Scan-Based Immersed Isogeometric Flow Analysis

Verhoosel, Clemens V.; Harald van Brummelen, E.; Divi, Sai C.; de Prenter, Frits

Citation (APA)
Verhoosel, C. V., Harald van Brummelen, E., Divi, S. C., & de Prenter, F. (2023). Scan-Based Immersed Isogeometric Flow Analysis. In T. E. Tezduyar (Ed.), *Modeling and Simulation in Science, Engineering and Technology* (pp. 477-512). (Modeling and Simulation in Science, Engineering and Technology; Vol. Part F1665). Birkhauser Verlag Basel. https://doi.org/10.1007/978-3-031-36942-1_14

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Scan-Based Immersed Isogeometric Flow Analysis

Clemens V. Verhoosel, E. Harald van Brummelen, Sai C. Divi, and Frits de Prenter

Abstract This chapter reviews the work conducted by our team on scan-based immersed isogeometric analysis for flow problems. To leverage the advantageous properties of isogeometric analysis on complex scan-based domains, various innovations have been made: (i) A spline-based segmentation strategy has been developed to extract a geometry suitable for immersed analysis directly from scan data; (ii) A stabilized equal-order velocity-pressure formulation for the Stokes problem has been proposed to attain stable results on immersed domains; (iii) An adaptive integration quadrature procedure has been developed to improve computational efficiency; (iv) A mesh refinement strategy has been developed to capture small features at a priori unknown locations, without drastically increasing the computational cost of the scan-based analysis workflow. We review the key ideas behind each of these innovations, and illustrate these using a selection of simulation results from our work. A patient-specific scan-based analysis case is reproduced to illustrate how these innovations enable the simulation of flow problems on complex scan data.

1 Introduction

The rapid developments in the field of scientific computing have opened the doors to performing computational analyses on data obtained using advanced scanning technologies (e.g., tomography or photogrammetry). Such analyses are of particular interest in applications pertaining to non-engineered systems, which are common in, for example, biomechanics, geomechanics and material science.

C. V. Verhoosel (✉) · E. Harald van Brummelen · S. C. Divi
Eindhoven University of Technology, Eindhoven, The Netherlands
e-mail: c.v.verhoosel@tue.nl; e.h.v.brummelen@tue.nl; s.c.divi@tue.nl

F. de Prenter
Delft University of Technology, Delft, The Netherlands
e-mail: f.deprenter@tudelft.nl

477

For scan-based simulations, the data sets from which the geometric models are constructed are typically very large, and the obtained models can be very complex in terms of both geometry and topology (see Fig. 1). In the context of standard finite element analyses (FEA), scan-based simulations require image segmentation and meshing techniques to produce high-quality analysis-suitable meshes that fit to the boundaries of the domain of interest. The construction of a FEA-suitable computational domain can be an error-prone and laborious process, involving manual geometry clean-up and mesh repairing and optimization operations. Such
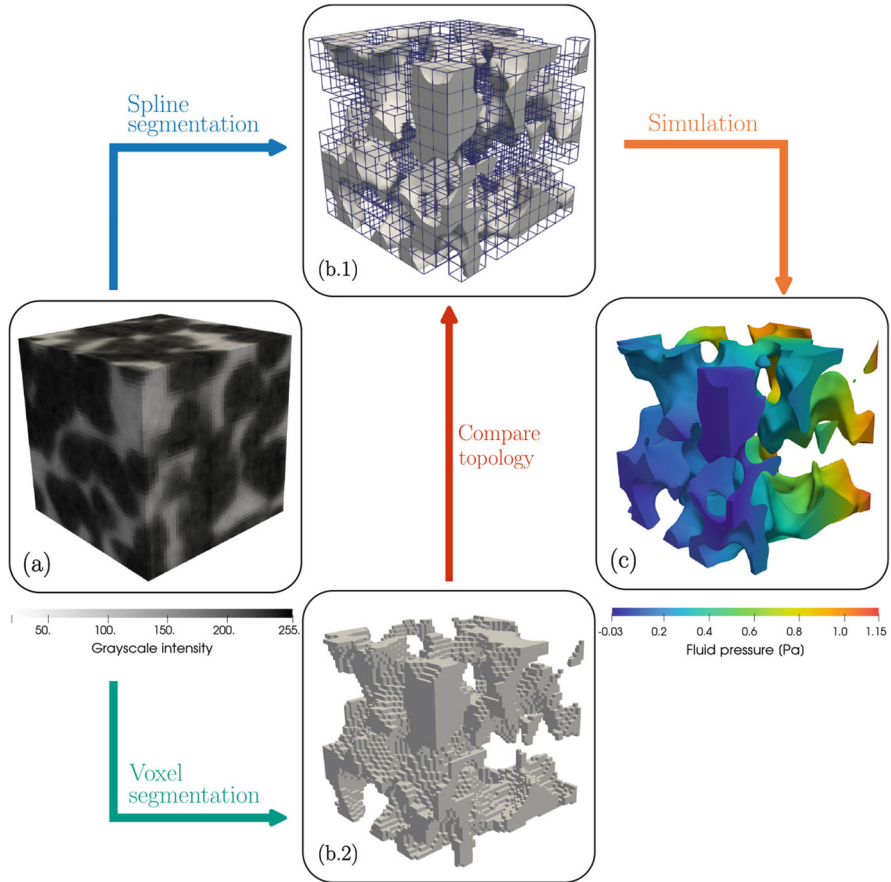
Fig. 1 Illustration of the developed scan-based simulation workflow, considering a sintered glass specimen [28] as a typical example. The grayscale scan data is shown in panel (**a**). A smooth reconstruction of the geometry (representing the void space) based on a spline segmentation is shown in panel (**b.1**), along with the computational mesh in which the geometry is submersed. The directly segmented voxel image, which is used to assess the topological correctness of the spline segmentation, is shown in panel (**b.2**). A typical immersed isogeometric analysis result is shown in panel (**c**)

operations can account for the majority of the total computational analysis time and form a bottleneck in the automation of scan-based simulation workflows [1].

The challenges associated with the simulation workflow for complex problems sparked the development of the isogeometric analysis (IGA) paradigm by Hughes and co-workers in 2005 [2]. The pivotal idea of IGA is to directly employ the geometry interpolation functions used in computer-aided design (e.g., B-splines and NURBS [3]) for the discretization of boundary value problems, thereby circumventing the problems associated with meshing. Besides the advantage of avoiding the meshing procedure and eliminating mesh-approximation errors, the use of higher-order continuous splines for the approximation of the solution has been demonstrated to yield accurate results using relatively few degrees of freedom for many (smooth) problems (see Ref. [4] for an overview). While isogeometric analysis has been successfully applied to complex three-dimensional problems based on (multi-patch) CAD objects (see, e.g., Refs. [5–10]), its application to scan-based simulations is hindered by the absence of analysis-suitable spline-based geometry models. Although spline preprocessors have been developed over the years for a range of applications [11–13], the robust generation of analysis-suitable boundary-fitting volumetric splines for scan-based analyses is beyond the scope of the current tools on account of the geometrical and topological complexity typically inherent to scan data.

To still leverage the advantageous approximation properties of splines in scan-based simulations, IGA is often used in combination with immersed methods. In immersed methods, a non-boundary-fitting mesh is considered, in which the computational domain is submersed. Since the immersed domain does not align with the computational grid, some of the elements in the grid are cut by the immersed boundary and require a special treatment. The immersed approach has been considered in the finite element setting in the context of the Finite Cell Method (FCM) [14–16] and CutFEM [17–19], amongst others. The immersed concept has also been used in combination with IGA [20–22], a strategy which is sometimes referred to as immersogeometric analysis [23, 24]. The versatility of immersed isogeometric analysis techniques with respect to the geometry representation—in the sense that the analysis procedure is not strongly affected by the complexity of the physical domain—makes it particularly attractive in the scan-based analysis setting. Applications can nowadays be found in, for example, the modeling of trabecular bone [25–27], porous media [28], coated metal foams [29], metal castings [30] and additive manufacturing [31].

Over the past decade our team has developed an analysis workflow using the immersed isogeometric analysis paradigm. This workflow is illustrated in Fig. 1. In this chapter we review the key research contributions that made this analysis workflow applicable to scan-based flow simulations:

- A **spline-based geometry segmentation** technique was proposed in Verhoosel et al. [25], with further improvements being made by Divi et al. [32, 33]. The pivotal idea of the developed segmentation strategy is that the original scan data, which is usually non-smooth (i.e., a voxel representation), is smoothed using a spline

approximation. A segmentation procedure able to provide an accurate explicit parametrization of the smoothed geometry then provides a geometric description of the scan object which is suitable for immersed isogeometric analysis.

- A **stabilized immersed isogeometric analysis** formulation for flow problems was proposed by Hoang et al. [28]. The key idea of the proposed formulation is to use face-based stabilization techniques to make immersed simulations robust with respect to (unfavorably) cut elements, preventing the occurrence of oscillations in the velocity and pressure approximations. The stabilization terms also enable the consideration of equal-order discretizations of the velocity and pressure fields, which would otherwise cause inf-sup stability problems even in boundary-fitting finite elements [34].
- An **adaptive integration procedure** was developed by Divi et al. [32] to reduce the computational cost involved in the evaluation of integrals over cut elements, thereby improving the computational efficiency of the immersed analysis workflow. Based on Strang's lemma [35], an estimator for the integration error is derived, which is then used to optimally distribute integration quadrature points over cut elements.
- An **error-estimation-based adaptive refinement** procedure has been developed by Divi et al. [36] to capture small features without drastically increasing the computational cost of the scan-based workflow. Residual-based error estimators are constructed to perform local basis function refinements to increase the resolution of the spline basis in regions where this is particularly beneficial from an accuracy point of view, without prior knowledge of the locations of these regions.

Our scan-based immersed isogeometric analysis workflow has been applied to a range of real world data problems, mainly in the context of μCT-scans. In this chapter we illustrate the capabilities of our workflow in the context of patient-specific arterial flow problems. The analysis of porous medium flows as presented in Ref. [28], and illustrated in Fig. 1, forms another prominent application of our method.

This chapter is organized as follows. The essential innovations regarding each of the research contributions listed above are reviewed in Sects. 2–5. A typical application of the developed workflow will then be discussed in Sect. 6. We will conclude this chapter in Sect. 7 with an assessment of our scan-based analysis workflow, discussing its capabilities and current limitations.

2 Spline-Based Geometry Segmentation

In this section we review the spline-based image segmentation procedure that we have developed in the context of scan-based immersed isogeometric analysis [25, 32, 33]. In Sect. 2.1 we first discuss the spline-based level set construction to smoothen scan data. In Sect. 2.2 we review the algorithms used to construct

an explicit parametrization of the scan domain. An example is finally shown in Sect. 2.3, illustrating the effectivity of the topology-preservation procedure developed in Ref. [33].

2.1 B-spline Smoothing of the Scan Data

The spline-based level set construction is illustrated in Fig. 2. We consider a d-dimensional scan domain, $\Omega_{\text{scan}} = [0, L_1] \times \ldots \times [0, L_d]$ with volume $V_{\text{scan}} = \prod_{i=1}^{n_d} L_i$, which is partitioned by a set of m_{vox} voxels, as illustrated in Fig. 2a. We denote the voxel mesh by $\mathcal{T}_{\text{scan}}^{\Delta}$, with Δ the voxel size in each direction. The grayscale intensity function is then defined as

$$g : \mathcal{T}_{\text{scan}}^{\Delta} \to \mathcal{G}, \tag{1}$$

with $\mathcal{G}$ the range of the grayscale data (e.g., from 0 to 255 for 8 bit unsigned integers). An approximation of the object Ω can be obtained by thresholding the grayscale data,

$$\Omega \approx \{x \in \Omega_{\text{scan}} | g(x) > g_{\text{crit}}\} \subset \Omega_{\text{scan}}, \tag{2}$$

where g_{crit} is the threshold value. As a consequence of the piecewise definition of the grayscale data in Eq. (1), the boundary of the segmented object is non-smooth when the grayscale data is segmented directly. In the context of analysis, the non-smoothness of the boundary can be problematic, as irregularities in the surface may lead to non-physical features in the solution to the problem.

The spline-based segmentation procedure developed in Refs. [25, 32, 33] enables the construction of a smooth boundary approximation based on voxel data. The key idea of this spline-based segmentation technique is to smoothen the grayscale
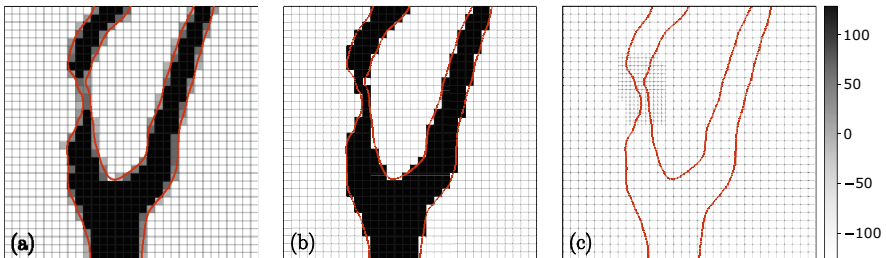
Fig. 2 A two-dimensional illustration of the spline-based segmentation procedure. (**a**) Grayscale data on a 32×32 voxel mesh, $\mathcal{T}_{\text{scan}}^{\Delta}$, as in (1). (**b**) The voxel segmentation obtained by thresholding the grayscale data as in (2). (**c**) The smooth level set function obtained using a THB-spline basis constructed on the voxel mesh with local refinements as in (3). The exact immersed boundary, which is in principle unknown, is shown in red for reference

function (1) by convoluting it using an n-dimensional spline basis, $\{N_{i,k}(\mathcal{T}_{\text{scan}}^h)\}_{i=1}^n$, defined over a mesh, $\mathcal{T}_{\text{scan}}^h$, with element size, $\boldsymbol{h} = (h_1, \ldots, h_d)$ (note that the mesh size can differ from the voxel size). The order k of the spline basis functions is assumed to be constant and isotropic. We consider THB-splines [37] for the construction of locally refined spaces. By considering full-regularity (C^{k-1}-continuous) splines of degree $k > 1$, a smooth level set approximation of (1) is obtained by the convolution operation

$$f(\boldsymbol{x}) = \sum_{i=1}^n N_{i,k}(\boldsymbol{x})a_i, \qquad a_i = \frac{\int_{\Omega_{\text{scan}}} N_{i,k}(\boldsymbol{x})g(\boldsymbol{x})\mathrm{d}\boldsymbol{x}}{\int_{\Omega_{\text{scan}}} N_{i,k}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}}, \qquad (3)$$

where $\{a_i\}_{i=1}^n$ are the coefficients of the discrete level set function. The smoothed domain then follows by thresholding of this level set function:

$$\Omega \approx \{\boldsymbol{x} \in \Omega_{\text{scan}} | f(\boldsymbol{x}) > f_{\text{crit}}\} \subset \Omega_{\text{scan}}. \qquad (4)$$

The spline level set function corresponding to the voxel data in Fig. 2a is illustrated in Fig. 2c for the case of a locally refined mesh $\mathcal{T}_{\text{scan}}^h$ and second order ($k = 2$) THB-splines. As can be seen, the object retrieved from the convoluted level set function more closely resembles the original geometry in Fig. 2a compared to the voxel segmentation in Fig. 2b. Also, as a consequence of the higher-order continuity of the spline basis, the boundaries of the domain are smooth, which is in closer agreement with reality.

The convolution operation (3) is computationally efficient, resulting from the fact that it is not required to solve a linear system of equations (in contrast to a (global) L^2-projection) and the restricted support of the convolution kernel. Moreover, the convolution strategy has various properties that are advantageous in the context of scan-based immersed isogeometric analysis (see Refs. [25, 33] for details):

Conservation of the Gray Scale Intensity
Under the condition that the spline basis, $\{N_{i,k}(\mathcal{T}_{\text{scan}}^h)\}_{i=1}^n$, satisfies the partition of unity property (e.g., B-splines, THB-splines), the smooth level set approximation (3) conserves the gray scale intensity of the original data in the sense that

$$\frac{1}{V_{\text{scan}}} \int_{\Omega_{\text{scan}}} f \, \mathrm{d}V = \frac{1}{V_{\text{scan}}} \int_{\Omega_{\text{scan}}} g \, \mathrm{d}V = \frac{1}{m_{\text{vox}}} \sum_{K \in \mathcal{T}_{\text{scan}}^\Delta} g(K). \qquad (5)$$

This property ensures that there is a direct relation between the threshold value, f_{crit}, for the smooth level set reconstruction (4) and that of the original data, g_{crit}, in Eq. (2).

Local Boundedness by the Original Data
On every voxel $K \in \mathcal{T}_{\text{scan}}^\Delta$, the level set function (3) is bound by the extrema of the voxel function over the support extension [38], $\widetilde{K}$, i.e.,

$$\min_{\boldsymbol{x}\in\tilde{K}} g(\boldsymbol{x}) \leq f(\boldsymbol{x}) \leq \max_{\boldsymbol{x}\in\tilde{K}} g(\boldsymbol{x}) \qquad\qquad \forall \boldsymbol{x} \in K. \qquad (6)$$

These bounds preclude overshoots and undershoots, which indicates that no spurious oscillations are created by the smoothing procedure (contrasting the case of an L^2-projection).

Approximate Gaussian Blurring

The spline-based convolution operation (3) can be written as an integral transform

$$f(\boldsymbol{x}) = \int_{\Omega_{\text{scan}}} \mathcal{K}(\boldsymbol{x}, \boldsymbol{y}) g(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y}, \qquad \mathcal{K}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} \frac{N_{i,k}(\boldsymbol{x}) N_{i,k}(\boldsymbol{y})}{\int_{\Omega_{\text{scan}}} N_{i,k}(\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z}}, \qquad (7)$$

where $\mathcal{K}(\boldsymbol{x}, \boldsymbol{y})$ is the kernel of the transformation.

The integral transform (7) acts as an approximate Gaussian filter [39]. We illustrate this behavior for the case of one-dimensional voxel data, which is smoothed using a B-spline basis defined on a uniform mesh, $\mathcal{T}^h_{\text{scan}}$, with mesh size h. Following the derivation in Ref. [25]—in which the essential step is to approximate the B-spline basis functions by rescaled Gaussians [40]—the integration kernel (7) can be approximated by

$$\mathcal{K}(x, y) \approx \kappa(x - y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-y)^2}{2\sigma^2}\right), \qquad (8)$$

where the width of the smoothing kernel is given by $\sigma = h\sqrt{\frac{k+1}{6}}$. Next, we consider an object of size ℓ, represented by the grayscale function

$$g(x) = \begin{cases} 1 & |x| < \ell/2 \\ 0 & \text{otherwise} \end{cases}. \qquad (9)$$

This object and the corresponding approximate level set function (3) are illustrated in Fig. 3 for various feature-size-to-mesh ratios, $\hat{\ell} = \ell/h = 2, 1, \frac{1}{2}$, and B-spline degrees, $k = 2, 3, 4$. Following the (Fourier) analysis in Ref. [33], the value of the smoothed level set function at $x = 0$ follows as

$$\hat{f}_1(0) = \hat{\ell}\sqrt{\frac{3}{\pi(k+1)}} \exp\left(-\frac{3\hat{\ell}^2}{16(k+1)}\right), \qquad (10)$$

which conveys that the maximum value of the smoothed level set depends linearly on the relative feature size $\hat{\ell}$ (for sufficiently small $\hat{\ell}$), and decreases with increasing B-spline order.

Figure 3a shows the case for which the considered feature is twice as large as the mesh size, i.e., $\hat{\ell} = 2$, illustrating that the sharp boundaries of the original grayscale
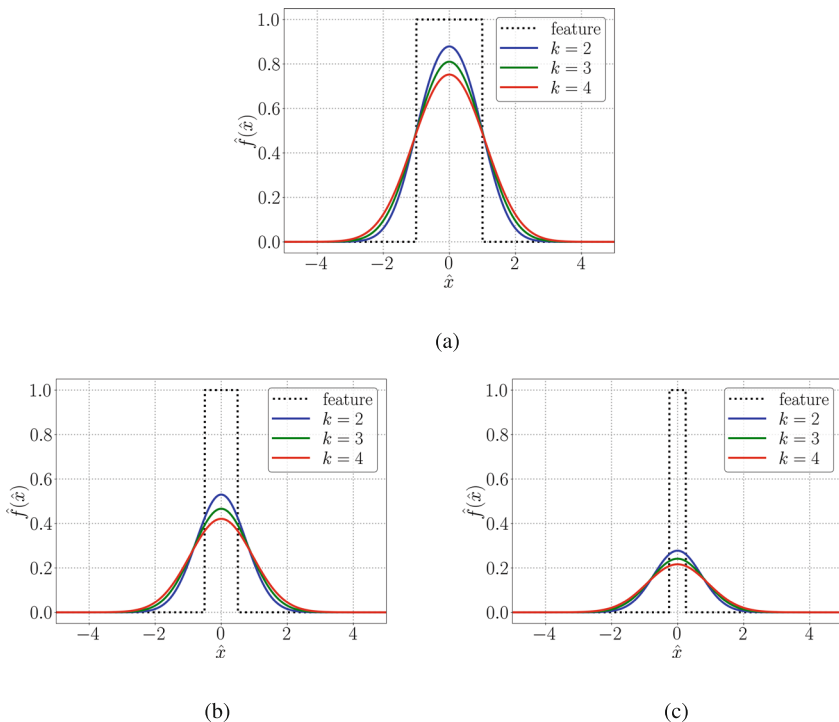
Fig. 3 Smoothed level set approximation (3) of a geometric feature in the spatial domain, with $\hat{x} = x/h$, for various feature-size-to-mesh ratios, $\hat{\ell} = \ell/h$, and B-spline degrees. (a) $\hat{\ell} = \frac{\ell}{h} = 2$. (b) $\hat{\ell} = \frac{\ell}{h} = 1$. (c) $\hat{\ell} = \frac{\ell}{h} = \frac{1}{2}$

function are significantly smoothed. The decrease in the maximum level set value as given by Eq. (10) is observed. When the level set function is segmented by a threshold of $g_{\mathrm{crit}} = 0.5$, a geometric feature that closely resembles the original one is recovered. Figure 3b-c illustrate cases where the feature length, ℓ, is not significantly larger than the mesh size, h. For the case where the feature size is equal to the size of the mesh, the maximum of the level set function drops significantly compared to the case of $\hat{\ell} = 2$. When considering second-order B-splines, the maximum is still marginally above $g_{\mathrm{crit}} = 0.5$. Although the recovered feature is considerably smaller than the original one, it is still detected in the segmentation procedure. When increasing the B-spline order, the maximum value of the level set drops below the segmentation threshold, however, indicating that the feature will no longer be detected. When decreasing the feature length further, as illustrated in Fig. 3c, the feature would be lost when segmentation is performed with $g_{\mathrm{crit}} = 0.5$, regardless of the order of the spline basis.

The implications of this smoothing behavior of the convolution operation (3) in the context of the spline-based segmentation will be discussed in Sect. 2.3.
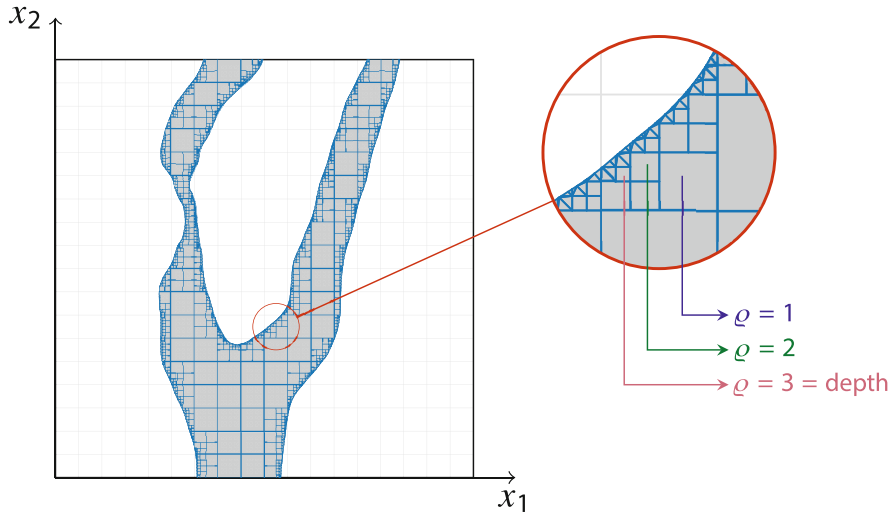
Fig. 4 Illustration of the octree-based tessellation procedure to acquire an explicit parametrization of the immersed computational domain. The number of element subdivisions is indicated by ϱ, which is equal to the octree *depth* at the lowest level of bisectioning

2.2 Octree-Based Tessellation Procedure

Our scan-based isogeometric analysis approach requires the construction of an explicit parametrization of the implicit level set domain (4). In this section we outline the segmentation procedure that we use to obtain an explicit parametrization of the domain and its (immersed) boundaries. This procedure, which is based on the octree subdivision approach introduced in the context of the Finite Cell Method in Ref. [15], is illustrated in Fig. 4. In Sect. 2.2.1 we first discuss the employed octree procedure, after which the tessellation procedure used at the lowest level of subdivision is detailed in Sect. 2.2.2. Without loss of generality, in the remainder we will assume that the level set function f is shifted such that $f_{\text{crit}} = 0$.

2.2.1 Octree Subdivision

In our analysis framework we consider a regular mesh $\mathcal{T}^h_{\text{scan}}$ that conforms to the scan domain Ω_{scan}. Each element in this mesh is a hyperrectangle (i.e., a line in one dimension, a rectangle in two dimensions, and a hexahedron in three dimensions) with size $\boldsymbol{h} = (h_1, \ldots, h_d)$ in each direction. Elements, K, that are intersected by the immersed boundary are trimmed, resulting in a partitioning, $\mathcal{P}_K$, of a cut element (see Fig. 4).

We employ the octree-based trimming procedure outlined in Algorithm 1. This procedure follows a bottom-up approach [41] in which the level set function (3)

Algorithm 1 Function that trims an element based on an evaluate level set function

---
Input: array of level set *values*, octree *depth*, *dimension* of the element to be trimmed
Output: element of type *Hyperrectangle ∪ Void ∪ WithChildren ∪ Mosaic*

1: **function** TRIM_ELEMENT(*values*, *depth*, *dimension*)
2:     **if all** *values* > 0 **then**   # All level set values are positive
3:         **return** *Hyperrectangle(dimension)*
4:     **else if all** *values* ≤ 0 **then**   # All level set values are non-positive
5:         **return** *Void(dimension)*
6:     **else**   # Both positive and non-positive level set values
7:         **if** *depth* > 0 **then**   # Recursively trim the element when *depth* > 0
8:             *withchildren_element* ← GET_WITHCHILDREN_ELEMENT(*dimension*)
9:             **for** *child* **in** *withchildren_element* **do**
10:                 *child_values* ← GET_CHILD_VALUES(*child*, *values*)
11:                 *child* ← TRIM_ELEMENT(*child_values*, *depth* − 1, *dimension*)
12:             **end for**
13:             **return** *withchildren_element*
14:         **else**   # Terminate the recursion when *depth* = 0
15:             **return** GET_MOSAIC_ELEMENT(*values*, *dimension*)
16:         **end if**
17:     **end if**
18: **end function**

---

is sampled at the $2^{depth} + 1$ vertices of the octree in each direction for each element, where *depth* is the number of subdivision operations performed to detect the immersed boundary.

The trimming procedure takes the evaluated level set *values*, the subdivision *depth* and the *dimension* of the element as input arguments. If all level set *values* are positive (L2), the TRIM_ELEMENT function retains the complete element in the mesh. If all level set values are non-positive (L4), the element is discarded. When some of the level set *values* are positive and some are non-positive (L6), this implies that the element is intersected by the immersed boundary. In this case, the element is subdivided (bisected) into 2^d children (L8). For each *child* a recursive call to the TRIM_ELEMENT function is made (L11).

The recursive subdivision routine is terminated at the lowest level of subdivision, i.e., at *depth* = 0 (L14). Since our analysis approach requires the evaluation of functions on the immersed boundary, it is convenient to also obtain an explicit parametrization of this boundary. To obtain this parametrization, at the lowest level of subdivision we consider a tessellation procedure based on the level set values. The function GET_MOSAIC_ELEMENT that implements this procedure is discussed in Sect. 2.2.2.

2.2.2 Midpoint Tessellation

At the lowest level of subdivision of the octree procedure, we perform a tessellation based on the 2^d level set values at that level. In the scope of our work, an important

requirement for the tessellation procedure is that it is suitable for the consideration of interface problems. Practically, this means that if the procedure is applied to the negated level set function, a partitioning of the complementary part of the cell is obtained, with an immersed boundary that matches with that of the tessellation based on the original level set function (illustrated in Fig. 5i). Standard tessellation procedures, specifically Delaunay tessellation [42], do not meet this requirement, as the resulting tessellation is always convex. When an immersed boundary tessellation is convex from one side, it is concave from the other side, meaning that it cannot be identically represented by the Delaunay tessellation. Another complication of Delaunay tessellation is its lack of uniqueness [43], which in our applications results in non-matching interface tessellations.

Algorithm 2 Function that constructs a mosaic element based on level set values

---

    Input: array of level set *values*, *dimension* of the element to be constructed
    Output: *element* of type *Mosaic ∪ Hyperrectangle ∪ Void*
1: **function** GET_MOSAIC_ELEMENT(*values*, *dimension*)
2:     *hyperrectangle* ← GET_HYPERRECTANGLE(*dimension*)
3:     **if all** *values* > 0 **then**   # All level set values are positive
4:         **return** *hyperrectangle*
5:     **else if all** *values* ≤ 0 **then**   # All level set values are non-positive
6:         **return** *Void*
7:     **else**   # Both positive and non-positive level set values
8:         *boundaries* ← GET_BOUNDARIES(*hyperrectangle*)
9:         **for** *boundary* **in** *boundaries* **do**
10:            *boundary_values* ← GET_BOUNDARY_VALUES(*boundary*, *values*)
11:            *boundary* ← GET_MOSAIC_ELEMENT(*boundary_values*, *dimension* − 1)
12:         **end for**
13:         *zero_points* ← GET_ZERO_POINTS(*boundaries*)
14:         *midpoint* ← AVERAGE(*zero_points*)
15:         *mosaic_element* ← EXTRUDE_TO_POINT(*boundaries*, *midpoint*)
16:         **return** *mosaic_element*
17:     **end if**
18: **end function**

To overcome the deficiencies associated with Delaunay tessellation, we have developed a dedicated tessellation procedure that suits the needs of our immersed analysis approach [32]. We refer to the developed procedure as *midpoint tessellation*, which is illustrated in Figs. 5 and 6 for the two-dimensional and three-dimensional case, respectively. The GET_MOSAIC_ELEMENT function outlined in Algorithm 2 implements our midpoint tessellation procedure. This function is called by the octree algorithm at the lowest level of subdivision (Algorithm 1, L15), taking the 2^d level set *values* at the octree vertices as input. The function returns a tessellation of the octree cell that (approximately) fits to the immersed boundary.

To explain Algorithm 2, we first consider the two-dimensional case, which is illustrated in Fig. 5. The midpoint tessellation procedure commences with looping over all the edges of the element and recursively calling the GET_MOSAIC_ELEMENT function to truncate the edges that are intersected by
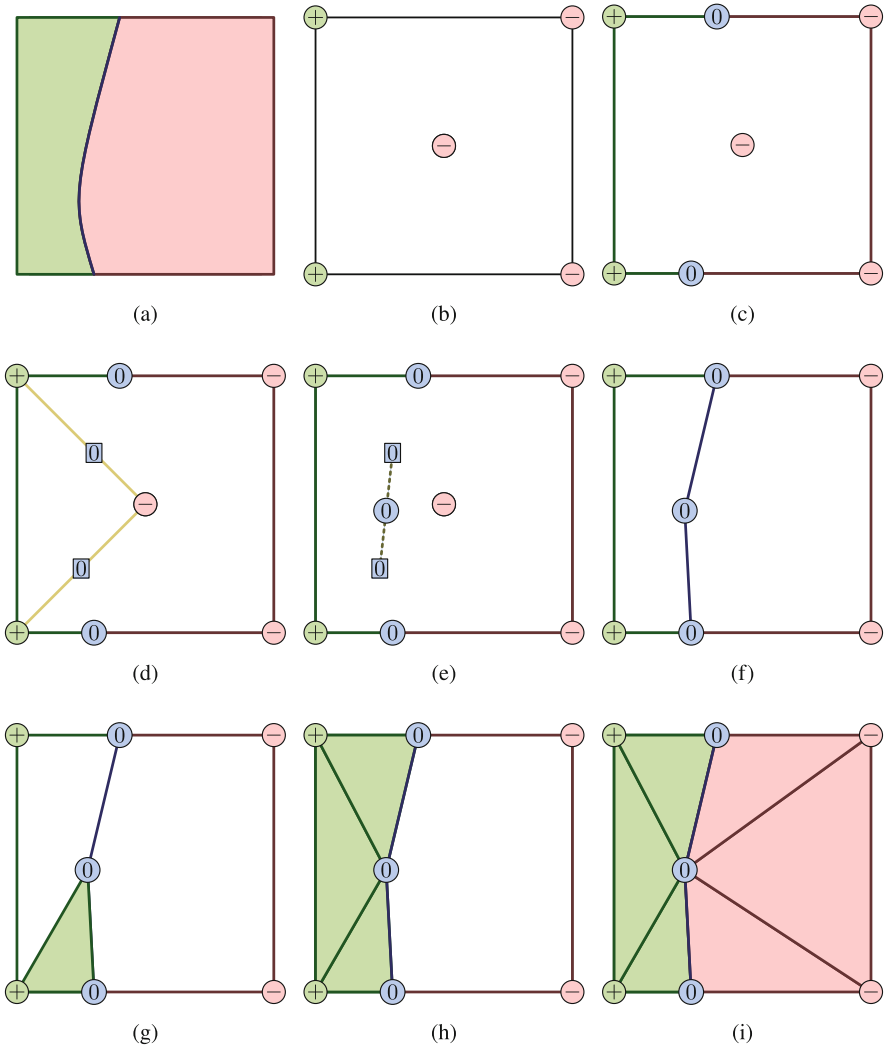
Fig. 5 Schematic representation of the midpoint tessellation procedure for a two-dimensional case

the immersed boundary (L8–L12, Fig. 5c). A set of *zero_points* is then computed by linear interpolation of the level set function across the diagonals between the centroid of the rectangle and its vertices (L13, Fig. 5d), and the arithmetic average of these points is defined as the *midpoint* (L14, Fig. 5e). The tessellation is then created by extruding the (truncated) edges toward this *midpoint* (Fig. 5g–h). Note that if this procedure is applied to the negated level set values, a tessellation of the complementary part of the rectangular element with a coincident immersed boundary is obtained (Fig. 5i).
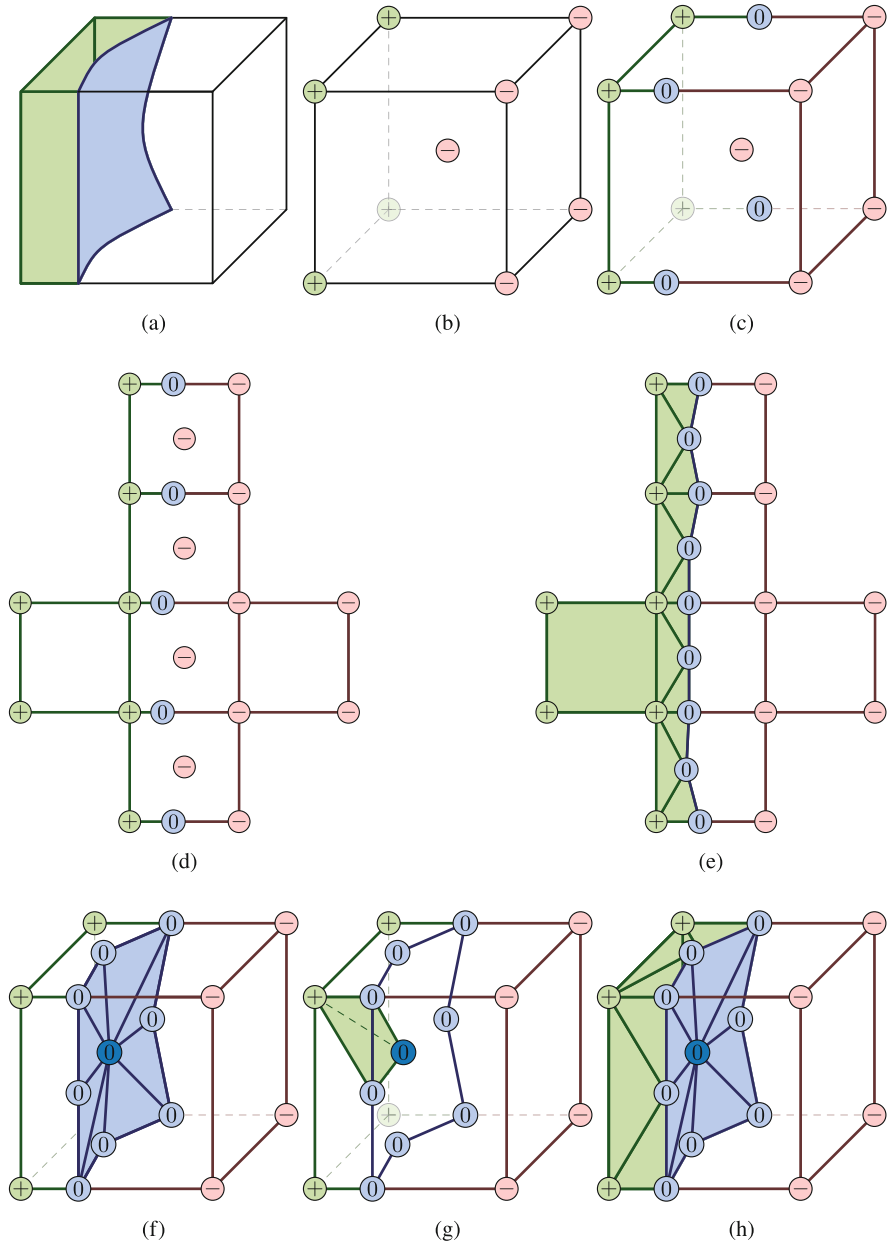
Fig. 6 Schematic representation of the midpoint tessellation procedure for a three-dimensional case

Since the tessellation algorithm recursively traverses the dimensions of the element, it can directly be extended to the three-dimensional case, as illustrated in Fig. 6. In three dimensions, all six faces of the element are tessellated by calling the GET_MOSAIC_ELEMENT function (L8–L12, Fig. 6e). Based on the diagonals between the centroid of the element and its vertices, zero level set points (L13) and a corresponding *midpoint* (L14) are then computed. The three-dimensional tessellation is finally constructed by extrusion of all (truncated) faces toward the *midpoint* (Fig. 6g–h). As in the two dimensional case, a conforming interface is obtained when the procedure is applied to the negated level set values.

Remark 1 (Generalization to Non-rectangular Elements) The algorithms presented in this section are presented for the case of hyperrectangles, i.e., a rectangle in two dimensions and a hexagon in three dimensions. The algorithms can be generalized to a broader class of element shapes (e.g., simplices, similar to Ref. [41]). For the algorithms to be generalizable, the considered element must be able to define *children* (of the same type). Moreover, the element and its faces must be convex, so that the *midpoint* is guaranteed to be in the interior of the element. Note that this convexity requirement pertains to the shape of the untrimmed element, and not to the trimmed element.

2.3 Topology Preservation

The spline-based segmentation procedure has been demonstrated to yield analysis-suitable domains for a wide range of test cases (see, e.g., Refs. [25, 28, 32, 33, 36, 44, 45]). An example from Ref. [33] is shown in Fig. 7. This example shows a carotid artery, obtained from a CT-scan. The scan data consists of 80 slices, separated by a distance of $400\,\mu$m. Each slice image consists of 85×70 voxels of size $300 \times 300\mu\text{m}^2$.

When the spline-based segmentation procedure is performed using a B-spline level set function defined on the voxel grid, the result shown in Fig. 7b is obtained. Although the smoothing characteristic of the technique is overall beneficial, in the sense that it leads to smooth boundaries, comparison to the original voxel data in Fig. 7a shows that in this particular case the topology of the object is altered by the segmentation procedure. This can occur in cases where the features of the object to be described are not significantly larger in size than the voxels (i.e., the Nyquist criterion is not satisfied; see Sect. 2.1). In many cases, the altering of the topology of an object fundamentally changes the problem under consideration, and is therefore generally undesirable.

To avoid the occurrence of topological anomalies due to smoothing, a topology-preservation strategy has been developed in Ref. [33]. The developed strategy follows directly from the smoothing analysis presented in Sect. 2.1, which shows that features with a small relative length scale, $\hat{\ell} = \ell/h$, may be lost upon smoothing, *cf.* Eq. (10). Hence, topological features may be lost when the mesh
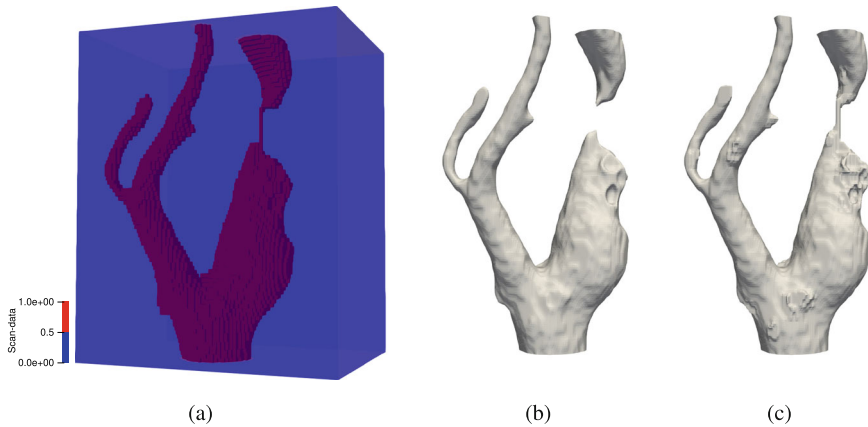
Scan-data

1.0e+00

0.5

0.0e+00

(a)   (b)   (c)

Fig. 7 Example of the spline-based segmentation procedure for a scan-based image of a carotid artery. (**a**) The original voxel data; (**b**) the spline segmentation based on the voxel mesh, resulting in a topologically erroneous domain; and (**c**) the spline segmentation after application of the topology-preservation algorithm

size on which the B-spline level set is constructed is relatively large compared to the voxel size. The pivotal idea of the strategy proposed in Ref. [33] is to detect topological anomalies by comparison of the segmented image (Fig. 7b) with the original voxel data (Fig. 7a) through a moving-window technique. In places where topological anomalies are detected, the mesh on which the smooth level set function is constructed is then refined locally (using THB-splines [37]). This locally increases the relative feature length scale, $\hat{\ell}$, such that the topology is restored (Fig. 7c).

3   Immersed Isogeometric Flow Analysis

In this section we introduce an immersed discretization of the Stokes flow problem solved on a domain $\Omega \subset \mathbb{R}^d$ according to (4), attained through the scan-based segmentation procedure outlined above. The boundary, $\partial\Omega$, as illustrated in Fig. 8, is (partly) immersed, in the sense that it does not coincide with element boundaries.

The Stokes flow problem reads

$$\begin{cases} -\nabla \cdot (2\mu\nabla^s \boldsymbol{u}) + \nabla p = \boldsymbol{f} & \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} = 0 & \text{in } \Omega, \\ \boldsymbol{u} = \boldsymbol{g} & \text{on } \partial\Omega_D, \\ 2\mu\left(\nabla^s \boldsymbol{u}\right)\boldsymbol{n} - p\boldsymbol{n} = \boldsymbol{t} & \text{on } \partial\Omega_N, \end{cases} \tag{11}$$

with velocity $\boldsymbol{u}$, pressure p, constant viscosity μ, body force $\boldsymbol{f}$, Dirichlet data $\boldsymbol{g}$ and Neumann data $\boldsymbol{t}$. The boundary is composed of a Neumann part, $\partial\Omega_N$, and a Dirichlet part, $\partial\Omega_D$, such that $\overline{\partial\Omega_N} \cup \overline{\partial\Omega_D} = \partial\Omega$ and $\partial\Omega_N \cap \partial\Omega_D = \emptyset$. The vector $\boldsymbol{n}$ in the last line denotes the outward-pointing unit normal to the boundary.

When discretizing the Stokes problem (11), the immersed setting poses various challenges: (i) Since the Dirichlet (e.g., no-slip) boundary is (partly) immersed, Dirichlet boundary conditions cannot be imposed strongly (i.e., by constraining degrees of freedom) [46, 47]; (ii) stability and conditioning issues can occur on unfavorably cut elements [19, 25, 44, 48, 49]; and (iii) elements which are known to be inf-sup stable in boundary-fitted finite elements (e.g., Taylor-Hood elements) can lose stability when being cut, resulting in oscillations in the velocity and pressure fields [34].

To enable scan-based immersed isogeometric analyses, we have developed a stabilized formulation that addresses these challenges. In this formulation, Dirichlet boundary conditions are imposed weakly through Nitsche's method [46, 47]. Ghost stabilization [17] is used to avoid conditioning and stability problems associated with unfavorably cut elements, and skeleton-stabilization is used to avoid inf-sup stability problems. Skeleton-stabilization also allows us to consider equal-order discretizations of the velocity and pressure spaces, simplifying the analysis framework. In Sect. 3.1 we first formalize the immersed analysis setting, after which the developed formulation is detailed in Sect. 3.2.

3.1 Immersed Analysis Setting

The physical domain is immersed in the (cuboid) scan domain, i.e., $\Omega_{\text{scan}} \supset \Omega$, on which a locally refined scan mesh $\mathcal{T}^h_{\Omega_{\text{scan}}}$ with elements K is defined. Locally refined meshes can be constructed by sequential bisectioning of selections of elements in the mesh, starting from a Cartesian mesh, which will be discussed in Sect. 5.

Elements that do not intersect with the physical domain can be omitted, resulting in the locally refined (active) background mesh

$$\mathcal{T}^h := \{K \mid K \in \mathcal{T}^h_{\Omega_{\text{scan}}}, K \cap \Omega \neq \emptyset\}. \tag{12}$$

The (active) background mesh is illustrated in Fig. 8b. By cutting the elements that are intersected by the immersed boundary $\partial\Omega$, a mesh that conforms to the physical domain Ω is obtained:

$$\mathcal{T}^h_\Omega := \{K \cap \Omega \mid K \in \mathcal{T}^h\}. \tag{13}$$

The tessellation procedure discussed in Sect. 2.2 provides a polygonal approximation of the immersed boundary $\partial\Omega$ through the set of boundary faces

$$\mathcal{T}^h_{\partial\Omega} := \{E \subset \partial\Omega \mid E = \partial K \cap \partial\Omega, \ K \in \mathcal{T}^h_\Omega\}. \tag{14}$$
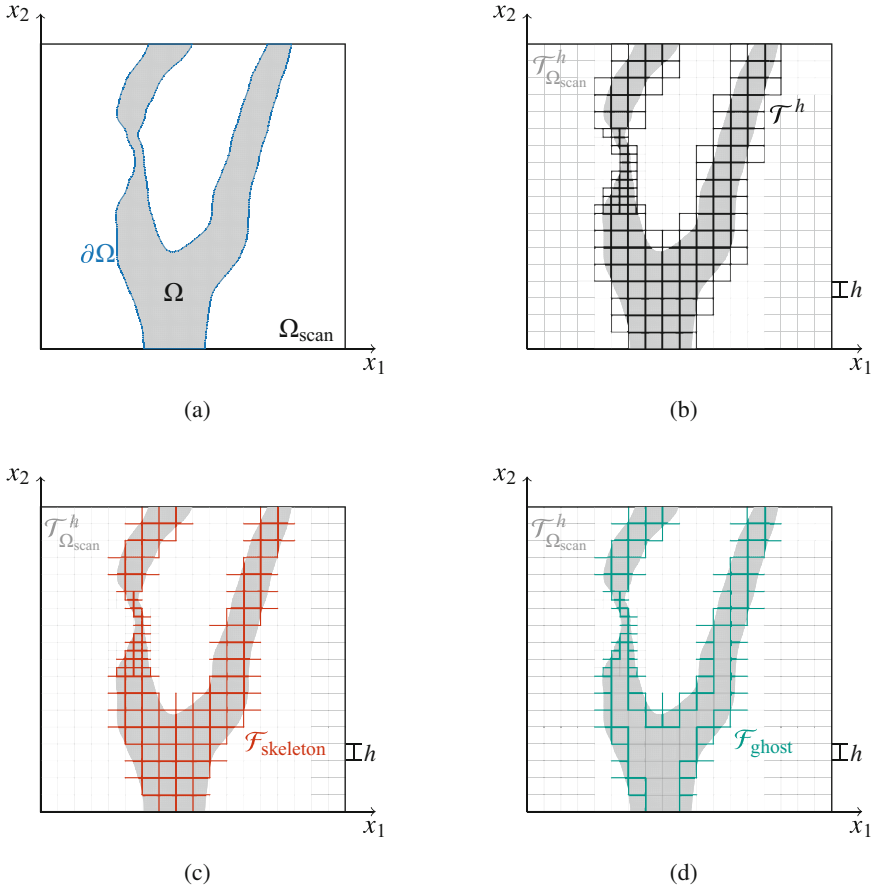
Fig. 8 (**a**) A physical domain Ω (gray), with boundary $\partial\Omega$ (blue), is embedded in the scan domain Ω_{scan} (white). (**b**) The background mesh $\mathcal{T}^h$, which consists of all elements that intersect the physical domain, is constructed by locally refining the ambient domain mesh $\mathcal{T}^h_{\Omega_{\text{scan}}}$. The skeleton mesh, $\mathcal{F}_{\text{skeleton}}$, and ghost mesh, $\mathcal{F}_{\text{ghost}}$, are shown in panels (**c**) and (**d**), respectively

The considered formulation (see Sect. 3.2) incorporates stabilization terms formulated on the edges of the background mesh, which we refer to as the skeleton mesh

$$\mathcal{F}_{\text{skeleton}} := \{F = \partial K \cap \partial K' \mid K, K' \in \mathcal{T}^h, K \neq K'\}. \tag{15}$$

Note that the faces $F \in \mathcal{F}_{\text{skeleton}}$ can be partially outside of the domain Ω and that the boundary of the background mesh is not part of the skeleton mesh. This skeleton mesh is illustrated in Fig. 8c.

In addition to the skeleton mesh, we define the ghost mesh as the subset of the skeleton mesh with the faces that belong to an element intersected by the domain

boundary, i.e.,

$$\mathcal{F}_{\text{ghost}} := \{F \cap \partial K \mid F \in \mathcal{F}_{\text{skeleton}}, K \in \mathcal{G}\}, \tag{16}$$

where $\mathcal{G} := \{K \in \mathcal{T}^h \mid K \cap \partial\Omega \neq \emptyset\}$ is the collection of elements in the background mesh that are crossed by the immersed boundary. The ghost mesh is illustrated in Fig. 8d.

3.2 Stabilized Formulation

To solve the Stokes problem (11) we discretize the velocity and pressure fields using truncated hierarchical B-splines [37, 50]. THB-splines form a basis of degree k and regularity α constructed over the locally-refined background mesh, $\mathcal{T}^h$, spanning the spline space

$$S_\alpha^k(\mathcal{T}^h) = \{N \in C^\alpha(\mathcal{T}^h) : N|_K \in Q^k(K), \ \forall K \in \mathcal{T}^h\}, \tag{17}$$

with $Q^k(K)$ the set of d-variate polynomials on the element K constructed by the tensor-product of univariate polynomials of order k. We consider equal-order discretizations of the velocity and pressure spaces with optimal regularity THB-splines, i.e., $\alpha = k - 1$:

$$\boldsymbol{u}^h \in \boldsymbol{V}^h = [S_{k-1}^k]^d \subset [H^1]^d, \qquad p^h \in Q^h = S_{k-1}^k \subset L^2. \tag{18}$$

Note that the superscript h is used to indicate that these fields are approximations obtained on a mesh with (local) element size h.

We consider the stabilized Bubnov-Galerkin formulation

$$\begin{cases} \text{Find } \boldsymbol{u}^h \in \boldsymbol{V}^h \text{ and } p^h \in Q^h \text{ such that:} \\ a(\boldsymbol{u}^h, \boldsymbol{v}^h) + b(p^h, \boldsymbol{v}^h) + s_{\text{nitsche}}(\boldsymbol{u}^h, \boldsymbol{v}^h) + s_{\text{ghost}}(\boldsymbol{u}^h, \boldsymbol{v}^h) = f(\boldsymbol{v}^h) \quad \forall \boldsymbol{v}^h \in \boldsymbol{V}^h \\ b(q^h, \boldsymbol{u}^h) - s_{\text{skeleton}}(p^h, q^h) = g(q^h) \qquad\qquad\qquad\qquad\quad \forall q^h \in Q^h \end{cases} \tag{19}$$

where the bilinear and linear operators are defined as (see Ref. [28] for details)

$$a(\boldsymbol{u}^h, \boldsymbol{v}^h) := 2\mu(\nabla^s \boldsymbol{u}^h, \nabla^s \boldsymbol{v}^h)$$
$$- 2\mu\left[\langle(\nabla^s \boldsymbol{u}^h)\boldsymbol{n}, \boldsymbol{v}^h\rangle_{\partial\Omega_D} + \langle(\nabla^s \boldsymbol{v}^h)\boldsymbol{n}, \boldsymbol{u}^h\rangle_{\partial\Omega_D}\right] \tag{20a}$$

$$b(p^h, \boldsymbol{v}^h) := -(p^h, \nabla \cdot \boldsymbol{v}^h) + \langle p^h, \boldsymbol{v}^h \cdot \boldsymbol{n}\rangle_{\partial\Omega_D} \tag{20b}$$

$$f(\boldsymbol{v}^h) := (\boldsymbol{f}, \boldsymbol{v}^h) + \langle\boldsymbol{t}, \boldsymbol{v}^h\rangle_{\partial\Omega_N} - 2\mu\langle(\nabla^s \boldsymbol{v}^h)\boldsymbol{n}, \boldsymbol{g}\rangle_{\partial\Omega_D} \tag{20c}$$

$$g(q^h) := \langle q^h, \boldsymbol{g} \cdot \boldsymbol{n} \rangle_{\partial \Omega_D} \tag{20d}$$

$$s_{\text{nitsche}}(\boldsymbol{u}^h, \boldsymbol{v}^h) := \langle \beta \mu h^{-1} (\boldsymbol{u}^h - \boldsymbol{g}), \boldsymbol{v}^h \rangle_{\partial \Omega_D} \tag{20e}$$

$$s_{\text{ghost}}(\boldsymbol{u}^h, \boldsymbol{v}^h) := \sum_{F \in \mathcal{F}_{\text{ghost}}} \int_F \gamma_g \mu h_F^{2k-1} [\![\partial_n^k \boldsymbol{u}^h]\!] \cdot [\![\partial_n^k \boldsymbol{v}^h]\!] \, \mathrm{d}S \tag{20f}$$

$$s_{\text{skeleton}}(p^h, q^h) := \sum_{F \in \mathcal{F}_{\text{skeleton}}} \int_F \gamma_s \mu^{-1} h_F^{2k+1} [\![\partial_n^k p^h]\!] [\![\partial_n^k q^h]\!] \, \mathrm{d}S \tag{20g}$$

where $(\cdot, \cdot)$ denotes the inner product in $L^2(\Omega)$, $\langle \cdot, \cdot \rangle_{\partial \Omega}$ denotes the inner product in $L^2(\partial \Omega)$, and $[\![\cdot]\!]$ denotes the interface jump operator. The parameters β, γ_g, and γ_s denote the penalty constants for the Nitsche term, the ghost-stabilization term, and the skeleton-stabilization term, respectively.

To ensure stability and optimal approximation, the Nitsche stabilization term (20e) scales with the inverse of the (background) mesh size parameter, h [19]. The Nitsche stability parameter β should be selected appropriately, being large enough to ensure stability, while not being too large to cause locking-type effects (see, e.g., Refs. [48, 49, 51]). The ghost-penalty operator in (20f) controls the kth-order normal derivative jumps over the interfaces of the elements which are intersected by the domain boundary $\partial \Omega$. Since in this contribution splines of degree k with C^{k-1}-continuity are considered, only the jump in the kth normal derivative is non-vanishing at the ghost mesh. The ghost-stabilization term scales with the size of the faces as h_F^{2k-1}. Appropriate selection of the parameter γ_g corresponding with the Nitsche parameter, β, assures the stability of the formulation independent of the cut-cell configurations. To avoid loss of accuracy, the ghost-penalty parameter, γ_g, should also not be too large [52].

The skeleton-stabilization operator (20g), proposed in Ref. [28], penalizes jumps in higher-order pressure gradients. This ensures inf-sup stability of the equal-order velocity-pressure discretization, and resolves spurious pressure oscillations caused by cut elements. This spline-based skeleton-stabilization technique can be regarded as the higher-order continuous version of the interior penalty method proposed in Ref. [53]. To ensure stability and optimality, the operator (20g) scales with h_F^{2k+1}. The parameter γ_s should be selected such that oscillations are suppressed, while the influence of the additional term on the accuracy of the solution remains limited. It is noted that since the inf-sup stability problem is not restricted to the immersed boundary, the skeleton stabilization pertains to all interfaces of the background mesh.

In our scan-based analysis workflow it is, from a computational effort point of view, generally impractical to evaluate the (integral) operators (20) exactly. The error of the Galerkin solution with inexact integration, $u_Q^h = (\boldsymbol{u}_Q^h, p_Q^h)$, is then composed of two parts, *viz.*: (i) the discretization error, defined as the difference between the analytical solution, $u = (\boldsymbol{u}, p)$, and the approximate Galerkin solution in the absence of integration errors, $u^h = (\boldsymbol{u}^h, p^h)$; and (ii) the inconsistency error related to the integration procedure, which is defined as the difference between

the approximate solution in the absence of integration errors, u^h, and the Galerkin solution with integration errors, u_Q^h. In practice, one needs to control both these error contributions in order to ensure the accuracy of a simulation result. From the perspective of computational effort, it is in general not optimal to make either one of the contributions significantly smaller than the other.

The decoupling of the geometry description from the analysis mesh provides the immersed (isogeometric) analysis framework with the flexibility to locally adapt the resolution of the solution approximation without the need to reparametrize the domain. To leverage this flexibility in the scan-based analysis setting, it is essential to automate the adaptivity procedure, as manual selection of adaptive cut-element quadrature rules and mesh refinement regions is generally impractical on account of the complex volumetric domains that are considered.

In our work we have developed error-estimation-based criteria that enable adaptive scan-based analyses. In Sect. 4 we first discuss an adaptive octree quadrature procedure used to reduce the computational cost associated with cut element integration. In Sect. 5 we then discuss a residual-based error estimator to refine the THB-spline approximation of the field variables only in places where this results in substantial accuracy improvements.

4 Adaptive Integration of Cut Elements

From the perspective of computational effort, a prominent challenge in immersed finite element methods is the integration of the cut elements. While quadrature points can be constructed directly on all octree sub-cells (Sect. 2.2.1), this generally results in very expensive integration schemes, especially for three-dimensional problems [32]. A myriad of techniques have been developed to make cut-element integration more efficient, an overview of which is presented in, e.g., Refs. [32, 54]. In the selection of an appropriate cut element integration scheme one balances robustness (with respect to cut element configurations), accuracy, and expense.

In the context of scan-based analyses, we have found it most suitable to leverage the robustness of the octree procedure as much as possible. To improve the computational efficiency of the resulting quadrature rules, we have developed a procedure that adapts the number of integration points on each integration sub-cell, similar to the approach used in Ref. [55], as lower-order integration on very small sub-cells does not significantly reduce the accuracy.

4.1 Integration Error Estimate

The pivotal idea of our adaptive octree quadrature procedure is to optimally distribute integration points over the sub-cells using an error estimator based on

Strang's first lemma [35] (see also [56, Lemma 2.27]). In the immersed analysis setting, this lemma provides an upper bound for the error $u - u_Q^h$. Following the derivation in Ref. [32] (to which we refer the reader for details), this error bound can be expressed in abstract form as

$$
\left\| u - u_Q^h \right\|_{W(h)} \leq \left(1 + \frac{\left\| a^h \right\|_{W(h), V^h}}{\alpha^h} \right) \left\| u - \mathcal{I}^h u \right\|_{W(h)} + \frac{1}{\alpha^h} \sum_{K \in \mathcal{T}_\Omega^h} \left(e_K^a + e_K^f \right),
$$

(21)

where α^h denotes the inf-sup constant associated with the (aggregate) bilinear form $a^h : W^h \times V^h \to \mathbb{R}$, with trial and test velocity-pressure spaces W^h and V^h, with $W(h) = \text{span}\{u\} \oplus W^h$ the linear space containing the weak solution, u. The element-integration-error indicators associated with the (aggregate) bilinear form a^h and (aggregate) linear form f^h are respectively elaborated as

$$
e_K^a = \sup_{v_K^h \in V_K^h} \frac{\left| \int_K A_\Omega^h(\mathcal{I}^h u, v_K^h)(\boldsymbol{x}_K) \, \mathrm{d}V - \sum_{l=1}^{l_K} \omega_K^l A_\Omega^h(\mathcal{I}^h u, v_K^h)(\boldsymbol{x}_K^l) \right|}{\left\| v_K^h \right\|_{V_K^h}},
$$

(22a)

$$
e_K^f = \sup_{v_K^h \in V_K^h} \frac{\left| \int_K F_\Omega^h(v_K^h)(\boldsymbol{x}) \, \mathrm{d}V - \sum_{l=1}^{l_K} \omega_K^l F_\Omega^h(v_K^h)(\boldsymbol{x}_K^l) \right|}{\left\| v_K^h \right\|_{V_K^h}},
$$

(22b)

where A_Ω^h and F_Ω^h are the integrands corresponding to the volumetric terms in the bilinear form and linear form in the Galerkin problem, respectively, and where for each element K, the set $\{(\boldsymbol{x}_K^l, \omega_K^l)\}_{l=1}^{l_K}$ represents a quadrature rule. The norm $\| \cdot \|_{V_K^h}$ corresponds to the restriction of the space V^h to the element K. We note that it has been assumed that integration errors associated with the boundary terms in the Galerkin problem are negligible in comparison to the errors in the volumetric terms, which is in line with the goal of optimizing the volumetric quadrature rules of cut elements.

It is desirable to apply a single integration scheme for all terms in the bilinear and linear forms and, hence, to treat the element-integration errors (22) in the same way. To do this, we note that the integrals between the absolute bars in the numerators of (22) constitute linear functionals on V_K^h. By the Riesz-representation theorem [56], there exist functions $T^a, T^f \in V_K^h$ such that

$$
\int_K T^a v_K^h \, \mathrm{d}V = \int_K A_\Omega^h \left(\mathcal{I}^h u, v_K^h \right) \mathrm{d}V, \qquad \int_K T^f v_K^h \, \mathrm{d}V = \int_K F_\Omega^h \left(v_K^h \right) \mathrm{d}V,
$$

(23)

for all $v_K^h \in V_K^h$. Assuming that the difference in applying the integral quadrature to the left- and right-hand-side members of (23) is negligible, it then holds that

$$e_K^a \leq \|T^a\|_{L^2(K)} \sup_{T^a, v_K^h \in V_K^h} \frac{\left| \int_K T^a v_K^h \, \mathrm{d}V - \sum_{l=1}^{l_K} \omega_K^l \left(T^a v_K^h \right) (\boldsymbol{x}_K^l) \right|}{\|T^a\|_{L^2(K)} \|v_K^h\|_{V_K^h}}, \tag{24a}$$

$$e_K^f \leq \|T^f\|_{L^2(K)} \sup_{T^f, v_K^h \in V_K^h} \frac{\left| \int_K T^f v_K^h \, \mathrm{d}V - \sum_{l=1}^{l_K} \omega_K^l \left(T^f v_K^h \right) (\boldsymbol{x}_K^l) \right|}{\|T^f\|_{L^2(K)} \|v_K^h\|_{V_K^h}}. \tag{24b}$$

with both T^a (resp. T^f) and v_K^h in the polynomial space V_K^h, the product $T^a v_K^h$ (resp. $T^f v_K^h$) resides in the double-degree (normed) polynomial space Q_K^{2k}. It then follows that

$$e_K^a \lesssim \|T^a\|_{L^2(K)} e_K^p, \qquad\qquad e_K^f \lesssim \|T^f\|_{L^2(K)} e_K^p, \tag{25}$$

with the uniformly applicable polynomial integration error defined as

$$e_K^p = \sup_{p_K \in Q_K^{2k}} \frac{\left| \int_K p_K(\boldsymbol{x}_K) \, \mathrm{d}V - \sum_{l=1}^{l_K} \omega_K^l p_K(\boldsymbol{x}_K^l) \right|}{\|p_K\|_{P_K}}$$

$$= \left| \int_K p_{K,\max}(\boldsymbol{x}_K) \, \mathrm{d}V - \sum_{l=1}^{l_K} \omega_K^l p_{K,\max}(\boldsymbol{x}_K^l) \right|. \tag{26}$$

The supremizer can be evaluated in terms of a polynomial basis $\boldsymbol{\Phi}$ for the space Q_K^{2k} as (see Ref. [32] for a detailed derivation)

$$p_{K,\max} = \frac{\boldsymbol{\Phi}^T \mathbf{G}^{-1} (\boldsymbol{\xi} - \bar{\boldsymbol{\xi}})}{\|\boldsymbol{\xi} - \bar{\boldsymbol{\xi}}\|_{\mathbf{G}^{-1}}}, \tag{27}$$

where $\boldsymbol{\xi} = \int_K \boldsymbol{\Phi} \, \mathrm{d}V$, $\bar{\boldsymbol{\xi}} = \sum_{l=1}^{l_K} \omega_K^l \boldsymbol{\Phi}(\boldsymbol{x}_K^l)$, and $\mathbf{G}$ is the (positive definite) Gramian matrix associated with the inner product with which the polynomial space is equipped.

4.2   Quadrature Optimization Algorithm

The computable error definition (26) and the corresponding computable "worst possible" function (27) form the basis of our adaptive integration procedure, which we summarize in Algorithm 3 (a detailed version is presented in Ref. [32]). The developed optimization procedure is intended as a per-element preprocessing operation, which results in optimized *quadrature* rules for all cut elements in a mesh. Besides the *partition*, $\mathcal{P}_K$, of element, K, the procedure takes the *order* of the monomial basis, Q_K, and stopping *criterion* (e.g., a prescribed number of integration points) as input.

Algorithm 3 Function to optimize the distribution of cut element quadrature points

---

    Input: element *partition*, basis function order *order*, stopping *criterion*
    Output: optimized *quadrature* rule
1: **function** OPTIMIZE_QUADRATURE(*partition*, *order*, *criterion*)
2:     *basis* = GET_MONOMIAL_BASIS(*order*)
3:     *xi_exact* = EXACT_INTEGRATION(*basis*, *order*)
4:     *gramian* = GET_GRAMIAN_MATRIX(*basis*, *order*)
5:     *quadrature* = INITIALIZE_QUADRATURE(*partition*)
6:     **while not** *criterion* **do**   # Adapt quadrature until the stopping criterion is met
7:         *xi_quadrature* = QUADRATURE_INTEGRATION(*basis*, *quadrature*)
8:         *worst_function* = GET_WORST_FUNCTION(*xi_exact*, *xi_quadrature*, *gramian*, *basis*)
9:         *indicators* = INITIALIZE_INDICATORS(*partition*)
10:        **for** *subcell*, *indicator* **in** ZIP(*partition*, *indicators*) **do**   # Iterate over the sub-cells
11:            *error* = GET_SUBCELL_ERROR(*worst_function*, *quadrature*)
12:            *cost* = GET_SUBCELL_COST(*quadrature*)
13:            *indicator* = GET_SUBCELL_INDICATOR(*subcell_error*, *subcell_cost*)
14:        **end for**
15:        *marking* = MARK_SUBCELLS(*indicators*)   # Mark based on marking strategy
16:        *quadrature* = UPDATE_QUADRATURE(*quadrature*, *marking*)
17:     **end while**
18:     **return** *quadrature*
19: **end function**

    The procedure commences with the determination of the polynomial *basis*, $\boldsymbol{\Phi}$ (L2), the evaluation of the basis function integrals, $\boldsymbol{\xi}$ (L3), the computation of the *gramian* matrix, **G** (L4), and the initialization of the partition *quadrature* rule (L5). This initial quadrature rule corresponds to the case where the lowest order (one point) integration rule is used on each sub-cell in the partition. It is noted that the integral computations with full Gaussian quadrature for the *basis* and *gramian* are relatively expensive, but that the computational efficiency gains from the optimized integration scheme outweigh these costs when used multiple times.

    The error-estimation-based quadrature optimization is then performed in an incremental fashion (L6), until the stopping *criterion* is met. Given the considered *quadrature* rule, the approximate basis integrals (L7) and worst possible function to integrate (27) (L8) are determined. Subsequently, for each sub-cell, $\wp$, in the

partition (L10), on L11 the sub-cell error indicators

$$e_\wp^p = \left| \int_\wp p_{K,\max}(\boldsymbol{x}_K)\,\mathrm{d}V - \sum_{l\in\mathcal{I}_\wp} \omega_K^l\, p_{K,\max}(\boldsymbol{x}_K^l) \right| \tag{28}$$

are computed. In this expression, $\mathcal{I}_\wp$, is the set of indices corresponding to integration points on the sub-cell $\wp$. Note that the sum of the sub-cell errors, $e_\wp^p$, provides an upper bound for the element integration error (26). Sub-cell *indicators* are then computed (L13) by weighing the sub-cell errors with the *costs* associated with increasing the quadrature order on a particular sub-cell, as evaluated on L12 (see Ref. [32] for details).

Once the *indicators* have been computed for all sub-cells in the *partition*, the sub-cells with the largest indicators are marked for increasing the number of integration points (L15). We propose two marking strategies, *viz.* a sub-cell marking strategy in which only the sub-cell with the highest indicator is marked, and an octree-level marking strategy in which all sub-cells in the octree level with the highest error are marked. After marking, the quadrature order on the marked sub-cells is increased (L16).

4.3 Optimized Quadrature Results

A detailed study of the error-estimation-based quadrature optimization scheme is presented in Ref. [32]. We here reproduce a typical result for a unit square with a circular exclusion, as illustrated in Fig. 9. To assess the performance of the developed adaptive integration technique, we study its behavior in terms of integration accuracy versus the number of integration points.

Figure 10a displays the integration error as evolving during the optimization procedure when using the sub-cell marking strategy. The non-optimized case in which the same integration scheme is considered on each sub-cell is displayed for reference. As can be seen, the error associated to the same number of integration points is substantially lowered using the adaptive integration procedure. For example, for the case where 144 integration points are considered, the error corresponding to the non-optimized second-order Gauss scheme is equal to 2.52×10^{-2}, while the error corresponding to the optimized quadrature is equal to 1.00×10^{-3}, i.e., a factor 25 reduction in error. Figure 9 displays the distribution of the integration points over the sub-cells for the equal-order Gauss scheme and the optimized case, which clearly demonstrates that the significant reduction in error is achieved by assigning more integration points to the larger sub-cells before introducing additional points in the smaller sub-cells. From Fig. 10a it is also observed that when the optimization algorithm is terminated at a fixed error of, e.g., $e_K^p \approx 1 \cdot 10^{-2}$, the number of integration points N using the optimized integration scheme is reduced substantially
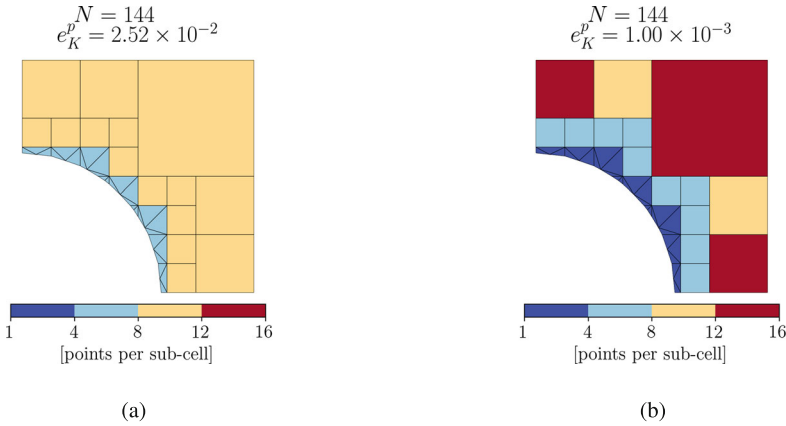
Fig. 9 Distribution of integration points over a cut element with 144 points, comparing the case of (**a**) an equal-order Gauss scheme, and (**b**) optimally distributed Gauss points using the sub-cell marking strategy. Note that the error is reduced by a factor of 25 by using the sub-cell marking strategy. (**a**) Equal-order Gauss. (**b**) Optimal integration
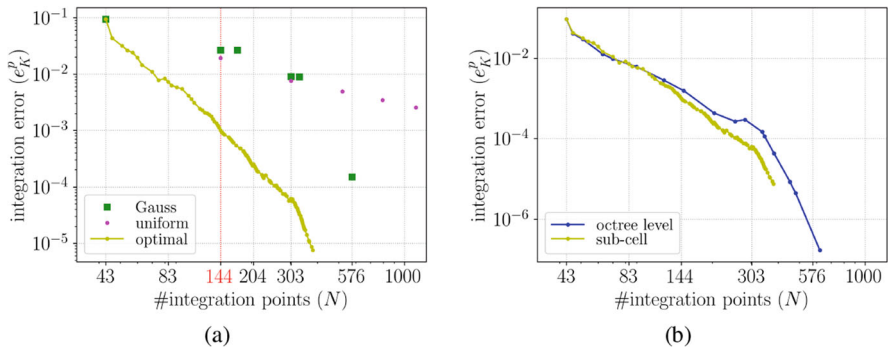

Fig. 10 Integration error *vs.* the number of integration points. (**a**) Comparison of the optimized quadrature results with (non-optimized) equal-order integration. (**b**) Comparison of the sub-cell and octree-level marking strategies.

(in this case from 303 to 83, i.e., a factor of almost 4). Even substantially bigger gains are observed in three-dimensional cases [32].

From the quadrature updating patterns that emerge from the sub-cell marking strategy it is observed that, as a general trend, integration orders are increased on a per-octree-level basis. This is explained by the fact that the indicators scale with the volume of the sub-cells. Based on this observation it was anticipated that an octree-level marking strategy could be very efficient, in the sense that it would yield a similar quadrature update pattern as the sub-cell marking, but that it would need fewer iterations by virtue of marking a larger number of sub-cells per step. Figure 10b compares the marking strategies, conveying that the octree-level marking indeed closely follows the sub-cell marking.

Although the computational effort of the quadrature optimization algorithm is worthwhile when one wants to re-use a quadrature rule multiple times, considerable computational effort is involved. In addition, one has to set up a suitable code to determine the optimal distributions for arbitrarily cut elements. Considering this, one may not be interested in obtaining the optimized distributions of the points, but may instead want a simple rule of thumb to select the quadrature on a cut element; see, e.g., Refs. [55, 57]. Using our quadrature optimization algorithm, in Ref. [32] we studied the effectivity of rules of thumb in which the order of integration is lowered with the octree depth. Although the rule-of-thumb schemes are, as expected, outperformed by the optimized schemes, they generally do provide an essential improvement in accuracy per integration point compared to equal-order integration. This observed behavior is explained by the fact that the rules of thumb qualitatively match the results of the optimization procedure.

5   Adaptive THB-spline Refinement

To leverage the flexibility of the immersed simulation paradigm with respect to refining the mesh independent of the geometry, an automated mesh adaptivity strategy is required. Various adaptivity strategies have been considered in the context of immersed methods, an overview of which is presented in, e.g., Ref. [36]. These refinement strategies can be categorized as either feature-based methods (refinements are based on, e.g., sharp gradients in the solution or high-curvature boundary regions) or methods based on error estimates (e.g., residual-based or goal-oriented methods).

To develop a generic adaptive procedure for scan-based analyses, we have constructed a residual-based a posteriori error estimator. In our isogeometric analysis approach we employ truncated hierarchical B-splines (THB-splines) [37, 50] to locally refine the (volumetric) background mesh.

5.1   Residual-Based Error Estimation

On account of the immersed boundary terms in the formulation (Sect. 3), it is not well-posed in the infinite dimensional setting. Upon appropriate selection of the stabilization parameters, the (mixed) Galerkin formulation of the Stokes problem is well-posed with respect to the mesh-dependent norm (see Ref. [36] for details)

$$\left\|\left| v^h \right\|\right|^2 = \left\|\left| (\boldsymbol{v}^h, q^h) \right\|\right|^2 = \left\|\left| \boldsymbol{v}^h \right\|\right|_u^2 + \left\|\left| q^h \right\|\right|_p^2, \tag{29}$$

with

$$\left\| \boldsymbol{v}^h \right\|_u^2 := \| \mu^{\frac{1}{2}} \nabla^s \boldsymbol{v}^h \|_{L^2(\mathcal{T})}^2 + \| \beta^{-\frac{1}{2}} h^{\frac{1}{2}} \mu^{\frac{1}{2}} \partial_n \boldsymbol{v}^h \|_{L^2(\partial\Omega_D)}^2$$

$$+ \| \beta^{\frac{1}{2}} h^{-\frac{1}{2}} \mu^{\frac{1}{2}} \boldsymbol{v}^h \|_{L^2(\partial\Omega_D)}^2 + \sum_{F \in \mathcal{F}_{\text{ghost}}} \| \gamma_g^{\frac{1}{2}} h_F^{k-\frac{1}{2}} \mu^{\frac{1}{2}} [\![\partial_n^k \boldsymbol{v}^h]\!] \|_{L^2(F)}^2,$$

$$\tag{30a}$$

$$\left\| q^h \right\|_p^2 := \| \mu^{-\frac{1}{2}} q^h \|_{L^2(\mathcal{T})}^2 + \sum_{F \in \mathcal{F}_{\text{skeleton}}} \| \gamma_s^{\frac{1}{2}} h_F^{k+\frac{1}{2}} \mu^{-\frac{1}{2}} [\![\partial_n^k q^h]\!] \|_{L^2(F)}^2. \tag{30b}$$

We refer to this mesh-dependent norm as the *energy norm* and use it to construct an a posterior error estimator for the discretization error, $u - u^h$.

Since, in the considered immersed setting, stability can only be shown in the discrete setting, we define the solution error with respect to the solution in the order-elevated space $\widehat{u}^h \in \widehat{V}^h$. The space $\widehat{V}^h$ is defined on the same mesh and with the same regularity as the space V^h, but with the order of the basis elevated in such a way that $\widehat{V}^h \supset V^h$. It is then assumed that

$$\left\| \widehat{u}^h - u^h \right\| \approx \left\| u - u^h \right\|. \tag{31}$$

We note that additional stabilization terms are required to retain stability in the order-elevated space. In principle this means that the operators (20) need to be augmented, but we assume that for the solution in the order-elevated space these terms are negligible. Similar assumptions, referred to as saturation assumptions, have been considered in, e.g., Refs. [58–60]. Note that the refined space is only used to provide a proper functional setting for the error estimator and that it is not required to perform computations in this space.

To construct an estimator for the error (31), it can be bound from above by

$$\left\| \widehat{u}^h - u^h \right\| \lesssim \sup_{\widehat{v}^h \in \widehat{V}^h \setminus \{0\}} \frac{a^h(\widehat{u}^h - u^h, \widehat{v}^h)}{\left\| \widehat{v}^h \right\|} = \sup_{\widehat{v}^h \in \widehat{V}^h \setminus \{0\}} \frac{r^h(\widehat{v}^h)}{\left\| \widehat{v}^h \right\|}, \tag{32}$$

where the aggregate residual (i.e., the combined velocity-pressure residual) is defined as

$$r^h(\widehat{v}^h) := r^h(u^h)(\widehat{v}^h) := f^h(\widehat{v}^h) - a^h(u^h, \widehat{v}^h), \tag{33}$$

with aggregate operators a^h and f^h (see Ref. [36] for details).

We propose an error estimator pertaining to the background mesh, $\mathcal{T}^h$, which bounds the error in the energy norm (32) as

$$\mathcal{E} = \sqrt{\sum_{K \in \mathcal{T}^h} \eta_K^2} \gtrsim \sup_{\widehat{v}^h \in \widehat{V}^h \setminus \{0\}} \frac{r^h(\widehat{v}^h)}{\left\| \widehat{v}^h \right\|} \gtrsim \left\| \widehat{u}^h - u^h \right\|, \tag{34}$$

where the element-wise error indicators, η_K, will serve to guide an adaptive refinement procedure.

To derive the error indicators, the residual (33) is considered with the operators defined as in (20). Following the derivation of Ref. [36], the indicators are defined as

$$
\begin{aligned}
\eta_K^2 = {}& \mu^{-1} h_K^2 \|r_{\mathrm{int},\boldsymbol{u}}^h\|_{L^2(K\cap\Omega)}^2 + \mu \|r_{\mathrm{int},p}^h\|_{L^2(K\cap\Omega)}^2 \\
& + \mu^{-1} h_K \|r_{\mathrm{neumann}}^h\|_{L^2(K\cap\partial\Omega_N)}^2 + 9\mu h_K^{-1} \|r_{\mathrm{nitsche}}^h\|_{L^2(K\cap\partial\Omega_D)}^2 \\
& + \sum_{F\in\mathcal{F}_{\mathrm{skeleton}}} \mu^{-1} h_K \|r_{\mathrm{jump}}^h\|_{L^2(\partial K\cap F)}^2 + \mu\beta^2 h_K^{-1} \|r_{\mathrm{nitsche}}^h\|_{L^2(K\cap\partial\Omega_D)}^2 \\
& + \sum_{F\in\mathcal{F}_{\mathrm{ghost}}^K} \mu\gamma_g^2 h_F^{2k-1} \|r_{\mathrm{ghost}}^h\|_{L^2(\partial K\cap F)}^2 \\
& + \sum_{F\in\mathcal{F}_{\mathrm{skeleton}}^K} \mu^{-1}\gamma_s^2 h_F^{2k+1} \|r_{\mathrm{skeleton}}^h\|_{L^2(\partial K\cap F)}^2,
\end{aligned}
\tag{35}
$$

where

$$
r_{\mathrm{int},\boldsymbol{u}}^h := \boldsymbol{f} + \nabla\cdot\left(2\mu\nabla^s\boldsymbol{u}^h\right) - \nabla p^h, \tag{36a}
$$

$$
r_{\mathrm{int},p}^h := \nabla\cdot\boldsymbol{u}^h, \tag{36b}
$$

$$
r_{\mathrm{neumann}}^h := \boldsymbol{t} - \left(2\mu\nabla^s\boldsymbol{u}^h\right)\boldsymbol{n} + p^h\boldsymbol{n}, \tag{36c}
$$

$$
r_{\mathrm{nitsche}}^h := \boldsymbol{g} - \boldsymbol{u}^h, \tag{36d}
$$

$$
r_{\mathrm{jump}}^h := \tfrac{1}{2}[\![\left(2\mu\nabla^s\boldsymbol{u}^h\right)\boldsymbol{n}]\!], \tag{36e}
$$

$$
r_{\mathrm{ghost}}^h := \tfrac{1}{2}[\![\partial_n^k\boldsymbol{u}^h]\!], \tag{36f}
$$

$$
r_{\mathrm{skeleton}}^h := \tfrac{1}{2}[\![\partial_n^k p^h]\!]. \tag{36g}
$$

The error indicator (35) reflects that the total element error for all elements that do not intersect the boundary of the domain is composed of the interior residuals associated with the momentum balance and mass balance, and the residual terms associated with the derivative jumps on the skeleton mesh. For elements that intersect the Neumann boundary, additional error contributions are obtained from the Neumann residual and the ghost penalty residual, while additional Nitsche-related contributions appear for elements intersecting the Dirichlet boundary.

5.2 Mesh Adaptivity Algorithm

The residual-based error estimator (34) is used in an iterative mesh refinement procedure, which is summarized in Algorithm 4. The procedure takes the stabilized immersed isogeometric *model* as outlined in Sect. 3 as input, as well as an initial *mesh* and stopping *criterion*. Once the stopping *criterion* is met, the algorithm returns the optimized *mesh* and the corresponding *solution*.

Algorithm 4 Function to perform residual-based error estimation and adaptivity

---

Input: immersed isogeometric *model*, initial background *mesh*, stopping *criterion*
Output: optimized *mesh* and *solution*
1: **function** ERROR_ESTIMATION_AND_ADAPTIVITY(*model*, *mesh*, *criterion*)
2:     **while not** *criterion* **do**   # Adapt the mesh until the stopping criterion is met
3:         *solution* = SOLVE(*model*, *mesh*)   # Immersed IGA formulation of Sect. 3
4:         *indicators* = INITIALIZE_INDICATORS(*mesh*)
5:         **for** *element*, *indicator* **in** ZIP(*mesh*, *indicators*) **do**   # Iterate over the elements
6:             *indicator* = GET_ELEMENT_INDICATOR(*element*, *model*, *solution*)
7:         **end for**
8:         *marking* = MARK_ELEMENTS(*indicators*)   # Dörfler marking
9:         *marking* = ENSURE_REFINEMENT(*marking*, *mesh*)
10:        *mesh* = UPDATE_MESH(*mesh*, *marking*)
11:     **end while**
12:     **return** *mesh*, *solution*
13: **end function**

For each step of the adaptivity procedure, for the given mesh the *solution* of the Galerkin problem (19) is computed (L3). For each element (L5), the error *indicator* (35) is then evaluated (L6). Dörfler marking [61]—targeting reduction of the estimator (34) by a fixed fraction—is used to select elements for refinement (L8). For THB-splines, refining elements does not necessarily result in a refinement of the approximation space [50, 62]. To ensure that the approximation space is refined, an additional refinement mask is applied to update the element *marking* (L9).

In our implementation the geometry approximation is not altered during mesh refinement. A consequence of this implementation choice is that an element can only be refined up to the octree depth. Elements requiring refinement beyond this depth are discarded from the *marking* list, and the adaptive refinement procedure is stopped if there are no more elements that can be refined. We refer the reader to Ref. [36] for details.

5.3 Mesh Adaptivity Results

Before considering the application of the developed residual-based error estimation and adaptivity procedure in the context of scan-based analysis in Sect. 6, we here
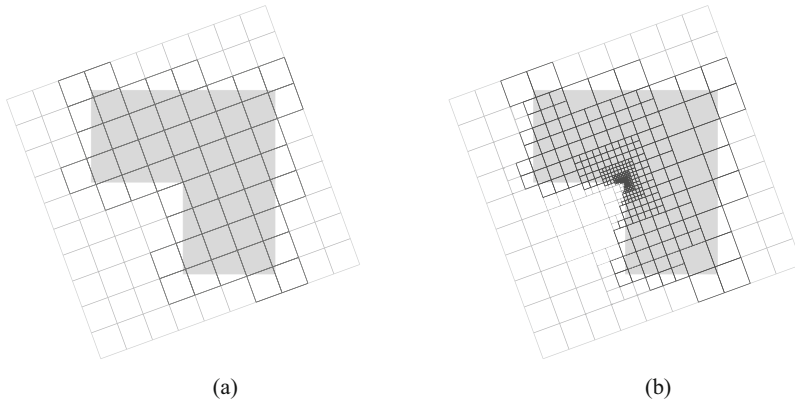
Fig. 11 Evolution of the mesh using the adaptive refinement procedure for the Stokes problem on a re-entrant corner domain using $k = 2$. (**a**) Initial mesh. (**b**) Step 5
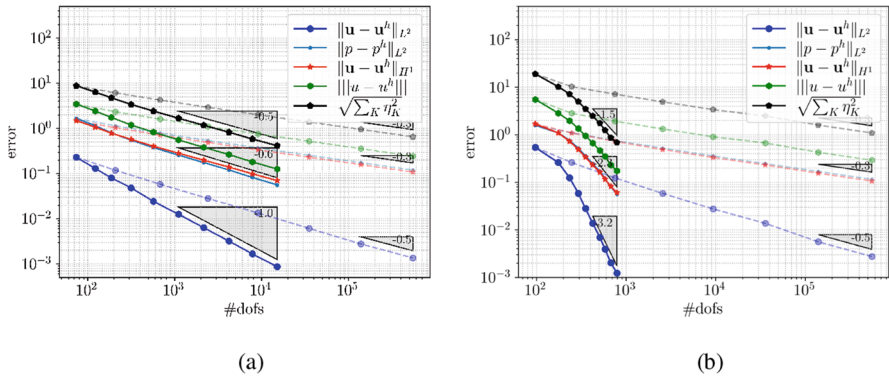

Fig. 12 Error convergence results for the Stokes problem on a re-entrant corner domain under residual-based adaptive refinement (solid) and uniform refinement (dashed) for linear ($k = 1$) and quadratic ($k = 2$) basis functions. (**a**) $k = 1$. (**b**) $k = 2$

first reproduce a benchmark case from Ref. [36]. We consider the Stokes problem (11) on a re-entrant corner domain (Fig. 11a) with mixed Dirichlet and Neumann boundaries. The method of manufactured solutions is considered with the (weakly singular) exact solution taken from Ref. [63]. We refer to Ref. [36] and references therein for a full specification of the benchmark.

Figure 12 displays the error convergence results obtained using uniform and adaptive refinements, for both linear and quadratic THB-splines. The convergence rate when uniform refinements are considered is suboptimal, limited by the weak singularity at the re-entrant corner. Using adaptive mesh refinement results in a recovery of the optimal rates in the case of linear basis functions, with even higher rates observed for the quadratic splines on account of the highly-focused refinements resulting from the residual-based error estimator as observed in Fig. 11b.

6 Scan-Based Flow Simulations

To demonstrate the scan-based analysis workflow reviewed in this work, we consider the blood flow (viscosity $\mu = 4\,\text{mPa} \cdot \text{s}$) through the patient-specific μCT-based carotid artery introduced in Sect. 2.3 (Fig. 7). Neumann conditions are imposed on the inflow (bottom) and outflow (top) boundaries, with the traction on the inflow boundary corresponding to a pressure of 17.3 kPa (130 mmHg) and a zero-traction condition on the outflow boundary. Homogeneous Dirichlet conditions are imposed along the immersed boundaries to impose a no slip condition. The presented results are based on second-order ($k = 2$) THB-splines. For details regarding the simulation setup we refer the reader to Ref. [36], from which the results presented here are reproduced.

We consider an initial scan-domain mesh consisting of $24 \times 24 \times 24$ elements, with a scan size of $25.6 \times 21.1 \times 32.0\,\text{mm}^3$. The octree depth is set to three. In this setting, after two refinements, an element is of a similar size as the voxels. The need to substantially refine beyond the voxel size is, from a practical perspective, questionable, as the dominant error in the analysis will then be related to the scan resolution and the segmentation procedure. In this sense, the constraint of not being able to refine beyond the octree depth is not a crucial problem in the considered simulations.

The initial mesh and final refinement result are shown in Fig. 13. The adaptive refinement procedure focuses on the regions where the errors are largest, i.e., near the stenosed section (i.e., the narrow region at the right artery) and at the outflow section of the left artery, such that important details of the solution are resolved. After the final refinement step, the adaptive simulation uses 12,816 DOFs, which is substantially lower than the approximately 100,000 DOFs that would have resulted from uniform mesh refinements up to the same level [33].

7 Concluding Remarks

In this contribution, we have reviewed the four key research contributions of our team with respect to scan-based immersed isogeometric flow analysis, *viz.*: (i) A spline-based image segmentation procedure, encompassing a voxel-data smoothing procedure, an octree-based procedure to obtain an explicit parametrization of the computational domain and its (immersed) boundary, and a topology-preservation strategy to restore smoothing-induced anomalies; (ii) A stabilized immersed formulation for (*a.o.*) Stokes flow, which ensures robustness with respect to unfavorably cut elements and enables the consideration of equal-order velocity-pressure discretizations without the loss of inf-sup stability; (iii) An adaptive procedure to optimize the distribution of integration points over cut elements, based on Strang's first lemma; (iv) A mesh refinement procedure based on rigorous residual-based
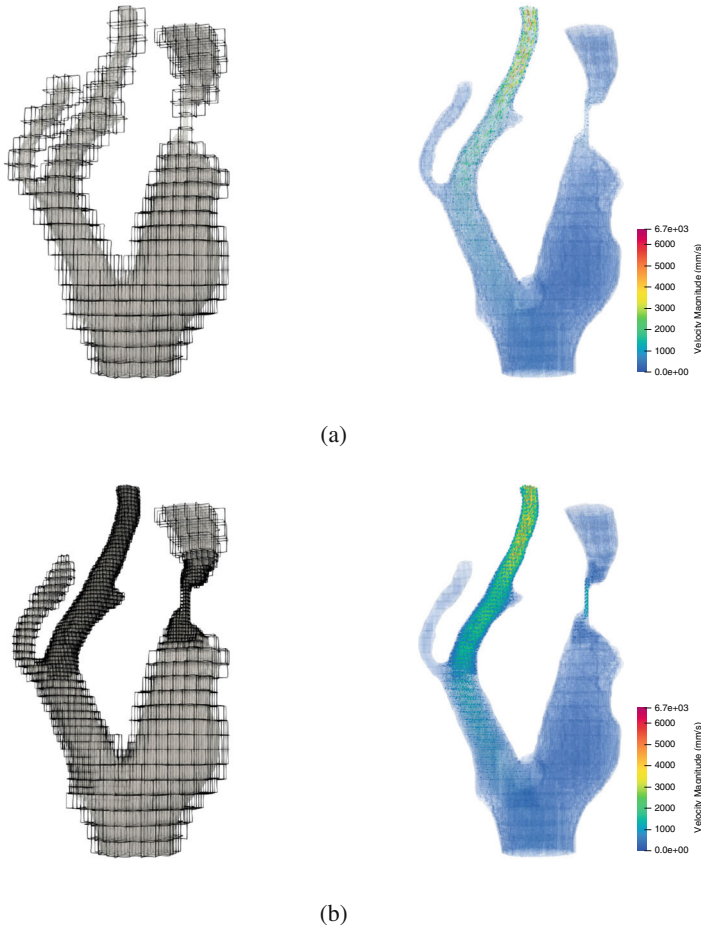
(a)

(b)

Fig. 13 Initial (**a**) and final (**b**) mesh (left) and velocity magnitude (right) for the patient-specific flow problem. (**a**) Initial mesh with 3158 #DOFs. (**b**) Step 3 with 12467 #DOFs

error estimates to refine the computational mesh in places where this results in significant accuracy improvements.

An important aspect of immersed (finite element) methods is the ill-conditioning associated with small (i.e., with a small volume fraction) or unfavorably cut (e.g., sliver-like) elements. Although not reviewed in this work, over the past decade our team has contributed to solving the challenges associated with ill-conditioning. The origin of the ill-conditioning problem was studied in detail by De Prenter et al. [44], which led to a scaling relation for the condition number with the smallest cut-element volume fraction. Dedicated preconditioning techniques, to be used in conjunction with iterative solvers, were developed based on the insights from this work, e.g., Refs. [27, 44, 45]. We consider these (preconditioned) solver

developments an important step in unlocking the potential of high-performance computing for immersed finite element methods [30]. Note that the ghost- and skeleton-stabilization terms employed in the formulation in this chapter, which are primarily added to ensure well-posedness of the weak form, also resolve the conditioning problems, such that preconditioning techniques are not essential in this work.

The innovations in computational procedures and problem formulations yield a highly robust immersed isogeometric analysis workflow for scan-based analyses. Error-controlled simulations can be performed directly based on scan data, without the need for extensive user interactions. The effectivity of the framework is not fundamentally affected by the geometric and topological complexity of the scan data, on account of the decoupling of the geometry and computational mesh in immersed methods. The robustness of the framework derives from the rigorous mathematical underpinning of the considered methods.

Further developments to the scan-based workflow are required to enable the consideration of more advanced problems/formulations, such as higher Reynolds number flows (requiring additional stabilization), fluid-structure interactions and complex fluid models. Further improvements are also required to enhance the computational performance of the developed workflow. This mainly pertains to algorithm and code optimization, which is required to apply the developed workflow to, e.g., larger scans, time-dependent problems and non-linear problems. Detailed recommendations for specific further developments can be found in our referenced work; see Ref. [64] for a summary of these.

References

1. Y. Zhang, Challenges and advances in image-based geometric modeling and mesh generation, in *Image-Based Geometric Modeling and Mesh Generation*, ed. by Y. Zhang. Lecture Notes in Computational Vision and Biomechanics (Springer Netherlands, Dordrecht, 2013), pp. 1–10
2. T.J.R. Hughes et al., Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Comput. Methods Appl. Mech. Eng. **194**, 4135–4195 (2005)
3. D.F. Rogers, An introduction to NURBS: with historical perspective, in *The Morgan Kaufmann Series in Computer Graphics* (Morgan Kaufmann Publishers, San Francisco, 2001)
4. J.A. Cottrell et al., *Isogeometric Analysis: Toward Integration of CAD and FEA* (Wiley, New York, 2009), 356 pp.
5. J.A. Cottrell et al., Isogeometric analysis of structural vibrations. Comput. Methods Appl. Mech. Eng. **195**, 5257–5296 (2006). John H. Argyris Memorial Issue. Part II
6. Y. Zhang et al., Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. Comput. Methods Appl. Mech. Eng. **196**, 2943–2959 (2007)
7. M. Ruess et al., Weak coupling for isogeometric analysis of non-matching and trimmed multi-patch geometries. Comput. Methods Appl. Mech. Eng. **269**, 46–71 (2014)

8. Y. Yu et al., Anatomically realistic lumen motion representation in patient-specific space–time isogeometric flow analysis of coronary arteries with time-dependent medical-image data. Comput. Mech. **65**, 395–404 (2020)

9. T.J.R. Hughes et al., Chapter 8—Smooth multi-patch discretizations in Isogeometric Analysis, in *Handbook of Numerical Analysis*, ed. by A. Bonito, R.H. Nochetto. Geometric Partial Differential Equations—Part II, vol. 22. (Elsevier, Amsterdam, 2021), pp. 467–543

10. M. Bucelli et al., Multipatch Isogeometric Analysis for electrophysiology: simulation in a human heart. Comput. Methods Appl. Mech. Eng. **376**, 113666 (2021)

11. Y. Zhang et al., Solid T-spline construction from boundary representations for genus-zero geometry, in *Computer Methods in Applied Mechanics and Engineering*. Higher Order Finite Element and Isogeometric Methods 249–252 (2012), pp. 185–197

12. M.-C. Hsu et al., An interactive geometry modeling and parametric design platform for isogeometric analysis. Computers and Mathematics with Applications. **70**, 1481–1500 (2015). High-Order Finite Element and Isogeometric Methods

13. B. Urick et al., Review of patient-specific vascular modeling: template-based isogeometric framework and the case for CAD. Arch. Comput. Meth. Eng. **26**, 381–404 (2019)

14. J. Parvizian et al., Finite cell method: h- and p-extension for embedded domain problems in solid mechanics. Comput. Mech. **41**, 121–133 (2007)

15. A. Düster et al., The finite cell method for three-dimensional problems of solid mechanics. Comput. Methods Appl. Mech. Eng. **197**, 3768–3782 (2008)

16. 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. Meth. Eng. **22**, 391–455 (2015)

17. E. Burman, Ghost penalty. C.R. Math. **348**, 1217–1220 (2010)

18. E. Burman, P. Hansbo, Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method. Appl. Numer. Math. **62**, 328–341 (2012). Third Chilean Workshop on Numerical Analysis of Partial Differential Equations

19. E. Burman et al., CutFEM: Discretizing geometry and partial differential equations. Int. J. Numer. Methods Eng. **104**, 472–501 (2015)

20. E. Rank et al., Geometric modeling, isogeometric analysis and the finite cell method. Comput. Methods Appl. Mech. Eng. **249–252**, 104–115 (2012). Higher Order Finite Element and Isogeometric Methods

21. D. Schillinger et al., 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. Eng. **249–252**, 116–150 (2012). Higher Order Finite Element and Isogeometric Methods

22. M. Ruess et al., Weakly enforced essential boundary conditions for NURBS-embedded and trimmed NURBS geometries on the basis of the finite cell method. Int. J. Numer. Methods Eng. **95**, 811–846 (2013)

23. D. Kamensky et al., An immersogeometric variational framework for fluid–structure interaction: Application to bioprosthetic heart valves. Comput. Methods Appl. Mech. Eng. **284**, 1005–1053 (2015). Isogeometric Analysis Special Issue

24. M.-C. Hsu et al., Direct immersogeometric fluid flow analysis using B-rep CAD models. Comput. Aided Geom. Des. **43**, 143–158 (2016). Geometric Modeling and Processing 2016

25. C.V. Verhoosel et al., Image-based goal-oriented adaptive isogeometric analysis with application to the micro-mechanical modeling of trabecular bone. Comput. Methods Appl. Mech. Eng. **284**, 138–164 (2015). Isogeometric Analysis Special Issue

26. M. Ruess et al., The finite cell method for bone simulations: verification and validation. Biomech. Model. Mechanobiol. **11**, 425–437 (2012)

27. F. de Prenter et al., Multigrid solvers for immersed finite element methods and immersed isogeometric analysis. Comput. Mech. **65**, 807–838 (2020)

28. T. Hoang et al., Skeleton-stabilized immersogeometric analysis for incompressible viscous flow problems. Comput. Methods Appl. Mech. Eng. **344**, 421–450 (2019)

29. A. Düster et al., Numerical homogenization of heterogeneous and cellular materials utilizing the finite cell method. Comput. Mech. **50**, 413–431 (2012)
30. J.N. Jomo et al., Robust and parallel scalable iterative solutions for large-scale finite cell analyses. Finite Elem. Anal. Des. **163**, 14–30 (2019)
31. M. Carraturo et al., Modeling and experimental validation of an immersed thermo-mechanical part-scale analysis for laser powder bed fusion processes. Addit. Manuf. **36**, 101498 (2020)
32. S.C. Divi et al., Error-estimate-based adaptive integration for immersed isogeometric analysis. Comput. Math. Appl. **80**, 2481–2516 (2020). High-Order Finite Element and Isogeometric Methods 2019
33. S.C. Divi et al., Topology-preserving scan-based immersed isogeometric analysis. Comput. Methods Appl. Mech. Eng. **392**, 114648 (2022)
34. T. Hoang et al., Mixed Isogeometric Finite Cell Methods for the Stokes problem. Comput. Methods Appl. Mech. Eng. **316**, 400–423 (2017)
35. G. Strang, G. Fix, *An Analysis of the Finite Element Method* (Wellesley-Cambridge Press, Cambridge, 2008), 400 pp.
36. S.C. Divi et al., Residual-based error estimation and adaptivity for stabilized immersed isogeometric analysis using truncated hierarchical B-splines. J. Mech. **38**, 204–237 (2022)
37. C. Giannelli et al., THB-splines: the truncated basis for hierarchical splines. Comput. Aided Geom. Des. **29**, 485–498 (2012). Geometric Modeling and Processing 2012
38. Y. Bazilevs et al., Isogeometric analysis: approximation, stability and error etimates for h-refined meshes, in *Mathematical Models and Methods in Applied Sciences*, vol. 16 (World Scientific Publishing Co., Singapore, 2006), pp. 1031–1090
39. G. Deng, L. Cahill, An adaptive Gaussian filter for noise reduction and edge detection, in *1993 IEEE Conference Record Nuclear Science Symposium and Medical Imaging Conference* (1993), pp. 1615–1619
40. M. Unser et al., On the asymptotic convergence of B-spline wavelets to Gabor functions. IEEE Trans. Inf. Theory **38**, 864–872 (1992)
41. V. Varduhn et al., The tetrahedral finite cell method: higher-order immersogeometric analysis on adaptive non-boundary-fitted meshes. Int. J. Numer. Methods Eng. **107**, 1054–1079 (2016)
42. B. Delaunay et al., Sur la sphere vide. Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk **7**, 1–2 (1934)
43. M. de Berg et al. *Computational Geometry Algorithms and Applications* (Spinger, Berlin, 2008)
44. F. de Prenter et al., Condition number analysis and preconditioning of the finite cell method. Comput. Methods Appl. Mech. Eng. **316**, 297–327 (2017). Special Issue on Isogeometric Analysis: Progress and Challenges
45. F. de Prenter et al., Preconditioning immersed isogeometric finite element methods with application to flow problems. Comput. Methods Appl. Mech. Eng. **348**, 604–631 (2019)
46. A. Hansbo, P. Hansbo, An unfitted finite element method, based on Nitsche's method, for elliptic interface problems. Comput. Methods Appl. Mech. Eng. **191**, 5537–5552 (2002)
47. A. Embar et al., Imposing Dirichlet boundary conditions with Nitsche's method and spline-based finite elements. Int. J. Numer. Methods Eng. **83**, 877–898 (2010)
48. F. de Prenter et al., A note on the stability parameter in Nitsche's method for unfitted boundary value problems. Comput. Math. Appl. **75**, 4322–4336 (2018)
49. F. de Prenter et al., Stability and conditioning of immersed finite element methods: analysis and remedies. Arch. Comput. Methods Eng. 1–40 (2023). https://link.springer.com/article/10.1007/s11831-023-09913-0
50. E.H. van Brummelen et al., An adaptive isogeometric analysis approach to elasto-capillary fluid-solid interaction. Int. J. Numer. Methods Eng. **122**, 5331–5352 (2021)
51. S. Badia et al., Mixed aggregated finite element methods for the unfitted discretization of the Stokes problem. SIAM J. Sci. Comput. **40**, B1541–B1576 (2018). Publisher: Society for Industrial and Applied Mathematics
52. S. Badia et al., Linking ghost penalty and aggregated unfitted methods. Comput. Methods Appl. Mech. Eng. **388**, 114232 (2022)

53. E. Burman, P. Hansbo, Edge stabilization for the generalized Stokes problem: a continuous interior penalty method. Comput. Methods Appl. Mech. Eng. **195**, 2393–2410 (2006)
54. A. Abedian et al., Performance of different integration schemes in facing discontinuities in the finite cell method. Int. J. Comput. Methods **10**, 1350002 (2013). Publisher: World Scientific
55. A. Abedian et al., The finite cell method for the J2 flow theory of plasticity. Finite Elem. Anal. Des. **69**, 37–47 (2013)
56. A. Ern, J.-L. Guermond, *Theory and Practice of Finite Elements: 159* (Springer, New York, 2004), 540 pp.
57. A. Abedian, A. Düster, Equivalent Legendre polynomials: numerical integration of discontinuous functions in the finite element methods. Comput. Methods Appl. Mech. Eng. **343**, 690–720 (2019)
58. R. Becker et al., A finite element method for domain decomposition with non-matching grids. ESAIM: Mathematical Modelling and Numerical Analysis **37**, 209–225 (2003). Number: 2 Publisher: EDP Sciences
59. M. Juntunen, R. Stenberg, Nitsche's method for general boundary conditions. Math. Comput. **78**, 1353–1374 (2009)
60. F. Chouly et al., Residual-based a posteriori error estimation for contact problems approximated by Nitsche's method. IMA J. Numer. Anal. **38**, 921–954 (2018)
61. W. Dörfler, A convergent adaptive algorithm for poisson's equation. SIAM J. Numer. Anal. **33**, 1106–1124 (1996). Publisher: Society for Industrial and Applied Mathematics
62. G. Kuru et al., Goal-adaptive Isogeometric Analysis with hierarchical splines. Comput. Methods Appl. Mech. Eng. **270**, pp. 270–292 (2014)
63. R. Verfürth, A review of a posteriori error estimation and adaptive mesh refinement techniques, in *Advances in Numerical Mathematics* (Wiley-Teubner, Chichester; New York, 1996)
64. S.C. Divi, Scan-based immersed isogeometric analysis. ISBN: 9789038654690. PhD thesis (Eindhoven University of Technology, Eindhoven, 2022)
65. G.v. Zwieten et al., *Nutils*. Version 7.0. (2022)