Region extraction in mesh intersection

Pablo Antolin\textsuperscript{a}, Annalisa Buffa\textsuperscript{a,b}, Emiliano Cirillo\textsuperscript{a,*}

\textsuperscript{a}Institute of Mathematics, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, 1015, Switzerland
\textsuperscript{b}Istituto di Matematica Applicata e Tecnologie Informatiche "Enrico Magenes" del CNR, 27100 Pavia, Italy

Abstract

Region extraction is a very common task in both Computer Science and Engineering with several applications in object recognition and motion analysis, among others. Most of the literature focuses on regions delimited by straight lines, often in the special case of intersection detection among two unstructured meshes. While classical region extraction algorithms for line drawings and mesh intersection algorithms have proved to be able to deal with many applications, the advances in Isogeometric Analysis require a generalization of such problem to the case in which the regions to be extracted are bounded by an arbitrary number of curved segments. In this work we present a novel region extraction algorithm that allows a precise numerical integration of functions defined in different spline spaces. The presented algorithm has several interesting applications in contact problems, mortar methods, and quasi-interpolation problems.

Keywords: Isogeometric analysis, Mortar methods, Mesh intersection, Numerical integration

1. Introduction

Given \( n \) planar curves \( c_1, \ldots, c_n \), the region extraction problem consists in finding the regions bounded by the \( n \) curves and their intersections, see Figure 1. Such problem is a very common task in both Computer Science and Engineering with several applications in object recognition, motion analysis and stereopsis, among others.

Most of the existing literature focuses on the case of a line drawing, that is a class of pictorial data where the information is conveyed by the edges and the vertices of a planar graph, or to the case of intersection detection among two unstructured mesh. In the first setting, Jiang and Bunke \cite{Jiang} proposed an approach based on the arrangement of groups of edges of a planar graph in a counter-clockwise order. The ordered edges are then scanned linearly in order to create wedges representing the area between two consecutive edges. The list of wedges is again sorted according to their vertices and appended together in order to build the different regions in \( O(n \log n) \) steps, where \( n \) is the number of edges of the line drawing. A similar algorithm was proposed by Shih \cite{Shih}, with the difference that the region extraction step of the algorithm is performed with a tagging process that results in assigning the same tag to the wedges belonging to the same region. This process allows to speed-up the number of steps to \( O(n) \). In \cite{Shih} instead, the authors use an adjacency matrix \( M \) with entries

\[
M_{i,j} = \begin{cases} 
1, & \text{if there is an edge between vertices } i \text{ and } j \\
0, & \text{otherwise.}
\end{cases}
\] (1)

and the clockwise angles between adjacent edges of the line drawing in order to walk the faces to be extracted in a counter-clockwise direction. Many different algorithms have instead been proposed for the identification of solid’s faces in a wireframe, see \cite{Jiang} \cite{Shih} \cite{Shih} and references therein, but all of these algorithms rely on the fact that wireframes can be represented as planar graphs.

A planar graph is a pair \((V, E)\), where \( V = \{v_1, \ldots, v_n\} \) is a set of vertices and \( E \) is a set of paired vertices called edges. Planar graphs can be represented using an adjacency matrix as in \cite{Jiang} and therefore cannot fully depict a situation as the one represented in Figure 1. If the boundary of the regions are indeed curvilinear segments, two vertices can be connected by more than one edge, see regions \( \psi_1, \psi_2, \psi_7, \) and \( \psi_8 \) in Figure 1 a relation that cannot be represented using an adjacency matrix.

A problem similar to region extraction arises in the context of mesh intersection methods as well. In order to intersect two unstructured meshes, Gander and Japhet \cite{Gander} proposed to use an approach that can be split in two steps. Given two triangular meshes \( M_1 \) and \( M_2 \) and two triangles \( T_1 \in M_1 \) and \( T_2 \in M_2 \), the first step consists in the identification of the intersections between \( T_1 \) and \( T_2 \). During this operation, a list of neighbor triangles of \( T_2 \) that intersect with \( M_1 \) is retrieved as well. The second step makes use of this additional information in order to extract the regions with an advancing front technique, defining the mesh intersections of \( M_1 \) and \( M_2 \). Another advancing front al-
A method to overcome this issue is represented by the so-called **segment-based integration schemes**. The idea behind these schemes is that the restrictions of the splines to their elements, are simple polynomials and therefore can be integrated using standard quadrature techniques.

Segments-based integration schemes are based on the approximation of the image of the regions in which the splines are simple polynomials. In the context of isogeometric mortar methods, for example, the product of two B-spline functions needs to be integrated over (part of) the boundary of a surface, or a solid, called **interface**. Seitz and colleagues [12] considered four-sided (linear) quadrilateral approximations of the regions in which the splines are polynomials. Quadrilaterals corresponding to the different splines are coupled and projected to a common auxiliary plane, where their intersection is identified via a clip polygon algorithm. The obtained polygon is then triangulated and the quadrature points are defined inside each triangle. The obtained quadrature points are finally mapped back to one of the original curved regions in order to evaluate the desired integral. Hesh and Betsch [13] used instead a different approach in the context of domain decomposition methods. The authors proposed to project the control points of both splines on one of the two surfaces. Then the intersection of the edges of the two splines are identified and the set of obtained intersection points are triangulated. Quadrature points are again defined in each obtained triangle.

While both approaches are quite efficient, their main drawback is that they cannot represent exactly the regions in which the involved splines are polynomials. If the regions involved are heavily concave a non-negligible amount of quadrature points will still be defined outside the desired region, affecting the quality of the numerical integration. In order to overcome this issue, an algorithm that automatically recognizes and extracts the curved regions in which the splines have the desired order of continuity is necessary. In this work we present a region extraction algorithm that can be applied to this context as well. This work is divided as follows. In Section 2 we...
give the definition of a curvilinear drawing, a generalization of the line drawing treated in [1] and [2]. Our region extraction algorithm is presented in the same section. In Section [3] we present several applications of the newly introduced algorithm in the context of mesh intersection for B-spline trivariate solids. Finally, some conclusive remarks are given in Section [4].

2. The region extraction algorithm

Before introducing our algorithm, we first need to set some notations and definitions. The idea of line drawing can be easily generalized by the following.

Definition 1. Let $c_1, \ldots, c_n$ be $n$ curves

$$c_j : [0, 1] \to \mathbb{R}^2, \quad j = 1, \ldots, n,$$

such that $c_j \in C^2[0, 1]$ and their intersection points form a discrete set. Then the set of curves $c_1, \ldots, c_n$, together with their intersection points, is said to be a curvilinear drawing.

An example of curvilinear drawing with curves $c_1, \ldots, c_{13}$ is shown in Figure 1. The requirement of the curves $c_1, \ldots, c_n$ to be $C^2$ is not strictly necessary but is here enforced in order to simplify the discussion. The goal of this work is to extract the planar regions delimited by the (restriction of the) curves $c_1, \ldots, c_n$. We remark that the condition about the discreteness of the intersection points is enforced in order to avoid curves that are partially coincident. For the purpose of this work we assume that we receive in input only the curves and that their restrictions need first to be found via curve-curve intersection.

To this end we loop over each curve in order to find its intersections with all the others, that are the vertices of the curvilinear drawing.

Definition 2. Let $v \in \bar{\Omega}$ be the intersection of at least two curves in a curvilinear drawing. Then we say that $v$ is a vertex of the curvilinear drawing.

Considering again Figure 1, the curvilinear drawing has ten vertices, denoted with $v_1, \ldots, v_{10}$, respectively.

Let $C$ and $V$ be the lists containing the curves and the currently identified vertices of a line drawing. Every time a new vertex $v$ has been identified we add it to the list $V$ and update two maps. The first map

$$V_c : V \to C$$

keeps track of the curves defining each vertex in $V$. The second map

$$C_v : C \to V$$

helps to identify the vertices lying on the same curve. For each curve $c$ we store the vertices lying on $c$ following the order given by the corresponding parameters. Assuming that all the curves in Figure 1(a) are parameterized as shown in Figure 1(b), Table 1 shows the corresponding maps $V_c$ and $C_v$ for each vertex and each curve. Notice that $C_v(c_{11})$ and $C_v(c_{12})$ contain twice the same vertex, indicating that $c_{11}$ and $c_{12}$ are closed curves that should be traversed in both directions. When all the vertices of the curvilinear drawing have been identified, $V_c$ and $C_v$ allow us to find the corresponding edges.

Definition 3. Let $v_i, v_j$, and $e$ be respectively two vertices and a curve in a curvilinear drawing such that there exist two parameters $t_i, t_j, t_i < t_j$, with $e(t_i) = v_i$ and $e(t_j) = v_j$. Then the restriction $e = e|_{[t_i, t_j]}$ is an (oriented) edge of the curvilinear drawing.

The edges of the curvilinear drawing in Figure 1(a), together with their orientation, are shown in Figure 1(b). Using a rather standard notation, here and in the following, we denote with $-e_i$ the edge $e_i$ traversed following its inverse parameterization. With these definitions at hand, a region of the curvilinear drawing can be represented as a closed trail.

Definition 4. A closed trail is a sequence of pairs of vertices and edges

$$\{ (v_{i_1}, e_{i_1}), (v_{i_2}, e_{i_2}), \ldots, (v_{i_m}, e_{i_m}) \}$$

such that each $e_{i_j}$ is an edge between $v_{i_j}$ and $v_{i_{j+1}}$ and $e_{i_m}$ is an edge between $v_{i_m}$ and $v_{i_1}$.

Note that the edges in this definition are oriented. As an example, let us consider again the drawing in Figure 1. The region $\psi_3$ can be represented as

$$\psi_3 = \{ (v_1, e_5), (v_4, e_{11}), (v_5, -e_7), (v_3, -e_3) \}.$$
Algorithm 1 ExtractRegions:
Extracts the regions from the curvilinear drawing

| i     | Π   |
|-------|-----|
| 1     | {e_1, e_2, e_3, e_5} |
| 2     | {−e_1, −e_2, −e_4, e_9} |
| 3     | {e_3, e_1, e_7} |
| 4     | {−e_3, e_8, e_{11}} |
| 5     | {e_7, e_8, −e_{11}, e_{12}} |
| 6     | {e_9, e_{10}, e_{13}, e_{12}} |
| 7     | {−e_9, e_{14}, e_{16}, e_{17}, e_{18}, e_{19}} |
| 8     | {−e_8, −e_{14}, e_{15}, e_{16}, −e_{17}} |
| 9     | {−e_{10}, −e_{15}} |
| 10    | {−e_{13}} |

Table 2: List of unvisited paths for the vertices v_1, ..., v_{10} in Figure 1

and

\[ \psi_9 = \{(v_7, -e_{19})\}, \]

respectively.

As remarked in Section 1, a line drawing can be efficiently represented as the adjacency matrix \( M \) in (1). Since this is not the case for a curvilinear drawing with loops (e.g. \( \psi_9 \) and \( v_{10} \) in Figure 1) and multiedges (e.g. \( e_1, e_2, e_{14}, e_{16}, e_{17} \) in Figure 1), we store its connectivity as lists of unvisited paths, one for each vertex of the curvilinear drawing. Given a vertex \( v \), its unvisited path list \( \Pi \) contains the (oriented) edges that originate in \( v \).

For instance, the initial list of unvisited paths for vertex \( v_5 \) in Figure 1 is given by

\[ \Pi_5 = \{-e_7, e_{12}, -e_{11}, e_8\}. \] (2)

Using the maps \( V_c \) and \( C_i \) in Table 1 (a) and (b), we can easily build the initial lists of unvisited paths for the drawing in Figure 1 see Table 2. By Definition 3, the list corresponding to vertex \( v_{10} \) contains only the path \(-e_{13}\), as curve \( e_{13} \) in Figure 1 has no further vertices. Vertices like \( v_{10} \) are called dangling nodes and are easily recognizable as vertices having only one open edge among their unvisited paths.

Our region extraction algorithm is described in Algorithm 1. Line 2 calls a simple routine that takes care of recognizing and discarding the dangling nodes in the curvilinear drawing, see Algorithm 2. We remark that, assigning an orientation to the edges, we implicitly duplicate the edges as each edge belongs to exactly two vertices and is hence traversed once in each direction in Algorithm 1.

Given a curvilinear drawing, the main idea behind Algorithm 1 is to construct a corresponding rotation system [14] and to use the obtained topological information to extract the regions bounded by its halfedges. For any vertex \( v \) in a given multigraph, a rotation system associates an ordering to the edges originating in \( v \). Such ordering is implicitly defined by the orientation of the surface in which the multigraph lies. Assigning a counter-clockwise ordering to all the edges that originate in \( v \), it is then possible to extract the regions around that vertex, by simply walking all closed trails starting and ending in \( v \).

In order to clarify how the region extraction algorithm works, let us consider again the drawing in Figure 1. Without loss of generality, we assume that \( v_{i_1} = v_5 \) in Line 3 of Algorithm 1. At the beginning of the algorithm the list of unvisited paths for \( v_5 \) is given in (2). The algorithm assigns \(-e_9\) to \( e_{i_1} \) in Line 5 and picks its end-point \( v_3 \). Therefore it sets \( v_{i_2} = v_3 \) and looks in the corresponding list \( \Pi_3 \), see Table 2, for the edge that forms the maximum counter-clockwise angle with \( e_{i_1} \). To this end we use the following definition of angle between two edges.

Definition 5. Let \( v \) be a vertex and let \( \bar{e} \) and \( \bar{e} \) be two edges originating in \( v \). Then the angle in \( v \) between edges \( \bar{e} \) and \( \bar{e} \) is given by the counterclockwise angle between \( t(v) \) and \( \bar{t}(v) \), where \( t(p) \) and \( \bar{t}(p) \) denote the tangents of \( \bar{e} \) and \( \bar{e} \) in a point \( p \in \Omega \), respectively. The orientation of the tangents are chosen so that they always point toward the interior of the corresponding edges.

Therefore we set \( e_{i_2} = -e_3 \) and we initialize the region \( \psi = \{(v_5, -e_7)\} \). Since \( e_{i_2} \neq e_{i_0} \), we add the pair \((v_3, -e_3)\) to \( \psi \) and we remove \(-e_3\) from the list of unvisited paths of \( v_3 \). The algorithm then sets \( v_{i_3} = v_1 \) and \( e_{i_3} = e_5 \), since \( e_5 \) is the edge forming again the maximum angle with \( e_{i_2} \). The pair \((v_1, e_5)\) is added to \( \psi \) and \( e_5 \) is removed from the list of unvisited paths of \( v_1 \). Since the end-point of \( e_5 \) is \( v_4 \), Lines 13 and 14 set \( v_{i_4} = v_4 \) and \( e_{i_4} = e_{11} \). The list \( \psi \) is then updated with the pair
(v_4, e_{11}) and e_{11} is removed by Π_4. Now the end-point of e_{11} is the initial vertex v_5 and the edge with maximal angle is −e_7 that is still in the list of unvisited paths Π_1 = Π_5. Therefore, we set v_6 = v_5 and e_{16} = −e_7. Since e_{16} = e_{a_6}, the algorithm terminates the inner loop, deletes −e_7 from Π_6 and adds the identified face ψ_3 to the list Ψ of extracted regions. After the identification of ψ_3, the list of unvisited paths for v_5, v_3, v_1, and v_4 are

\[ Π_5 = \{ e_8, −e_{11}, e_{12} \}, \]
\[ Π_3 = \{ e_4, e_7 \}, \]
\[ Π_1 = \{ e_1, e_2, e_3 \}, \]

and
\[ Π_4 = \{ −e_5, e_6 \}, \]

respectively. Algorithm 1 proceeds then with the extraction of the regions ψ_1, ψ_4 and ψ_5 corresponding to v_1 = v_5 and e_1 = e_{a_8} e_{a_{12}} and −e_{11}, respectively. Once all the regions surrounding v_5 have been extracted, Algorithm 1 finds a different vertex with a non empty unvisited paths list, until all regions in the curvilinear graph have been recognized.

Let us remark some features about Algorithm 1. An edge in a curvilinear drawing is adjacent to exactly two regions, one for each direction in which we can traverse it, while any region adjacent to a vertex is always bounded by at least one of the edges originating at said vertex. Therefore, Lines 7 and 13 in Algorithm 1 are always guaranteed to succeed. On the other hand, for any closed region in a curvilinear graph there exists at least one edge that is adjacent to it and hence the algorithm is guaranteed to extract all of them. Finally, the algorithm is guaranteed to terminate when all the edges of the drawing have been traversed in both directions.

The detection of the maximal angle extensively used in Algorithm 1 can be complicated by the fact that two or more edges can have same tangent directions in a vertex. In order to simplify the discussion, we did not consider this case in the description of the region extraction algorithm. Nevertheless, these corner cases can be treated by considering beforehand the curvature of the edges having same tangents in a single vertex.

Among the regions extracted by Algorithm 1 there is also the external unbounded region ψ_3 in Figure 1. Whether similar regions need to be purged or not can be application dependent but it is possible to easily identify them as the regions for which the list of consecutive vertices and edges follows a clockwise direction. In order to avoid further operations, it is possible to compute the angles between the edges in the interval [−π, π] and to sum them up. Once a region has been extracted, it can be purified if the sum of the angles is positive.

**Algorithm 2 PurgeDanglingNodes**: Recognizes and delete the dangling nodes and the relative edges.

```
Input Vertices list V = \{v_1, \ldots, v_m\};
Input Unvisited paths lists, Π = \{Π_1, \ldots, Π_m\};
Output Updated V and Π;
1: L = m; /* Initial number of vertices. */
2: for v_i ∈ V do
3:   if \|Π_i\| = 1 then /* v_i is a dangling node. */
4:     e ← edge in Π_i;
5:     delete e from all unvisited paths lists in Π;
6:     V = V \ \{v_i\};
7:   end if
8: end for
9: if \|V\| < L then
10:   PurgeDanglingNodes(V, Π);
11: end if
```

### 3. Applications

As mentioned in Section 1, Algorithm 1 can be used as a tool to numerically compute integrals involving spline functions with a finite order of continuity and their product. Usually, standard quadrature rules provide accurate approximations of the integrals only if the integrand can be well approximated by polynomials. In order to numerically compute the integral of splines functions it is therefore necessary to identify the sub-regions of their domain in which they are polynomials, and apply the quadrature rule separately in each sub-region. Algorithm 1 represents a robust tool to automatically recognize and extract such regions in the case of an integrand represented by a product of spline functions.

In this section we further develop this idea in the context of trivariate B-spline functions defined over three-dimensional solids. Despite we here restrict ourselves to the case of n = 2 intersecting solids, the methods described in the rest of this section can be easily generalized to the case n ≥ 3, using the domain decomposition presented in [15].

Let us denote with B_{i,d} = B_{i,d,t} the i-th tensor-product trivariate B-spline basis function of degrees d = (d_u, d_v, d_w) with knot vectors t = (t_u, t_v, t_w) and with \delta_{d,t} the corresponding linear space spanned by these basis functions. Moreover, for every bounded domain Ω ∈ R^3, we denote with \Sigma its closure.

Let us formalize our setting. Let \Omega ⊆ R^3 be a connected, bounded domain such that there exist two possibly overlapping domains Ω_1 and Ω_2 such that Ω = Ω_1 ∪ Ω_2. The domain Ω can be partitioned as \{Ω_1, Ω_2\}, where

\[
Ω_1 = Ω_1^* \cup Ω_2^* \cup Ω_3^*; \quad Ω_2 = Ω_2^* \setminus Ω_1^*.
\]
We define the interface \( \Gamma_{1,2} \) as
\[
\Gamma_{1,2} = \partial \Omega_1^* \cap \bar{\Omega}_2^*.
\] (4)

Figure 2 shows an example of such a domain decomposition. In Figure 2(a) the two overlapping domains \( \Omega_1^* \) and \( \Omega_2^* \) are shown. Figure 2(b) shows the partition \( \{ \Omega_1^*, \Omega_2^* \} \) as defined in (3), while (c) shows the interface \( \Gamma_{1,2} \) as defined in (4). Let \( T_1 \in S_{d_1,t_1} \) and \( T_2 \in S_{d_2,t_2} \) be two trivariate B-spline parameterizations of \( \Omega_1^* \) and \( \Omega_2^* \), respectively, see Figure 3. Here and in the rest of this section we finally denote with \( \Gamma_{1,2} \) the preimage of \( \Gamma_{1,2} \) in the parametric space of \( T_1 \). In this section we are going to face three different problems. In Section 3.1 we are going to use Algorithm 1 in order to create a suitable quadrature rule for the approximate computation of integrals defined over the interface of two solids. Two numerical experiments are carried out, one concerning the integration of a smooth function and one regarding the integration of the product of splines defined in different spline spaces.

In Sections 3.2 and 3.3 we apply the quadrature rule described in Section 3.1 in order to enforce weak continuity constraints to volumetric objects. The constraints are imposed differently, depending on the relative position of the two objects. In one of the cases we are able to reproduce the deformation of \( \Omega_2^* \) in the spline space \( S_{d_1} \), provided that the latter is sufficiently refined. In the second case, the appearance of oscillations does not allow us to obtain the same results but we approximate the deformation of \( \Omega_2^* \) using a convolution based strategy.

Finally, in Section 3.4 we use our algorithm to solve the Poisson’s problem for two bodies in a contact position, using a mortar-like approach as described in [16].

3.1. Precise computation of integrals over the interface

In this section we are going to compute integrals over the interface \( \Gamma_{1,2} \). We propose two different examples, one with a smooth function and one with a product of splines belonging to the spline spaces of \( T_1 \) and \( T_2 \), respectively. In order to numerically compute the integral in the latter case it is necessary to find the mesh intersection between the mesh inherited by the two splines. By construction, see Equation (4), \( \Gamma_{1,2} \) is always part of the boundary of \( \Omega_1^* \) and therefore it inherits from \( T_1 \) its mesh information. Our goal is therefore to find out how the mesh of \( T_2 \) intersects with the natural one of \( \Gamma_{1,2} \). To perform this operation we follow three steps.

1. Extract the knots isoparametric surfaces of \( T_2 \) in Euclidean space, see Figure 4(a);
2. Find the intersection curves of each isoparametric surface with the interface \( \Gamma_{1,2} \). Note that not all the isoparametric surfaces in Step 1 have necessarily an intersection with \( \Gamma_{1,2} \), see Figure 4(b);
3. Pull-back the obtained curves in \( \Gamma_{1,2} \) via \( T_1^{-1} \), see Figure 4(c).

The list of pull-back curves, the parametric grid of \( \Gamma_{1,2} \), and the boundary curve of \( \Gamma_{1,2} \), together with their intersection points compose now a curvilinear drawing as in Definition 1 and we can hence apply Algorithm 1 in order to extract the list of regions \( \Psi \) that represents the mesh intersection of the interface \( \Gamma_{1,2} \), see colored regions in Figure 4(c). We remark that several of the vertices of the curvilinear drawing are represented by the intersections of curves with straight lines that are parallel to the \( u \) and \( v \) directions and therefore can be retrieved very efficiently.

In both the examples of this section we consider the geometric setting in Figure 5(a). The solids \( \Omega_1^* \) and \( \Omega_2^* \) are parameterized by two trivariate B-splines \( T_1 \in S_{d_1,t_1} \) and \( T_2 \in S_{d_2,t_2} \), respectively. Figure 5(b) shows the pull-back of the intersection curves between the isoparametric surfaces of \( \Omega_2^* \) and \( \Gamma_{1,2} \) in the parametric space of \( T_1 \).

In the first example we compute the integral
\[
\int_{\Gamma_{1,2}} \sin \left( \frac{\pi x}{2} \right) \cos(\pi y) e^x, \tag{5}
\]
using our algorithm and comparing the result with the one
obtained using standard quadrature rules. Denoting with
\( f(x, y) = \sin\left(\frac{\pi}{2} x\right)\cos(\pi y)e^x, \)
Equation (5) can be rewritten as
\[
\int_{\Gamma_{1,2}} f = \int_{\hat{\Gamma}_{1,2}} f \circ S_{1,2} \det(\nabla S_{1,2}), \tag{6}
\]
where \( S_{1,2} : \hat{\Gamma}_{1,2} \to \Gamma_{1,2} \) is a parameterization of the interface \( \Gamma_{1,2} \), that is \( S_{1,2} = T_{1|\hat{\Gamma}_{1,2}}. \) Since \( f \) is an analytic function, Equation (6) can be numerically computed easily with standard quadrature rules. We denote with \( I_f \) the value of (5) obtained in such a way, using an overkill number of quadrature points. In order to test our algorithm, we further write (6) as
\[
\int_{\Gamma_{1,2}} f \circ S_{1,2} \det(\nabla S_{1,2}) = \sum_{\psi \in \Psi} \sum_{j=1}^{L_\psi} \int_{[0,1]^2} f \circ S_{1,2} \circ \pi_j \det(\nabla \pi_j) \det(\nabla S_{1,2}), \tag{7}
\]
where \( \Psi \) contains all the regions extracted by Algorithm 1, see Figure 3(b). Finally, in order to compute numerically the integrals in (7), we need to create a suitable quadrature rule for each region \( \psi \). To this end, we apply the untrimming algorithm proposed in \cite{17} in order to split \( \psi \) in a list of four-sided, non overlapping, free-form quadrilaterals parameterized as planar parametric patches \( \pi_1, \ldots, \pi_{L_\psi} \)
\[
\pi_j : [0, 1]^2 \to \psi, \quad j = 1, \ldots, L_\psi.
\]
Each \( \pi_j \) is guaranteed to be a Bézier patch and the union of their images is a partition of \( \psi \) in \( \hat{\Gamma}_{1,2} \). Therefore, we can finally compute (5) as
\[
\int_{\Gamma_{1,2}} f \circ S_{1,2} \det(\nabla S_{1,2}) = \sum_{\psi \in \Psi} \sum_{j=1}^{L_\psi} \int_{[0,1]^2} f \circ S_{1,2} \circ \pi_j \det(\nabla \pi_j) \det(\nabla S_{1,2}), \tag{8}
\]
which can be computed with standard quadrature techniques. We remark that the algorithm proposed in \cite{17} minimizes the number of quadrilaterals necessary to partition each region \( \psi \) with a greedy algorithm and therefore \( \psi \) is rarely split in more than two patches.

The integrals in (8) are computed separately using \( 2^j \) quadrature points per tile direction and we denote with \( I_f^{(j)} \) the obtained value of the integral in (5). These results are compared with \( I_f \) by considering the error
\[
E_f^{(j)} = |I_f - I_f^{(j)}|. \tag{9}
\]
In the second test we want to show that Algorithm 1 is a suitable method for computing the integral of piecewise polynomials defined over \( \hat{\Gamma}_{1,2} \). To this end, we consider...
the function
\[ s(u, v) = s_1(u, v)\tilde{s}_2(u, v), \]
where \( s_1 \in S_{d_1, t_1} \) and
\[ \tilde{s}_2 = s_2 \circ T_2^{-1} \circ T_1, \]
for some \( s_2 \in S_{d_2, t_2} \).

As remarked in Section 3, in order to integrate \( s \) over \( \Gamma_{1,2} \) it is necessary to identify the regions in which \( s_1 \) and \( \tilde{s}_2 \) have maximum order of continuity and the algorithm presented in this work allows us to easily recognize such regions. We therefore express the integral of \( s \) as
\[
\int_{\Gamma_{1,2}} s = \sum_{\phi \in \Psi} \sum_{j=1}^{L_\phi} \int_{[0,1]^2} (s_1 \tilde{s}_2) \circ \pi_j \det(\nabla \pi_j) \]
(11)
and we numerically compute each integral in (11) using an overkill number of quadrature points for each tile. Denoting such value with \( I_s \), we define
\[
E^{(j)}_s = |I_s - I_s^{(j)}| \]
(12)
where \( I_s^{(j)} \) is the value of the integral approximated using \( 2^j \) quadrature points per tile direction.

In both experiments, we stop the computation of the quadrature rule if two subsequent approximated values of the integrals are close enough, that is when
\[
|I_s^{(j)} - I_s^{(j-1)}| < 10^{-12}, \quad \alpha = f, s. \]
The results of the integrations for both numerical tests are visible in Table 3 and Figure 6.

We remark that in the computation of \( I_{s}^{(j)} \) geometric operations such as surface-surface intersections, pull-back of curves and curve-curve intersections are of utmost importance. If a pull-back curve is computed coarsely, the image of a quadrature point through \( T_2^{-1} \circ T_1 \) is not guaranteed to be in the right knot element of \( s_2 \). If this happen,
Table 3: Error values for the integrations in Section 3.1. The table reports the values of $E_j^{(j)}$ and $E_s^{(j)}$ in (9) and (12), respectively, for $j = 0, \ldots, 5$.

| $j$ | 0     | 1     | 2     | 3     | 4     | 5     |
|-----|-------|-------|-------|-------|-------|-------|
| $E_f^{(j)}$ | 7.90e−02 | 1.00e−03 | 1.49e−06 | 6.13e−12 | 1.27e−12 | 1.27e−12 |
| $E_s^{(j)}$ | 2.21e−03 | 6.53e−05 | 1.77e−07 | 1.41e−09 | 1.46e−09 | 1.46e−09 |

In Figure 6, the semi-log plot of $E_f^{(j)}$ and $E_s^{(j)}$ in (9) and (12), respectively, for $j = 0, \ldots, 5$. See also Table 3.

3.2. Weak continuity

The algorithm proposed in this work can be used to enforce weak continuity constraints to solids geometries. Weak continuity constraints have been imposed to bidimensional geometries by Zou and colleagues in [22]. Their approach is based on a newly defined Bézier projector. A similar result can be achieved with a Lee-Lyche-Mørken quasi-interpolant [23], taking advantage of the presented algorithm. We remark that also the method presented in [22] can be implemented using our algorithm for improved accuracy.

Let us denote with $\Omega_1^*$ and $\Omega_2^*$ two solids parameterized by two trivariate B-splines $T_1$ and $T_2$ and with $\hat{\Omega}$ their Boolean union. Let $\Omega_2^*$ be a deformation of $\Omega_2^*$ such that there exists a trivariate displacement B-spline $\delta T_2$ such that $T_2 + \delta T_2$ is a parameterization of the closure of $\hat{\Omega}_2^*$. Our goal is to find a corresponding trivariate displacement $\delta T_1$ such that $T_2 + \delta T_2$ and $T_1 + \delta T_1$ form a weakly continuous piecewise parameterization of

$$\hat{\Omega} = \hat{\Omega}_1^* \cup \hat{\Omega}_2^*,$$

where $\hat{\Omega}_1^*$ is the domain parameterized by $T_1 + \delta T_1$, representing the corresponding deformation of $\Omega_1^*$.

To this end we find $\delta T_1$ as the Lee–Lyche–Mørken quasi-interpolant of $\delta T_2$, that is $\delta T_1 = I_{LLM}\delta T_2$. This family of quasi-interpolants is a widely used method for the local projection a function $f \in \mathcal{L}_2(D)$, for some domain $D$, into a given spline space $\mathcal{S}_{d,t}$

$$I_{LLM} : \mathcal{L}_2(D) \rightarrow \mathcal{S}_{d,t}$$

In their work [23], Lee, Lyche, and Mørken proposed a procedure to build such quasi-interpolants that is based on the use of local spline projectors, see Algorithm 3. There are two main advantages in using a Lee-Lyche-Mørken approach in this setting. On the one hand, being based on local spline projectors, only the elements of $\Omega_1$ close to the interface are to be influenced and therefore a refinement step can help us at controlling the influence that $\delta T_2$ has on $\Omega_1^*$. On the other hand, we are sure of exactly reproducing $\delta T_2$, provided that the spline space $\mathcal{S}_{d,t}$ is large enough.

Figure 7 describes the geometric setting of this numerical experiment. In Figure 7(a), the domains $\Omega_1^*$ and $\Omega_2^*$ are represented together with the isoparametric knot surfaces of $\Omega_2$. In this case there is no intersection between the bodies $\Omega_1$ and $\Omega_2$ and therefore $\Omega_2^* = \Omega_i$, $i = 1, 2$, while the interface $\Gamma_{1,2}$ is simply the intersection of the boundaries of the solids. The intersections of these surfaces with the interface $\Gamma_{1,2}$ are pulled-back in $T_i$’s parametric space and form the regions shown in Figure 7(b).

In their work [23], Lee, Lyche, and Mørken proposed a procedure to build such quasi-interpolants that is based on the use of local spline projectors, see Algorithm 3. There are two main advantages in using a Lee-Lyche-Mørken approach in this setting. On the one hand, being based on local spline projectors, only the elements of $\Omega_1$ close to the interface are to be influenced and therefore a refinement step can help us at controlling the influence that $\delta T_2$ has on $\Omega_1^*$. On the other hand, we are sure of exactly reproducing $\delta T_2$, provided that the spline space $\mathcal{S}_{d,t}$ is large enough.

Figure 7 describes the geometric setting of this numerical experiment. In Figure 7(a), the domains $\Omega_1^*$ and $\Omega_2^*$ are represented together with the isoparametric knot surfaces of $\Omega_2$. In this case there is no intersection between the bodies $\Omega_1$ and $\Omega_2$ and therefore $\Omega_2^* = \Omega_i$, $i = 1, 2$, while the interface $\Gamma_{1,2}$ is simply the intersection of the boundaries of the solids. The intersections of these surfaces with the interface $\Gamma_{1,2}$ are pulled-back in $T_i$’s parametric space and form the regions shown in Figure 7(b).

Algorithm 3 is a general procedure and allows us to choose freely both the knot interval $K_t$ in Line 3 and the local projector $P_{K_t}$ in Line 3. Since the only condition about $K_t$ is to have a non-empty intersection with the
trivariate functions over a planar domain. Therefore both integrals in Equation (13) are integrals of the boundary surface containing the local interface $\Gamma$. We remark that, thanks to Equation (3), the knot interval $\ell$ of the basis function in $\Lambda$ for any function $f$ is given by

$$L = \{ t : K_t \cap \text{supp}(B_{t,d}) \neq \emptyset \}.$$ 

We remark that, thanks to Equation (3), the knot interval $K_t$ of the basis function in $\Lambda$ is a knot interval of the boundary surface containing the local interface $\Gamma$. Therefore both integrals in Equation (13) are integrals of trivariate functions over a planar domain.

Let us denote with $L$ the cardinality of $\Lambda$ and with $i$ the index of the basis function in $\Lambda$ corresponding to $i$, for any $i$ such that $1 \leq i \leq L$. By writing explicitly

$$P_t = \sum_{i=1}^{L} \lambda_i B_{i,d},$$

Equation (13) reads

$$\sum_{i=1}^{L} \lambda_i \int_{K_t} B_{i,d} B_{t,d} = \int_{K_t} \delta T B_{t,d}, \quad l \in \Lambda_t, \quad (14)$$

for some unknown coefficients $\lambda_i$. In order to keep the notation as simple as possible, here and in the rest of this section we denote with $\delta T$ both the function $\delta T_2 \in S_{d_2}$ and the corresponding function as defined in (10). While this is formally incorrect, we think that no confusion is likely to arise as all integrations are performed in the parametric space of $T_1$.

In order to determine the $\lambda_i$s it is necessary to solve the (sparse) linear system

$$G_\ell \lambda = P,$$

where $\lambda = (\lambda_1, \ldots, \lambda_L)^T$,

$$P = \left( \int_{K_t} \delta T_2 B_{1,d}, \ldots, \int_{K_t} \delta T_2 B_{L,d} \right)^T \quad (15)$$

and $G_\ell$ is the Gibbs matrix with entries

$$(G_\ell)_{i,j} = \int_{K_t} B_{i,d} B_{j,d}, \quad i, j = 1, \ldots, L.$$ 

While the entries of the Gibbs matrix $G_\ell$ are trivial to compute numerically, as $K_t$ is an element for the spline space in which the basis functions $B_{t,d}$ are defined, the components of the vector $P$ require a special treatment. $\delta T_2$ indeed does not belong to the same spline space of the B-spline basis functions $B_{1,d}, \ldots, B_{L,d}$. We are therefore

---

**Algorithm 3 LLM quasi-interpolant**: Computes a quasi-interpolant for a given function $f$; see [23].

```
Input Degrees $d = (d_u, d_v, d_w)$;
Input Knots vectors $t = (t_u, t_v, t_w)$;
Input Function $f$;
Output $I_{LLM}$;
1: $N \leftarrow$ number of degrees of freedom in $S_{d,t}$;
2: for $\ell = 0, \ldots, N$ do
3: $K_t \leftarrow$ knot interval such that $K_t \cap \text{supp}(B_{t,d}) \neq \emptyset$;
4: $S_{d,t,K_t} \leftarrow$ restriction of $S_{d,t}$ to interval $K_t$;
5: $P_{K_t} \leftarrow$ local projector on the space $S_{d,t,K_t}$;
6: $f_{K_t} \leftarrow$ restriction of $f$ to interval $K_t$;
7: $c_\ell \leftarrow$ corresponding coefficient in $P_{K_t} f_{K_t}$;
8: end for
9: return $\sum_{\ell=0}^N c_\ell B_{t,d,t}$;
```

The support of $B_{t,d}$, in our work we simply set

$$K_t = \text{supp}(B_{t,d}),$$

while we use as the local projector $P_{K_t}$ in Line 5 the usual $L^2$ projector that satisfies

$$\int_{K_t} P_{K_t}[f] B_{t,d} = \int_{K_t} f B_{t,d}, \quad l \in \Lambda_t, \quad (13)$$

for any function $f$, where

$$\Lambda_t = \{ l : K_t \cap \text{supp}(B_{t,d}) \neq \emptyset \}.$$ 

Figure 7: Example described in Section 3.2. The original geometries $\Omega_1$ and $\Omega_2$ are shown in (a). The pull-back curves of the intersection between the isosurfaces of $\Omega_2$ and the interface $\Gamma_{1,2}$ are shown in (b). The regions to be extracted are marked with different colors.
left with the key problem of the creation of the quadrature mesh along the preimage of the interface \( \Gamma_{1,2} \). In order to apply the same strategy as in Section 3.1, the isoparametric surfaces of \( T_2 \) are extended in the three parametric directions before computing their intersection with the interface \( \Gamma_{1,2} \). The regions representing the mesh intersection of \( T_1 \) and \( T_2 \) are shown in Figure 5(b). Therefore, we proceed to write the \( i \)-th entry of \( P \) as

\[
P_i = \int_{K_i} \delta T_2 B_i d_1 = \sum_{\psi \in \Psi_i} \int_{K_i} \delta T_2 B_i d_1,
\]

where \( \Psi_i = \{ \psi \in \Psi : \psi \subset K_i \} \). In each region \( \psi \in \Psi_i \), the restriction of \( B_i d_1 \) is a polynomial of degree \( d_1 \), since \( \psi \) is guaranteed to be contained in a single knot span of \( S_{t_1, d_1} \), while \( \delta T_2 \) is a polynomial of degree \( d_2 \) on \( T_2^{-1} \circ T_1(\psi) \), as the latter is fully contained in a knot span of \( S_{t_2, d_2} \).

Figure 8 (a) shows the influence that a displacement \( \delta T_2 \) has on \( \Omega_1 \) by representing \( \delta T_1 \) and \( \delta T_2 \) as scalar fields with values between \([0, 1]\), where \( 0 \) represents no deformation and \( 1 \) represents maximal deformation. As expected, the maximal deformation of \( \Omega_1 \) is localized around the interface \( \Gamma_{1,2} \). Nevertheless, when we deform \( \Omega_1 \) and \( \Omega_2 \) according to the respective displacements, it is visible a discrepancy between the selecting deformed models, see Figure 8 (d). In order to improve this result we refine the mesh of \( T_1 \) in two different steps. In the first step we insert a new knot in the middle of each span for each knot vector of \( T_1 \) in the three parametric directions, while, in the second, we perform the same operation but only for two parametric directions, see Figures 8 (b) and (c), respectively. The results of this procedure are visible in Figures 8 (e) and (f), respectively. We notice that, already after the first refinement, the behavior of \( \delta T_1 \) reproduces much closely the distortion \( \delta T_2 \) and improves even further with the next refinement step.

### 3.3. Weak continuity with non-conforming interface

Let us now consider the situation in Figure 9. If the interface \( \Gamma_{1,2} \) is not the image of a whole face of both \( \Omega_1 \) and \( \Omega_2 \), we say that the solids are in a non-conforming geometrical setting.

Enforcing weak continuity constraints in such cases is more complicated due to the fact that the function \( \delta T_2 \) that needs to be projected can have discontinuities along the trimming curve of \( \Gamma_{1,2} \). Using the same approach as in Section 3.2 would therefore result in oscillations in the final results due to the Gibbs phenomenon. Gibbs phenomenon could be avoided by carefully choosing the element \( K_i \) in which to perform the \( L^2 \) projection but at the cost of obtaining a ‘block’, pixelized behavior.

In order to prevent this unpleasant effect, we here propose an approach that enforces weak continuity among the two solids with a mass lumping strategy.

Figure 9 shows the setting of this numerical experiment. In Figure 9 (a), the domains \( \Omega_1 \) and \( \Omega_2 \) are shown, together with the isoparametric surfaces of \( \Omega_2 \). These surfaces are intersected with the interface \( \Gamma_{1,2} \) and produce six intersection curves, whose pull-backs, together with the parametric grid inherited by \( \Gamma_{1,2} \) are shown in Figure 9 (b). In order to impose weak continuity constraints in this setting, we introduce the auxiliary function

\[
\delta T_{1,2} = \begin{cases} 
\delta T_2, & \text{inside } \hat{\Gamma}_{1,2} \\
0, & \text{otherwise},
\end{cases}
\]

and the domain

\[
\Theta = \bigcup_{i \in \Lambda} \text{supp}(B_i d_1) \subseteq \hat{\Gamma}_{1,2},
\]

where

\[
\Lambda = \{ i : \hat{\Gamma}_{1,2} \cap \text{supp}(B_i d_1) \neq \emptyset \}.
\]

The spline-based level set function \( [27] \) is then defined as

\[
\delta T_1 = \sum_{i=0}^{n_1} B_i d_1 p_i,
\]

where \( n_1 \) is the number of control points of \( M_1 \) and

\[
p_i = \int_\Theta B_i d_1 \delta T_{1,2}.
\]

There are several reasons for imposing weak continuity constraints using a spline-based level set approach. Using similar arguments as the one proposed in [27], it can be shown that \( \delta T_1 \) satisfies

\[
\int_\Theta \delta T_1 = \int_\Theta \delta T_{1,2},
\]

and therefore \( \delta T_1 \) preserves the average value of \( \delta T_{1,2} \) over \( \Theta \). Moreover it can be shown [27] that \( [17] \) is bounded from above and from below by the maximum and minimum of \( \delta T_{1,2} \) and therefore no wild oscillations due to the Gibbs phenomenon are to be expected. Finally, the computation of \( [17] \) does not necessitate the resolution of a linear system as defined in Section 3.2 and therefore the new material specification can be computed much more efficiently.

In order to compute the \( i \)-th control point we first note that the denominator in \( [18] \) can be computed using standard numerical techniques. As for the numerator instead, we write

\[
\int_\Theta B_i d_1 \delta T_{1,2} = \int_{K_i \cap \hat{\Gamma}_{1,2}} B_i d_1 \delta T_2,
\]

with \( K_i = \text{supp}(B_i d_1) \). We notice that the right hand-side integral in \( [19] \) can be computed using a similar technique as used in Section 3.2.

In order to simulate the effect of a deformation of \( \Omega_2 \), we translate \( \Omega_2 \) over one of the main directions of a con-
stant quantity. Without loss of generality we assume that
\[ \delta T_2 = (0, c, 0)^T, \quad c \in \mathbb{R}. \]

The effects of this deformation are represented in Figure 10 for the different refinements of \( T_1 \). As visible in Figure 10(d), for a very coarse mesh of \( \Omega_1 \) the convolution-based strategy does not approximate well enough the deformation of \( \Omega_2 \). This is due to the fact that (17) reproduces the average of the distortion of \( \delta T_2 \) in \( \Theta \). From (18) it is clear that, in order to get a better approximation of \( \delta T_2 \), the support of the basis functions \( K_i \) should be small enough, so to guarantee that the average of \( \delta T_2 \) on \( K_i \) is a good approximation of the behavior of \( \delta T_2 \) over the same domain. In order to improve this result we refine the mesh of \( T_1 \) in two different steps. In each step we insert a new knot in the middle of each span for each knot vector of \( T_1 \) in two of the three parametric directions. This is equivalent to splitting each element of \( T_1 \) into four sub-elements. The results of these refinement steps on the behavior of \( \delta T_2 \) are visible in Figures 10(e) and (f). Notice that, as the mesh gets finer, the influence of the distortion gets more localized around the trimming curve of the interface \( \Gamma_{1,2} \), as well. Figures 10(a)–(c) show instead the influence of the distortion of \( \Omega_2 \) over the first body, showing that only the elements that are closer to the interface are in practice affected by this procedure.

### 3.4. Poisson’s problem

In this section we use our algorithm to solve the Poisson’s equation in a contact problem context. Let us consider the geometric setting as in Figure 11(a). The two boxes \( \Omega_1 \) and \( \Omega_2 \) touch along the interface but do not intersect and are parameterized by two trivariate B-splines \( T_1 \in \mathbb{S}_{d_1} \) and \( T_2 \in \mathbb{S}_{d_2} \), where we assume the degrees in \( d_1 \) to satisfy \( d_1^{(\alpha)} \geq 2, \alpha = u, v, w \). In this geometric setting, the strong form of the Poisson’s problem can be stated as follows. Given a domain \( \Omega = \Omega_1 \cup \Omega_2 \), find \( u : \Omega \rightarrow \mathbb{R} \) such that

\[
-\Delta u_i = f \quad \text{in } \Omega_i, \quad i = 1, 2, \\
\partial_n u_1 + \partial_n u_2 = 0 \quad \text{on } \Gamma_{1,2}, \\
u_1 - u_2 = 0 \quad \text{on } \Gamma_{1,2}, \\
u = 0 \quad \text{on } \Gamma_D, \\
\nabla u \cdot n = 0 \quad \text{on } \Gamma_N, \\
\]

\[
(20)
\]
where $u_i = u_i|_{\Omega_i}$, $i = 1, 2$, $f: \Omega \to \mathbb{R}$ and $n$ is the outward normal of $\partial\Omega$. We assume that
\[
\Gamma_D \cup \Gamma_N = \partial\Omega
\]
\[
\Gamma_D \cap \Gamma_N = \emptyset
\]
The definition of $\Gamma_D$ and $\Gamma_N$ for the considered volumetric model is shown in Figure 11 (c) for a horizontal cross section of $\Omega$. The upper and lower faces of both $\Omega_1$ and $\Omega_2$ are considered as Neumann boundaries.

In order to present the weak formulation of (20), we first introduce the function spaces
\[
V_h = \{v_{1,h} \in S_{d_1,t_1}, v_{1,h}|_{\Gamma_D \cap \partial\Omega_1} = 0\} \oplus \{v_{2,h} \in S_{d_2,t_2}, v_{2,h}|_{\Gamma_D \cap \partial\Omega_2} = 0\}
\]
and
\[
\Lambda_h = \{\lambda_h \in tr(S_{d_1-2,t_1'})\},
\]
where we use the notation $d - k = (d_u - k, d_v - k, d_w - k)$ and we denote with $tr(\cdot)$ the trace operator over $\Gamma_{1,2}$ and with $t_1'$ a knot vector obtained from $t_1$ by removing the first and the last two knots in each parametric direction. The function space $\Lambda_h$ is where the Lagrange multipliers for the resolution of the Poisson problem are going to be defined. This particular choice of $\Lambda_h$ guarantees the inf-sup stability of the Lagrange multipliers [16] and is the reason for the hypothesis on the degrees of the trivariate $T_1$.

Hence, the discrete, weak form of Poisson’s problem can be stated as follows. Find $u_h \in V_h$ and $\lambda_h \in \Lambda_h$ such that
\[
a(u_h, v_h) + b(u_h, \lambda_h) = f(v_h) \quad v_h \in V_h
\]
\[
b(u_h, \mu_h) = 0 \quad \mu_h \in \Lambda_h
\]
where
\[
a(u, v) = \int_\Omega \nabla u \cdot \nabla v,
\]
\[
b(v, \mu) = \int_{\Gamma_{1,2}} [v]\mu,
\]
\[f(v) = \int_\Omega fv,
\]
and $[u] = u_1 - u_2$ is the usual jump operator from $\Omega_1$ to $\Omega_2$ over $\Gamma_{1,2}$.

Denoting with $u_i,h = u_i|_{\Omega_i}$, $i = 1, 2$, $b(u,h, \mu_h)$ reads
\[
b(u_h, \mu_h) = \int_{\Gamma_{1,2}} \mu_h u_{1,h} - \int_{\Gamma_{1,2}} \mu_h u_{2,h},
\]
and therefore, it appears clear that, in order to obtain a good approximation for $b(u_h, \mu_h)$ it is needed to compute the mesh intersection between the meshes inherited by $\mu_h$ and $u_{2,h}$.

To this end, we use our region extraction algorithm in order to identify the regions visible in Figure 11 (b). In order to visualize the solution of the Poisson’s problem,
we associate to \( \Omega_2 \) the sinusoidal source

\[
f(x, y, z) = \left( \frac{\pi}{10} \right)^6 \sin \left( \frac{\pi}{10} x \right)
\]

and we enforce equality over the interface \( \Gamma_{1,2} \) with the Lagrange multipliers method. Figure 12 shows \( u_1 \) and \( u_2 \) according to three different refinements of the mesh of \( \Omega_1 \).

4. Conclusion

In this work we have presented a novel region extraction algorithm for curvilinear drawings that allows to easily identify the regions bounded by a set of planar curves. The algorithm has shown to be a powerful tool to precisely compute integrals over the interfaces of solids of piecewise polynomials defined in different meshes. In the literature, this kind of integrals are computed by simply approximating the regions in which the splines are represented by polynomials. These approximations polluted the quality of the integration and, in all our numerical tests, resulted in a lower precision than the one shown in Section 3.1. The precise computation of integrals of this type come in handy in many practical operations.

Three of such operations have been discussed in Section 3. In the context of weak continuity enforcement, different results can be achieved using different projectors for the Lee-Lyche-Mørken quasi-interpolant or a different quasi-interpolant altogether. Different choices could be, for example, the use of the standard \( L^2 \) projector or yet the use of the projector proposed in [22]. Both of these choices would take advantage of our algorithm for a precise computation of the integrals. Other choices of the local knot interval \( K_\ell \) different than the one taken in Section 3.2 can also influence the behavior of the result. Different approaches can be taken according to the needs of the specific application (locality, reproduction of the method etc.) and it is therefore our idea that the best results can be achieved only using application tailored methods.

Among the applications discussed, the enforcement of the weak continuity is probably the more interesting from a geometric modeling point of view. In a recent work [28], Masalha and colleagues proposed a procedure for creating heterogeneous parametric trivariate fillets. Despite the interesting geometric algorithms proposed, the obtained fillets are not connected to the original objects and therefore new fillets need to be created every time a deformation applies to the input objects. The weak continuity constraints proposed in this work can overcome this issue, creating a unique multipatch geometry for the entire volumetric model.

Among the possible applications of Algorithm 1 that have not been presented in this work, there are several...
ones that are more tailored to the context of IGA, such as mortar methods and contact problems. These are promising research areas that we are currently investigating and will be the arguments of forthcoming works.

5. Acknowledgment

P. Antolin and A. Buffa are partially supported by the ERC AdG project CHANGE n. 694515. P. Antolin, A. Buffa and E. Cirillo are partially supported by the SNSF through the project “Design-through-Analysis (of PDEs): the litmus test” n. 40B2-0 187094 (BRIDGE Discovery 2019). These supports are gratefully acknowledged.

References

[1] X. Jiang, H. Bunke, An optimal algorithm for extracting the regions of a plane graph, Pattern Recognit. Lett. 14 (1993) 553–558.
[2] Z. Shih, A systolic algorithm for extracting regions from a planar graph, Comput. Vis. Graph. Image Process. 47 (1989) 227–242.
[3] K.-C. Fan, C.-Y. Chang, Surface extraction from line drawings of a polyhedron, Pattern Recognit. Lett. 12 (1991).
[4] R. D. Dutton, R. C. Brigham, Efficiently identifying the faces of a solid, Comput. Graph. 7 (1983) 143–147.
[5] J. A. Brewer, S. M. Courter, Automated conversion of curvilinear wire-frame models to surface boundary models; a topological approach, in: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH ’86, Association for Computing Machinery, New York, NY, USA, 1986, p. 171–178.
[6] M. Kuo, Automatic extraction of quadric surfaces from wire-frame models, Comput. Graph. 25 (2001) 109–119.
[7] M. J. Gander, C. Japhet, Algorithm 932: Pang: Software for nonmatching grid projections in 2d and 3d with linear complexity, ACM Trans. Math. Softw. 40 (2013).
[8] P. Lee, C.-H. Yang, J.-R. Yang, Fast algorithms for computing self-avoiding walks and mesh intersections over unstructured meshes, Adv. Eng. Softw. 35 (2004) 61–73.
[9] R. Löhner, Some useful data structures for the generation of unstructured grids, Commun. Appl. Numer. Meth. 4 (1988) 123–135.
[10] R. Löhner, Adaptive remeshing for transient problems, Comput. Methods. Appl. Mech. Engrg. 75 (1989) 195–214.
[11] S. J. Plimpton, B. Hendrickson, J. R. Stewart, A parallel rendezvous algorithm for interpolation between multiple grids, J. Parallel Distrib. Comput. 64 (2004) 266–276.
[12] A. Seitz, P. Farah, J. Kremheller, B. I. Wohlmuth, W. A. Wall,
A. Popp, Isogeometric dual mortar methods for computational contact mechanics, Comput. Methods. Appl. Mech. Engrg. 301 (2016) 259–280.

[13] C. Hesch, P. Betsch, Isogeometric analysis and domain decomposition methods, Comput. Methods. Appl. Mech. Engrg. 213-216 (2012) 104–112.

[14] L. Heffter, Ueber das problem der nachbargebiete, Math. Ann. 38 (1891).

[15] P. Antolin, A. Buffa, R. Puppi, X. Wei, Overlapping multi-patch isogeometric method with minimal stabilization, SIAM J. Comput. 43 (2021) A330 – A354.

[16] E. Brivadis, A. Buffa, B. Wohlmuth, L. Wunderlich, Isogeometric mortar methods, Comput. Methods. Appl. Mech. Engrg, 284 (2015) 292–319.

[17] X. Wei, B. Marussig, P. Antolin, A. Buffa, Immersed boundary-conformal isogeometric method for linear elliptic problems, Comput. Mech. 68 (2021) 1385–1405.

[18] Open Cascade - software development company, OCCT surface and solid modeling tool, 2021. URL: https://www.opencascade.com/, accessed: 2021-10-01.

[19] G. Elber, Irit 12 user’s manual, 2021. URL: https://www.cs.technion.ac.il/~irit/, accessed: 2021-10-01.

[20] P. Antolin, A. Buffa, M. Martinelli, Isogeometric analysis on v-reps: first results, Comput. Methods. Appl. Mech. Engrg. 355 (2019) 976–1002.

[21] P. Antolin, T. Hirschler, Quadrature-free immersed isogeometric analysis, 2021. arXiv:2107.09024.

[22] Z. Zou, M. A. Scott, M. J. Borden, D. C. Thomas, W. Dornisch, E. Brivadis, Isogeometric Bézier dual mortaring: Refinable higher-order spline dual bases and weakly continuous geometry, Comput. Methods. Appl. Mech. Engrg. 333 (2018) 497–534.

[23] B.-G. Lee, T. Lyche, K. Mørken, Some examples of quasi-interpolants constructed from local spline projectors, in: Mathematical Methods in CAGD: Oslo 2000, Vanderbilt, University Press, 2000, pp. 243–252.

[24] J. W. Gibbs, Fourier’s series, Nature 59 (1898) 200–200.

[25] J. W. Gibbs, Fourier’s series [3], Nature 59 (1899) 606–606.

[26] H. Wilbraham, On a certain periodic function, Cambridge and Dublin Math. J. 3 (1848) 198–201.

[27] C. Verhoosel, G. van Zwieten, B. van Rietbergen, R. de Borst, Image-based goal-oriented adaptive isogeometric analysis with application to the micro-mechanical modeling of trabecular bone, Comput. Methods. Appl. Mech. Engrg. 284 (2015) 138–164.

[28] R. Masalha, E. Cirillo, G. Elber, Heterogeneous parametric trivariate fillets, Comput. Aided Geom. Des. 86 (2021).