order elements that belong to a feature surface and are not adjacent to a feature curve or vertex (*inner surface elements*) interpolate an almost everywhere quartic $\mathcal{C}^2$-continuous surface. These properties provide regularity guarantees in the output mesh that are discussed in Sect. 4.3.

2.2 Method: Hierarchical Subdivision and Blending

The curved high-order mesh generation procedure proposed in this work is composed of four main steps:

1. **Approximate a surrogate boundary**. Given a linear tetrahedral mesh, Fig. 1(a), we extract its boundary. The boundary is a linear triangular mesh with its entities marked, see Fig. 1(b) and Fig. 1(c), and by means of the hierarchical subdivision of its elements we generate a curved high-order triangular surface mesh. The curved surface mesh approximates a surrogate boundary composed of feature surfaces with an interior that is $\mathcal{C}^1$-continuous and $\mathcal{C}^2$-continuous almost everywhere.

   This surrogate is determined by the subdivision of the curves and surfaces, and preserves the sharp features and smooth regions marked on the boundary of the initial volume mesh. See details in Sect. 4.

2. **Substitute the boundary of the volume mesh**. We increase the polynomial degree of the volume mesh and replace the straight-sided boundary of the current high-order volume mesh by the high-order surface mesh obtained in the first step. See details in Sect. 5.1.

3. **Accommodate the curvature of the boundary**. We accommodate the curvature of the curved surface mesh to the boundary volume elements using an explicit hierarchical blending, see Fig. 1(d). See details in Sect. 5.2.

4. **Local untangling**. If necessary, we optimize the inverted elements locally following the approach detailed in [8, 9].

In some applications, it may be desired to perform a sharp-to-smooth modeling of the geometry. Therefore, as a preprocess, we can recast the non-desired geometry features accordingly to the list of features to remove provided as input to obtain a smoother surrogate geometry. The recasting process is detailed in Sect. 5.1.

3. PRELIMINARIES: CURVE AND SURFACE MESH SUBDIVISION

In this section, we present the approximative subdivision algorithms that we use to refine the boundary

of a tetrahedral volume mesh. The boundary mesh is composed of vertices, curves, and surfaces. Then, the subdivision is performed hierarchically, that is, vertices remain fixed, curves are refined using a curve subdivision scheme and surfaces are subdivided using a surface subdivision scheme. Our goal is to generate a finer boundary mesh that targets a smooth limit surface. Therefore, the curve subdivision scheme has to generate new points consistently with the subdivision surface scheme.

In Sect. 3.1, we detail the curve subdivision algorithm we use in this work, and Sect. 3.2 recalls Loop's subdivision surface scheme. Although these methods are approximative, in Sect. 3.3 we explain how they can be modified to interpolate the initial mesh points preserving the same continuity properties.

3.1 Curve Subdivision Scheme

The curve subdivision scheme used in this work was originally presented in [21]. Successive applications of the algorithm generate finer polygons, all of them with the same limit curve determined by the initial mesh, denoted as control mesh. This curve is parametrized by a polynomial of degree three and it is $\mathcal{C}^2$-continuous.

Given a polygon, the curve subdivision scheme consists of three steps: generate a new node for each edge, update the position of the original nodes, and define the new elements of the finer mesh. A new edge point is generated at the midpoint of the two endpoints of the segment. Specifically, at refinement level $l+1$, the edge connecting nodes i and $i+1$ is divided and the position of the new edge node, $\boldsymbol{x}_{(i,i+1)}$, is given by

$$\boldsymbol{x}_{(i,i+1)} = \frac{1}{2}\left(\boldsymbol{x}_i^l + \boldsymbol{x}_{i+1}^l\right), \tag{1}$$

where $\boldsymbol{x}_j^k$ denotes the position of node j at level k.

The position of the original nodes is also modified and they are relocated to a linear combination of their position and the position of their neighboring nodes at level l,

$$\boldsymbol{x}_i^{l+1} = \frac{1}{8}\left(\boldsymbol{x}_{i-1}^l + 6\boldsymbol{x}_i^l + \boldsymbol{x}_{i+1}^l\right). \tag{2}$$

As subdivision proceeds, the refined polygons tend to a cubic $\mathcal{C}^2$-continuous curve. We remark that the initial control mesh determines the limiting curve, so all the refined polygons converge to the same curve. Furthermore, it is possible to compute the position in the limit curve of the nodes at any refinement step from the following expression

$$\boldsymbol{x}_i^{l,\infty} = \frac{1}{6}\left(\boldsymbol{x}_{i-1}^l + 4\boldsymbol{x}_i^l + \boldsymbol{x}_{i+1}^l\right), \tag{3}$$

where $\boldsymbol{x}_j^{k,\infty}$ denotes the limiting position of the node j at refinement level k.

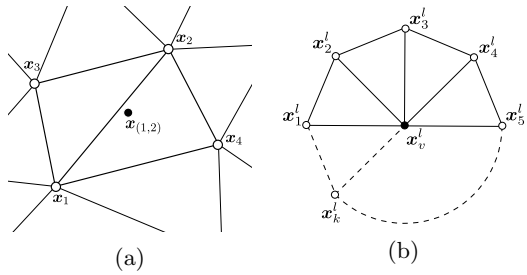


Figure 2: Edge and point configurations for Loop's subdivision surfaces. (a) A new node, $\boldsymbol{x}_{(1,2)}$, is generated on the edge $(1,2)$. (b) Neighbor nodes $\{\boldsymbol{x}_i^l\}$ of a node $\boldsymbol{x}_v^l$.

3.2 Surface Subdivision Scheme

In this work, we subdivide a given triangular surface mesh using Loop's subdivision algorithm [22]. Loop's subdivision scheme consists of three steps: generate a new node for each edge, update the position of the original nodes, and refine the original triangle into four smaller ones. Successive applications of the algorithm generate finer triangular meshes, all of them with the same limit surface determined by the initial control mesh. This surface is $\mathcal{C}^2$-continuous almost everywhere.

Before detailing Loop's subdivision process, we present several definitions related to neighbor elements and nodes. The *neighbor elements* of a node v are the elements incident to v, and the *neighbor nodes* of v are the vertices of these elements. For the case of simplices, there exists an equivalent definition based on edges. The *neighbor edges* of a node v are the edges incident to v, and the *neighbor nodes* of v are the vertices of these edges. We say that a surface node is *regular* if it has six neighbor nodes. Otherwise, we say the node is *irregular*. Around regular nodes, the limit surface is parametrized by a polynomial of degree four and is $\mathcal{C}^2$-continuous, while on irregular nodes it is of class $\mathcal{C}^1$[23].

In the Loop subdivision algorithm, a new node is generated for each edge. Note that in a surface mesh an edge connects two nodes and belongs to two elements. Let $\boldsymbol{x}_1$, ..., $\boldsymbol{x}_4$ be the position of four points in $\mathbb{R}^3$ that define two triangles $(1,2,3)$ and $(1,4,2)$. These triangles share the edge $(1,2)$, as shown in Fig. 2(a). In this edge, a new node, with position denoted by $\boldsymbol{x}_{(1,2)}$, is generated using the expression

$$\boldsymbol{x}_{(1,2)} = \frac{3}{8}\left(\boldsymbol{x}_1 + \boldsymbol{x}_2\right) + \frac{1}{8}\left(\boldsymbol{x}_3 + \boldsymbol{x}_4\right). \tag{4}$$

In addition to generating the new mid-edge nodes, Loop's scheme also modifies the location of the vertices of the initial mesh. At refinement level l, the

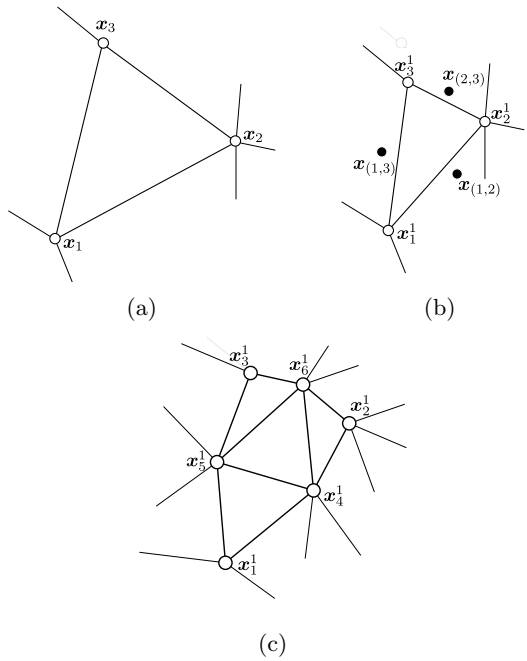


Figure 3: Subdivision process: (a) element of the original mesh; (b) original element with updated vertex nodes (white dots), and new edge nodes (black dots); and (c) subdivided element after one iteration of the process.

position of an existing node, $\boldsymbol{x}_v^l$, with neighbor nodes $\{\boldsymbol{x}_i^l\}_{i=1,\dots,k}$, Fig. 2(b), is modified according to

$$\boldsymbol{x}_v^{l+1} = (1 - k\omega_k)\, \boldsymbol{x}_v^l + \omega_k \sum_{i=1}^{k} \boldsymbol{x}_i^l, \tag{5}$$

where ω_k is defined as

$$\omega_k = \frac{1}{k} \quad \frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4} \cos\left(\frac{2\pi}{k} \right) \right)^2 \Bigg).$$

If we connect and relabel the three new nodes generated for each edge, the subdivided surface mesh is obtained, see Fig. 3. Note that all the nodes generated on the edges of the mesh are regular. This is so since each edge is shared by two triangles, and the new elements are defined by connecting the mid-edge nodes. Therefore, since the percentage of regular nodes increases with each iteration, the discontinuity on the derivatives around irregular nodes can be mitigated applying successively the scheme, if desired.

We remark that the subdivision scheme defines a hierarchy of control meshes, all of them converging to the same limit surface. Moreover, we can compute the limiting location for the nodes at any refinement level. In particular, given a linear mesh, the limit position for the node v at any refinement level l, denoted by $\boldsymbol{x}_v^{l,\infty}$, is

$$\boldsymbol{x}_v^{l,\infty} = (1 - k\chi_k)\, \boldsymbol{x}_v^l + \chi_k \sum_{i=1}^{k} \boldsymbol{x}_i^l, \tag{6}$$

where $\{\boldsymbol{x}_i^l\}_{i=1,\dots,k}$ are the positions of the neighbor nodes of v at level l, and where the weights are computed as

$$\chi_k = \frac{1}{k + \frac{3}{8\omega_k}}.$$

We highlight that there exists an explicit parametrization of the limit surface on the elements in which all the nodes are regular. This parametrization is indeed a piecewise polynomial of degree four, and its explicit expression can be found in [15, 24].

3.3 Non-interpolative to Interpolative

Given a control mesh, the schemes in Sect. 3.1 and Sect. 3.2 for mesh subdivision generate a hierarchy of subdivided meshes all of them tending to the same limit manifold (curve or surface). These schemes do not preserve the position of the initial vertices of the mesh. However, a new control mesh can be computed so that the limit manifold contains the nodes of the initial mesh [15]. That is, new points can be found such that the limit curves and surfaces interpolate the given data points.

Specifically, let φ be the operator that maps the position of the initial mesh nodes, $\boldsymbol{X}_0$, onto their limit position. Then, we compute a new control mesh, with nodes position denoted by $\boldsymbol{X}_C$, such that

$$\varphi\left(\boldsymbol{X}_C\right) = \boldsymbol{X}_0.$$

In the case of the subdivision schemes considered in this work, φ is a linear application with rows given by the coefficients in Eq. (3) or Eq. (6), depending on the point to be updated. This matrix is sparse and the solution of the linear system can be computed using a sparse direct solver. In our Python implementation, we call the sparse direct solver of the SuperLU library [25, 26] through the Python SciPy package [27]. Recall that this operation is performed on the boundary of a tetrahedral mesh. Therefore, the dimension of the linear system is of the order of the number of boundary nodes and not of the order of nodes of the volume mesh.

4. APPROXIMATE SURROGATE GEOMETRY: CURVED SURFACE MESH

In this section, we detail the procedure based on subdivision that we propose in this work to generate a

Algorithm 1 High-order surface mesh.

Input: MarkedMesh $\mathcal{M}$, PolynomialDegree p
Output: MarkedMesh $\mathcal{M}_p$
1: **function** GENERATEHOSURFACEMESH
2: $\mathcal{M}_C \leftarrow$ GENERATENEWCONTROLMESH($\mathcal{M}$)
3: **if** p is 2 **then**
4: $\mathcal{M}' \leftarrow$ SURFACESUBDIVISION($\mathcal{M}_C$, 1)
5: **else if** p is 4 **then**
6: $\mathcal{M}' \leftarrow$ SURFACESUBDIVISION($\mathcal{M}_C$, 2)
7: **end if**
8: $\{\boldsymbol{x}_i^\infty\} \leftarrow$ MAPNODESONTOLIMITPOSITION($\mathcal{M}'$)
9: $\{\boldsymbol{e}_i^p\} \leftarrow$ GENERATEHOTOPOLOGY($\mathcal{M}'$, $\mathcal{M}_C$, p)
10: $\mathcal{M}_p \leftarrow$ SETMESH($\{\boldsymbol{x}_i^\infty\}$, $\{\boldsymbol{e}_i^p\}$)
11: **return** $\mathcal{M}_p$
12: **end function**

curved surface mesh that approximates a smooth surrogate boundary.

In Algorithm 1, we describe the main steps of this process. First, in Line 2, we cast the triangular surface mesh to a new control mesh to ensure that the location of the initial vertices is preserved in the high-order mesh, as detailed in Sect. 3.3. Given the new control mesh, the subdivision method determines a limit manifold that in this work corresponds to the surrogate geometry for the curving procedure. In addition, we exploit the structure of the subdivision surface method to determine the new high-order elements. In particular, four subelements of a linear triangle, once subdivided, determine one element of polynomial degree $p = 2$. Similarly, sixteen linear elements obtained after applying two iterations of the subdivision algorithm to a linear element determine a unique element of polynomial degree $p = 4$. Therefore, in Lines 4 and 6 we call the function `SurfaceSubdivision` to perform the number of subdivision required for the generation of the high-order mesh. The surface subdivision process with feature preservation proposed in this work, `SurfaceSubdivision`, is detailed in Algorithm 2 from Sect. 4.1. Following, the nodes are mapped onto its limit position, Line 8, interpolating the surrogate geometry. Finally, in Line 9, the subdivided mesh is cast to a high-order mesh by reinterpreting the children of each element as a new high-order element. The casting of the subdivision to a high-order mesh and the approximation of the surrogate geometry are detailed in Sect. 4.2.

We highlight that the surrogate geometry for the generation of the high-order mesh is determined by the initial linear mesh and the given geometry features. However, as it will be detailed in Sect. 4.3, prior subdivision steps to the ones performed in Lines 4 and 6 improve the smoothness of the curved high-order mesh that approximates the surrogate. Therefore, if desired, after computing the new control mesh in Line 2 a new finer straight-sided mesh could be generated applying several subdivision steps. From this point, the curving procedure would continue as detailed in Algorithm 1.

In this work, we have favored a subdivision scheme that preserves the location of the vertices of the original linear mesh. To this aim, in Line 2 of Algorithm 1 a new control mesh for the subdivision procedure is computed. If the standard approximative version of the subdivision schemes is preferred, the same procedure applies but the generation of the new control mesh can be omitted.

4.1 Surrogate Geometry: Surface Mesh Subdivision

This work is devoted to curving linear meshes when no geometry is available. Thus, surface mesh subdivision converges to a limit manifold that determines the surrogate geometry for the mesh curving problem. In addition, in the interior of the feature curves it is $\mathcal{C}^2$-continuous, whereas in the interior of the feature surfaces it is $\mathcal{C}^1$-continuous becoming even $\mathcal{C}^2$-continuous in regular regions of the mesh. In this section, we propose a hierarchical subdivision process that allows preserving the sharp features marked in the mesh.

We consider a triangular surface mesh with its entities characterizing the geometry features to preserve. Following Algorithm 2, three main steps are performed. First, in Line 5, we generate the points of the subdivided mesh. Next, in Line 6, the elements of the finer mesh are generated. Finally, the subdivided mesh inherits the marks from the coarse mesh, Line 8. These steps are repeated for all the specified subdivision iterations.

The generation of the points of the hierarchical subdivision scheme is performed looping through the unique edges and nodes of the mesh, as detailed in Algorithm 3. Since each node of the mesh is given a unique identifier, an edge is uniquely defined by the nodes it connects. Consider an edge $e = (i, j)$. If the edge belongs to a feature curve, the position of the new edge node, $\boldsymbol{x}_e^{\text{new}}$, is at the midpoint of the edge points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, see Eq. (1), Line 5. In contrast, if e belongs to a feature surface, in order to generate the new edge point, we need to evaluate Eq. (4) and thus, we need access to the nodes of the elements sharing the edge e but opposite to it, Line 9. We denote the position of these nodes by $\boldsymbol{x}_{t_1}$ and $\boldsymbol{x}_{t_2}$. Therefore, the position of the new edge node is given accordingly to Line 10.

In order to update the position of the original nodes, we loop through the nodes of the mesh. Consider a node v. If v is a feature vertex, its position does not change, so $\boldsymbol{x}_v^{\text{new}} = \boldsymbol{x}_v$, see Line 16. To update the position of a point that belongs to a curve by the evaluation of Eq. (2), we need access to the two neighbor

Algorithm 2 Surface subdivision.

Input: MarkedMesh $\mathcal{M}$, NumberOfSubdivisions N
Output: MarkedMesh $\mathcal{M}'$

1: **function** SurfaceSubdivision
2: $\mathcal{M}' \leftarrow \mathcal{M}$
3: **for** $j \in \{1, \dots, N\}$ **do**
4: $\mathcal{M}_0 \leftarrow \mathcal{M}'$
5: $\{\boldsymbol{x}_i\} \leftarrow$GeneratePointsSubdivision$(\mathcal{M}_0)$
6: $\{\boldsymbol{e}_i\} \leftarrow$GenerateFinerTopology$(\mathcal{M}_0)$
7: $\mathcal{M}' \leftarrow$SetMesh$(\{\boldsymbol{x_i}\}, \{\boldsymbol{e_i}\})$
8: $\mathcal{M}' \leftarrow$InheritMarks$(\mathcal{M}', \mathcal{M}_0)$
9: **end for**
10: **return** $\mathcal{M}'$
11: **end function**

nodes in the curve. We denote the position of these two nodes by $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, see Line 19. Now, the updated position of node v is given accordingly to Line 20. Analogously, we update the position of a node that belongs to a surface. In Line 24, we get the position of its neighbor nodes $\{\boldsymbol{x}_i\}_{1\dots k}$, and the position of node v is updated using Eq. (5) in Line 25.

We highlight that Algorithm 3 combines two types of subdivision algorithms. Therefore, the limit manifold, which determines our surrogate geometry, inherits different smoothness guarantees from the original subdivision algorithms that are discussed in Sect. 4.3.

4.2 Cast Subdivision to Curved High-order Surface Mesh and Interpolate Surrogate Geometry

The hierarchical subdivision procedure presented in Sect. 4.1 generates a new point for each edge of the mesh and four elements for each refined linear triangle. Therefore, in the curving process in Algorithm 1, we exploit this structure and reinterpret the topology from the children of each original element into a new high-order element. An element of polynomial degree two is obtained after one application of the subdivision algorithm to a linear element. Two iterations of the subdivision scheme lead to a quartic element. Therefore, high-order element generation based on subdivision defines elements of polynomial degree $p = 2^k$, where k is the number of iterations of the subdivision schemes performed. The limit surface is parametrized by a polynomial of degree four around regular nodes, therefore, we focus on quartic meshes since then the limit surface and its regularity is exactly captured around regular nodes for this polynomial degree.

The main steps of the reinterpretation of the subdivided elements as a unique high-order element are stated in Algorithm 4. For each element of the initial linear mesh, $\boldsymbol{e}_i$, we obtain the elements of the finer mesh generated from the subdivision of $\boldsymbol{e}_i$, Line 3.

Algorithm 3 Generate points of surface subdivision.

Input: MarkedMesh $\mathcal{M}$
Output: Points $\{\boldsymbol{x_i}^{\text{new}}\}$

1: **function** GeneratePointsSubdivision
2: **for** each edge $e = (i, j)$ of the mesh **do**
3: **if** e belongs to a curve **then**
4: ▷ New edge point using Eq. (1)
5: $\boldsymbol{x}_e^{\text{new}} \leftarrow \frac{1}{2}(\boldsymbol{x}_i + \boldsymbol{x}_j)$
6: **else**
7: ▷ e belongs to a surface
8: ▷ New edge point using Eq. (4)
9: $\boldsymbol{x}_{t_1}, \boldsymbol{x}_{t_2} \leftarrow$ GetOppositeNodes(e)
10: $\boldsymbol{x}_e^{\text{new}} \leftarrow \frac{3}{8}(\boldsymbol{x}_i + \boldsymbol{x}_j) + \frac{1}{8}(\boldsymbol{x}_{t_1} + \boldsymbol{x}_{t_2})$
11: **end if**
12: **end for**
13: **for** each node v of the mesh **do**
14: **if** v is a vertex point **then**
15: ▷ Point remains fixed
16: $\boldsymbol{x}_v^{\text{new}} \leftarrow \boldsymbol{x}_v$
17: **else if** v belong to a curve **then**
18: ▷ Update position using Eq. (2)
19: $\boldsymbol{x}_i, \boldsymbol{x}_j \leftarrow$ GetNeighborsInCurve(v)
20: $\boldsymbol{x}_v^{\text{new}} \leftarrow \frac{1}{8}(\boldsymbol{x}_i + \boldsymbol{x}_j) + \frac{6}{8}\boldsymbol{x}_v$
21: **else**
22: ▷ v belongs to a surface
23: ▷ Update position using Eq. (5)
24: $\{\boldsymbol{x}_i\}_{i=1\dots k} \leftarrow$ GetNeighborsInSurface(v)
25: $\boldsymbol{x}_v^{\text{new}} \leftarrow (1 - k\omega_k)\,\boldsymbol{x}_v + \omega_k \sum_{i=1}^{k} \boldsymbol{x}_i$
26: **end if**
27: **end for**
28: **return** $\{\boldsymbol{x_i}^{\text{new}}\}$
29: **end function**

When used to generate meshes of polynomial degree two, this set contains four linear elements, whereas when used to generate a new mesh of degree four, this set contains sixteen elements. Finally, in Line 4, this set of linear elements is reinterpreted as a unique high-order element.

Specifically, given an element of the linear mesh, Fig. 4(a), one iteration of the hierarchical subdivision scheme generates three edge nodes and modifies the position of the vertices, Fig. 4(b). We denote by 4, 5 and 6 the edge nodes created from the subdivision of edge $(1, 2)$, $(1, 3)$ and $(2, 3)$, respectively. Now, the nodes are mapped onto its limiting position using Eq. (3) or Eq. (6). Finally, instead of connecting the edge nodes and subdividing the element as in Fig. 4(c), a curved element of polynomial degree two is defined by considering the new generated nodes as the mid-edge nodes that are required to define an element of polynomial degree two, see Fig. 4(d).

This process is repeated for all the elements, obtaining a curved surface mesh of polynomial degree two that approximates the surrogate geometry with third order

Algorithm 4 Reinterpret finer mesh as a high-order mesh.

Input: FineMesh $\mathcal{M}'$, CoarseMesh $\mathcal{M}$, PolynomialDegree p
Output: Elements $\{\boldsymbol{e}_i^p\}$

```
1: function GenerateHOTopology
2:   for each element e_i of M do
3:     {e_{i,j}} ←GetChildren(e_i, M')
4:     e_i^p ←Reinterpret({e_{i,j}}, p)
5:   end for
6:   return {e_i^p}
7: end function
```

accuracy, see Sect. 4.3 for more details on the regularity of the high-order mesh. This high-order mesh has the same number of nodes as the subdivided mesh but it has the same number of elements as the original linear mesh.

For quartic polynomial degree, $p = 4$, each element is defined by fifteen nodes. The procedure to generate the curved mesh of degree four from a given linear mesh is similar to the quadratic case. Applying twice the subdivision process fifteen new nodes are obtained, see Fig. 5(c). Next, these nodes are mapped onto its limiting position using Eq. (3) or Eq. (6). Finally, we reinterpret the topology of the sixteen elements from the subdivision into a unique element of polynomial degree four, see Fig. 5(d). Specifically, nodes 7, 8, 9, 10, 11, 12, 13, 14 and 15 are the edge nodes created from the subdivision of edges $(1,4)$, $(1,5)$, $(2,4)$, $(2,6)$, $(3,5)$, $(3,6)$, $(4,5)$, $(4,6)$ and $(5,6)$, respectively.

This process is repeated for all the elements, obtaining a curved surface mesh of polynomial degree four with the same number of nodes as after applying two times the subdivision scheme to the original linear mesh, although the number of elements is the same than in the linear mesh.

4.3 Smoothness of the Surrogate Geometry and Curved Surface Mesh

In this section, we analyze the smoothness of the surrogate geometry and of the high-order meshes obtained using the approach presented in Sect. 4.2.

The surrogate boundary is composed of the union of the limit curves and the limit surfaces determined by the control mesh. On the one hand, the curve subdivision scheme, see Sect. 3.1, ensures that in the inner edges of the feature curves (not adjacent to a feature vertex) the limit curve is cubic and $\mathcal{C}^2$-continuous. On the contrary, curve edges of the control mesh that are incident to a vertex point determine the region where the limit curve is of class $\mathcal{C}^0$.

On the other hand, Loop's subdivision scheme, see

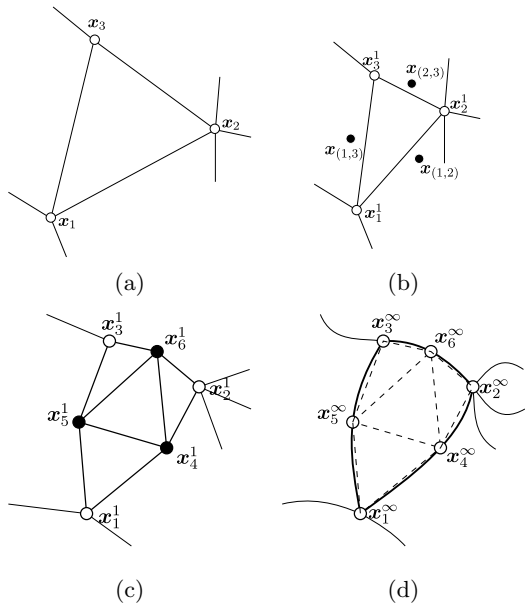


Figure 4: Generation of an element of polynomial degree two from a linear one. (a) Straight-sided element. (b) Original element once new nodes (black dots) have been generated and the position of the old ones (white dots) have been modified by the hierarchical subdivision scheme. (c) Subdivision of the linear element. (d) Curved element of polynomial degree two, displaying with dashed lines the four elements from the subdivision scheme.

Sect. 3.2, ensures that in the inner triangles of the feature surfaces (not adjacent to a feature curve or vertex), the limit surface is $\mathcal{C}^1$-continuous, and $\mathcal{C}^2$-continuous almost everywhere. In particular, in the inner triangles, this surface is of class $\mathcal{C}^2$ everywhere except at the position of the irregular vertices of the initial control mesh. At these points, the surface is strictly $\mathcal{C}^1$-continuous. The surface triangles that are adjacent to a vertex point or a curve determine the region where the limit surface is $\mathcal{C}^0$-continuous. Moreover, in those triangles where the limit surface is of class $\mathcal{C}^2$, it can be parametrized by quartic polynomials. We remark that this discontinuity in the derivatives is confined. Fig. 6(a) shows a regular mesh featuring inner triangles in dark gray and triangles adjacent to a feature curve (bold) in light gray. In this configuration, the limit surface is of class $\mathcal{C}^2$ only on the dark gray region. The limit curve determined by the bold edges is also of class $\mathcal{C}^2$.

Regarding the smoothness of the obtained high-order meshes, curve (surface) meshes of polynomial degree $p = 2$ approximate the cubic (quartic) limit curve (surface) with third-order accuracy. That is, meshes of polynomial degree 2 are strictly $\mathcal{C}^0$-continuous, and no

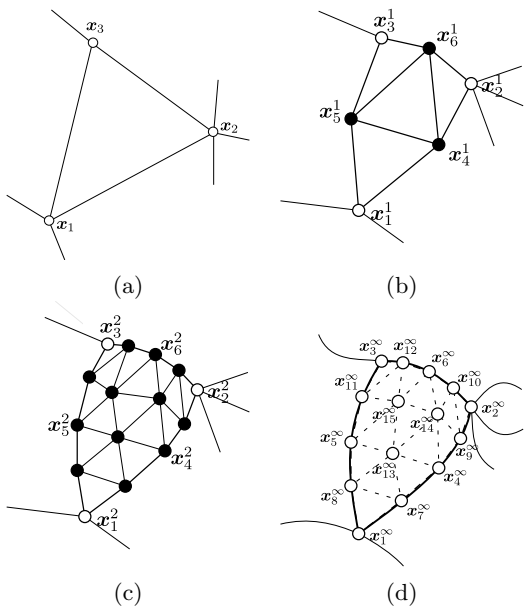


Figure 5: Generation of an element of polynomial degree four from a linear one. (a) Straight-sided element. (b) First subdivision step. (c) Second subdivision step. (d) Curved element of polynomial degree four, displaying with dashed lines the sixteen elements from the subdivision scheme.

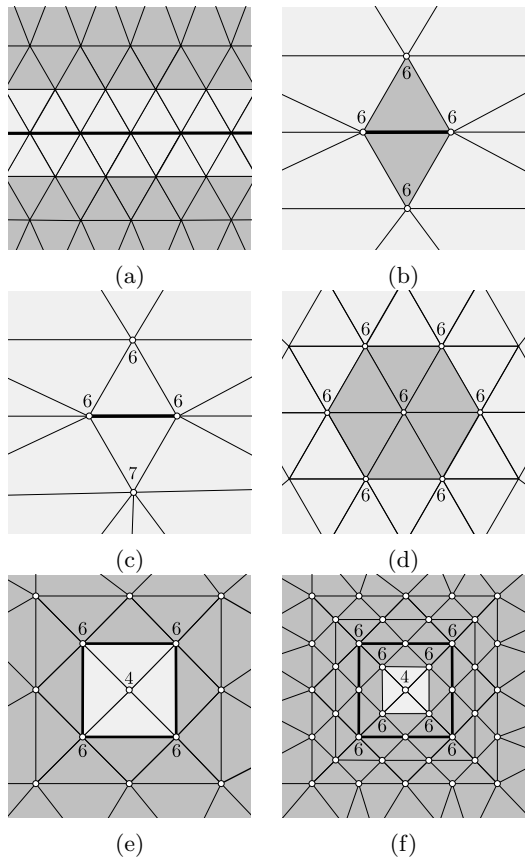


Figure 6: Dark gray regions indicate $\mathcal{C}^2$ smoothness, while light gray color indicate regions where $\mathcal{C}^2$-continuity cannot be ensured. (a) Regular mesh around a feature curve (bold). (b) Mesh with a regular edge (bold). (c) Mesh with an irregular edge (bold). (d) Regular patch around a regular vertex. (e) Irregular patch (light gray) around an irregular vertex. (f) Mitigation of the irregular patch after subdividing the mesh.

guarantees of the $\mathcal{C}^2$-continuity are given by the proposed subdivision-based curving method. However, some prior subdivisions can be applied to the initial linear mesh using the subdivision method presented in Sect. 4.1. Next, this refined mesh can be curved with the subdivision-based curving method proposed in Algorithm 1. This new finer high-order mesh determines a better approximation of the surrogate geometry, and consequently, its smoothness is also improved.

Following, we analyze the smoothness of the meshes of polynomial degree four. First, the inner edges of the feature curves of the high-order mesh exactly capture the $\mathcal{C}^2$-continuous limit curve. This is so since the limit curve is parametrized by a third degree polynomial, while the elements are described by shape functions of degree four. Similarly to the curve case, the nodes of the surface mesh also interpolate the limit surface. However, the surface mesh does not inherit the smoothness of the limit surface straight-forwardly as in the curve case. This is so since the limit surface is parameterized element-wise, but the parameterization is of degree four only in a regular element, that is, in an element where its three vertices have six neighbors.

In the interior of an element, the mesh is of class $\mathcal{C}^\infty$. Therefore, the smoothness of the surface mesh has to be analyzed along the edges (interfaces between two inner surface elements) and vertices (interfaces between more than two inner surface elements). We first analyze the case between two elements that share an edge. We say an edge is a *regular edge* if all the vertices of the two triangles that share such edge are regular, Fig. 6(b), *i.e.* if the vertices have six neighbors. In this case, the two elements of degree four that share the edge interpolate exactly the quartic limit surface. Hence both elements are exactly equal to the limit surface and, since the limit surface is $\mathcal{C}^2$-continuous, the interface (edge) between the two elements also is. In general, no guarantee of the continuity of the derivatives can be deduced along an edge that is not regular, see Fig. 6(c).

Following, we analyze the smoothness of the mesh

around the vertices of the inner surface elements. Given a regular vertex (with six neighbors), if all the edges incident to it are regular then the surface mesh is of class $\mathcal{C}^2$ around such vertex. In particular, if all the edges incident to a regular vertex are regular, then all its neighbor vertices are also regular, as observed in Fig. 6(d). In such regions, colored in Fig. 6(d) in dark gray, the surface mesh captures exactly the limit surface and inherits all its features. The presence of an irregular vertex, as illustrated in Fig. 6(e), implies the surface mesh to approximate the limit surface, rather than exactly capturing it. Therefore, on the one hand, around regular patches, we are able to interpolate and exactly capture the limit surface and obtain a $\mathcal{C}^2$-continuous surface mesh. On the other hand, around irregular patches, the limit surface is interpolated and approximated with fifth order accuracy but not matched exactly.

In order to improve the smoothness of the high-order mesh and mitigate this issue, the structure induced in the mesh by Loop's subdivision process can be exploited. All the new vertices generated by Loop's subdivision are regular. Therefore, the linear mesh, Fig. 6(e), can be subdivided before generating the mesh of degree four with the presented procedure. As observed in Fig. 6(f) in contrast to Fig. 6(e), the light gray irregular region where the limit surface (and subsequently its smoothness) is not exactly captured is reduced. Exploiting prior refinements of the linear mesh, the regions where the high-order surface mesh is not $\mathcal{C}^2$-continuous can be successively reduced.

5. CURVED VOLUME MESH APPROXIMATING A SURROGATE BOUNDARY

In this section, we detail how a linear tetrahedral mesh with marked boundary entities is curved while preserving the sharp features. In Sect. 5.1, we detail the sharp-to-smooth modeling of the geometry features and the replacement of the straight-edged boundary of the linear mesh by the curved boundary mesh. In Sect. 5.2, the curvature on the boundary is accommodated to the interior using a blending technique. This procedure leads to a high-order tetrahedral mesh where its boundary approximates a surrogate geometry composed of feature surfaces with an interior that is $\mathcal{C}^1$-continuous and $\mathcal{C}^2$-continuous almost everywhere. In addition, the vertices of the high-order mesh are kept in the same position than in the initial linear mesh.

Algorithm 5 Curve volume mesh recasting features.

Input: MarkedMesh $\mathcal{M}$, PolynomialDegree p, FeaturesToRecast R
Output: MarkedMesh $\mathcal{M}_p$
1: **function** CurveMarkedVolumeMesh
2: $\mathcal{M} \leftarrow$RecastFeatures($\mathcal{M}$, R)
3: $\mathcal{M}_p \leftarrow$ GenerateHOVolumeMesh($\mathcal{M}$, p)
4: **return** $\mathcal{M}_p$
5: **end function**

Algorithm 6 Recast geometry features.

Input: MarkedMesh $\mathcal{M}$, FeaturesToRecast R
Output: MarkedMesh $\mathcal{M}$
1: **function** RecastFeatures
2: **for** each feature f in R **do**
3: $\mathcal{M} \leftarrow$ RemoveFeatureFromList($\mathcal{M}$, f)
4: $\mathcal{M} \leftarrow$ MergeIncidentFeatures($\mathcal{M}$, f)
5: **end for**
6: **return** $\mathcal{M}$
7: **end function**

5.1 Substitute the Boundary of the Volume Mesh

In this section, we detail the subdivision-based curving of a high-order mesh. This process is presented in Algorithm 5. First, in Line 2, we recast the desired feature entities. Then, in Line 3, we generate a high-order volume mesh preserving the sharp features provided by the new model once the original entities have been recast. These processes, denoted as `RecastFeatures` and `GenerateHOVolumeMesh`, are next detailed in Algorithms 6 and 7.

The first step to curve the volume mesh is to recast, if necessary, the geometry features present in the original model. Recall that vertices, curves, and surfaces are characterized by a unique identifier. Thus, in order to recast a sharp feature, it is enough to provide its identifier. That is, in Algorithm 6, the variable `FeaturesToRecast` contains a list of the identifiers of the feature vertices and curves to be recast. Specifically, the recasting of a feature is composed of two steps: remove the feature from the list of features to preserve, Line 3; and merge the features incident to such feature, Line 4. Since each feature is described by a unique identifier, the process of merging the incident features reduces to assigning the same identifier to these features.

Next, the curving method based on hierarchical subdivision and blending is performed. The generation of a high-order volume mesh is described in Algorithm 7. First, given a linear tetrahedral mesh with the recast features, Fig. 7(a), we extract its boundary, Line 2. The boundary is a surface mesh that inherits the geometry features of the volume mesh. Next, in Line

Algorithm 7 High-order volume mesh.

Input: MarkedMesh $\mathcal{M}$, PolynomialDegree p
Output: MarkedMesh $\mathcal{M}_p$

```
1: function GenerateHOVolumeMesh
2:   ∂M ← ExtractBoundary(M)
3:   ∂M_p ← GenerateHOSurfaceMesh(∂M, p)
4:   M_p ← IncreasePolynomialDegree(M)
5:   M_p ← ReplaceBoundary(M_p, ∂M_p)
6:   M_p ← AccommodateCurvature(M_p, M)
7:   if M_p has invalid elements then
8:     M_p ← OptimizeMesh(M_p)
9:   end if
10:  return M_p
11: end function
```

3, we call the function `GenerateHOSurfaceMesh` described in Algorithm 1 to generate a surface mesh of polynomial degree p preserving the sharp features. Third, we generate a straight-edged high-order volume mesh, Line 4, illustrated in Fig. 7(b). Following, in Line 5, we replace the boundary of the straight-edged mesh by the curved surface mesh, see Fig. 7(c). Then, in Line 6, the curvature of the surface is accommodated to the elements adjacent to boundary, using a blending technique to be described in Sect. 5.2. Finally, if the mesh contains tangled elements, it is optimized using [8, 9], see Line 8.

The methodology proposed in this work can also be used to generate, given an initial linear mesh, finer linear meshes that successively improve the approximation of the surrogate geometry. To do so, the generated high-order mesh can be reinterpreted as a linear mesh by the decomposition of each high-order element into linear elements. Specifically, the reference high-order element is decomposed into several structured linear elements determined by the high-order nodes. In particular, each quadratic element is decomposed into eight linear tetrahedra, while an element of polynomial degree four leads to sixty-four linear elements. If the linear mesh contains tangled elements, the optimization procedure described in [8] is applied to ensure a valid mesh.

5.2 Accommodate the Curvature of the Boundary

In Algorithm 7, after replacing the curved boundary in the straight-sided high-order mesh, Line 5, the obtained high-order mesh may contain low-quality or tangled elements, see Fig. 8(a) and 8(b). Since the curvature information is provided by the surface mesh, only boundary tetrahedra are affected and therefore, the number of invalid elements is small compared with the scale of the generated meshes. Thus, similarly to [19], as an attempt to improve the mesh quality

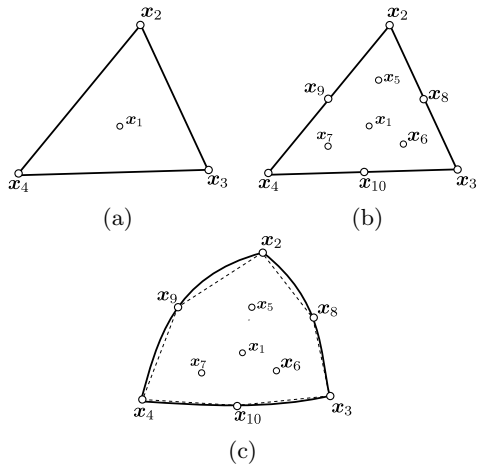


Figure 7: Curving of the boundary for a boundary element of polynomial degree $p = 2$. (a) Linear physical element, where the face (2 3 4) belongs to the boundary. (b) Straight-edged physical element of polynomial degree two. (c) Curved boundary element of polynomial degree two, displaying with dashed lines the four elements from the subdivision scheme applied to the boundary.

in a fast and explicit manner, in Line 6 in function `AccommodateCurvature`, we use Transfinite Interpolation (TFI) [28] to accommodate the curved surface to those entities of the boundary elements not present in the surface mesh. Specifically, given a boundary element, we use transfinite interpolation hierarchically on its entities to accommodate the curving of the boundary. That is, first we relocate the nodes on edges, then nodes on faces, and finally, nodes in the interior of tetrahedra.

First, consider an edge of a high-order boundary element with one of its endpoints on the curved surface. The new location of the edge nodes is given by the linear isoparametric mapping between the one-dimensional reference domain and the physical edge, denoted as ϕ^1. Specifically, the new position of the kth node of the edge, $\boldsymbol{x}_k$, for $k = 0, \ldots, p$, is determined as

$$\boldsymbol{x}_k = \phi^1(\boldsymbol{\xi}_k) = \sum_{l=0}^{1} \boldsymbol{x}_{v_l} N_{v_l}(\boldsymbol{\xi}_k)$$

where $\boldsymbol{\xi}_k$ is the position of the kth node of the reference domain, $\boldsymbol{x}_{v_l}$ is the position of the lth endpoint of the edge, and N_{v_l} is the linear nodal shape function of the interval associated to the lth endpoint. In Fig. 8(c), we illustrate the relocation of the nodes of the edges in a triangle when one of its edges (in bold) is on the boundary.

Now, we are interested in relocating the nodes on the interior of a face of a boundary tetrahedron. On the

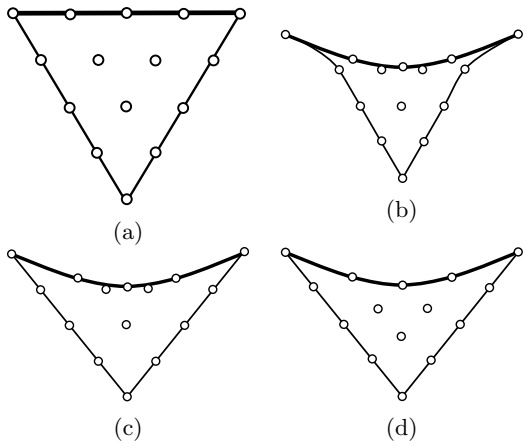


Figure 8: Accommodating the curvature to a triangular element of polynomial degree $p = 4$. (a) Straight-edged triangle with a boundary edge (in bold). (b) Triangle with curved boundary. (c) Transfinite interpolation applied to the edges, (d) and to the face.

one hand, if such face belongs to the boundary, its nodes have been already relocated by the subdivision scheme, see bold edge in Fig. 8(c). On the other hand, if such face does not belong to the boundary, its edge nodes have been modified using the transfinite interpolation for edges explained above, see non-bold edges in Fig. 8(c). For the latter, we apply the transfinite interpolation for faces, that is, we accommodate the deformation of the curving of the edges to the interior of the triangular faces.

We denote by f_i the edge of the triangle opposed to vertex i, $i = 1, 2, 3$. Let us denote as $\boldsymbol{x}_{f_i,k}$ the coordinates of the kth node of the edge f_i in the physical element, $k = 0, \ldots, p$. These nodes are fixed now, since as previously detailed their location has already been computed. Therefore, each curved or relocated edge can be parametrized using the restriction to such edge of the two-dimensional isoparametric mapping of degree p, ϕ^p, as:

$$\phi_{f_i}(\boldsymbol{\xi}) := \phi^p|_{f_i}(\boldsymbol{\xi}) = \sum_{k=0}^{p} \boldsymbol{x}_{f_i,k} N_{f_i,k}(\boldsymbol{\xi}) \qquad (7)$$

where $N_{f_i,k}(\xi)$ is the high-order nodal shape function of the triangle associated to the kth node of the edge f_i. In particular, the boundary of the triangle is fixed and parameterized by the three mappings $\{\phi_{f_i}\}_{i=1,2,3}$.

Consider a point $\boldsymbol{x}$ in the physical triangle, to which we want to compute its displaced position in terms of the location of the boundary edge nodes. Denote by $\boldsymbol{\xi}$ the position of the point in the reference triangle expressed in cartesian coordinates such that $\phi^p(\boldsymbol{\xi}) = \boldsymbol{x}$. Now, denote by $\boldsymbol{\lambda}$ the same point in the reference domain expressed in barycentric coordinates $(\lambda_1, \lambda_2, \lambda_3)$, $\sum_{i=1}^{3} \lambda_i = 1$. Following [28], we compute the projection of the point to the edges. A point on an edge can be parametrized as a function of the barycentric coordinates of the two vertices of the triangle defining the edge. Therefore, two different projections ($\boldsymbol{\lambda}^j_{f_*}$ for $j \in f_*$) are computed for each one of the three edges of the triangle (rows, $\boldsymbol{\lambda}^*_{f_i}$ for $i = 1, 2, 3$):

$$f_1 = (2,3): \begin{cases} \boldsymbol{\lambda}^2_{f_1} = (0, 1-\lambda_3, \lambda_3), \\ \boldsymbol{\lambda}^3_{f_1} = (0, \lambda_2, 1-\lambda_2), \end{cases}$$

$$f_2 = (1,3): \begin{cases} \boldsymbol{\lambda}^1_{f_2} = (1-\lambda_3, 0, \lambda_3), \\ \boldsymbol{\lambda}^3_{f_2} = (\lambda_1, 0, 1-\lambda_1), \end{cases}$$

$$f_3 = (1,2): \begin{cases} \boldsymbol{\lambda}^1_{f_3} = (1-\lambda_2, \lambda_2, 0), \\ \boldsymbol{\lambda}^2_{f_3} = (\lambda_1, 1-\lambda_1, 0). \end{cases}$$

Note that $\boldsymbol{\lambda}^j_{f_i}$ belongs to edge f_i and has the jth component expressed as a function of the others. Then, we express these six projections of the point at the edges, computed in barycentric coordinates, back in the reference coordinates $\boldsymbol{\xi}$. As previously remarked, we denote the change from barycentric coordinates of a point $\boldsymbol{\lambda}^j_{f_i}$ to reference coordinates as $\boldsymbol{\xi}^j_{f_i}$, for $i = 1, 2, 3$. Since these points are on the edges of the triangle, they can be mapped onto the physical triangle through the mappings ϕ_{f_i} of the edges, $i = 1, 2, 3$, presented in Eq. (7) as:

$$\boldsymbol{x}^j_{f_i} := \phi_{f_i}\left(\boldsymbol{\xi}^j_{f_i}\right)$$

We highlight that given a point $\boldsymbol{x}$, $\boldsymbol{x}^j_{f_i}$ corresponds to the coordinates on the physical element of the projection j of the point to the edge f_i, $i = 1, 2, 3$.

Finally, the new position of point $\boldsymbol{x}$ in the physical triangle, denoted as $\hat{\boldsymbol{x}}$, is given in [28] as:

$$\begin{aligned} \hat{\boldsymbol{x}} = {} & \lambda_1\left(\boldsymbol{x}^1_{f_2} + \boldsymbol{x}^1_{f_3} - \boldsymbol{x}_{v_1}\right) + \lambda_2\left(\boldsymbol{x}^2_{f_1} + \boldsymbol{x}^2_{f_3} - \boldsymbol{x}_{v_2}\right) \\ & + \lambda_3\left(\boldsymbol{x}^3_{f_1} + \boldsymbol{x}^3_{f_2} - \boldsymbol{x}_{v_3}\right) \end{aligned}$$

where $\boldsymbol{x}_{v_j}$ is the position of the jth vertex of the physical triangle. We highlight that the transfinite interpolation for triangles can be expressed as a function of the isoparametric mapping of the edges and the location of the vertices of the triangle.

In order to relocate the nodes in the interior of the high-order physical faces, the steps detailed above are applied to the nodes in the interior of the high-order reference triangle, see Fig. 8(d). This procedure is repeated for all the faces with a boundary node or edge.

Lastly, we follow an analogous approach to modify the position of the nodes in the interior of the boundary tetrahedra. The boundary of a tetrahedron is composed of four faces and six edges. These faces and edges have already been curved with the procedures

detailed above. Therefore, we relocate the interior nodes according to the curved boundary already accommodated to the edges and faces. Similarly to the triangle case, the transfinite interpolation for tetrahedra can be expressed as a function of the isoparametric mapping of the edges, the isoparametric mapping of the faces, and the location of the vertices of the tetrahedron.

Denote by T the set of vertices of a tetrahedron. We define the entity $f_{i_1,\dots,i_k}$ as the entity of dimension $d-k$, $d=3$, with vertices given by the nodes $T \setminus \{i_1,\dots,i_k\}$. Note that the face opposed to node i is denoted by f_i, and the edge shared by the faces f_i and f_k is f_{ik}. Given the three-dimensional isoparametric mapping of degree p, ϕ^p, analogously to Eq. (7), we denote the restriction to the face f_i as ϕ_{f_i}, and the restriction to the edge f_{ik} as $\phi_{f_{ik}}$.

Similarly to the two-dimensional case, consider a point $\boldsymbol{x}$ in the physical tetrahedron, and denote by $\boldsymbol{\xi}$ and $\boldsymbol{\lambda}$ its preimage in the reference tetrahedron expressed in cartesian and barycentric coordinates, respectively. Now, denote by $\boldsymbol{\lambda}^j_{f_i}$ the projection of the point to the face f_i that has the jth component expressed as a function of the others. $\boldsymbol{\lambda}^j_{f_{ik}}$ denotes the projection of the point to the edge f_{ik} that has the jth component expressed as a function of the others. These projections in the reference domain expressed in cartesian coordinates are denoted by $\boldsymbol{\xi}^j_{f_i}$ and $\boldsymbol{\xi}^j_{f_{ik}}$, respectively.

Since these points are on the faces and edges of the tetrahedron, they can be mapped onto the physical element through the mappings ϕ_{f_i} on the faces and $\phi_{f_{ik}}$ on the edges as:

$$\boldsymbol{x}^j_{f_i} := \phi_{f_i}\left(\boldsymbol{\xi}^j_{f_i}\right), \quad \boldsymbol{x}^j_{f_{ik}} := \phi_{f_{ik}}\left(\boldsymbol{\xi}^j_{f_{ik}}\right)$$

Finally, the new position of point $\boldsymbol{x}$ in the physical triangle, denoted as $\hat{\boldsymbol{x}}$, is given in [28] as:

$$\begin{aligned}\hat{\boldsymbol{x}} = {} & \lambda_1\left(\boldsymbol{x}^1_{f_2}+\boldsymbol{x}^1_{f_3}+\boldsymbol{x}^1_{f_4}-\boldsymbol{x}^1_{f_{23}}-\boldsymbol{x}^1_{f_{24}}-\boldsymbol{x}^1_{f_{34}}+\boldsymbol{x}_{v_1}\right)\\ & +\lambda_2\left(\boldsymbol{x}^2_{f_1}+\boldsymbol{x}^2_{f_3}+\boldsymbol{x}^2_{f_4}-\boldsymbol{x}^2_{f_{13}}-\boldsymbol{x}^2_{f_{14}}-\boldsymbol{x}^2_{f_{34}}+\boldsymbol{x}_{v_2}\right)\\ & +\lambda_3\left(\boldsymbol{x}^3_{f_1}+\boldsymbol{x}^3_{f_2}+\boldsymbol{x}^3_{f_4}-\boldsymbol{x}^3_{f_{12}}-\boldsymbol{x}^3_{f_{14}}-\boldsymbol{x}^3_{f_{24}}+\boldsymbol{x}_{v_3}\right)\\ & +\lambda_4\left(\boldsymbol{x}^4_{f_1}+\boldsymbol{x}^4_{f_2}+\boldsymbol{x}^4_{f_3}-\boldsymbol{x}^4_{f_{12}}-\boldsymbol{x}^4_{f_{13}}-\boldsymbol{x}^4_{f_{23}}+\boldsymbol{x}_{v_4}\right)\end{aligned}$$

We remark that this method does not guarantee to repair the invalid elements, neither ensures an increase of the element quality. However, it is an explicit and fast formulation, which in practice represents a good initial condition for mesh curving methods when no geometry is available [9, 29, 12, 30, 4]. In all the tested applications, see Sect. 6, the procedure improves significantly the quality of the meshes. Once the TFI-based relocation process is finalized, if low quality or inverted elements are present, we perform the non-linear quality optimization procedure presented in [9].

	Min Q	# inv	Time
Boundary	0.93	0	283 s
Volume (no TFI)	0	169	961 s
Volume (TFI)	0.80	0	367 s
			1611 s

Table 1: Quality statistics of a mesh of polynomial degree $p=4$ for Sierra del Escudo (Spain).

6. RESULTS

In this section, we present several examples to illustrate the main features of the methods presented in this work. As a proof of concept, the proposed algorithms have been developed in Anaconda Python [31]. The prototyping code is sequential (one execution thread) and non-vectorized. All the examples have been run on a MacBook Pro (with one dual-core Intel Core i5 CPU, a clock frequency of 2.3 GHz, and a total memory of 16 GBytes).

Although not specified in the previous sections, if desired, we perform a straight subdivision in straight curves and planar surfaces. The straight subdivision scheme does not modify the position of a node and generates the new mid-edge nodes at the midpoint of the edge to subdivide.

In all the examples, we validate both the high-order boundary and volume meshes using the Jacobian-based distortion measure proposed in [32, 8]. In particular, the quality of a high-order element is computed with respect to its straight-sided original element in the linear mesh.

6.1 Regular Mesh: Matching the $\mathcal{C}^2$ Surrogate Topography

In this example, we illustrate the features of our method with a linear mesh that discretizes a real topography. The original data is provided as a level curve map and thus, a CAD model is not available.

We consider a tetrahedral mesh composed of a discretization of a topography that defines the bottom surface and a planar top surface located at the desired height. The tetrahedral mesh is regular and is generated using the mesher presented in [1, 2]. Thus, all the nodes of the surface mesh are regular. From the linear mesh, we generate a curved high-order mesh of polynomial degree four using the procedure detailed in Sect. 5. Since the surface nodes are regular, the high-order topography surface mesh is $\mathcal{C}^2$-continuous.

In Fig. 9(a), we show the initial linear mesh composed of $4.8\cdot10^5$ nodes and $2.6\cdot10^6$ tetrahedra. The high-order boundary mesh is composed of $1.1\cdot10^6$ nodes and $1.4\cdot10^5$ triangles, does not contain tangled ele-

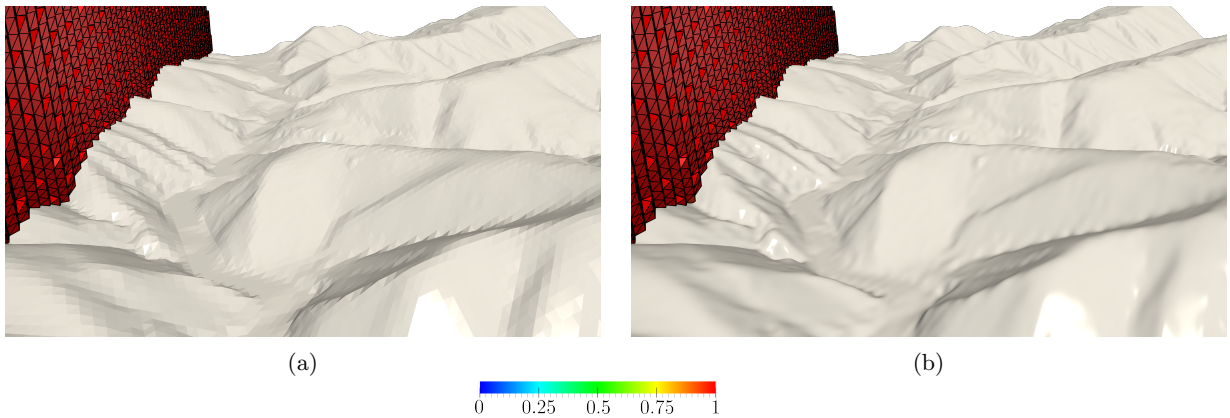


Figure 9: Curving of a tetrahedral mesh of Sierra del Escudo (Spain). Elements of the volume meshes are colored with their elemental quality. (a) Linear mesh. (b) Curved mesh of polynomial degree $p = 4$ with no invalid elements.

	Min Q	# inv	Time
Boundary	0.56	0	191 s
Volume (no TFI)	0	358	611 s
Volume (TFI)	0	24	237 s
Volume (TFI + Optimization)	0.70	0	60 s
			1099 s

Table 2: Quality statistics of a mesh of polynomial degree $p = 4$ for a Falcon aircraft.

ments, and is generated in 283 seconds, see Table 1. The curved high-order volume mesh is generated in 961 seconds and contains 169 inverted elements. The process of accommodating the curvature of the boundary detailed in Sect. 5.2 is performed to $4.1 \cdot 10^5$ elements abutting the boundary. This blending takes 367 seconds and untangles all the invalid elements, attaining a minimum quality of 0.8. Finally, in Fig. 9(b), we show the mesh of polynomial degree four composed of $2.9 \cdot 10^7$ nodes and $2.6 \cdot 10^6$ elements.

6.2 Sharp-to-smooth Modeling

In this example, we illustrate the capability of our method to perform a sharp-to-smooth modeling in different regions of the geometry. In particular, we recast some of the features entities present in the original model, and thus provide a new model improving the smoothness of the surrogate geometry. Each feature vertex (node of the mesh), curve (set of edges of the mesh) and surface (set of triangles of the mesh) is associated with a unique identifier. Therefore, to recast a feature, it is enough to know its identifier.

We consider a linear tetrahedral mesh from a CAD legacy model of a simplified Falcon aircraft. The boundary entities are marked defining 28 surfaces, 54 curves, and 34 vertices. As shown in Fig. 10(a) and Fig. 10(b), the main part of the fuselage is composed of two surfaces and a curve. However, such curve is not desirable since, ideally, we would desire a smooth mesh along each section of the fuselage. Therefore, we recast the curve indicating its unique identifier. The first step consists in removing the curve from the list of feature curves. Following, the two surfaces initially incident to this curve, see Fig. 10(a), are merged by identifying the id's of the two surfaces as a unique one, see Fig. 10(c). Thus, the whole fuselage is modeled as a smoother virtual surface.

Similarly, we observe that each section of the wing is described by two surfaces: one at the top and one at the bottom; and two curves: one on the leading edge and one on the trailing edge. In order to obtain a model with an improved smoothness on the leading edge, we decide to recast the feature curve describing the leading edge. This way, the surface at the top and the bottom are merged, and join smoothly in the front part of the wing. We highlight that the curve describing the trailing edge is maintained, and thus this sharp feature is preserved.

Note that the lateral wing joins the fuselage in a profile described by two curves (top and bottom) and two vertex points (front and back). We recast the feature vertex in the front. Therefore, this vertex is removed from the list of feature vertices, and the two curves are merged by identifying their id's as a unique one. As a result, we obtain a single closed curve with a sharp endpoint on the trailing edge.

Similar changes are made in similar parts of the mesh to generate a more convenient mesh model for flow simulation, see Fig. 10(c) and Fig. 10(d). As highlighted in Sect. 5.1, once the id's of all the features to recast are identified, the recasting process is straight-

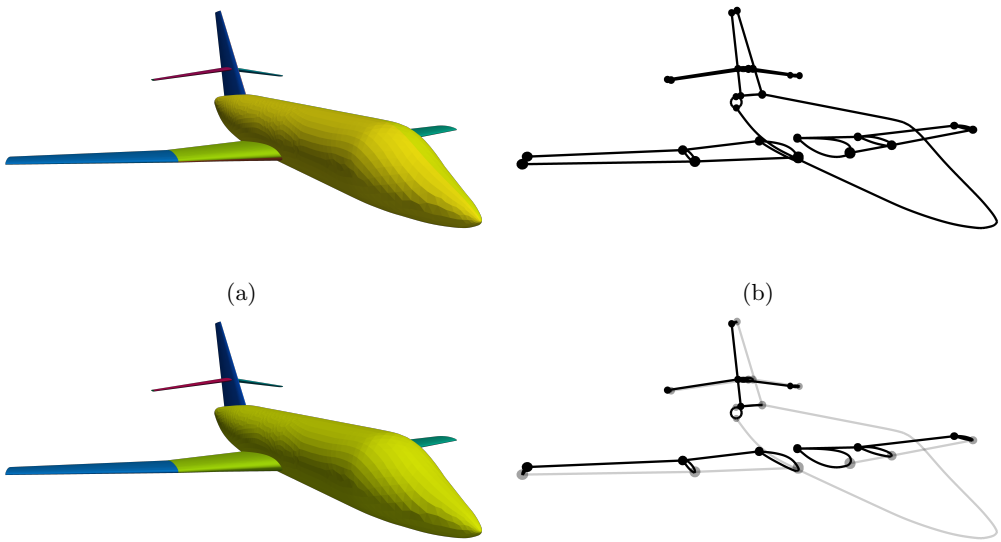


Figure 10: Initial and final linear mesh model of a Falcon aircraft. Marks on boundary entities of the initial model: (a) boundary triangles, and (b) curve and vertex features. Marks of the final model: (c) virtual surface features, (d) curve and vertex features recast (gray) and preserved (black).

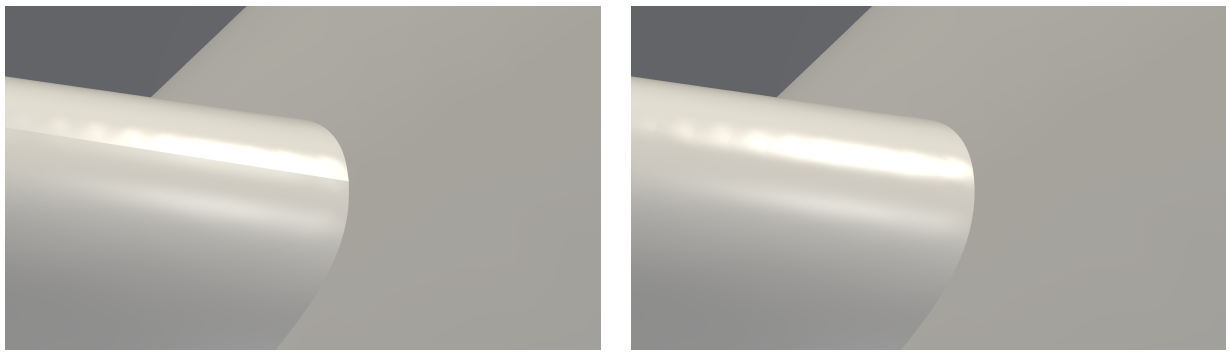
forward. Given the list of identifiers, the recasting process consists in removing these features from the list of features to preserve and automatically identify the features adjacent to the recast feature as a single one. We remark that the recasting of some of the geometry features does not modify the mesh, only the number of vertex, curve, and surface features changes. Specifically, the original model of the presented Falcon aircraft contains 34 vertex points, 54 curves, and 28 surfaces; while the model with the recast features contains 20 vertices to preserve, 32 curves, and 20 surfaces.

In order to illustrate the difference between these two models, we take a close look at the leading edge of the wing. In Fig. 11(a), we show the mesh of polynomial degree $p = 4$ generated with the initial marks. We observe a discontinuity in the normal vector of the wing along the leading edge. In Fig. 11(b), we show a mesh generated with a model in which the leading edge has been recast. The nodes originally present in the leading edge are still on the leading edge, but the new points are generated to interpolate the almost everywhere $\mathcal{C}^2$-continuous surrogate geometry. Those regions where the features have been recast are smoother in the second mesh model than in the original one. The leading edge now belongs to the interior of the surface, and therefore, all the nodes interpolate a $\mathcal{C}^1$-continuous surface.

A summary of the mesh quality can be found in Table 2. The linear mesh is composed of $1.3 \cdot 10^5$ nodes and $1.7 \cdot 10^6$ tetrahedra, and the volume mesh of polynomial degree $p = 4$ is generated in 611 seconds. This mesh, prior to the blending technique, contains 358 tangled elements. In this example, there are $3.0 \cdot 10^5$ boundary elements and the TFI reduces to 24 the number of invalid elements, that is, in 237 seconds a 93% of the invalid elements have been untangled. Now, we apply the optimization technique presented in [8, 9] to optimize locally the quality of the inverted elements. Since the mesh after the TFI is close to be optimal, it is a good initial condition for the implicit optimization and in 60 seconds the mesh becomes valid achieving a minimum quality of 0.7. In Fig. 12, we show the valid curved tetrahedral mesh of polynomial degree $p = 4$ composed of $1.8 \cdot 10^7$ nodes and $1.7 \cdot 10^6$ elements.

7. CONCLUDING REMARKS

The obtained results show that we can generate, from an initial straight-edged mesh, successively refined piece-wise linear, quadratic and quartic meshes, that target smooth curves and surfaces, while preserving the initially marked sharp features and smooth regions. The interior of the obtained limit curves is of class $\mathcal{C}^2$, and the interior of the surfaces is at least $\mathcal{C}^1$-continuous, being of class $\mathcal{C}^2$ when the surface mesh is structured.



(a) (b)

Figure 11: Close look at the wing of a Falcon aircraft. Mesh of polynomial degree four with (a) the initial model, and (b) the final model with the leading edge recast.

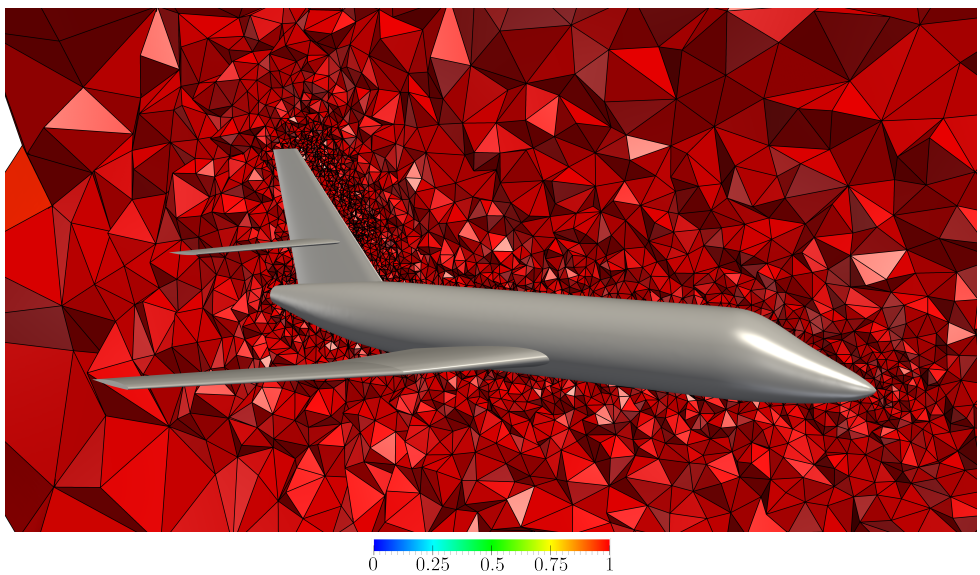


Figure 12: Curved tetrahedral mesh of polynomial degree $p = 4$ of a Falcon aircraft with no invalid elements.

Our method incorporates a unique sharp-to-smooth modeling capability not fully available in standard CAD packages. This capability allows removing sharp features, *i.e.* vertices and curves, and smoothly merge the incident entities, *i.e.* curves and surfaces. The resulting surrogate geometry features $\mathcal{C}^1$-continuity along the merging region. On the contrary, standard CAD packages use NURBS curve and surface modeling and thus, do not feature all these sharp-to-smooth modeling combinations. Note that with NURBS, it is possible to impose different levels of continuity between adjacent NURBS curves (surfaces) sharing a common point (curve) but only when using non-trimmed NURBS. Furthermore, NURBS modeling does not allow determining $\mathcal{C}^1$-continuity on a point shared by more than four non-trimmed quadrilateral surfaces.

Recall that in topographical applications, it is standard to have a structured mesh representation, and thus, this methodology leads to $\mathcal{C}^2$-continuous topography surface meshes. We also found that since the method is fast and explicit, even a non-vectorized Anaconda Python implementation, is competitive with whole mesh curving methods that might need parallel implementations for fine meshes.

The sharp curves and surfaces of the initial mesh determine the target geometries, and thus, when higher is the refinement level, closer to the target geometries is the resulting mesh. This capability of matching limit curves and surfaces is required to perform convergence studies to validate and verify the in-house flow solver.

If after successive refinement, the computational geometry would not lead to a limit geometry, the flow solution would not converge, too. Note that, after successive refinement, the meshes become nested and thus, the incorporation of a geometrical multigrid capability into the flow solver could be adequate.

The proportion of invalid elements is small compared to the size of the meshes, and thus, we showed that it can be fixed using local untangling and curving without the need for a global solver. That would not be the case for meshes for viscous laminar flow where highly stretched elements are needed to resolve boundary layers. Nevertheless, the meshes obtained with the proposed approach are well suited for those applications where isotropically but graded meshes are needed, such as in inviscid flow simulation and unsteady large eddy simulation.

In conclusion, it is possible to refine and curve a volume mesh and obtain smooth surfaces while preserving sharp features determined by vertices and polylines, the latter targeting smooth limit curves. For manifolds with boundaries, only a straight-edged mesh with boundary triangles marked with surface identifiers is required. Then, the method automatically computes the boundary curves and vertices from the triangle marks. Finally, the technique is well-suited to curve large quadratic and quartic meshes in low-memory configurations, *e.g.* curving a topographic mesh composed of two million and a half quartic elements using a laptop equipped with 16 GBytes of main memory.

ACKNOWLEDGMENTS

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 the third author has been partially supported by the Spanish Ministerio de Economía y Competitividad under the personal grant agreement RYC-2015-01633. Special thanks to Eloi Ruiz-Gironés.

References

[1] Gargallo-Peiró A., Avila M., Owen H., Prieto L., Folch A. "Mesh Generation for Atmospheric Boundary Layer Simulation in Wind Farm Design and Management." *Procedia Engineering*, vol. 124, 239–251, 2015

[2] Gargallo-Peiró A., Avila M., Owen H., Prieto-Godino L., Folch A. "Mesh generation, sizing and convergence for onshore and offshore wind farm Atmospheric Boundary Layer flow simulation with actuator discs." *Journal of Computational Physics*, vol. 375, 209–227, 2018

[3] Gargallo-Peiró A., Folch A., Roca X. "Representing urban geometries for unstructured mesh generation." *Procedia engineering*, vol. 163, 175–185, 2016

[4] Persson P.O., Peraire J. "Curved Mesh Generation and Mesh Refinement using Lagrangian Solid Mechanics." 12 2008

[5] Chaurasia H., Roca X., Persson P., Peraire J. "A coarse-to-fine approach for efficient deformation of curved high-order meshes." *Research Notes, 21st Int. Meshing Roundtable, Springer International Publishing*, pp. 1–5, 2012

[6] Johnen A., Remacle J.F., Geuzaine C. "Geometrical validity of curvilinear finite elements." *Journal of Computational Physics*, vol. 233, 359–372, 2013

[7] Gargallo-Peiró A. *Validation and generation of curved meshes for high-order unstructured methods.* Ph.D. thesis, Universitat Politècnica de Catalunya. Departament de Matemàtica Aplicada III, July 2014

[8] Gargallo-Peiró A., Roca X., Peraire J., Sarrate J. "Distortion and quality measures for validating and generating high-order tetrahedral meshes." *Engineering with Computers*, vol. 31, no. 3, 423–437, Jul 2015

[9] Gargallo-Peiró A., Roca X., Peraire J., Sarrate J. "Optimization of a regularized distortion measure to generate curved high-order unstructured tetrahedral meshes." *International Journal for Numerical Methods in Engineering*, vol. 103, no. 5, 342–363, 2015

[10] Ruiz-Gironés E., Sarrate J., Roca X. "Generation of curved high-order meshes with optimal quality and geometric accuracy." *Procedia engineering*, vol. 163, 315–327, 2016

[11] Ruiz-Gironés E., Roca X., Sarrate J. "High-order mesh curving by distortion minimization with boundary nodes free to slide on a 3D CAD representation." *Computer-Aided Design*, vol. 72, 52–64, 2016

[12] Moxey D., Ekelschot D., Keskin Ü., Sherwin S., Peiró J. "High-order curvilinear meshing using a thermo-elastic analogy." *Computer-Aided Design*, vol. 72, 130 – 139, 2016. 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation

[13] Jiao X., Wang D. "Reconstructing high-order surfaces for meshing." *Engineering with Computers*, vol. 28, no. 4, 361–373, Oct 2012

[14] Ims J., Duan Z., Wang Z.J. "meshCurve: an automated low-order to high-order mesh generator." *22nd AIAA computational fluid dynamics conference*, p. 2293. 2015

[15] Persson P.O., Aftosmis M.J., Haimes R. "On the Use of Loop Subdivision Surfaces for Surrogate Geometry." P.P. Pébay, editor, *Proceedings of the 15th International Meshing Roundtable*, pp. 375–392. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006

[16] Jiménez Ramos A. *Incorporating curvature to the boundary of linear and highorder meshes when a target geometry is unavailable.* Master's thesis, Universitat Politècnica de Catalunya, 2018

[17] Dyn N., Levine D., Gregory J.A. "A butterfly subdivision scheme for surface interpolation with tension control." *ACM transactions on Graphics (TOG)*, vol. 9, no. 2, 160–169, 1990

[18] Yang H.Q., Zhou X., Harris R.E., Yang S. "An Open Source, Geometry Kernel Based High-Order Element Mesh Generation Tool." *AIAA Scitech 2019 Forum*, p. 1719. 2019

[19] Johnen A., Roca X., Toulorge T., Remacle J. "A new framework for curving structured boundary-layer meshes." *International Conference on Adaptive Modeling and Simulation.* 2018

[20] Gargallo-Peiró A., Houzeaux G., Roca X. "Subdividing triangular and quadrilateral meshes in parallel to approximate curved geometries." *Procedia Engineering*, vol. 203, 310 – 322, 2017. 26th International Meshing Roundtable, IMR26, 18-21 September 2017, Barcelona, Spain

[21] Lane J.M., Riesenfeld R.F. "A theoretical development for the computer generation and display of piecewise polynomial surfaces." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, , no. 1, 35–46, 1980

[22] Loop C. *Smooth Subdivision Surfaces Based on Triangles.* Master's thesis, Department of Mathematics, The University of Utah, Masters Thesis, January 1987

[23] Zorin D. "A method for analysis of C 1-continuity of subdivision surfaces." *SIAM Journal on Numerical Analysis*, vol. 37, no. 5, 1677–1708, 2000

[24] Stam J. "Evaluation of Loop Subdivision Surfaces." 01 1998

[25] Demmel J.W., Eisenstat S.C., Gilbert J.R., Li X.S., Liu J.W.H. "A supernodal approach to sparse partial pivoting." *SIAM J. Matrix Analysis and Applications*, vol. 20, no. 3, 720–755, 1999

[26] Li X., Demmel J., Gilbert J., iL. Grigori, Shao M., Yamazaki I. "SuperLU Users' Guide." Tech. Rep. LBNL-44289, Lawrence Berkeley National Laboratory, September 1999. `http://crd.lbl.gov/~xiaoye/SuperLU/`. Last update: August 2011

[27] Jones E., Oliphant T., Peterson P., et al. "SciPy: Open source scientific tools for Python.", 2001. [Online; last accessed September-2018]

[28] Perronnet A. "Interpolation transfinie sur le triangle, le tétraèdre et le pentaèdre. Application à la création de maillages et à la condition de Dirichlet." *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, vol. 326, no. 1, 117–122, 1998

[29] Ruiz-Gironés E., Gargallo-Peiró A., Sarrate J., Roca X. "Automatically imposing incremental boundary displacements for valid mesh morphing and curving." *Computer-Aided Design*, 2019

[30] Toulorge T., Geuzaine C., Remacle J.F., Lambrechts J. "Robust untangling of curvilinear meshes." *Journal of Computational Physics*, vol. 254, 8–26, 2013

[31] Van Rossum G., Drake Jr F.L. *Python reference manual.* Centrum voor Wiskunde en Informatica Amsterdam, 1995

[32] Gargallo-Peiró A., Roca X., Peraire J., Sarrate J. "A distortion measure to validate and generate curved high-order meshes on CAD surfaces with independence of parameterization." *International Journal for Numerical Methods in Engineering*, vol. 106, no. 13, 1100–1130, 2016