Robust numerical integration on curved polyhedra based on folded decompositions

Pablo Antolin[a],[*], Xiaodong Wei[b], Annalisa Buffa[a],[c]

[a] *Institute of Mathematics, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland*
[b] *University of Michigan-Shanghai Jiao Tong University Joint Institute, Shanghai Jiao Tong University, Shanghai, China*
[c] *Istituto di Matematica Applicata e Tecnologie Informatiche "Enrico Magenes" del CNR, Pavia, Italy*

Abstract

We present a novel method to perform numerical integration over curved polyhedra enclosed by high-order parametric surfaces. Such a polyhedron is first decomposed into a set of triangular and/or rectangular pyramids, whose certain faces correspond to the given parametric surfaces. Each pyramid serves as an integration cell with a geometric mapping from a standard parent domain (e.g., a unit cube), where the tensor-product Gauss quadrature is adopted. As no constraint is imposed on the decomposition, certain resulting pyramids may intersect with themselves, and thus their geometric mappings may present negative Jacobian values. We call such cells *folded cells* and refer to the corresponding decomposition as a folded decomposition. We show that folded cells do not cause any issues in practice as they are only used to numerically compute certain integrals of interest. The same idea can be applied to planar curved polygons as well. We demonstrate both theoretically and numerically that folded cells can retain the same accuracy as the cells with strictly positive Jacobians. On the other hand, folded cells allow for a much easier and much more flexible decomposition for general curved polyhedra, on which one can robustly compute integrals. In the end, we show that folded cells can flexibly and robustly accommodate real-world complex geometries by presenting several examples in the context of immersed isogeometric analysis, where involved sharp features can be well respected in generating integration cells.
© 2022 Elsevier B.V. All rights reserved.

Keywords: Numerical integration; Curved polyhedra; Negative-Jacobian cells; Isogeometric analysis; Trimmed volumes; Complex geometric features

1. Introduction

Accurate, efficient, and robust numerical integration methods over curved polygons/polyhedra are required in various applications across computer-aided engineering (CAE), computer-aided design (CAD) [1], and computer graphics and animation [2]. Particularly in CAE, novel integration schemes are often needed when the mesh for approximating solutions does not conform to the geometric representation, such as extended finite element methods [3], cut finite element methods (cutFEM) [4], virtual element methods [5], finite cell methods (FCM) [6], discontinuous-Galerkin-based immersed boundary methods [7], and immersed isogeometric analysis (IGA) [8]. The

---

key idea of such non-conformal methods is to decouple geometric representation from the discretization for solution approximation to significantly simplify mesh generation. For instances, in FCM and immersed IGA, it involves embedding the geometric representation into a simple background grid (e.g., a Cartesian grid), leading to cut (or trimmed) elements of complex shapes. Properly dealing with such cut elements poses several challenges such as numerical integration, stabilization regardless of cutting [9–12], imposition of the Dirichlet boundary conditions [13], and so forth. We focus on the numerical integration in this paper.

There exist several major groups of methods to perform numerical integration over curved polygons/polyhedra, such as (1) application of the divergence theorem to reduce the dimension of integrals, (2) moment fitting, and (3) domain decomposition into simplices. First, the divergence theorem can be applied to convert an integral to its low-dimension counterpart [14–18]. For instance, a volume integral can be reduced to a surface integral, and even to a point evaluation [16,18]. To be able to apply the divergence theorem, it is critical to find a suitable antiderivative for the integrand. Accordingly, one needs to either assume integrands to be in a certain form (e.g., homogeneous functions) to have analytic antiderivatives [16] or employ certain numerical methods to obtain approximations [17].

Second, moment fitting generally solves a local system of equations to find a quadrature rule (i.e., the locations and/or weights of quadrature points) that can exactly integrate certain polynomial basis functions [19–24]. When point locations are taken as a priori, the system becomes linear and the unknowns are only weights. Recently, moment fitting was further explored in nonlinear problems [25–27]. The advantage of moment fitting is to retain the minimum number of integration points via solving a linear system for each cut element. In fact, it needs to be used together with other groups of methods to accurately compute the right-hand-side vectors. In other words, moment fitting moves the computation of integrals over complex domains to the preprocessing step such that later it can be used in a fast way.

Third, decomposition-based methods represent a popular way of computing integrals over complex domains. The basic idea is to partition a given domain into manageable "cells" where standard quadrature can be applied. Such methods can be divided into low-order and high-order ones. A typical example of low-order decomposition is adaptive quad/octree subdivision widely adopted in FCM [28], where a cut element is adaptively subdivided into sub-cells to capture the domain boundary for integration. Essentially, it is merely a piecewise constant approximation to the given geometry. Such a low-order approximation leads to a large number of quadrature points aggregated around the boundary and thus renders it computationally expensive especially in nonlinear problems or in 3D [29–31]. Alternatively, the low-order approximation can be used together with a Taylor series correction to achieve high-order accuracy [32,33]. On the other hand, high-order decomposition features a high-order approximation to the geometry of interest, and thus it yields much fewer quadrature cells than the adaptive subdivision; see [34–38] for several methods that work in 3D. Due to its high accuracy per quadrature point, it is particularly suitable for high-order methods such as immersed IGA. However, a high-order decomposition generally involves a high-order triangulation [37] or other sophisticated partitions [34]. The procedure can be complicated and time-consuming especially when the resulting integration cells are required to have positive Jacobian. However, we will show that the positive-Jacobian constraint is in fact not necessary for the integration purpose.

In this paper, we propose a novel decomposition-based method that allows also for integration cells that present negative Jacobian values. We call such cells *folded cells* and refer to the corresponding decomposition as a *folded decomposition*. In other words, no constraint is imposed to the Jacobians of the integration cells when we create a decomposition for a given domain. Allowing for folded cells greatly increases the flexibility and guarantees that the decomposition is always successful. The method works for both low-order and high-order decompositions. We show that the performance of folded decompositions is comparable to that of strictly positive-Jacobian cells in terms of integration accuracy and convergence. Moreover, when applied to solving linear elliptic partial differential equations, our strategy yields expected error convergence rates, and due to its intrinsic flexibility, it can robustly accommodate real-world complex geometries. While we focus applications and present examples in immersed IGA in this paper, the proposed integration scheme can be applied to other methods such as virtual element methods [5], finite cell methods [6], discontinuous-Galerkin-based immersed boundary/interface methods [7,39], and the NURBS-Enhanced finite element methods [40]. We note that an independent work [41] recently reported a related idea but it is limited to 2D domains and only involves *negative-Jacobian cells* (non-positive Jacobian everywhere), a special case of folded cells. Its numerical tests also validate the accuracy and efficiency of this particular type of cells on 2D convex and non-convex domains with affine and curved boundaries.

The remainder of the paper is organized as follows. Section 2 presents the problem setting. Next in Sections 3 and 4, we discuss the folded decomposition for both curved polygons and curved polyhedra. Section 5 discusses how to

apply the proposed method to immersed IGA. Numerical examples are then presented in Sections 6 and 7, where Section 6 is devoted to demonstrating the efficacy and robustness of folded decomposition, whereas Section 7 shows that folded decomposition can robustly accommodate real-world complex geometries in engineering simulations. We conclude our work in Section 8.

2. Problem setting: Integration over general domains

We aim to find an accurate and robust quadrature rule to approximate the integral of a certain function $f(\mathbf{x})$ over a compact domain $\Omega \subset \mathbb{R}^d$ ($d = 2, 3$, the spatial dimension),

$$\int_{\Omega} f(\mathbf{x}) \, d\mathbf{x}. \tag{1}$$

In finite element analysis (FEA), $f(\mathbf{x})$ usually represents quantities contributing to mass or stiffness matrices. The challenge of computing (1) originates from the complexity of the domain Ω.

In this paper, Ω is assumed to be a curved polygonal domain in 2D or a curved polyhedral domain in 3D with an orientable boundary, and it is sufficient for Ω to only have a boundary representation (B-rep). More precisely, a curved polygon or polyhedron is a domain $\Omega \subset \mathbb{R}^d$ ($d = 2$ for a plane or $d = 3$ a volume) such that its boundary $\partial \Omega$ is represented by the union of a set of parameterized edges (for $d = 2$), $\{\mathbf{C}_i(u) \in \mathbb{R}^2, u \in [0, 1], i = 1, \ldots, N_e\}$, or a set of parameterized faces (for $d = 3$), $\{\mathbf{S}_i(u, v) \in \mathbb{R}^3, (u, v) \in \mathcal{D}_i \subseteq [0, 1]^2, i = 1, \ldots, N_f\}$. Note that $\mathcal{D}_i$, the definition domain of $\mathbf{S}_i(u, v)$, generally is a subset of the entire parametric domain $[0, 1]^2$.

The aim of this paper is to design robust and cheap approaches to compute Eq. (1) for a known regular function $f : \mathbb{R}^d \to \mathbb{R}$. For the sake of clarity, let us introduce a few terms we use in this paper.

Definition 1 (*Cell*). We call a compact set $\mathcal{T} \subset \mathbb{R}^d$ a *cell* if it is defined by a mapping $\mathbf{T} : [0, 1]^d \to \mathcal{T}$, i.e., $\mathbf{x} = \mathbf{T}(\mathbf{u})$, such that its Jacobian, $\mathrm{Jac}_{\mathcal{T}}(\mathbf{x}) \equiv \det(\nabla \mathbf{T}(\mathbf{u}))$, is well defined everywhere. The interior and the boundary of $\mathcal{T}$ are denoted as $\mathring{\mathcal{T}}$ and $\partial \mathcal{T}$, respectively, and clearly $\mathcal{T} = \mathring{\mathcal{T}} \cup \partial \mathcal{T}$.

Definition 2 (*J^+ Cell*). A cell $\mathcal{T}$ is called a *J^+ cell* if it has $\mathrm{Jac}_{\mathcal{T}} \geq 0$ everywhere and $\max(\mathrm{Jac}_{\mathcal{T}}) > 0$.

Definition 3 (*J^+ Decomposition*). A set of non-overlapping cells, $\{\mathcal{T}_\ell\}_{\ell=1}^L$ where $\mathring{\mathcal{T}}_\ell \cap \mathring{\mathcal{T}}_{\ell'} = \varnothing$ for any $\ell \neq \ell'$, is called a *J^+ decomposition* of Ω if $\Omega = \cup_{\ell=1}^L \mathcal{T}_\ell$ and every $\mathcal{T}_\ell$ is a *J^+ cell*.

With a J^+ decomposition, Eq. (1) can be computed as

$$\int_{\Omega} f(\mathbf{x}) d\mathbf{x} = \sum_{\ell=1}^L \int_{\mathcal{T}_\ell} f(\mathbf{x}) d\mathbf{x} = \sum_{\ell=1}^L \int_{[0,1]^d} f(\mathbf{T}_\ell(\mathbf{u})) \det(\nabla \mathbf{T}_\ell(\mathbf{u})) d\mathbf{u} \approx \sum_{\ell=1}^L \sum_{q=1}^{N_g} f(\mathbf{T}_\ell(\mathbf{u}_q^g)) \det(\nabla \mathbf{T}_\ell(\mathbf{u}_q^g)) \, w_q^g, \tag{2}$$

where N_g is the number of quadrature points, $\mathbf{u}_q^g$ and w_q^g are tensor-product Gauss quadrature points and weights, respectively, defined in $[0, 1]^d$. Letting

$$\mathbf{x}_q^\ell := \mathbf{T}_\ell(\mathbf{u}_q^g) \quad \text{and} \quad w_q^\ell := \det(\nabla \mathbf{T}_\ell(\mathbf{u}_q^g)) \, w_q^g, \tag{3}$$

we obtain the typical quadrature formula for a decomposition-based method,

$$\int_{\Omega} f(\mathbf{x}) d\mathbf{x} \approx \sum_{\ell=1}^L \sum_{q=1}^{N_g} f(\mathbf{x}_q^\ell) \, w_q^\ell, \tag{4}$$

where we notice that all the weights w_q^ℓ are strictly positive. While it is possible to use different numbers of quadrature points in different cells, for the sake of notation simplicity, we adopt the same N_g for all cells throughout the paper.

In this paper, we aim at generalizing this formula to accommodate more general decompositions by allowing for "folded" cells, i.e., cells that may present negative Jacobian values. Equivalently speaking, the weights w_q^ℓ in Eq. (4) may be negative. More specifically, we introduce the following definitions to facilitate our discussion.
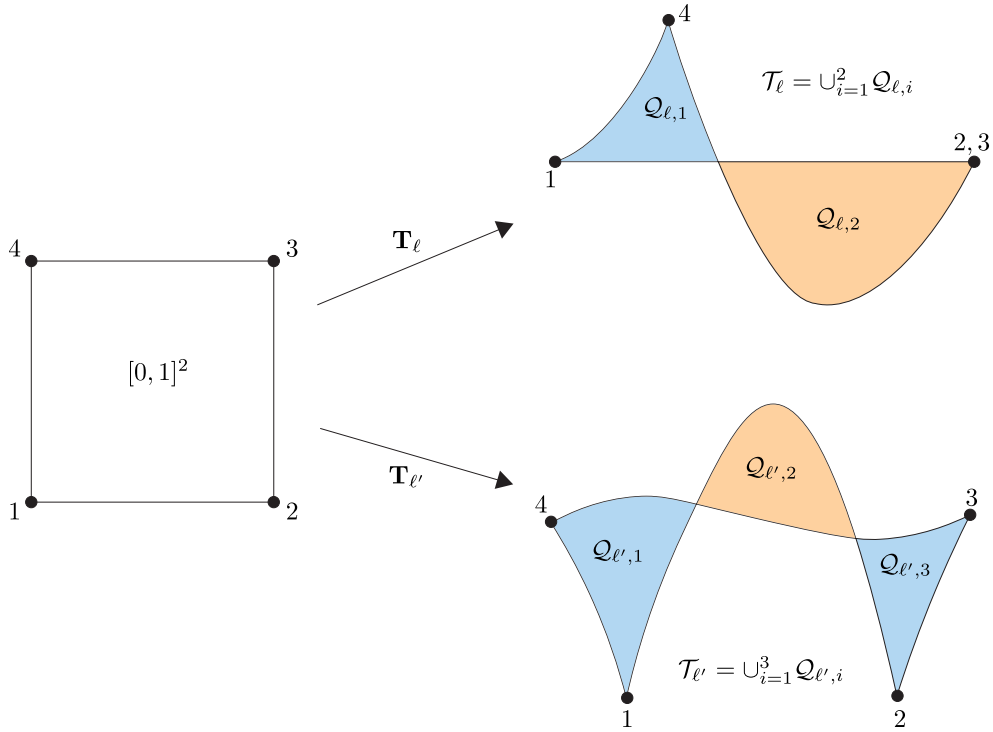
Fig. 1. Split of folded cells into subsets, inside which the Jacobian does not change sign. The blue and orange regions represent J^+ and J^- cells, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Definition 4 (J^- *Cell*). A cell $\mathcal{T}$ is called a J^- cell if it has $\mathrm{Jac}_{\mathcal{T}} \leq 0$ everywhere and $\min(\mathrm{Jac}_{\mathcal{T}}) < 0$.

Definition 5 (*Folded Cell*). We say that a cell $\mathcal{T}$ is a *folded cell* if it has $\min(\mathrm{Jac}_{\mathcal{T}}) < 0$ in $\mathring{\mathcal{T}}$, but $\mathcal{T}$ can be split into a finite number of subdomains, $\mathcal{T} = \cup_{i=1}^{N} \mathcal{Q}_i$, such that every $\mathcal{Q}_i$ is either a J^+ cell or a J^- cell.

According to Definitions 4 and 5, we note that J^- cells are a special case of folded cells. A few examples of folded cells are shown in Fig. 1.

Definition 6 (*Folded Decomposition*). A set of overlapping cells $\{\mathcal{T}_\ell\}_{\ell=1}^L$ (where there exists $\ell \neq \ell'$ such that $\mathring{\mathcal{T}}_\ell \cap \mathring{\mathcal{T}}_{\ell'} \neq \varnothing$) is called a *folded decomposition* of Ω if $\Omega = \sqcup_{\ell=1}^L \overline{\mathring{\mathcal{T}}_\ell}$ and there exists at least one cell that is a folded cell, where the upper bar denotes the closure of a domain,

$$\sqcup_{\ell=1}^L \mathring{\mathcal{T}}_\ell := \{\mathbf{x} \in \cup_{\ell=1}^L \mathring{\mathcal{T}}_\ell : \quad \sum_{\ell=1}^L \mathrm{Count}_{\mathcal{T}_\ell}(\mathbf{x}) = 1\}, \tag{5}$$

and

$$\mathrm{Count}_{\mathcal{T}_\ell}(\mathbf{x}) := \begin{cases} 1 & \text{if } \mathbf{x} \in \mathring{\mathcal{T}}_\ell \text{ and } \mathrm{Jac}_{\mathcal{T}_\ell}(\mathbf{x}) > 0, \\ -1 & \text{if } \mathbf{x} \in \mathring{\mathcal{T}}_\ell \text{ and } \mathrm{Jac}_{\mathcal{T}_\ell}(\mathbf{x}) < 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Eqs. (5) and (6) in Definition 6 essentially collect the points that only constitute the regions of "net" positive Jacobians. For example, if a point $\mathbf{x}$ only belongs to a J^+ cell, then $\sum_{\ell=1}^L \mathrm{Count}_{\mathcal{T}_\ell}(\mathbf{x}) = 1$ holds and thus $\mathbf{x} \in \sqcup_{\ell=1}^L \mathring{\mathcal{T}}_\ell$. On the other hand, if a point $\mathbf{x}$ belongs to two cells, $\mathcal{T}_\ell$ and $\mathcal{T}_{\ell'}$, and moreover $\mathrm{Jac}_{\mathcal{T}_\ell}(\mathbf{x}) > 0$ and $\mathrm{Jac}_{\mathcal{T}_{\ell'}}(\mathbf{x}) < 0$, then
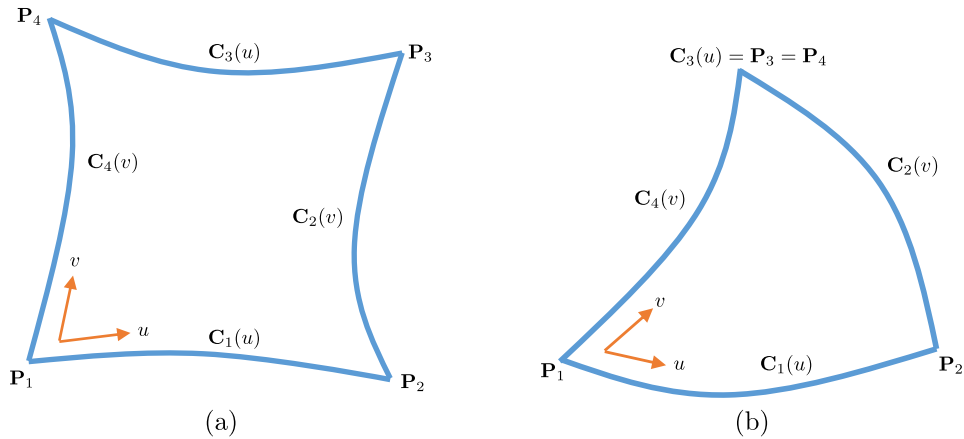
Fig. 2. Boundary curves and corners of Coons patches, where $\mathbf{C}_3(u)$ in (b) is degenerated.

$\sum_{\ell=1}^{L} \text{Count}_{\mathcal{T}_\ell}(\mathbf{x}) = 0$ and thus $\mathbf{x} \notin \sqcup_{\ell=1}^{L} \mathring{\mathcal{T}}_\ell$. In other words, the contributions from the positive- and negative-Jacobian regions are canceled out at this same point $\mathbf{x}$. We will discuss this mechanism in detail in the next section.

In what follows, we provide an efficient algorithm to compute possibly folded decomposition that never fails. We start our discussion for $d = 2$ in Section 3 and then we move to $d = 3$ in Section 4.

3. Folded decompositions for $d = 2$

In this section, we introduce two approaches to split 2D domains to generate folded decompositions.

3.1. Quadrilateral-dominant decomposition

The quadrilateral-dominant decomposition recursively splits a given polygon into quadrilaterals and/or triangles, which provides a means to minimize the number of resulting cells. Clearly, no decomposition is needed when a polygon is a quadrilateral or a triangle. When a polygon is a quadrilateral, a Coons patch $\mathbf{T}(u, v)$ [42] is created to interpolate its 4 edges,

$$\mathbf{T}(u, v) = (1 - v)\mathbf{C}_1(u) + v\mathbf{C}_3(u) + (1 - u)\mathbf{C}_4(v) + u\mathbf{C}_2(v)$$
$$- (1 - u)(1 - v)\mathbf{P}_1 - u(1 - v)\mathbf{P}_2 - uv\mathbf{P}_3 - (1 - u)v\mathbf{P}_4, \tag{7}$$

where $(u, v) \in [0, 1]^2$, and $\mathbf{C}_i : [0, 1] \to \mathbb{R}^2$ and $\mathbf{P}_i \in \mathbb{R}^2$ $(i = 1, \ldots, 4)$ are the corresponding curves and corners, respectively; see Fig. 2(a). When a polygon is a triangle, a Coons patch can still be constructed with a degenerated curve (i.e., a zero-length curve); see Fig. 2(b). Such (degenerate) Coons patches serve as cells for integration.

On the other hand, when a polygon has more than 4 edges (e.g., Fig. 3(a)), we split the polygon into two sub-polygons, one with 4 edges and the other with $n - 2$ edges, where n is the number of edges of the given polygon. Two new edges are added due to splitting; see $\mathbf{P}_4\mathbf{P}_1$ and $\mathbf{P}_1'\mathbf{P}_4'$ in Fig. 3(b). Any splitting is valid as long as a quadrilateral is obtained. We repeat this procedure for each resulting sub-polygon with more than 4 edges; see Fig. 3(c). As a result, all resulting sub-polygons are either quadrilaterals or triangles, and a Coons patch is constructed accordingly for each of them as a cell. This way, the polygonal domain is decomposed into the cells $\mathcal{T}_\ell$ $(\ell = 1, \ldots, L)$ whose associated mapping $\mathbf{T}_\ell : [0, 1]^2 \to \mathcal{T}_\ell$ is a (degenerate) Coons patch.

Edges of a curved polygonal domain generally correspond to high-order curves, and thus, an arbitrary decomposition may lead to folded cells. For example, the decomposition in Fig. 4 is deemed valid, where, however, the splitting edge (dashed) intersects a polygonal edge and thus one of the resulting cells is a folded cell that has negative Jacobian in the orange region. We will show that such cells are valid cells for the integration purpose.
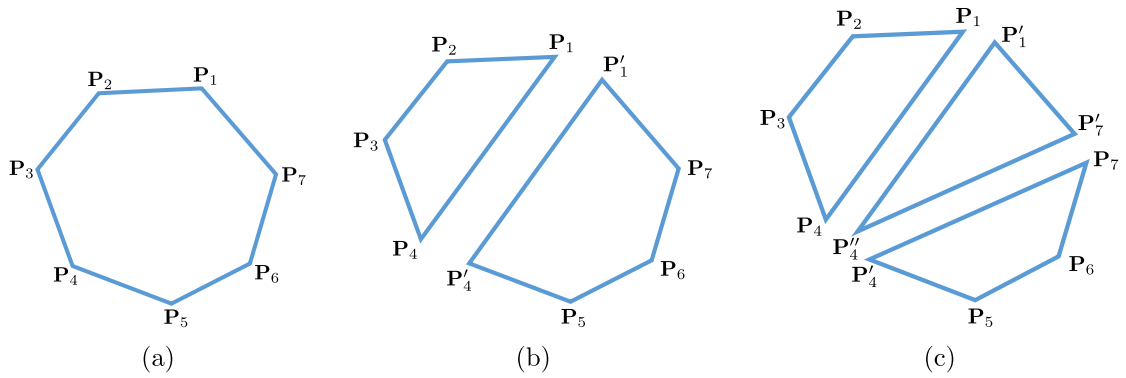
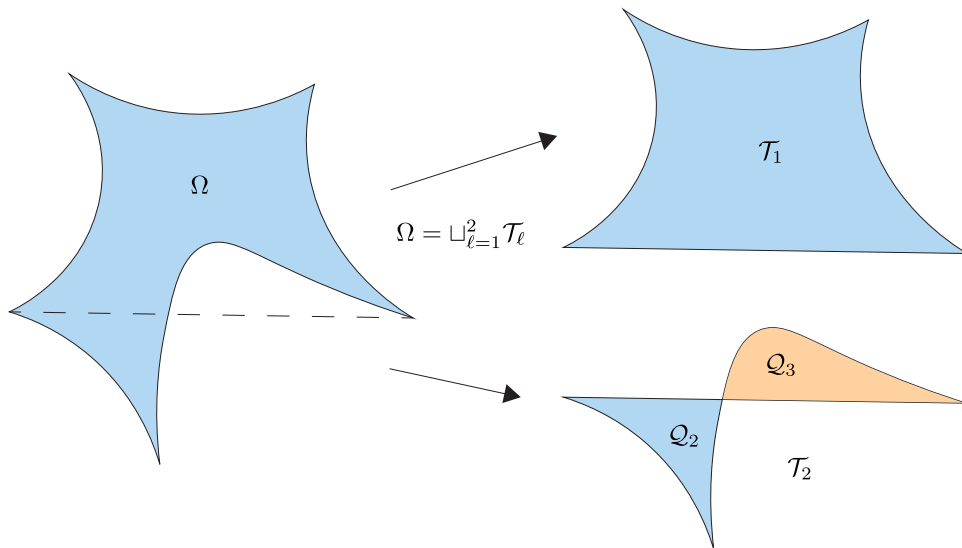**Fig. 3.** Recursively splitting a polygon into quadrilaterals and triangles.

Fig. 4. Example of a folded decomposition such that Ω is split into $\{\mathcal{T}_1, \mathcal{T}_2\}$, where $\mathcal{T}_1$ is a J^+ cell and $\mathcal{T}_2$ is a folded cell. $\mathcal{T}_2$ can be further split into $\{\mathcal{Q}_2, \mathcal{Q}_3\}$, where $\mathcal{Q}_2$ is a J^+ cell and $\mathcal{Q}_3$ is a J^- cell. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.2. Triangulation

We create 2D triangulations to randomly generate folded decompositions to test the robustness of our proposed method. In this paper, every triangle is treated as a Coons patch for simplicity. In principle, any triangulation is allowed for a given polygon whose edges are counterclockwise oriented. More specifically, a seed vertex $\mathbf{V}$ is first arbitrarily chosen in the bounding box of the polygon. A triangle is then created with every polygonal edge and $\mathbf{V}$.

$\mathbf{V}$ can be inside or outside the polygon; see the red vertices in Fig. 5(a, b). When $\mathbf{V}$ is inside, the generated decomposition may or may not have negative Jacobian, depending on whether or not a splitting edge intersects with any of the polygonal edges. However, when $\mathbf{V}$ is outside, some of the resulting cells must have negative Jacobian; see Fig. 5(b).

Among all the choices of $\mathbf{V}$, a straightforward and sometimes efficient way is to pick a vertex of the polygon; see Fig. 5(c). When an edge that shares this vertex is simply a straight segment, the resulting triangle has a zero area and thus will have no contribution to integration. In other words, fewer cells are generated in this case; compare Fig. 5(c) with Fig. 5(a, b).
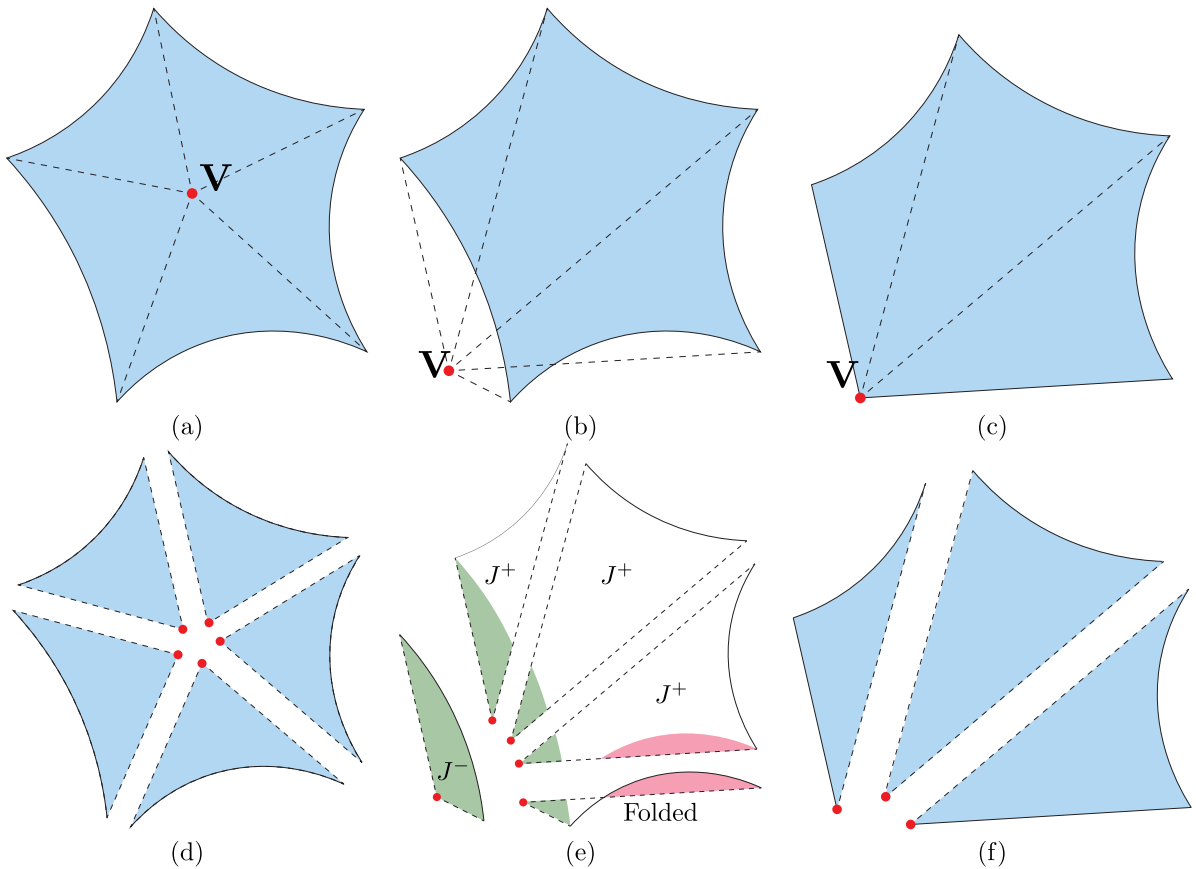
Fig. 5. Triangulations of curved polygons with different seed vertices **V**. (a) **V** inside the polygon, (b) **V** outside the polygon, (c) **V** being a vertex of the polygon, and (d–f) exploded views of (a–c). In (e), the regions of the same color overlap with one another. The boundary curves of the polygons are oriented counterclockwise. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Remark 1. While up to now we have assumed that the domain of interest is a planar domain, i.e., $\Omega \subset \mathbb{R}^2$, the proposed method also works for 2-manifold as long as it has a parametric representation. More specifically, Ω can be a parameterized surface of the form, $\mathbf{S} : \mathcal{D} \subseteq [0, 1]^2 \to \mathbb{R}^3$. When this is the case, the proposed decomposition is applied to $\mathcal{D}$ rather than to Ω. As a result, $\mathcal{D}$ is decomposed into (possibly folded) cells $\mathcal{T}_\ell$ ($\ell = 1, \ldots, L$), whose associated mappings $\mathbf{T}_\ell : [0, 1]^2 \to \mathcal{T}_\ell$ are (degenerate) Coons patches.

3.3. Theoretical study on folded decompositions

In this section, we start with an intuitive explanation to motivate how folded decompositions work for integration, and then we provide a formal argument.

We first consider to compute the area of a curved polygon by integrating the Jacobian over all the cells that are obtained through either quadrilateral-dominant decomposition or triangulation. We take decompositions in Figs. 3, 5 as examples. In the case of Fig. 3(c) or Fig. 5(a) where all the cells are J^+ cells, it is clear that integrating the Jacobian over such cells leads to the area of the polygon. On the other hand, when folded cells are involved, for example in Fig. 4 or Fig. 5(b), negative-Jacobian regions there overlap with certain positive-Jacobian regions in other cells, and thus their contributions are canceled out, leading to the desired area of the polygon; see the regions of matching colors in Fig. 5(e). This implies that when the integrand is a constant, we can use folded decompositions to appropriately integrate it over a complex domain Ω.

In fact, folded decompositions can be applied to much more general functions, which we will show in Section 6 through numerical examples. It is well known that canceling of positive and negative contributions may cause large deviations due to the round-off error. On the other hand, we never spot such an issue in any of our examples.

Remark 2. We note that for the cancellation to work, the boundary curves need to be oriented in a consistent manner (e.g., counterclockwise). Moreover, we notice that due to the presence of negative Jacobian, generally a folded decomposition leads to a set of cells where $\Omega \subseteq \cup_{\ell=1}^{L} \mathcal{T}_\ell$. In other words, the resulting cells may cover a larger region than the domain of interest, which, therefore, requires the integrand $f(\mathbf{x})$ to be well defined on the entire $\cup_{\ell=1}^{L} \mathcal{T}_\ell$. We will discuss more on the definition of $f(\mathbf{x})$ in Section 5.

Next, we theoretically study folded decompositions for integration under the assumption that first, the integrand in Eq. (1) is suitably defined over all the involved integration cells, and second, the antiderivative of the integrand exists. The key idea is to apply the divergence theorem to transform an integral to its boundary counterpart. We will show that the resulting boundary integral stays the same regardless of how the decomposition is achieved. We work with the 2D case for illustration and explanation, but the argument holds for both 2D and 3D. The following derivation is constructive rather than being comprehensive to provide insights on how folded decompositions work.

In particular, we consider the domain Ω shown in Fig. 4, where the decomposition leads to a J^+ cell $\mathcal{T}_1$ and a folded cell $\mathcal{T}_2$. Initially, the integral of interest is computed using $\mathcal{T}_1$ and $\mathcal{T}_2$,

$$\int_\Omega f(\mathbf{x})d\mathbf{x} = \int_{\mathcal{T}_1} f(\mathbf{x})d\mathbf{x} + \int_{\mathcal{T}_2} f(\mathbf{x})d\mathbf{x}. \tag{8}$$

We further split $\mathcal{T}_2$ into $\mathcal{T}_2 = \mathcal{Q}_2 \cup \mathcal{Q}_3$ such that $\mathcal{Q}_2$ is a J^+ cell and $\mathcal{Q}_3$ is a J^- cell; see Fig. 6. Specifically, letting $\mathbf{Q}_k : [0,1]^2 \to \mathcal{Q}_k$ $(k = 2, 3)$, we have

$$\int_{\mathcal{T}_2} f(\mathbf{x})d\mathbf{x} = \sum_{k=2}^{3} \int_{\mathcal{Q}_k} f(\mathbf{x})d\mathbf{x} = \sum_{k=2}^{3} \int_{[0,1]^2} f(\mathbf{Q}_k(\mathbf{u})) \det(\nabla \mathbf{Q}_k(\mathbf{u})) d\mathbf{u}, \tag{9}$$

where $\det(\nabla \mathbf{Q}_3(\mathbf{u})) \leq 0$. To attain cells of consistent Jacobian, we reparameterize $\mathcal{Q}_3$ as $\tilde{\mathcal{Q}}_3$ such that

$$\int_{\tilde{\mathcal{Q}}_3} f(\mathbf{x})d\mathbf{x} = \int_{[0,1]^2} f(\tilde{\mathbf{Q}}_3(\mathbf{s})) \det\left(\nabla \tilde{\mathbf{Q}}_3(\mathbf{s})\right) d\mathbf{s} = -\int_{[0,1]^2} f(\mathbf{Q}_3(\mathbf{u})) \det(\nabla \mathbf{Q}_3(\mathbf{u})) d\mathbf{u} = -\int_{\mathcal{Q}_3} f(\mathbf{x})d\mathbf{x}, \tag{10}$$

where $\tilde{\mathcal{Q}}_3 = \mathcal{Q}_3$, $\tilde{\mathbf{Q}}_3$ is the mapping of $\tilde{\mathcal{Q}}_3$, and $\det\left(\nabla \tilde{\mathbf{Q}}_3\right) \geq 0$. Eq. (8) becomes

$$\int_\Omega f(\mathbf{x})d\mathbf{x} = \int_{\mathcal{T}_1} f(\mathbf{x})d\mathbf{x} + \int_{\mathcal{Q}_2} f(\mathbf{x})d\mathbf{x} - \int_{\tilde{\mathcal{Q}}_3} f(\mathbf{x})d\mathbf{x}. \tag{11}$$

Now we assume the antiderivative of $f(\mathbf{x})$ to be $\mathbf{F}(\mathbf{x})$ such that $f(\mathbf{x}) = \nabla \cdot \mathbf{F}(\mathbf{x})$. Applying the divergence theorem, we have

$$\begin{aligned}
\int_\Omega f(\mathbf{x})d\mathbf{x} &= \int_{\partial \mathcal{T}_1} \mathbf{F}(\mathbf{x}) \cdot \mathbf{n}_1 d\mathbf{x} + \int_{\partial \mathcal{Q}_2} \mathbf{F}(\mathbf{x}) \cdot \mathbf{n}_2 d\mathbf{x} - \int_{\partial \tilde{\mathcal{Q}}_3} \mathbf{F}(\mathbf{x}) \cdot \tilde{\mathbf{n}}_3 d\mathbf{x}, \\
&= \sum_{j=1}^{4} \int_{\mathcal{C}_{1,j}} \mathbf{F}(\mathbf{x}) \cdot \mathbf{n}_1 d\mathbf{x} + \sum_{j=1}^{3} \int_{\mathcal{C}_{2,j}} \mathbf{F}(\mathbf{x}) \cdot \mathbf{n}_2 d\mathbf{x} - \sum_{j=1}^{2} \int_{\tilde{\mathcal{C}}_{3,j}} \mathbf{F}(\mathbf{x}) \cdot \tilde{\mathbf{n}}_3 d\mathbf{x},
\end{aligned} \tag{12}$$

where $\mathbf{n}_1$, $\mathbf{n}_2$, and $\tilde{\mathbf{n}}_3$ are the outwards normals to corresponding domains, and $\mathcal{C}_{i,j}$ are the boundary curves; see Fig. 6. Note that we have $\mathbf{n}_2 = -\mathbf{n}_1$ when $\mathbf{x} \in \mathcal{C}_{2,1}$, and $\tilde{\mathbf{n}}_3 = \mathbf{n}_1$ when $\mathbf{x} \in \tilde{\mathcal{C}}_{3,1}$. Therefore, contributions from common curves (i.e., $\mathcal{C}_{2,1}$ and $\tilde{\mathcal{C}}_{3,1}$ with $\mathcal{C}_{1,1}$) are canceled out, leading to

$$\int_\Omega f(\mathbf{x})d\mathbf{x} = \sum_{j=2}^{4} \int_{\mathcal{C}_{1,j}} \mathbf{F}(\mathbf{x}) \cdot \mathbf{n}_1 d\mathbf{x} + \sum_{j=2}^{3} \int_{\mathcal{C}_{2,j}} \mathbf{F}(\mathbf{x}) \cdot \mathbf{n}_2 d\mathbf{x} - \int_{\tilde{\mathcal{C}}_{3,2}} \mathbf{F}(\mathbf{x}) \cdot \tilde{\mathbf{n}}_3 d\mathbf{x}. \tag{13}$$

Regarding the last term in Eq. (13), after reversing the direction of $\tilde{\mathcal{C}}_{3,2}$, we have

$$\int_{\tilde{\mathcal{C}}_{3,2}} \mathbf{F}(\mathbf{x}) \cdot \tilde{\mathbf{n}}_3 d\mathbf{x} = -\int_{\mathcal{C}_{3,2}} \mathbf{F}(\mathbf{x}) \cdot \mathbf{n}_3 d\mathbf{x}, \tag{14}$$
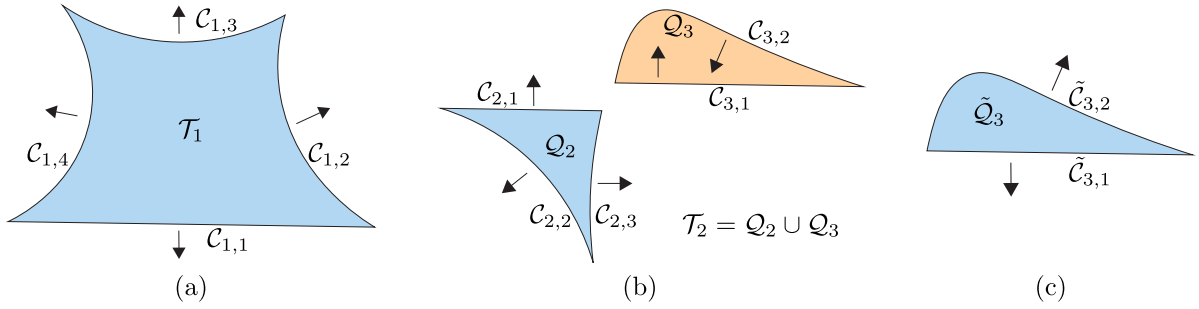
Fig. 6. The decomposed cells shown in Fig. 4 and notations of their boundary curves. (a) The J^+ cell, (b) the folded cell that consists of a J^+ subcell and a J^- subcell, and (c) reparameterization of $\mathcal{Q}_3$ to obtain a J^+ counterpart. The black arrows indicate the normal directions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where $\mathbf{n}_3 = -\tilde{\mathbf{n}}_3$. $\mathcal{C}_{3,2}$ has the same orientation as $\partial\Omega$ and $\mathbf{n}_3$ is the outwards normal to $\partial\Omega$ when $\mathbf{x} \in \mathcal{C}_{3,2}$. Therefore, examining Eqs. (13), (14) reveals that the integral over Ω reduces to the boundary integral,

$$\int_\Omega f(\mathbf{x})\mathrm{d}\mathbf{x} = \int_{\partial\Omega} \mathbf{F}(\mathbf{x}) \cdot \mathbf{n}\mathrm{d}\mathbf{x}, \tag{15}$$

where we note that $\partial\Omega = \mathcal{C}_{1,2} \cup \mathcal{C}_{1,3} \cup \mathcal{C}_{1,4} \cup \mathcal{C}_{2,2} \cup \mathcal{C}_{2,3} \cup \mathcal{C}_{3,2}$.

Therefore, Eqs. (8), (15) are equivalent. As the involved decomposition represents a generic case and the same argument can be straightforwardly applied to other decompositions, we conclude that the integral of interest leads to the same boundary integral regardless of how a decomposition is created, given that the above-mentioned two assumptions hold. In other words, if a J^+ decomposition exists for the domain shown in Fig. 4, then theoretically it leads to the same integration result as that using the folded decomposition. Indeed, we will show different kinds of numerical evidence later.

4. Folded decompositions for $d = 3$

Now we discuss the 3D case, that is, to use folded decompositions for integration over curved polyhedra. The idea is analogous to Section 3.2, but here our goal is to decompose a given polyhedron into a set of (triangular and/or rectangular) pyramids, which, in particular, can naturally deal with gaps/overlaps as well as sharp features in the input B-rep. Pyramids are treated as degenerated hexahedra. The decomposition needs two steps: (1) decomposition of the input B-rep following the previous section, and (2) search for a seed vertex to build pyramids.

As the input B-rep is essentially a 2-manifold, Step (1) immediately follows Section 3. We assume that the B-rep has the form $\partial\Omega = \{\mathbf{S}_i(u, v) \in \mathbb{R}^3, (u, v) \in \mathcal{D}_i \subseteq [0, 1]^2, i = 1, \ldots, N_f\}$. It is worth mentioning that the mesh is not required to be conformal on the common edge of any two faces. Such edges sometimes represent sharp features of $\partial\Omega$. Moreover, gaps/overlaps are allowed in the input B-rep as long as the topological information is correct. Such settings are supported because folded decompositions are generated independently for each $\mathbf{S}_i(u, v)$, or more precisely, for $\mathcal{D}_i$ in the parametric domain, leading to $\mathbf{R}_{i,j} : [0, 1]^2 \to \mathbb{R}^2, j = 1, \ldots, N_i$, where $\mathbf{R}_{i,j}$ is a (degenerate) Coons patch that may have negative Jacobian. As a result, each face $\mathbf{S}_i$ is decomposed as $\mathbf{S}_i \circ \mathbf{R}_{i,j}([0, 1]^2), j = 1, \ldots, N_i$. Each $\mathbf{S}_i \circ \mathbf{R}_{i,j}([0, 1]^2)$ is further approximated by a Bézier surface to obtain a polynomial representation.

Next in Step (2), we choose a seed vertex $\mathbf{V}$ to create target pyramids with all the resulting patches $\mathbf{S}_i \circ \mathbf{R}_{i,j}([0, 1]^2)$ from Step (1), where $\mathbf{V}$ and $\mathbf{S}_i \circ \mathbf{R}_{i,j}$ serve as the top vertex and the base surface, respectively. While $\mathbf{V}$ can be chosen in a rather arbitrary manner, we usually choose it to be one of the vertices of the polyhedron as it can yield fewer cells (similar to 2D).

Once $\mathbf{V}$ and $\mathbf{R}_{i,j}(u, v)$ are in place, a pyramidal cell $\mathbf{T}_{i,j}(u, v, w)$ is created as a ruled volume between $\mathbf{V}$ and each $\mathbf{R}_{i,j}(u, v)$,

$$\mathbf{T}_{i,j}(u, v, w) = (1 - w)\mathbf{S}_i \circ \mathbf{R}_{i,j}(u, v) + w\mathbf{V}, \quad (u, v, w) \in [0, 1]^3. \tag{16}$$

Therefore, the given curved polyhedron is decomposed into a set of such cells $\{\mathbf{T}_{i,j}(u, v, w)\}$ for integration.
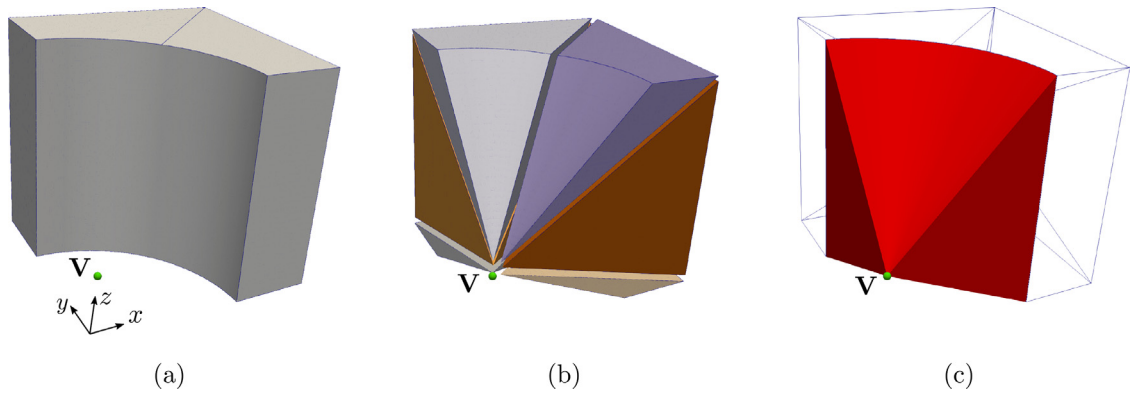
Fig. 7. Split of a curved polyhedron using an arbitrary vertex **V**. (a) The input B-rep (a cube cut with a cylinder); (b) six of the resulting cells present positive Jacobian; and (c) one of the cells has negative Jacobian. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

As an example, in Fig. 7 the decomposition of a 3D B-rep is shown. The polyhedra to decompose is a cube $[0, 1]^3$ cut with a cylinder of radius 0.65 and axis along the z direction (see Fig. 7(a)). A seed vertex **V** is (arbitrarily) chosen to be at $(0, 0, 0.3)$. The bottom and top faces of the B-rep (5-sided polygons) are decomposed into two quadrilaterals each, whereas all the other faces require a single quadrilateral. By generating pyramids between the generated surface patches and the vertex **V** a total of 7 cells are created. Six of them (Fig. 7(b)) present positive Jacobian everywhere except at the vertex **V**, where the Jacobian vanishes, so they are J^+ cells. On the other hand, the cell shown in Fig. 7(c) has negative Jacobian everywhere except at **V**, where the Jacobian vanishes, so it is a J^- cell.

Remark 3. The tensor-product Gauss quadrature is set up for every cell, even though it is not optimal on triangular or pyramidal cells. We adopt it mainly for a simple and unified implementation. Otherwise, much more effort has to be devoted to carefully distinguishing and tracking different types of cells in order to set up a corresponding sophisticated quadrature, which may complicate much the implementation, be error-prone, and thus hinder robustness.

Remark 4. The work [38] strives to find a so-called kernel point as the seed vertex such that all the resulting cells can have positive Jacobian. A kernel point is a point that is visible to all the points on the boundary. This is known as the three-dimensional *art gallery* problem. Generally, it involves recursively subdividing the given domain to find an appropriate kernel point for each subdomain, which complicates implementation and generates a large number of cells. Due to the great flexibility folded decompositions provide, we do not need a kernel point for decomposition in this work.

5. Application: immersed methods

One natural application of the integration method described above is isogeometric analysis (IGA) [43] with trimmed volumetric geometries. The proposed method can be also used in other applications such as virtual element methods [44]. One typical way of using such representations in IGA is through immersed methods [45], which embed a B-rep into a background Cartesian grid and thus leads to a non-conformal description of the computational domain; see Fig. 8(a) for example.

Numerical integration in IGA is performed at the element level, so we first classify different types of elements. Let $\mathcal{G}$ be a Cartesian grid, and $K \in \mathcal{G}$ be an element. The computational domain is Ω and $\Gamma = \partial\Omega$. Depending on how K relates to Γ and Ω, elements can be divided into trimmed elements ($K \cap \Gamma \neq \varnothing$), non-trimmed active elements ($K \cap \Omega = K$), and non-trimmed inactive elements ($K \cap \Omega = \varnothing$) that are not assembled in general. While it is straightforward to use standard Gauss quadrature for non-trimmed elements, trimmed elements need special treatment.

Every trimmed element ($K_a = K \cap \Omega \neq \varnothing$) is a curved polygon ($d = 2$) or a curved polyhedron ($d = 3$), to which we apply our decomposition strategy. In other words, K_a is now the domain of interest in Eq. (1). The boundary of K_a, e.g., $K \cap \Gamma$ in particular, generally does not have a polynomial representation, and instead it has a piecewise polynomial structure. Depending on the CAD packages that are used to create the trimming curve Γ, $K \cap \Gamma$ may be a piece of NURBS curve or a set of piecewise linear segments. To build integration cells, polynomial fitting is performed (in both 2D and 3D) to find a compact Bézier representation to approximate $K \cap \Gamma$. The degree of the Bézier representation is left as a parameter to be chosen to guarantee the desired level of accuracy; see, e.g., [37].

Remark 5. Recall that the tensor-product Gauss quadrature is defined on the parent domain $\Lambda = [0,1]^d$ ($d = 2, 3$) of each cell. The positions of quadrature points in a trimmed element K_a (whose parent element is K) are obtained by the geometric mapping $\mathbf{T}_{K_a} : [0,1]^d \rightarrow \mathcal{T}$ of the cell; see Eqs. (7), (16). Their quadrature weights in the domain of K_a are the product of the Jacobian of $\mathbf{T}$ with the corresponding weights of Gauss quadrature in Λ; see also Eq. (4). This implies that the weights can be negative or zero, which, however, is not an issue because they are only used in multiplication and never in division.

However, due to the nature of negative Jacobian, we may have $\mathbf{T}(\Lambda) \not\subset \Omega$, and thus some quadrature points may lie outside of Ω. When certain quantities of interest are not well defined outside Ω, one solution to this potential issue is through polynomial extension [11], which enables a stable way to evaluate/extrapolate quantities outside of the domain. More specifically, let a generic function f be the quantity of interest, which may be certain given data but not defined outside Ω. We now perform polynomial extension elementwise. Considering a trimmed element K_a that has folded cells, we first find a so-called *good neighbor* (K_g) to K_a such that $K_g \subset \Omega$ is (preferably) a non-trimmed active element. Next, we perform L^2 projection for f onto the space spanned by the Bernstein polynomials defined on K_g. As a result, f is expressed as a linear combination of these Bernstein polynomials. Finally, we extend such expression to the outside of Ω and evaluate f with the aid of the (extended) Bernstein polynomials.

Remark 6. Recall that when creating integration cells, we do not require their parameterization to be conformal to their common interfaces. This not only provides great flexibility on how we can decompose a given domain, but also supports B-reps that have artificial defects such as gaps and overlaps, thus opening the door to deal with dirty CAD geometries in a robust way.

Remark 7. While folded decompositions provide a significant flexibility in accommodating real-world applications, there are limitations due to their nature of exhibiting negative Jacobian. For instance, visualizing simulation results with folded cells poses an issue. The sign of Jacobian can be seen as the orientation of a local region. Change of sign means change of orientation, and thus visualization detects a reverse reflection on negative-Jacobian regions. As a result, negative-Jacobian regions are invisible to users. The remedy we adopt in this work is to create J^+ surface cells for visualization on the boundary of a volume, which is much easier to generate than volumetric J^+ cells. The boundary is decomposed according to quadrangulation [46] to ensure that all the surface cells have positive Jacobians, where quantities of interest are computed and visualized. This is done independently from creating volumetric cells, and it is only meant for visualization purposes.

6. Integration of regular functions

In this section, we demonstrate the efficacy and robustness of folded decompositions by integrating several regular functions. We first compare the performance of folded and J^+ decompositions in computing integrals of polynomials. Next, we study the convergence behaviors in integrating general functions that are not polynomials.

6.1. Integration of polynomials

The domain of interest is denoted as Ω. The boundary Γ of Ω is embedded into a Cartesian grid $\mathcal{G}$, where we are only interested in trimmed elements as polynomials can be easily integrated exactly in non-trimmed elements. For each trimmed element $K_a = K \cap \Omega \neq \varnothing$ ($K \in \mathcal{G}$ and $K \cap \Gamma \neq \varnothing$), we aim to compute

$$I_{K,i} = \int_{K_a} B_i^p(\mathbf{x}) d\mathbf{x},$$
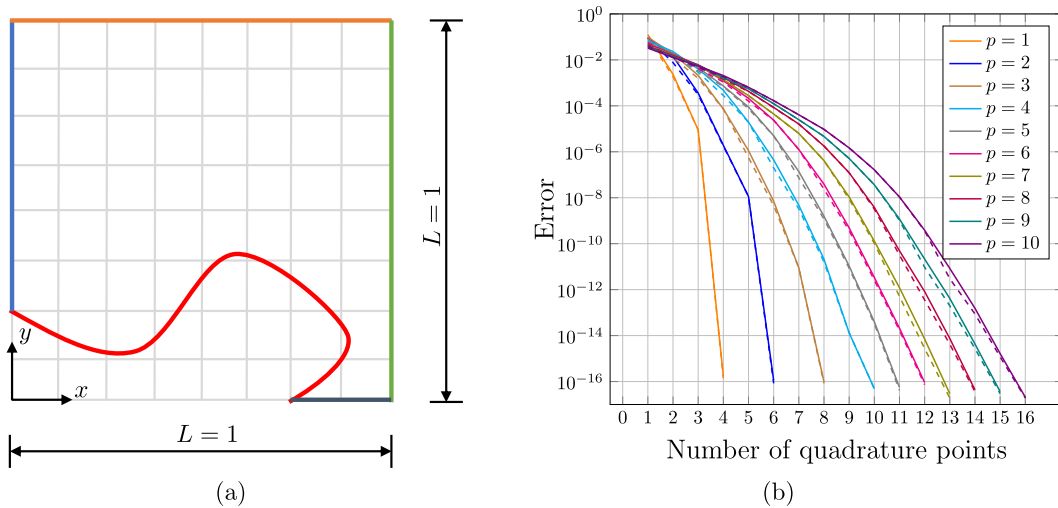
Fig. 8. Integration of polynomials on a 2D B-rep that involves a quadratic B-spline curve (red). (a) The input B-rep composed of 5 connected curves, and (b) relative integration error with respect to the number of quadrature points per direction, where different colors represent different degrees, and solid and dashed curves are the results using folded and J^+ decompositions, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where $B_i^p(\mathbf{x})$ is ith degree-p tensor-product Bernstein polynomial defined in K. Note that $\text{span}\{B_i^p(\mathbf{x})\}_i = \mathbb{Q}_{p,p}(K)$ in 2D and $\text{span}\{B_i^p(\mathbf{x})\}_i = \mathbb{Q}_{p,p,p}(K)$ in 3D. Generally, K_a is approximated with a set of integration cells $\{\mathbf{T}_{K,j}(\Lambda)\}_j$ ($\Lambda = [0,1]^d$), and thus $I_{K,i}$ is computed as

$$I_{K,i} = \int_{K_a} B_i^p(\mathbf{x})d\mathbf{x} \approx \sum_j \int_\Lambda B_i^p \circ \mathbf{T}_{K,j} \det\left(\nabla \mathbf{T}_{K,j}\right) d\mathbf{x}. \tag{17}$$

We use the tensor-product Gauss quadrature rule with n points per direction in Λ. An overkill result with 64 quadrature points per direction is taken as the reference ($I_{K,i}^{\text{ref}}$) to evaluate error,

$$\text{Err} = \max_{K \in \mathcal{G}, K \cap \Gamma \neq \varnothing} \max_i |I_{K,i}^h - I_{K,i}^{\text{ref}}|, \tag{18}$$

where $I_{K,i}^h$ is the approximate to $I_{K,i}$ obtained by (17) with Gauss quadrature.

6.1.1. B-rep in 2D with a B-spline curve

The geometry of the first 2D example is shown in Fig. 8(a). Its B-rep consists of 5 connected curves: 4 straight segments and a quadratic B-spline curve (red), whose knot vector is $\{0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1\}$ and its control points are $(0, 0.25)$, $(0.25, 0)$, $(0.5, 0.5)$, $(0.9, 0.25)$, $(0.8, 0.125)$, and $(0.75, 0)$. The B-rep is embedded into a 8×8 Cartesian grid. We consider two ways of decomposition for trimmed elements: the J^+ decomposition by quadrangulation [46], and the folded decomposition through triangulation in Section 3.2. In either case, the degree of all the Bézier patches (i.e., $\mathbf{T}_{K,j}$ in (17)) is set to be 2, the same as that of the B-spline curve.

In the case of folded decomposition, we choose different seed vertices to create a family of triangulations for each trimmed element K_a (whose parent element is K). Specifically, 11×11 points are sampled according to a uniform Cartesian distribution in K, and each of them serves as a seed vertex to create a corresponding triangulation. This is aimed to test whether folded decompositions can work robustly in a rather arbitrary setting. Different triangulations may yield different integration errors, among which the largest is taken as the error of folded decomposition.

We use different degrees of Bernstein polynomials (ranging from 1 to 10) to test the integration accuracy with respect to the number of quadrature points per direction. The results are summarized in Fig. 8(b). We observe that the results are almost indistinguishable between folded and J^+ cells. Moreover, according to [47] and (A.3) in Appendix, where we study the number of quadrature points required to exactly integrate a polynomial, we need at least $2(p+1)$ quadrature points (per direction) to exactly integrate $B_i^p(\mathbf{x})$. In particular, we observe this agreement in Fig. 8(b) for degrees from 1 to 4, where the number of quadrature points needed to achieve machine precision
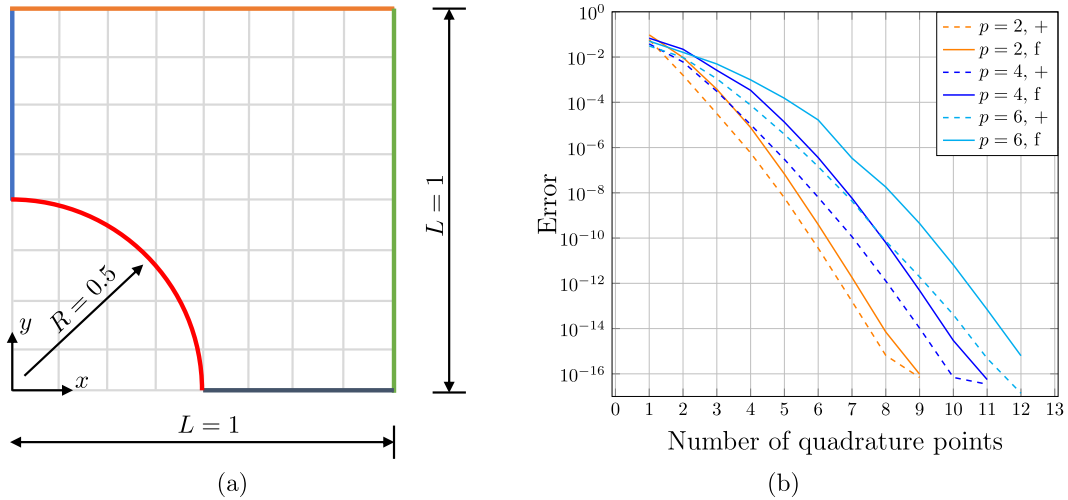
Fig. 9. Integration of polynomials on a 2D B-rep that involves a quadratic rational Bézier curve (red). (a) The input B-rep composed of 5 connected curves, and (b) relative integration error with respect to the number of quadrature points per direction, where different colors represent different degrees, and solid and dashed curves are the results using folded and J^+ decompositions, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

($\sim 10^{-16}$) is 4, 6, 8, and 10, respectively. However, for even higher degrees, machine precision is reached before $2(p+1)$ quadrature points are used, and further increasing the number of quadrature points will no longer improve the accuracy.

6.1.2. B-rep in 2D with a rational curve

Next, we study another 2D B-rep that involves a rational curve. The input is shown in Fig. 9(a), where the red curve is a quadratic rational Bézier curve exactly representing a quarter of a circle. Other than the difference in the B-rep, everything else follows the same as that in Section 6.1.1. The degrees of Bernstein polynomials for integration are chosen to be $p \in \{2, 4, 6\}$. The circle is approximated with polynomial Bézier curves of degree p, which are then used to create in either folded or J^+ decompositions. The results are summarized in Fig. 9(b). We observe the following. First, overall, J^+ decompositions yield more accurate results than folded decompositions, where fewer quadrature points are used to achieve the same accuracy. However, recall that we create 11×11 triangulations for each cut element, and we take the worst integration results to be the results using folded decompositions in Fig. 9(b). Second, with a fixed number of quadrature points, the relative accuracy difference between folded and J^+ decompositions is really not significant; compare solid and dashed curves when the relative error is 10^{-16}. Third, folded decompositions can also achieve machine precision by using slightly more quadrature points than J^+ decompositions. Fourth, J^+ and folded decompositions show the same convergence rates, while the latter may have a longer pre-asymptotic regime.

6.1.3. B-rep in 3D with a polynomial surface

Now we integrate Bernstein polynomials over a 3D domain. The corresponding B-rep is shown in Fig. 10(a), which consists of 4 axis-aligned planar surfaces and a bi-quadratic Bézier surface. The control points of the Bézier surface are given as $(1, 0.2, 0)$, $(1, 0.8, 0.5)$, $(1, 0.4, 1)$, $(0.5, 0.5, 0)$, $(0.5, 0.5, 0.5)$, $(0.25, 0.25, 1)$, $(0.2, 1, 0)$, $(0, 1, 0.5)$, and $(0.3, 1, 1)$. The B-rep is embedded into a $1 \times 1 \times 1$ Cartesian grid, i.e., a single element. Similarly, $11 \times 11 \times 11$ points are sampled as seed vertices to create a family of pyramidal decompositions, where the ones that yield the worst and best integration errors are chosen. The worst decomposition is guaranteed to have folded cells. A J^+ decomposition is created using the seed vertex as $(1, 1, 0.5)$. The degrees of Bernstein polynomials are set to be $p \in \{3, 6, 9\}$. The results are summarized in Fig. 10(b), and we observe the following. First, machine precision can be again achieved using both folded and J^+ decompositions with a sufficient number of quadrature points per direction. Second, machine precision is achieved with fewer quadrature points than the ones theoretically
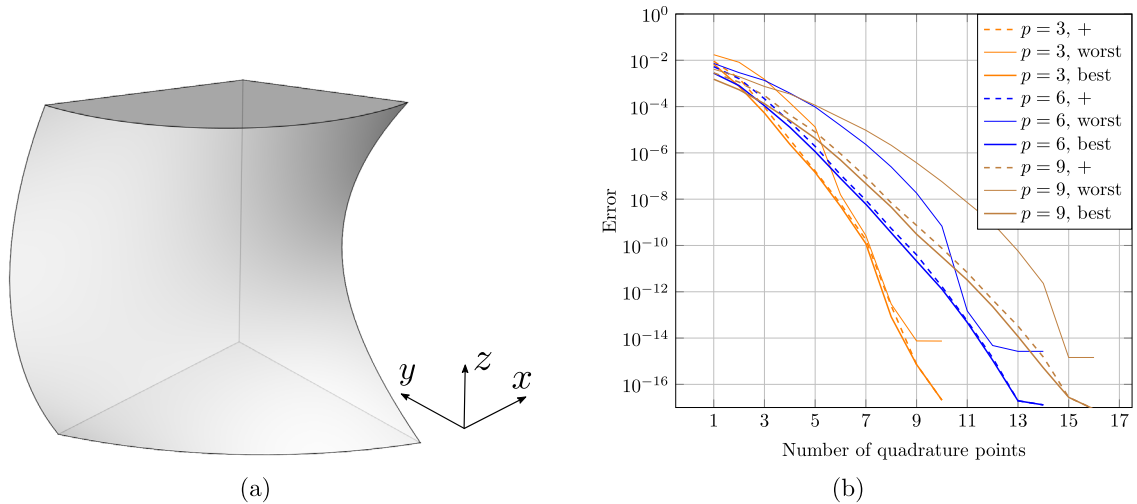
Fig. 10. Integration of polynomials on a 3D B-rep that involves a bi-quadratic Bézier surface. (a) The input B-rep composed of 4 axis-aligned planar surfaces and a bi-quadratic Bézier surface, and (b) relative integration error with respect to the number of quadrature points per direction, where different colors represent different degrees, and solid and dashed curves are the results using folded and J^+ decompositions, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

predicted for exact integration. In 3D, according to Eq. (A.5) in Appendix, we need $n \geq 3(p + 1)$ to exactly integrate polynomials. For instance, when $p = 6$, 21 quadrature points are needed in theory, but as shown in Fig. 10(b), 14 points already suffice to reach machine precision when the J^+ decomposition is used. Third, given the same number of quadrature points, the results using the J^+ decomposition fall between those of the best and the worst decompositions. Even in the worst scenario, the difference in accuracy can be within two orders provided that a sufficient number of quadrature points are used. In other words, the performance of a folded decomposition can be always comparable to that of J^+ decompositions.

6.2. Integration of general functions

In this section, we study to integrate several regular functions that are not polynomials. We use again the B-reps shown in Figs. 8(a) and 10(a) for 2D and 3D tests, respectively. The integrands of interest are the smooth functions:

$$f_1(x, y, z) = \exp(y) \sin(x) \cos(y) \cos(z) \tag{19a}$$

$$f_2(x, y, z) = \frac{\left(x^2 + y^2 + z^2\right)(x + y + z) + xyz}{x^4 + y^4 + z^4 + 1} \tag{19b}$$

where f_2 is a rational polynomial similar to the ones present in the case of rational B-spline basis functions. For the 2D case, we just set $z = 0$.

In 2D, the B-rep is embedded into a series of globally refined Cartesian grids, from 2×2 up to 128×128. Likewise in 3D, the Cartesian grids range from $1 \times 1 \times 1$ to $16 \times 16 \times 16$. Integration is performed elementwise. The B-reps are approximated in each involved integration cell with quadratic Bézier curves in 2D, or bi-quadratic Bézier surfaces in 3D. In each integration cell, we change the number of quadrature points per direction (denoted by n) and study the convergence with respect to mesh refinement. In either 2D or 3D, an overkill solution, which is obtained using a dense Cartesian grid and a large number of quadrature points, serves as the reference solution to evaluate the error.

In Fig. 11, we report the convergence of the integration error, computed as the absolute difference between the obtained values and the reference ones for the integrals computed over the whole domain. We compare folded and J^+ decompositions in 2D, whereas we only use folded decompositions in 3D. As in the previous section, for the folded decompositions we create all the possible decompositions by selecting the seed vertex in a 11×11 or
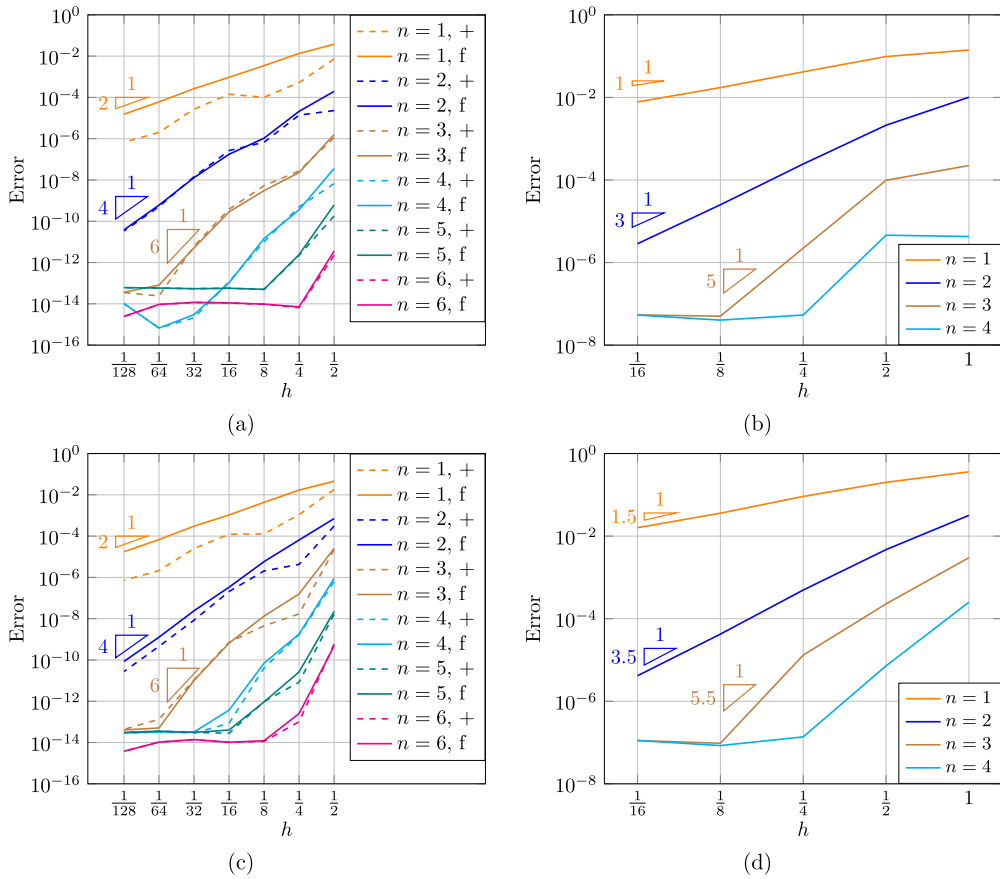
Fig. 11. Convergence tests of the absolute integration error over the whole domain for the integration of functions f_1 and f_2 defined in (19) with respect to the mesh size h. In (a,c) we report the 2D tests for the domain shown in Fig. 8(a) using the functions f_1 and f_2, respectively. Analogously, in (b,d), the 3D tests for the domain in Fig. 10(a) and the functions f_1 and f_2, respectively, are plotted. In (a,c), we compare the performance of (worst) folded and J^+ decompositions, whereas in (b,d), we only use (worst) folded decompositions. Solid and dashed curves in (a,c) are the results using (worst) folded and J^+ decompositions, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$11 \times 11 \times 11$ Cartesian grid, respectively. Among all the created decompositions, the worst error is reported in Fig. 11.

In 2D, we observe that the performance of the worst folded decompositions and J^+ decompositions is comparable, especially when $n \geq 2$. We also observe a convergence rate of $2n$ when $1 \leq n \leq 3$, which is expected using standard Gauss quadrature. When $n \geq 4$, the integration error quickly goes under machine precision, so we cannot observe the asymptotic convergence of the expected rate.

On the other hand, in 3D, when $1 \leq n \leq 3$ we observe convergence rates $2n - \epsilon$, with $\epsilon = 1$ for f_1 (Fig. 11(b)), and $\epsilon = 1/2$ for f_2 (Fig. 11(d)): one and half order less than expected, respectively. Recall that a certain surface of an integration cell, which has a Bézier representation, approximates the given trimming surface. As stated before in Section 5, and illustrated below in Section 7, this approximation, if performed with the same degree as the one used for the discretization of an elliptic PDE, guarantees the optimal approximation properties of the presented immersed approach. The suboptimal convergence may be due to the fact that the geometric error in creating integration cells is more prominent in 3D than that in 2D. Moreover, the convergence reaches a *plateau* when the accuracy is in the order of 10^{-8}. This is because surface–surface intersections (element boundaries with the trimming surface) are involved in defining cut elements, for which we employ OpenCASCADE [48] and the accuracy of related operations is limited to an untunable tolerance of 10^{-8}. Similar geometric artifacts were reported before in [18,37].
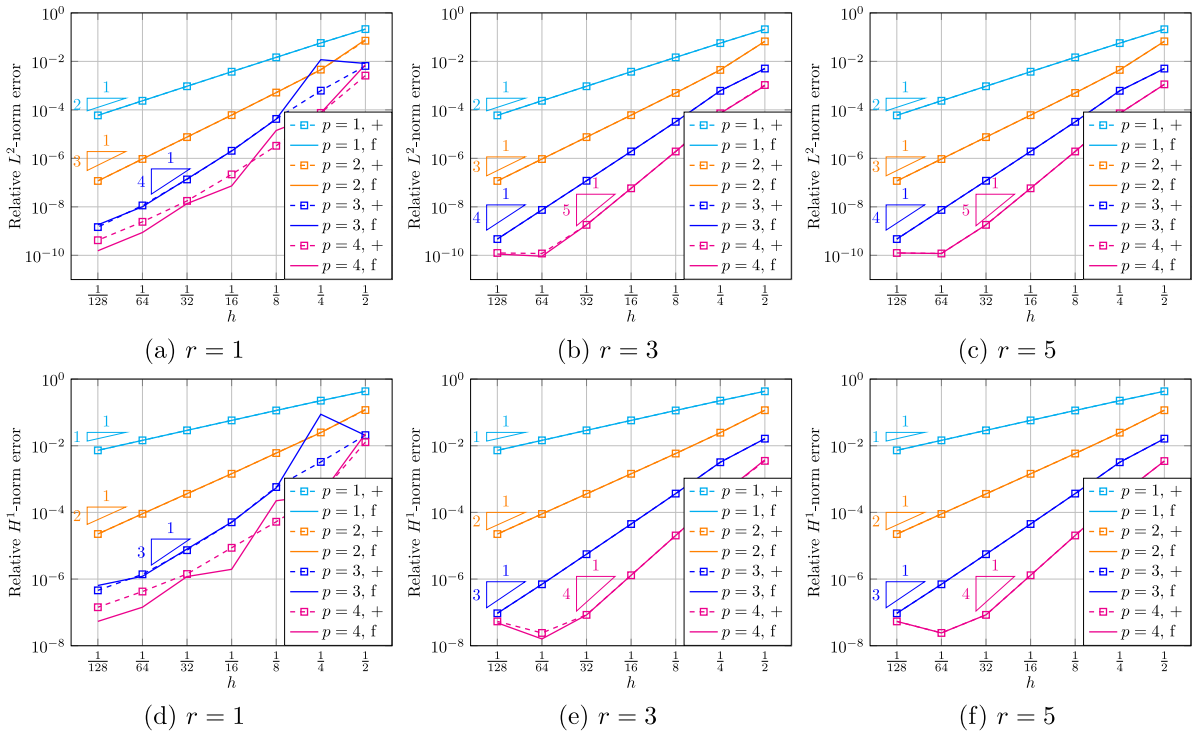
Fig. 12. The convergence plot in solving Poisson's problem on the domain shown in Fig. 8(a), where we compare the performance between folded and J^+ decompositions for different values of r. (a–c) Relative L^2-norm error, and (d–f) relative H^1-norm error, where h stands for the element size. Solid and dashed curves are the results using folded and J^+ decompositions, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7. Poisson's problem with immersed IGA

In this section, we solve Poisson's problem using IGA, where folded decompositions are used for numerical integration. We first perform convergence studies on trimmed planar geometries, where the results are compared with those of J^+ decompositions. Moreover, to show the flexibility and potential of folded decompositions, we solve the problem on several trimmed volumes, including an arbitrarily cut cube and a complex engine model. Note that the number of quadrature points per direction suffices as long as the integration error does not dominate simulation error. Therefore, we adopt the $(p+r)$-Gauss quadrature rule for a given spline discretization of degree p, where we vary $r \geq 1$ to check how it influences convergence. Moreover, recall that to create integration cells, approximation is often needed with Bézier curves (2D) or Bézier surfaces (3D). We choose the degree of each Bézier representation to be p, which suffices to deliver optimal convergence rates [37].

We take Poisson's problem as the model problem in all the following tests. The goal is to find $u \in \Omega$ such that

$$\Delta u = f \quad \text{in } \Omega\,, \tag{20a}$$

$$\nabla u \cdot \mathbf{n} = g \quad \text{on } \partial\Omega_N\,, \tag{20b}$$

$$u = 0 \quad \text{on } \partial\Omega_D\,, \tag{20c}$$

where $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$ and $\partial\Omega_D \cap \partial\Omega_N = \emptyset$, $\mathbf{n}$ is the outer unit normal on $\partial\Omega$, and g is the Neumann boundary condition. We adopt $u(x, y) = \sin(\pi x)\sin(\pi y)$ in 2D and $u(x, y, z) = \sin(\pi x)\sin(\pi y)\sin(\pi z)$ in 3D as manufactured solutions (except for the complex engine model). The choice of such regular functions as target solutions was motivated by the aim of focusing our study on the consistency error, mainly controlled by the numerical integration and geometric representation errors. The effect of the approximation error in the solution of elliptic PDEs using trimmed spline spaces has been previously addressed in [37].
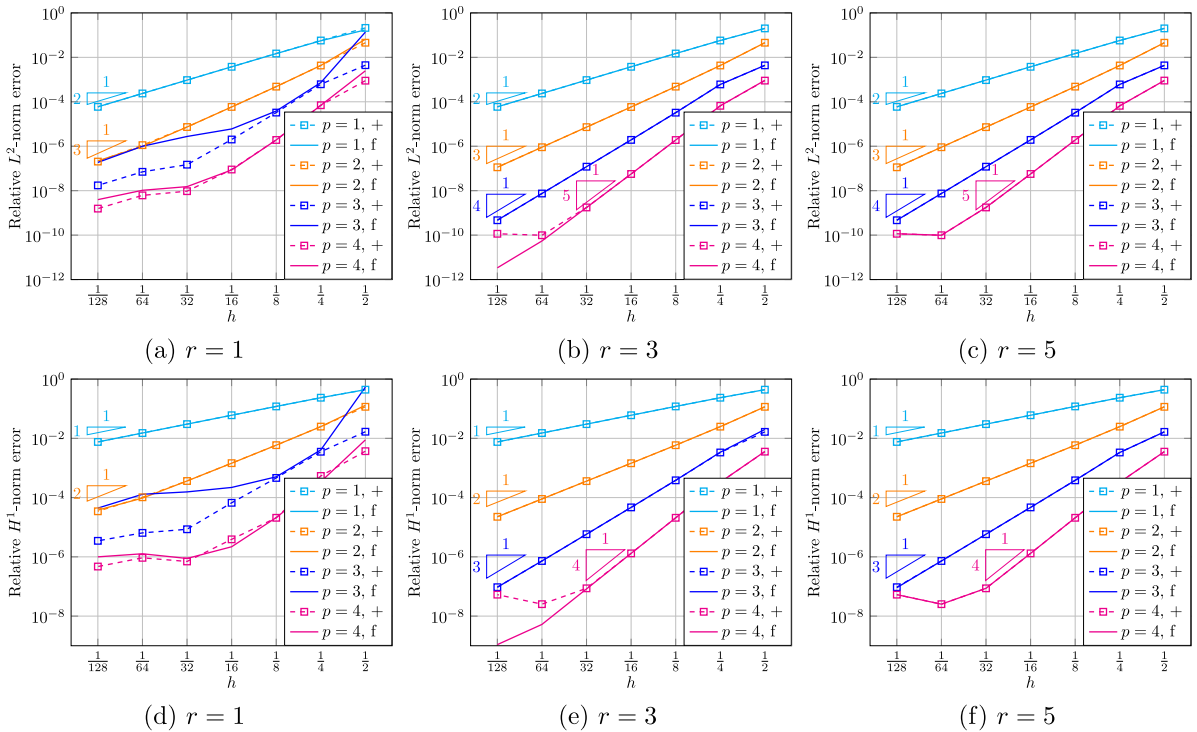
Fig. 13. The convergence plot in solving Poisson's problem on the domain shown in Fig. 9(a), where we compare the performance between folded and J^+ decompositions for different values of r. (a–c) relative L^2-norm error, and (d–f) relative H^1-norm error, where h stands for the element size. Solid and dashed curves are the results using folded and J^+ decompositions, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7.1. Trimmed planar geometries

We start with two planar geometries, whose B-reps are already shown in Figs. 8(a) and 9(a). The problem settings for these two domains are identical. The homogeneous Dirichlet boundary condition is imposed on the right boundary ($x = 1$), whereas the non-homogeneous Neumann boundary condition is imposed elsewhere. In each problem, the B-rep is embedded into a series of consecutively refined Cartesian grids, from 2×2 up to 128×128. Different degrees p of B-spline discretizations are defined on each Cartesian grid, with $p \in \{1, 2, 3, 4\}$. In each integration cell, $p + r$ quadrature points are used along each direction, where we use $r \in \{1, 3, 5\}$.

For these two geometries, the folded decomposition for each trimmed element is created using the element's origin as seed vertex for triangulation. This is meant to create highly distorted cells and it guarantees the presence of folded cells for this particular geometry. J^+ decompositions are generated using the quadrangulation method proposed in [46].

The convergence plots are shown in Figs. 12 and 13. We observe that with a sufficient number of quadrature points, both integration methods (folded and J^+ decompositions) yield identical optimal results for any degree. While, for $p \leq 2$ and $p + 1$ points the results of both methods are indistinguishable; $p + 3$ are needed for $p = 3$, and $p + 5$ for $p = 4$, in order to match the results of both methods even for the finest discretization considered. We remark here that, as discussed in [47] and in Appendix, the integration of reparameterized elements requires more than the classical $p + 1$ quadrature points per direction, being specially evident for $p \geq 3$ in Figs. 12 and 13.

We also observe in this case that the convergence curve hits a plateau when H^1-norm error reaches 10^{-8} (10^{-10} for the L^2-norm). This is caused by the geometric tolerances involved in the computation of surface–surface intersections required to locate trimmed elements, as also discussed in [18,37]. Once the simulation result reaches this level of accuracy, the geometric approximation error starts to dominate and thus leads to a plateau in convergence. These tests provide the numerical evidence that as long as a sufficient number of quadrature points are used, folded decompositions can yield the same convergence as J^+ decompositions.
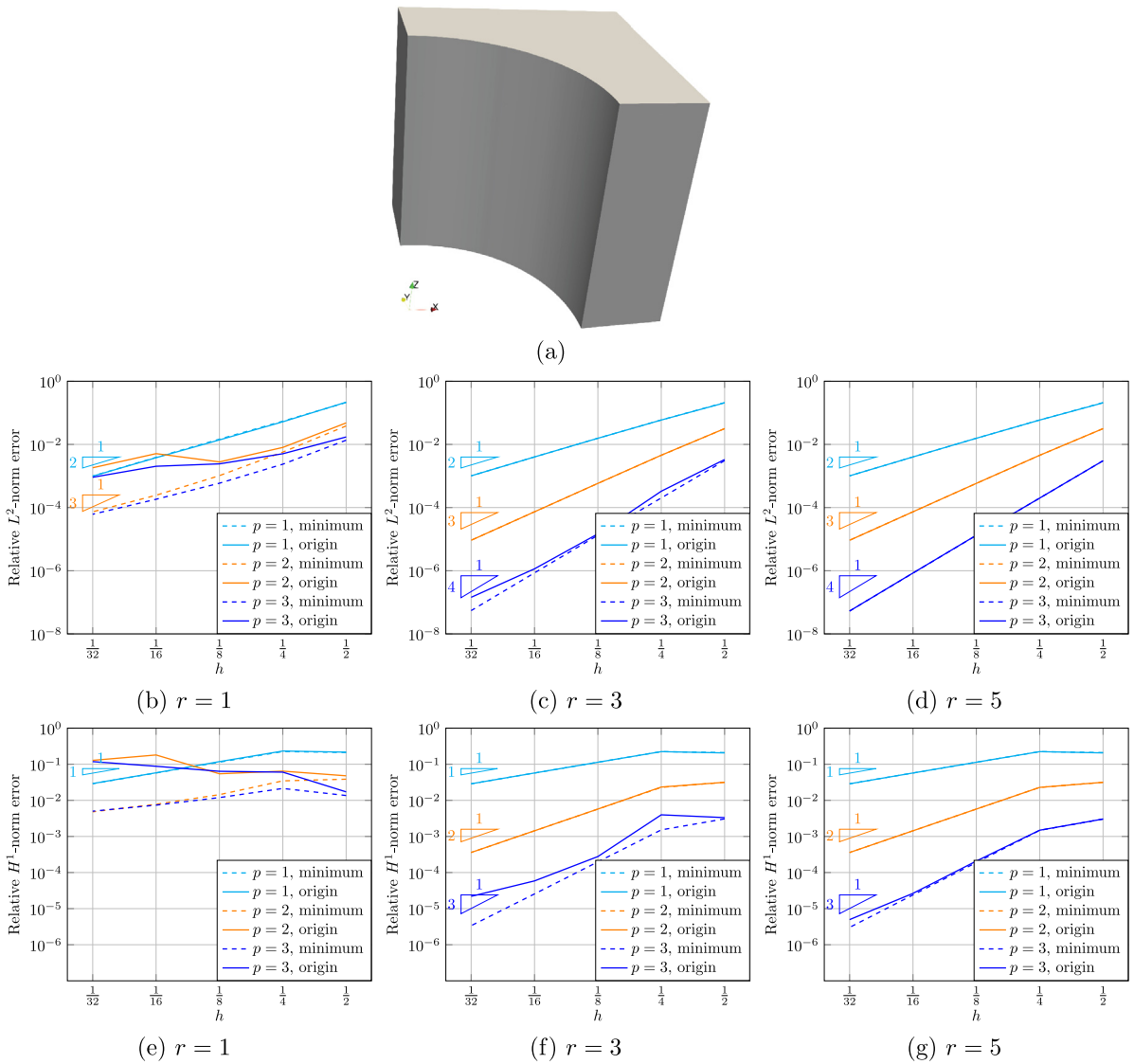
Fig. 14. Solving Poisson's problem on a domain obtained by cutting a cube with a cylinder. (a) The input B-rep, (b–d) relative L^2-norm error, and (e–g) relative H^1-norm error, for different numbers of quadrature points. Two different ways of decompositions are studied: one leading to the minimum number of cells (dashed lines) and the other choosing the element origin as the seed vertex (solid lines). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7.2. Trimmed volumes

Now we present several 3D examples where we do not constrain the type of decompositions to be generated. In other words, the decomposition algorithm terminates as long as a decomposition is achieved for a trimmed element, which can be either a J^+ decomposition or a folded decomposition. Nonetheless, folded decompositions are often generated in practice.

We start with a simple geometry as shown in Fig. 14(a), where a cube $[0, L]^3$ ($L = 1000$) is cut by a cylinder of radius $0.65L$, and the axis of the cylinder is along the z direction. A quarter of the cylinder is involved in the B-rep, which is exactly represented by a rational bi-quadratic Bézier surface. To perform a convergence study, the B-rep is embedded into a series of consecutively refined Cartesian grids, with the resolution from $2 \times 2 \times 2$ up to $32 \times 32 \times 32$. The homogeneous Dirichlet boundary is imposed on the (bottom) plane $x = L$, whereas the
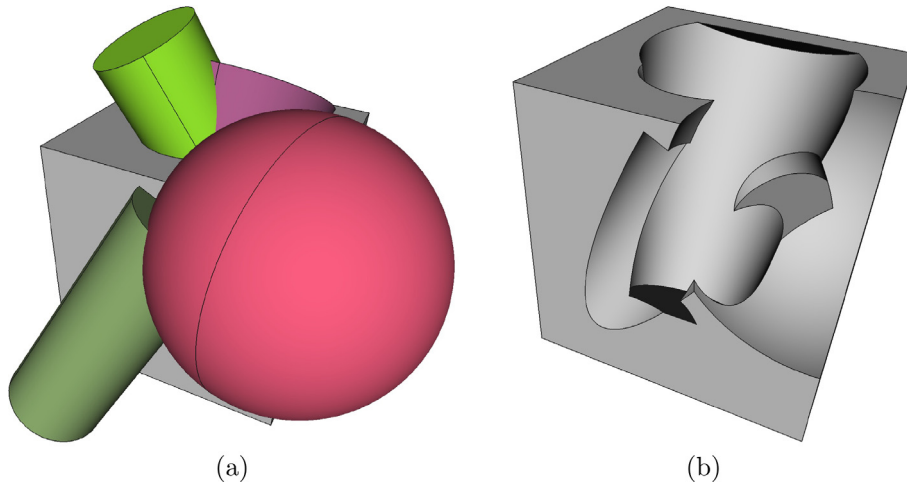
(a) (b)

Fig. 15. Construction of a complex geometry through trimming. (a) A cube to be cut by two cylinders (green), a cone (violet), and a sphere (pink), and (b) the resulting B-rep. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

non-homogeneous Neumann boundary condition is imposed elsewhere. On each Cartesian grid, we define B-spline discretizations of degrees $p \in \{1, 2, 3\}$. When creating integration cells for each trimmed element, we consider two options for the seed vertex: (1) one of the polyhedral vertices (recall that a trimmed element is a curved polyhedron), and (2) the origin of each element. Among all the possible decompositions in Option (1), we choose the one that leads to the minimum number of cells. Option (2) is meant to create highly distorted cells and it guarantees the presence of folded cells for this particular geometry. In each cell, $p + r$ ($r = 1, 3, 5$) Gauss quadrature points are used per direction. The convergence results are summarized in Fig. 14(b–g), where we compare the two options of decompositions. We observe that for $p \leq 2$, stable results are achieved with $r = 3$ and that both decompositions yield almost indistinguishable convergence results, while $r = 5$ is required for $p = 3$. Optimal convergence rates are achieved with all the B-spline discretizations. This implies that the accuracy and convergence are independent of how we create decompositions.

Next, we consider a complex geometry shown in Fig. 15, where a cube $[0, L]^3$ ($L = 10$) is cut by a sphere, a cone, and two cylinders. The corresponding B-rep is shown in Fig. 15(b), which contains several sharp curves/corners. Such features can be respected with integration cells for accurate integration. When creating a decomposition for each trimmed element, the seed vertex is chosen to be one of the element vertices. Among all the possible decompositions, we choose the one that leads to the minimum number of cells. The homogeneous Dirichlet boundary is imposed on the plane $z = 0$, whereas the non-homogeneous Neumann boundary condition is imposed elsewhere. All the other problem settings, including Cartesian grids, corresponding spline discretizations, and the number of quadrature points per direction, follow the same as those in the previous example.

In Fig. 16, we show a series of Cartesian grids and the corresponding trimmed elements. Each color represents a single trimmed element, which is generally decomposed into a collection of reparameterized cells for integration. We observe that such cells well align with the sharp features in the input B-rep. In addition, in Fig. 16(a–d) it can be seen that face-wise decompositions are not conformal. The solution field using the $16 \times 16 \times 16$ Cartesian grid is shown in Fig. 16(e). Moreover, the convergence plots are shown in Fig. 17, where we again observe optimal convergence rates for linear, quadratic and cubic spline discretizations.

As the last example, we present an engine model obtained from the gallery of OpenCASCADE [48] to show that our proposed method can robustly accommodate real-world geometries. Its B-rep has many detailed features that are represented by 918 trimmed spline surfaces in total. The B-rep is embedded into a series of Cartesian grids with resolutions $8 \times 8 \times 8$, $16 \times 16 \times 16$, and $32 \times 32 \times 32$, on which we solve Poisson's problem using tricubic B-splines. The numerical solutions are shown in Fig. 18. This demonstrates that with folded decompositions, we can flexibly reparameterize trimmed elements while retaining all the important geometric features.
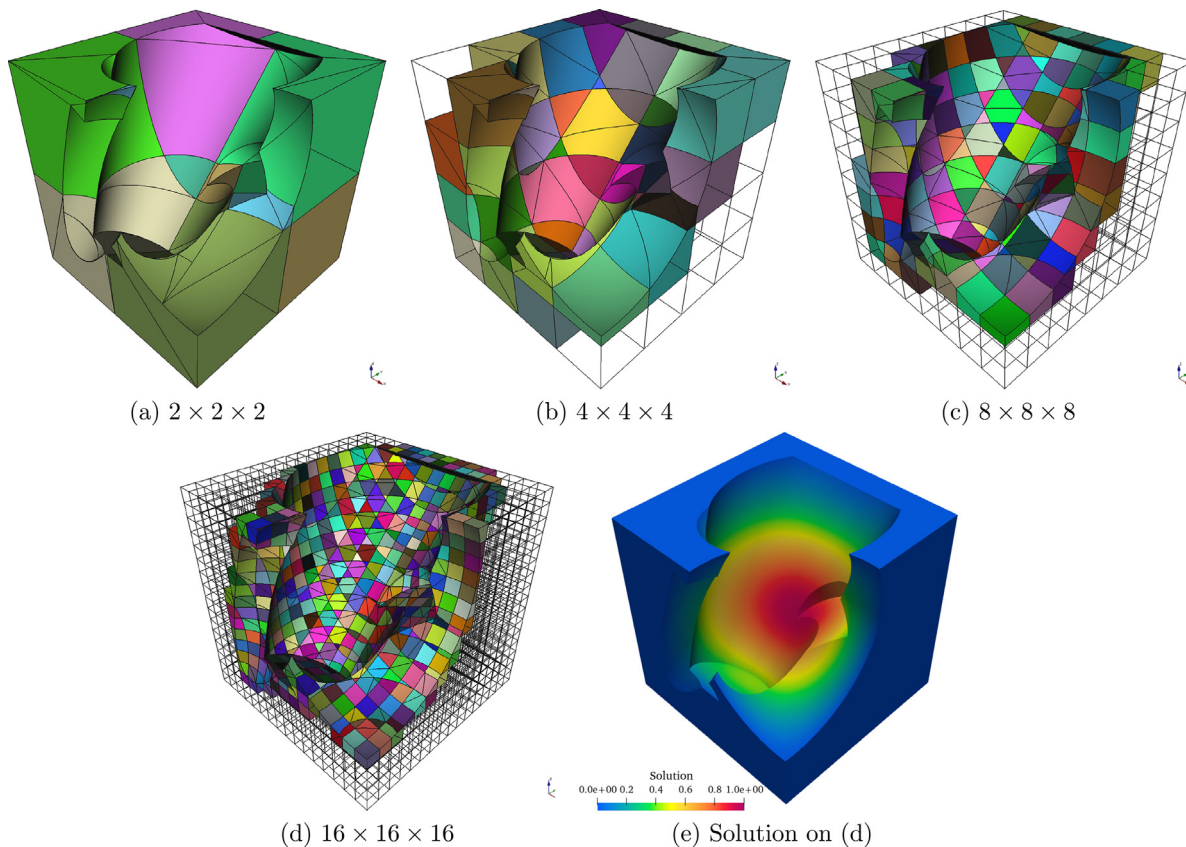
(a) $2 \times 2 \times 2$

(b) $4 \times 4 \times 4$

(c) $8 \times 8 \times 8$

(d) $16 \times 16 \times 16$

(e) Solution on (d)

Fig. 16. The B-rep embedded into a series of consecutively refined Cartesian grids (a–d), and the solution field (e) obtained using the mesh in (d). In (a–d), only trimmed elements are shown and each color represents a trimmed element. The black curves represent face-wise decompositions for the boundaries of trimmed elements as well as the outlines of non-trimmed elements. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(a)

(b)
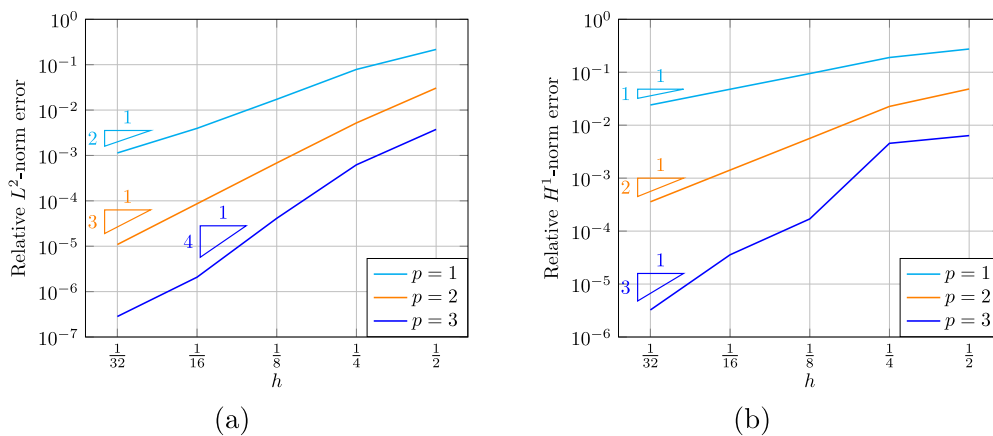
Fig. 17. Solving Poisson's problem on a domain shown in Fig. 15. (a, b) Convergence plots of relative L^2- and H^1-norm errors, respectively, with $r = 5$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
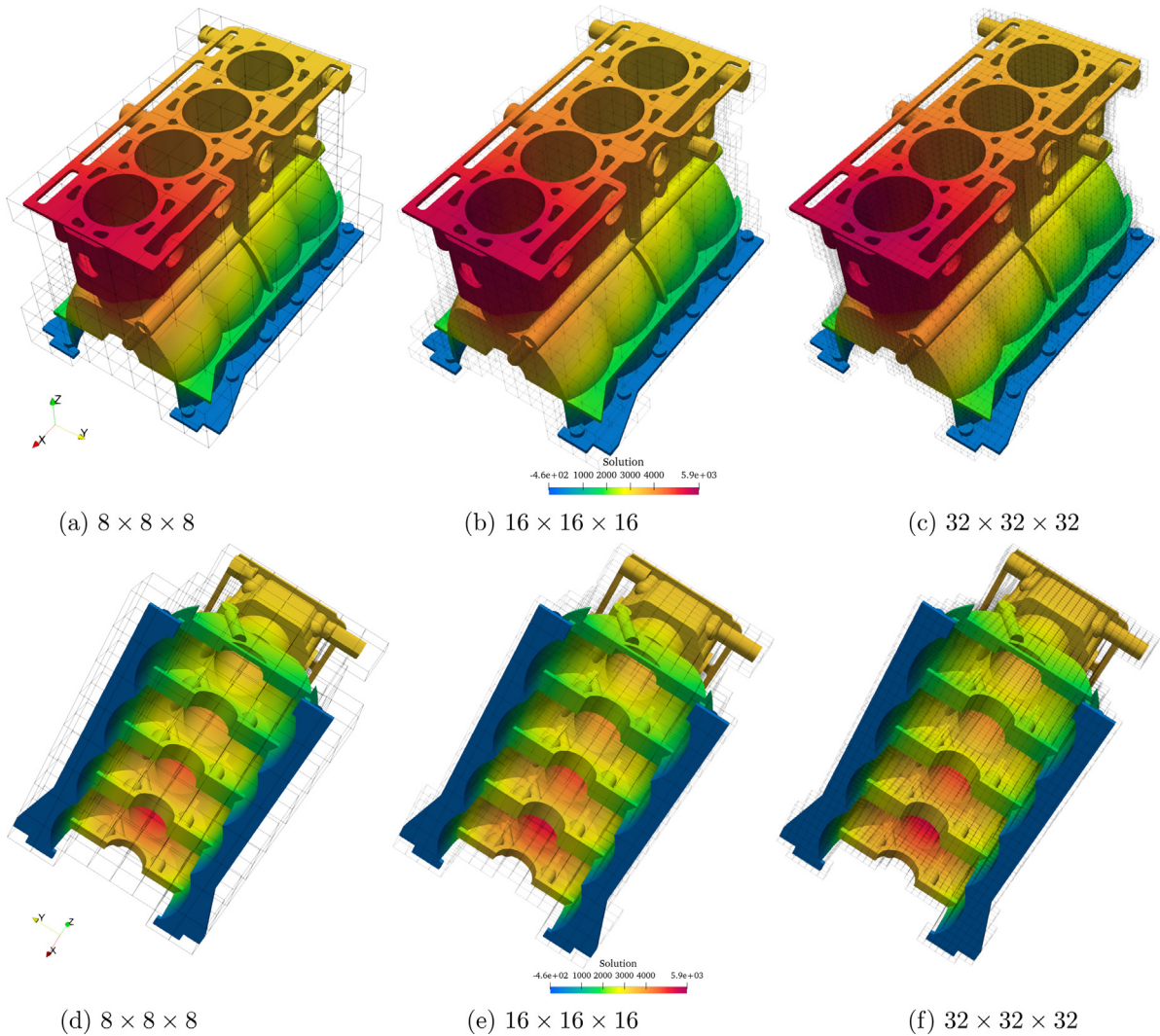
(a) $8 \times 8 \times 8$ (b) $16 \times 16 \times 16$ (c) $32 \times 32 \times 32$

(d) $8 \times 8 \times 8$ (e) $16 \times 16 \times 16$ (f) $32 \times 32 \times 32$

Fig. 18. Solving Poisson's problem on an engine model with different resolutions of Cartesian grids. (a–b) show a top view of the solution for different discretizations, whereas (d–f) show the solution seen from the bottom. The underlying embedding domain is represented with black lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

8. Conclusions and future work

In this paper, we propose to use folded decompositions for numerical integration over complex domains, which essentially eliminates constraints upon candidate decompositions and thus provides a significant flexibility to accommodate real-world geometries in engineering simulation, while retaining a relatively small number of integration cells. A domain of interest usually has a B-rep in 3D. A corresponding volumetric decomposition is created following two steps. First, the B-rep is decomposed into smaller surface patches, which respect possible sharp geometric features and do not need to be conformal. Such surface patches are generally approximated with Bézier surfaces. Second, the domain is decomposed into a collection of pyramidal cells using the surface patches and a seed vertex chosen in a rather arbitrary manner. The resulting pyramids serve as the integration cells where standard Gauss quadrature can be applied. Generally, such cells present negative Jacobian because there is no constraint on the choice of the seed vertex.

The validity of these unconstrained decompositions is theoretically guaranteed. We have performed two groups of numerical tests to numerically assess the validity and robustness of folded decompositions: integration of functions

and solving partial differential equations (PDEs) using isogeometric analysis (IGA). In both cases, we compare the performance of folded decompositions with that of J^+ decompositions. In the case of integrating polynomials, we have shown that with a sufficient number of quadrature points, machine precision can always be reached using folded decompositions. On the other hand, in the case of solving PDEs, we observe that the performance of folded and J^+ decompositions is almost indistinguishable in terms of accuracy and convergence, as long as a sufficient number of quadrature points are used. However, due to its great flexibility, folded decompositions can accommodate very complex geometries that involve various kinds of geometric features.

In summary, the proposed methodology allows the creation of high-order decompositions of B-reps for integration purposes in a very simple way, while yielding results analogous in terms of accuracy to the ones obtained using J^+ decompositions. Nevertheless, while folded decompositions provide a significant flexibility and robustness to accommodate real-world complex geometries, there are limitations, for instance, when we use them to visualize simulation results. Therefore, in the future, we plan to further exploit possible decomposition schemes such that we can minimize the number of folded integration cells while being able to deliver J^+ cells at its best.

Declaration of competing interest

Acknowledgments

Appendix. Number of quadrature points per direction

In this Appendix we study the number of quadrature points needed per direction to *exactly* integrate a given polynomial $f(\mathbf{x})$ over a certain domain Ω, i.e., to compute $I = \int_\Omega f(\mathbf{x})d\mathbf{x}$ (Eq. (1)). Let $\mathbb{Q}_{p,p}(\Omega)$ and $\mathbb{Q}_{p,p,p}(\Omega)$ denote bi-degree-p and tri-degree-p polynomial spaces defined on Ω, respectively, and we assume $f(\mathbf{x}) \in \mathbb{Q}_{p,p}(\Omega)$ in 2D or $f(\mathbf{x}) \in \mathbb{Q}_{p,p,p}(\Omega)$ in 3D. We note that a similar result was reported in 2D for NURBS-enhanced finite element methods [47].

We adopt the following three assumptions to facilitate the study of exact polynomial integration, which, however, are not needed elsewhere in the paper. First, we assume that Ω can be exactly represented by a set of cells, $\mathbf{T}_\ell : \Lambda = [0, 1]^d \to \mathcal{T}_\ell$. Second, $\mathbf{T}_\ell$ is assumed to be a Bézier patch that has a polynomial representation. In other words, $\mathbf{C}_i(u)$ in (7) and $\mathbf{S}_i \circ \mathbf{R}_{i,j}(u, v)$ in (16) are Bézier curves and surfaces, respectively. Third, for simplicity, $\mathbf{T}_\ell$ is further assumed to have an identical degree (q) in all directions, i.e., $\mathbf{T}_\ell \in \mathbb{Q}_{q,q}(\Lambda) \times \mathbb{Q}_{q,q}(\Lambda)$ for $d = 2$ or $\mathbf{T}_\ell \in \mathbb{Q}_{q,q,q}(\Lambda) \times \mathbb{Q}_{q,q,q}(\Lambda) \times \mathbb{Q}_{q,q,q}(\Lambda)$ for $d = 3$.

Therefore, according to (2), we have

$$I = \int_\Omega f(\mathbf{x})d\mathbf{x} = \sum_{\ell=1}^{L} \int_\Lambda f \circ \mathbf{T}_\ell \det(\nabla \mathbf{T}_\ell) \, d\mathbf{u}. \tag{A.1}$$

When $d = 2$, $f \circ \mathbf{T}_\ell \in \mathbb{Q}_{2pq,2pq}(\Lambda)$, $\det(\nabla \mathbf{T}_\ell) \in \mathbb{Q}_{2q-1,2q-1}(\Lambda)$, and thus we have

$$f \circ \mathbf{T}_\ell \det(\nabla \mathbf{T}_\ell) \in \mathbb{Q}_{2q(p+1)-1,2q(p+1)-1}(\Lambda). \tag{A.2}$$

To exactly compute (A.2) with the tensor-product Gauss quadrature defined in Λ, we need the number of quadrature points per direction n to be sufficiently large such that

$$2n - 1 \geq 2q(p + 1) - 1 \iff n \geq q(p + 1). \tag{A.3}$$

Likewise when $d = 3$, $f \circ \mathbf{T}_\ell \in \mathbb{Q}_{3pq,3pq,3pq}(\Lambda)$ and $\det(\nabla \mathbf{T}_\ell) \in \mathbb{Q}_{3q-1,3q-1,3q-1}(\Lambda)$. We have

$$f \circ \mathbf{T}_\ell \det(\nabla \mathbf{T}_\ell) \in \mathbb{Q}_{3q(p+1)-1,3q(p+1)-1,3q(p+1)-1}(\Lambda), \tag{A.4}$$

and

$$2n - 1 \geq 3q(p+1) - 1 \iff n \geq \frac{3}{2}q(p+1). \tag{A.5}$$

Eqs. (A.3), (A.5) imply that for an exact integration, n can easily become very large when either p or q increases, due to the composition $f \circ \mathbf{T}_\ell$. However, exact integration of a polynomial is often not needed as long as the integration error does not dominate the application of interest. We have such examples in Section 7.

References

[1] A. Krishnamurthy, S. McMains, Accurate GPU-accelerated surface integrals for moment computation, Comput. Aided Des. 43 (10) (2011) 1284–1295, http://dx.doi.org/10.1016/j.cad.2011.06.020.

[2] E. Guendelman, R. Bridson, R. Fedkiw, Nonconvex rigid bodies with stacking, ACM Trans. Graph. 22 (3) (2003) 871—878, http://dx.doi.org/10.1145/882262.882358.

[3] N. Moës, J. Dolbow, T. Belytschko, A finite element method for crack growth without remeshing, Internat. J. Numer. Methods Engrg. 46 (1) (1999) 131–150, http://dx.doi.org/10.1002/(SICI)1097-0207(19990910)46:1<131::AID-NME726>3.0.CO;2-J.

[4] E. Burman, S. Claus, P. Hansbo, M.G. Larson, A. Massing, CutFEM: Discretizing geometry and partial differential equations, Internat. J. Numer. Methods Engrg. 104 (7) (2015) 472–501, http://dx.doi.org/10.1002/nme.4823.

[5] L. Beirão da Veiga, F. Brezzi, A. Cangiani, G. Manzini, L.D. Marini, A. Russo, Basic principles of virtual element methods, Math. Models Methods Appl. Sci. 23 (1) (2013) 199–214, http://dx.doi.org/10.1142/S0218202512500492.

[6] J. Parvizian, A. Düster, E. Rank, Finite cell method, Comput. Mech. 41 (2007) 121–133, http://dx.doi.org/10.1007/s00466-007-0173-y.

[7] R. Rangarajan, A. Lew, G.C. Buscaglia, A discontinuous-Galerkin-based immersed boundary method with non-homogeneous boundary conditions and its application to elasticity, Comput. Methods Appl. Mech. Engrg. 198 (17) (2009) 1513–1534, http://dx.doi.org/10.1016/j.cma.2009.01.018.

[8] D. Kamensky, M.-C. Hsu, D. Schillinger, J.A. Evans, A. Aggarwal, Y. Bazilevs, M.S. Sacks, T.J.R. Hughes, An immersogeometric variational framework for fluid–structure interaction: Application to bioprosthetic heart valves, Comput. Methods Appl. Mech. Engrg. 284 (2015) 1005–1053, http://dx.doi.org/10.1016/j.cma.2014.10.040.

[9] B. Marussig, J. Zechner, G. Beer, T.P. Fries, Stable isogeometric analysis of trimmed geometries, Comput. Methods Appl. Mech. Engrg. 316 (2017) 497–521, http://dx.doi.org/10.1016/j.cma.2016.07.040.

[10] A. Johansson, B. Kehlet, M.G. Larson, A. Logg, Multimesh finite element methods: Solving PDEs on multiple intersecting meshes, Comput. Methods Appl. Mech. Engrg. 343 (2019) 672–689, http://dx.doi.org/10.1016/j.cma.2018.09.009.

[11] A. Buffa, R. Puppi, R. Vázquez, A minimal stabilization procedure for isogeometric methods on trimmed geometries, SIAM J. Numer. Anal. 58 (5) (2020) http://dx.doi.org/10.1137/19M1244718.

[12] P. Antolin, A. Buffa, R. Puppi, X. Wei, Overlapping multi-patch isogeometric method with minimal stabilization, SIAM J. Sci. Comput. 43 (1) (2021) A330–A354, http://dx.doi.org/10.1137/19M1306750.

[13] M. Ruess, D. Schillinger, Y. Bazilevs, V. Varduhn, E. Rank, Weakly enforced essential boundary conditions for NURBS-embedded and trimmed NURBS geometries on the basis of the finite cell method, Internat. J. Numer. Methods Engrg. 95 (10) (2013) 811–846, http://dx.doi.org/10.1002/nme.4522.

[14] H.G. Timmer, J.M. Stern, Computation of global geometric properties of solid objects, Comput. Aided Des. 12 (6) (1980) 301–304, http://dx.doi.org/10.1016/0010-4485(80)90093-7.

[15] Y. Sudhakar, J.P. Moitinho de Almeida, W.A. Wall, An accurate, robust, and easy-to-implement method for integration over arbitrary polyhedra: Application to embedded interface methods, J. Comput. Phys. 273 (2014) 393–415, http://dx.doi.org/10.1016/j.jcp.2014.05.019.

[16] E.B. Chin, N. Sukumar, An efficient method to integrate polynomials over polytopes and curved solids, Comput. Aided Geom. Design 82 (2020) 101914, http://dx.doi.org/10.1016/j.cagd.2020.101914.

[17] D. Gunderman, K. Weiss, J.A. Evans, Spectral mesh-free quadrature for planar regions bounded by rational parametric curves, Comput. Aided Des. 130 (2021) 102944, http://dx.doi.org/10.1016/j.cad.2020.102944.

[18] Pablo Antolin, Thibaut Hirschler, Quadrature-free immersed isogeometric analysis, Eng. Comput. In press (2022) http://dx.doi.org/10.1007/s00366-022-01644-3.

[19] H. Xiao, Z. Gimbutas, A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions, Comput. Math. Appl. 59 (2) (2010) 663–676, http://dx.doi.org/10.1016/j.camwa.2009.10.027.

[20] S.E. Mousavi, N. Sukumar, Numerical integration of polynomials and discontinuous functions on irregular convex polygons and polyhedrons, Comput. Mech. 47 (2011) 535—554, http://dx.doi.org/10.1007/s00466-010-0562-5.

[21] Y. Sudhakar, W.A. Wall, Quadrature schemes for arbitrary convex/concave volumes and integration of weak form in enriched partition of unity methods, Comput. Methods Appl. Mech. Engrg. 258 (2013) 39–54, http://dx.doi.org/10.1016/j.cma.2013.01.007.

[22] B. Müller, F. Kummer, M. Oberlack, Highly accurate surface and volume integration on implicit domains by means of moment-fitting, Internat. J. Numer. Methods Engrg. 96 (8) (2013) 512–528, http://dx.doi.org/10.1002/nme.4569.

[23] M. Joulaian, S. Hubrich, A. Düster, Numerical integration of discontinuities on arbitrary domains based on moment fitting, Comput. Mech. 57 (2016) 979—999, http://dx.doi.org/10.1007/s00466-016-1273-3.

[24] D. Gunderman, K. Weiss, J.A. Evans, High-accuracy mesh-free quadrature for trimmed parametric surfaces and volumes, Comput. Aided Des. 141 (2021) 103093, http://dx.doi.org/10.1016/j.cad.2021.103093.

[25] S. Hubrich, P. Di Stolfo, L. Kudela, S. Kollmannsberger, E. Rank, A. Schröder, A. Düster, Numerical integration of discontinuous functions: moment fitting and smart octree, Comput. Mech. 60 (5) (2017) 863–881, http://dx.doi.org/10.1007/s00466-017-1441-0.

[26] S. Hubrich, A. Düster, Numerical integration for nonlinear problems of the finite cell method using an adaptive scheme based on moment fitting, Comput. Math. Appl. 77 (7) (2019) 1983–1997, http://dx.doi.org/10.1016/j.camwa.2018.11.030.

[27] H.G. Bui, D. Schillinger, G. Meschke, Efficient cut-cell quadrature based on moment fitting for materially nonlinear analysis, Comput. Methods Appl. Mech. Engrg. 366 (2020) 113050, http://dx.doi.org/10.1016/j.cma.2020.113050.

[28] D. Schillinger, M. Ruess, The finite cell method: A review in the context of higher-order structural analysis of CAD and image-based geometric models, Arch. Comput. Methods Eng. 22 (2015) 391—455, http://dx.doi.org/10.1007/s11831-014-9115-y.

[29] A. Düster, J. Parvizian, Z. Yang, E. Rank, The finite cell method for three-dimensional problems of solid mechanics, Comput. Methods Appl. Mech. Engrg. 197 (45) (2008) 3768–3782, http://dx.doi.org/10.1016/j.cma.2008.02.036.

[30] S.C. Divi, C.V. Verhoosel, F. Auricchio, A. Reali, E.H. van Brummelen, Error-estimate-based adaptive integration for immersed isogeometric analysis, Comput. Math. Appl. 80 (11) (2020) 2481–2516, http://dx.doi.org/10.1016/j.camwa.2020.03.026.

[31] M. Petö, F. Duvigneau, S. Eisenträger, Enhanced numerical integration scheme based on image-compression techniques: Application to fictitious domain methods, Adv. Model. Simul. Eng. Sci. 7 (21) (2020) http://dx.doi.org/10.1186/s40323-020-00157-2.

[32] V. Thiagarajan, V. Shapiro, Adaptively weighted numerical integration over arbitrary domains, Comput. Math. Appl. 67 (9) (2014) 1682–1702, http://dx.doi.org/10.1016/j.camwa.2014.03.001.

[33] F. Scholz, A. Mantzaflaris, B. Jüttler, First order error correction for trimmed quadrature in isogeometric analysis, in: Advanced Finite Element Methods with Applications: Selected Papers from the 30th Chemnitz Finite Element Symposium 2017, Springer International Publishing, 2019, pp. 297–321, http://dx.doi.org/10.1007/978-3-030-14244-5_15.

[34] O. Marco, R. Sevilla, Y. Zhang, J.J. Ródenas, M. Tur, Exact 3D boundary representation in finite element analysis based on cartesian grids independent of the geometry, Internat. J. Numer. Methods Engrg. 103 (6) (2015) 445–468, http://dx.doi.org/10.1002/nme.4914.

[35] L. Kudela, N. Zander, S. Kollmannsberger, E. Rank, Smart octrees: Accurately integrating discontinuous functions in 3D, Comput. Methods Appl. Mech. Engrg. 306 (2016) 406–426, http://dx.doi.org/10.1016/j.cma.2016.04.006.

[36] C.-K. Im, S.-K. Youn, The generation of 3D trimmed elements for NURBS-based isogeometric analysis, Int. J. Comput. Methods 15 (7) (2018) 1850065, http://dx.doi.org/10.1142/S0219876218500652.

[37] P. Antolin, A. Buffa, M. Martinelli, Isogeometric analysis on V-reps: First results, Comput. Methods Appl. Mech. Engrg. 355 (2019) 976–1002, http://dx.doi.org/10.1016/j.cma.2019.07.015.

[38] F. Massarwi, P. Antolin, G. Elber, Volumetric untrimming: Precise decomposition of trimmed trivariates into tensor products, Comput. Aided Geom. Design 71 (2019) 1–15, http://dx.doi.org/10.1016/j.cagd.2019.04.005.

[39] C. Gürkan, M. Kronbichler, S. Fernández-Méndez, Extended hybridizable discontinuous Galerkin for incompressible flow problems with unfitted meshes and interfaces, Internat. J. Numer. Methods Engrg. 117 (7) (2019) 756–777, http://dx.doi.org/10.1002/nme.5975.

[40] R. Sevilla, S. Fernández-Méndez, A. Huerta, NURBS-enhanced finite element method (NEFEM): A seamless bridge between CAD and FEMs, Arch. Comput. Methods Eng. 18 (4) (2011) 441–484, http://dx.doi.org/10.1007/s11831-011-9066-5.

[41] E.B. Chin, N. Sukumar, Scaled boundary cubature scheme for numerical integration over planar regions with affine and curved boundaries, Comput. Methods Appl. Mech. Engrg. 380 (2021) 113796, http://dx.doi.org/10.1016/j.cma.2021.113796.

[42] S.A. Coons, Surfaces for Computer-Aided Design of Space Forms, Technical Report Technical Report MAC-TR-41, MIT, Massachusetts Institute of Technology, 1967.

[43] T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (39) (2005) 4135–4195, http://dx.doi.org/10.1016/j.cma.2004.10.008.

[44] L. Beirão da Veiga, A. Buffa, G. Sangalli, R. Vázquez, Analysis-suitable T-splines of arbitrary degree: Definition, linear independence and approximation properties, Comput. Methods Appl. Mech. Engrg. 23 (11) (2013) 1979–2003, http://dx.doi.org/10.1142/S0218202513500231.

[45] D. Schillinger, L. Dedè, M.A. Scott, J.A. Evans, M.J. Borden, E. Rank, T.J.R. Hughes, An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces, Comput. Methods Appl. Mech. Engrg. 249–252 (2012) 116–150, http://dx.doi.org/10.1016/j.cma.2012.03.017.

[46] X. Wei, B. Marussig, P. Antolin, A. Buffa, Immersed boundary-conformal isogeometric method for linear elliptic problems, Comput. Mech. (2021) http://dx.doi.org/10.1007/s00466-021-02074-6.

[47] R. Sevilla, S. Fernández-Méndez, Numerical integration over 2D NURBS-shaped domains with applications to NURBS-enhanced FEM, Finite Elem. Anal. Des. 47 (10) (2011) 1209–1220, http://dx.doi.org/10.1016/j.finel.2011.05.011.

[48] Open CASCADE SAS. Open CASCADE technology (version 7.5.0), 2020, https://www.opencascade.com/open-cascade-technology.