



Higher-order block-structured hex meshing of tubular structures

Domagoj Bošnjak¹ · Antonio Pepe² · Richard Schussnig³ · Dieter Schmalstieg² · Thomas-Peter Fries¹

Received: 17 January 2023 / Accepted: 24 April 2023
© The Author(s) 2023

Abstract

Numerical simulations of the cardiovascular system are growing in popularity due to the increasing availability of computational power, and their proven contribution to the understanding of pathodynamics and validation of medical devices with in-silico trials as a potential future breakthrough. Such simulations are performed on volumetric meshes reconstructed from patient-specific imaging data. These meshes are most often unstructured, and result in a brutally large amount of elements, significantly increasing the computational complexity of the simulations, whilst potentially adversely affecting their accuracy. To reduce such complexity, we introduce a new approach for fully automatic generation of higher-order, structured hexahedral meshes of tubular structures, with a focus on healthy blood vessels. The structures are modeled as skeleton-based convolution surfaces. From the same skeleton, the topology is captured by a block-structure, and the geometry by a higher-order surface mesh. Grading may be induced to obtain tailored refinement, thus resolving, e.g., boundary layers. The volumetric meshing is then performed via transfinite mappings. The resulting meshes are of arbitrary order, their elements are of good quality, while the spatial resolution may be as coarse as needed, greatly reducing computing time. Their suitability for practical applications is showcased by a simulation of physiological blood flow modelled by a generalised Newtonian fluid in the human aorta.

Keywords Mesh generation · Convolution surfaces · Block structure · Higher-order meshes · Transfinite mappings

1 Introduction

Domains with a tubular structure are featured in various applications, particularly in biomedically motivated ones, such as blood flow and bronchial airflow [1–3]. Such geometries can be reconstructed from medical imaging data for

different applications, including computational fluid dynamics (CFD) (Fig. 1). The discretization of these domains is most often performed through unstructured mesh generation, supported by its capabilities to robustly capture complex geometries [4, 5]. The lack of control over the number of nodes and elements in the mesh, as well as the number of elements sharing a given node [6], are among the reasons one may prefer *structured* meshing approaches. The difficulties in structured mesh generation for nontrivial geometries are well-known, though they may be alleviated by providing sufficient topological and geometrical domain information, as well as a priori identification of potential singularities or sharp jumps arising in simulations.

A prominent example for special solution features, which should be considered in the mesh generation, are *boundary layers* in computational fluid dynamics [7], where one expects very strong velocity gradients. Special treatment of boundary layers is vital for any fluid simulation, and it should therefore be performed on the mesh level. Hexahedra are the imminently suitable element type in this regard, bolstered by the ability to induce anisotropy via *mesh grading* [8], with a 2D example in Fig. 2b. Thereby, tailored mesh refinement is obtained, inducing thinner elements

✉ Domagoj Bošnjak
bosnjak@tugraz.at

Antonio Pepe
antonio.pepe@tugraz.at

Richard Schussnig
richard.schussnig@uni-a.de

Dieter Schmalstieg
schmalstieg@tugraz.at

Thomas-Peter Fries
fries@tugraz.at

¹ Institute of Structural Analysis, Graz University of Technology, 8010 Graz, Austria

² Institute of Computer Graphics and Vision, Graz University of Technology, 8010 Graz, Austria

³ High-Performance Scientific Computing, University of Augsburg, 86159 Augsburg, Germany

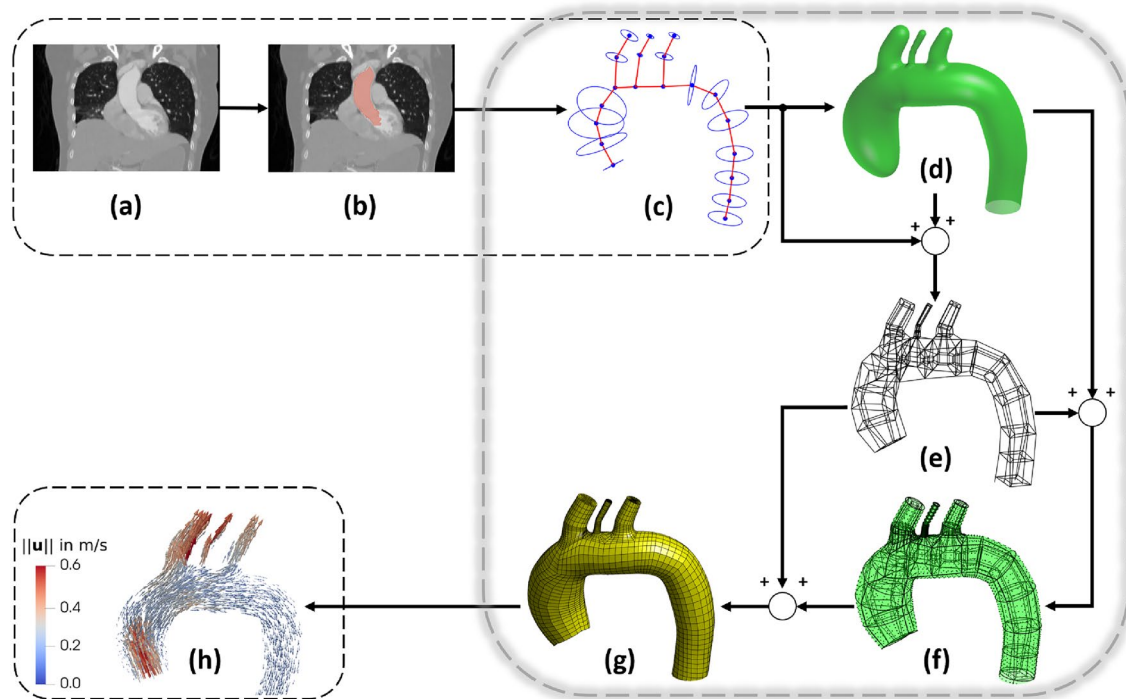


Fig. 1 The overall pipeline: **a** the CT scan, **b** the CT scan with segmentation, **c** the extracted skeleton, **d** the convolution surface, **e** the block-structure, **f** the higher-order surface mesh on top of the block-structure, **g** the volumetric mesh, **h** velocity u obtained via a fluid flow simulation

near the domain boundary, capturing the aforementioned phenomenon whilst retaining the number of mesh elements. Alternatively, tetrahedra are a common choice, given their flexibility and popularity in unstructured mesh generation [5], though hex or hex-dominant unstructured approaches are also available [9]. Also, meshes may consist of both hexahedra and tetrahedra, including prisms and pyramids for transition [10].

Automatic structured hexahedral mesh generation is commonly performed on specific classes of domains. Herein, we restrict ourselves to healthy blood vessels, with the emphasis on patient-specific aortas. Such structures naturally admit a *topological skeleton*, i.e., a centerline abstracting the topology of the domain. Skeleton-based mesh generation includes a wide variety of approaches. One approach might be to consider so-called quad layout extraction [11], which was also extended to three dimensions in [12]. An important feature is the placement and subdivision of cubes at the junction points of the skeleton. Additional mesh optimization is performed to guarantee element validity. In [13], a hex mesh generation based on filling a triangular surface mesh was presented, providing a level of control between surface fitting and minimum mesh quality. Generation of explicit and smooth cylindrical maps, i.e., maps between a tubular domain and a cylinder in the polar coordinate system, was performed in [14]. A recent skeleton-reliant method for so-called face extrusion quad meshes is given in [15]. Semiautomatic approaches

exist as well, e.g., for vascular modelling based on a block structure and a quadrilateral surface mesh [16]. High quality hexahedral meshing of large vascular structures based on a centerline input was performed by Ghaffari et al. [17]. A very recent and rather extensive survey of hex and hex-dominant mesh generation as well as mesh post-processing may be found in [18]. Furthermore, skeleton-based methods often go in the direction of quadrilateral *scaffolds*, i.e., coarse quadrilateral representations of the surface around the skeleton, such as scaffolding based on Voronoi diagrams [19], also mentioned by the same authors in [20], as well as a scaffolding approach in [21], citing a minimal number of quads for the given topological regularity. A somewhat similar three-dimensional concept is a *block-structure*, i.e., a division of the domain into coarse blocks. As the block structure captures the topology of the domain, and the skeleton is meant to abstract the very same topology, it was a natural choice to merge these two approaches. Relevant examples include a method based on a block structure enabling different element types presented in [22], and a survey of block-structured approaches in [23]. A comparison of computational fluid dynamics meshes generated based on various block structure generation methods was performed as well [24]. The block structure approach additionally enables the generation of rather coarse, but domain-conforming meshes. In particular, this enables applying the *geometric multigrid method* [25], a powerful iterative algorithm, which

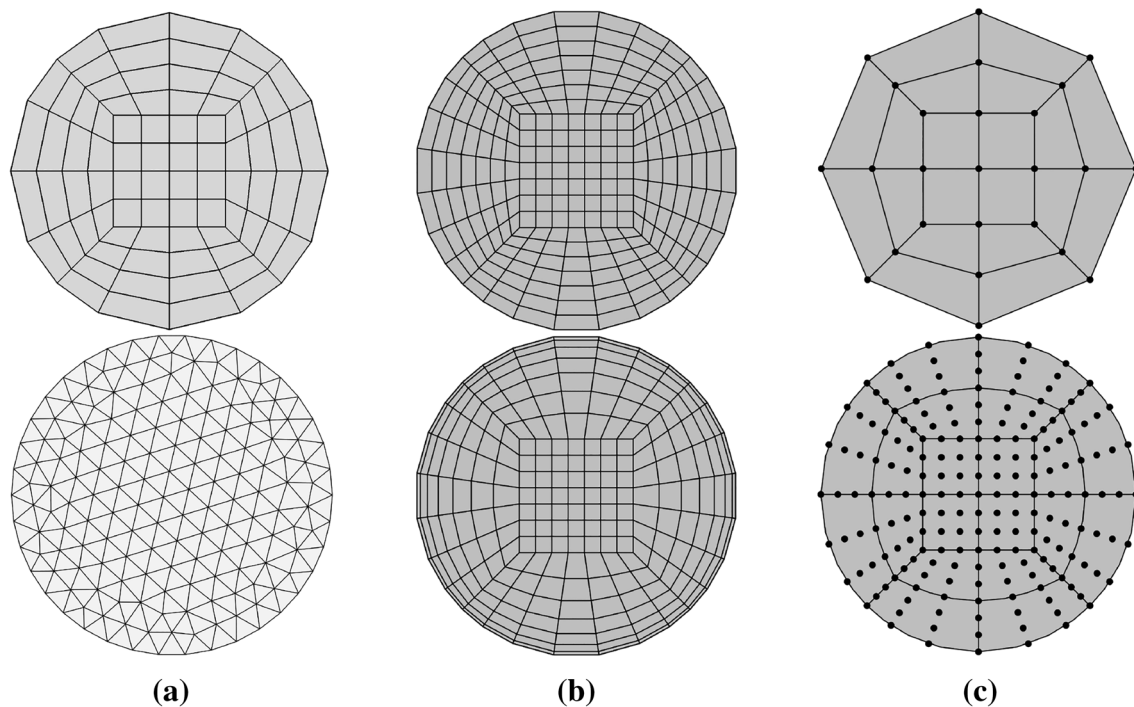


Fig. 2 Demonstration of mesh features: **a** a structured quad mesh and an unstructured triangle mesh, **b** an isotropic mesh and an anisotropic graded mesh, and **c** a linear and a higher-order mesh, where both have the same amount of elements

however requires the availability of several nested grids with varying spatial resolution of the same domain.

Most of the aforementioned work is focused on obtaining linear (first-order) meshes. Therein, geometric and approximation errors can be reduced by using more elements to discretise the domain, which is known as *h-FEM*. Instead of using *more* linear elements, one may rather improve the *order* of the elements, known as *p-FEM* [26]. Better performance is achieved by increasing the polynomial degree of the FE shape functions, and thus the number of nodes per element, as shown in Fig. 2c. A well-known example of a higher-order approach are the so-called *Taylor-Hood* elements [27]. They are particularly popular when solving incompressible flow problems, where we seek to determine pressure and velocity. Therein, elements of order k are used for velocity, and of order $k - 1$ for pressure, noting that equal orders lead to instabilities [28]. Many modern fluid flow solvers are turning to the higher-order paradigm as well [29]. Moreover, structured, higher-order, hexahedral meshes are the standard in isogeometric analysis [30, 31]. An early work by Zhang et al. offers a NURBS-based sweeping method for mesh generation [32]. The generation of truncated T-splines, showing desirable properties whilst enabling localized refinement, was the focus in [33]. Patient-specific flow with the focus on higher-order representation of realistic motion in a single lumen may be found in [34]. A review of patient-specific NURBS construction was given

in [35]. Finally, the process of going from scanned images all the way to applicable geometrical models was covered in detail in a recent book [36], and earlier in the review paper [37]. Further advantages of higher-order approaches are discussed extensively in [38], wherein the lack of appropriate higher-order mesh generators was particularly emphasized.

The generation of higher-order meshes should be accompanied with a suitable shape representation method. Herein, we utilise *convolution surfaces* [39], relying heavily on the fact they offer a smooth representation. A more detailed overview of the topic of shape modelling, as well as accompanying details regarding convolution surfaces, are given in Sect. 2.

Finally, we highlight topological restrictions of the domains we mesh. The tubular shapes we encounter in principle contain only bifurcations, and only rarely trifurcations. However, the algorithm is capable of resolving certain, modestly complex n -furcations where $n \leq 6$. A particularly problematic situation of interest, to an extent already mentioned in [17], is the occurrence of multiple bifurcations very close to each other. This is only amplified by the potential large differences in the radii of the vessels. Similar to [12], we place cubes at junctions, but the present approach differs in the cube schemes, refinement strategies and re-orientation algorithms. In particular, our tactic for skeleton-based block structure generation is only locally dependent on the skeleton, thus *loops* (i.e., a skeleton, or parts of it,

without any endpoints) do not present particular issues, as shown in examples in Sect. 4. The algorithm is furthermore capable of resolving loops in the skeleton, which might be encountered, e.g., when modelling the entire venal or arterial system. This aspect, however, is not of immediate interest within the present work.

1.1 Contribution

In this work, we introduce the following features:

- Automatic generation of structured hexahedral meshes of arbitrary order, utilising the convolution surfaces for surface representation, as the fact that they are smooth perfectly complements the notion of higher-order meshes.
- A skeleton-based block structure generation method yielding very coarse (valid) meshes, contained in an approach that is not limited to existing block structure templates, but instead flexible towards new ones.
- A block-structured mesh generator that offers straightforward *parametric* control over mesh grading, through which the user easily controls the elements representing the boundary layers, without introducing new elements or affecting the surface representation quality.

Figure 1 showcases the entire pipeline from a CT scan to a structured volume mesh, and its application in a numerical simulation.

2 Shape modelling of vascular structures

This section provides an overview of shape modelling techniques to represent vascular structures with an emphasis on the method used in our workflow (Fig. 1a–d). The aim is to represent vascular structures as smooth and differentiable shapes, a common requirement of different numerical

tasks, like gradient-based optimization, and that guarantees the absence of domain discontinuities in numerical simulations. For this, shape acquisition is a preliminary step and 3D imaging modalities like CTA represent the gold standard for acquisition and clinical assessment of deeper vascular structures [40]. After image acquisition, 3D reconstruction of such structures is done by segmenting the blood vessel in the CTA volume (Fig. 1a, b), typically after a windowing and denoising step as preprocessing [41]. The segmentation process generates a binary representation of the vascular structure inside the CTA grid. Recent reviews provide a broad overview of this topic [40, 41]. However, binary segmentations provide only a coarse and discrete representation. For visualisation purposes, it is common to build surface meshes by means of Marching Cubes (MC). However, MC is characterized by poor mesh quality [42] and supports only unstructured meshing (Fig. 3, left). Shape modelling can support the generation of higher quality meshes [43]. Distance transforms, which represent shapes as the distance from their surface, can be coupled with algorithms like Marching Tetrahedra to solve most issues of MC [44], yet the computed mesh is unstructured. Template deformations are a common technique to represent shapes with tailored meshes when topology is constant [45]. For tubular structures with variable topology, like blood vessels, skeleton-based representations are a valid alternative for shape representation and modelling [46]. Skeletons are generally 1D representations of the vascular structures that preserve their morphology. They can be combined with radial information to describe the cross-sectional shape of a blood vessel. Different cross-sectional priors have been considered, ranging from circles, to ellipses, polynomials, and Fourier descriptors [46, 47]. The skeleton provides information about blood flow direction and can support the generation of structured meshes. Sweeping and NURBS patches can be used to generate structured meshes from skeletal representations [32]. However, it can be challenging to connect different patches

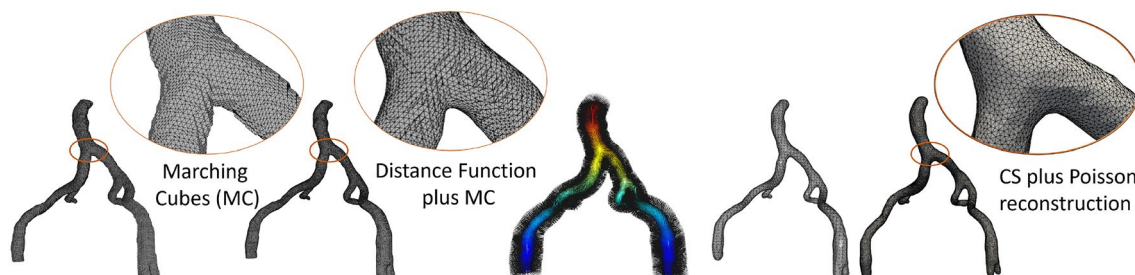


Fig. 3 Example of unstructured surface meshes of an Iliac bifurcation, representing the terminal part of the human aorta. From the left: Mesh explicitly generated using the Marching Cubes (MC) algorithm on the binary segmentation. Voxel artifacts are evident. The following mesh is generated using MC on a discrete distance function generated

from the binary volume. Voxel artifacts are less evident. Eventually, a level set of the convolution surface (CS) is generated from the skeleton of the Iliac bifurcation. The CS also provides the normal vectors for algorithms like Poisson reconstruction at different resolutions. No voxel artifacts are evident

[43]. Alternatively, implicit modelling techniques can be used to reconstruct the cross-sectional shape and interpolate between two cross sections by means of blending [46] or convolution surfaces [48] (Fig. 3). Implicit models represent a shape by means of level sets:

$$f(x, y, z) - C = 0. \tag{1}$$

In computational engineering, implicit descriptions of geometries and interfaces became popular under the label *level-set method* [49–51]. One may also incorporate (physical) concepts for the change or transport of the involved level-sets. Herein, we employ convolution surfaces for the implicit description of the geometry. The algorithm proposed herein also extends straightforwardly to other level-set representations, e.g. those based on signed distances.

2.1 Topological skeletons

After acquisition and segmentation, the topological skeleton provides the means to both model shape and structure (Fig. 1c). 3D shapes are typically represented either explicitly as surface meshes or implicitly as level sets. The latter also guarantee a degree of shape continuity, but neither representation provides information on shape *topology*. A way to analyze topology is to build the medial axis transform (MAT) of a given shape Σ [52]. The MAT is defined as the set of all spheres (C_i, r_i) that are maximally inscribed in Σ . The MAT is sensitive to noise as even small perturbations can change the size, number, or origin of the maximally inscribed spheres. For tubular structures, the MAT is approximated as the set of segments [48]

$$\Gamma = \{S_i = (C_i, C_j, r_i, r_j) : (C_i, r_i), (C_j, r_j) \in \text{MAT} \wedge i, j = 0, \dots, N - 1\},$$

where the number and length of the segments define the level of MAT approximation (Fig. 1c). While preserving the general shape, such shape representation allows easy identification of critical points such as bifurcations by analyzing the recurrence of a sphere origin C_i in the set. Bifurcation points are characterized by more than two occurrences, whereas an end point is characterized by a single occurrence. Different methods have been suggested to regress a shape to medial segments and build a topological skeleton [53], with mesh contraction being one of the most common ones [54].

2.2 Convolution surfaces

The MAT representation implicitly holds shape and topology information. While it is common to reconstruct shapes as the union of the medial axis spheres [55], implicit representations

of the medial spheres can guarantee smoothness and differentiability (Fig. 1d) [48, 56]. Particular examples of implicit representations include metaballs [57], and convolution surfaces [48]. Metaballs are defined as the convolution of N spherical Gaussian functions over an Euclidean distance metric, centered in O_i [57]:

$$f(P) = \sum_{i=0}^{N-1} b_i e^{-\sigma_i \|P - O_i\|_2^2}. \tag{2}$$

The chosen level C (Eq. 1) determines a geometric locus in the convolution domain. Convolution surfaces extend the concept of metaballs by replacing the sparse set of center points with a skeletal structure, which is often provided by a set of polylines [48] or Bézier curves [20] annotated with local thickness information along the skeleton. Given an arbitrary skeleton Γ , a convolution surface can be defined as

$$f(P) = \int_{\Gamma} e^{-\frac{1}{2} \|P - s\|_2^2} ds. \tag{3}$$

This formulation describes the locus of points P_i equally distant from Γ (Fig. 1d). Due to the linear properties of convolution, we can decompose Γ into non-overlapping regions such that $\Gamma = \bigcup_i \Gamma_i$, which lets us approximate the boundary surface of an arbitrary volume with a discrete set of primitives Γ_i (segments, curves, splines, etc.):

$$f(P) = \sum_i \int_{\Gamma_i} e^{-\frac{1}{2} \|P - s\|_2^2} ds. \tag{4}$$

2.3 Convolution surface with radius control

For accurate shape reconstruction, it is crucial to have full control and matching between the radii associated with the topological skeleton and the reconstructed shape surface. In its original formulation, the convolution kernel is an exponential function, as shown in Eq. 4 [48]. To describe tubular structures using their skeleton and radial thickness, we rely on the convolution kernel proposed by Fuentes Suárez et al. [20]. Given a skeleton represented with polylines, for a single skeleton segment S defined by points $\mathbf{a}, \mathbf{b} \in \mathbb{R}$, the convolution surface at $\mathbf{x} \in \mathbb{R}$ is given by

$$C_{\Gamma}^K(\mathbf{x}) = \int_0^1 K(g(\Gamma(s), \mathbf{x} - \Gamma(s)))g(\Gamma(s), \Gamma'(s)) ds, \tag{5}$$

$$C_{\Gamma}^K : \mathbb{R}^3 \rightarrow \mathbb{R},$$

where K denotes the kernel function and Γ denotes the parametrization of a skeleton segment $[a, b]$ defined as

$$\Gamma_s(s) = (1-s)a + sb, \quad \Gamma : [0, 1] \rightarrow \mathbb{R}^3. \quad (6)$$

We employ the same polynomial kernel function as Fuentes Suárez et al. [20], which is given by

$$K(x) = \begin{cases} \frac{35}{16}(1-x^2)^3, & x \in [0, 1], \\ 0, & \text{otherwise,} \end{cases} \quad K : \mathbb{R} \rightarrow \mathbb{R}, \quad (7)$$

as well as the same distance function g :

$$g(\mathbf{y}, \mathbf{x}) = \sqrt{\mathbf{x}^T G(\mathbf{y}) \mathbf{x}}, \quad (8)$$

where G is a positive definite symmetric matrix, recovered from its eigendecomposition $G = UDU^T$. The matrix U is the frame of the skeletal curve determining the convolution surface. As the skeleton elements are segments, each segment defines a constant matrix U . Moreover, D is a constant diagonal matrix since our focus is on circular cross-sections. Otherwise, for ellipsoidal cross-sections, D would need to be modified accordingly. Finally, the total convolution surface function is obtained as the sum of convolution surface functions of individual segments:

$$C^K(\mathbf{x}) = \sum_{s \in \mathcal{S}} C_{\Gamma_s}^K(\mathbf{x}). \quad (9)$$

The function C^K as well as $\nabla_x C^K$ are computed through numerical integration, adopting the Clenshaw–Curtis quadrature [58]. An important property of this kernel function is its finite support, meaning that individual segments in the skeleton affect only nearby segments. Hence, a more precise radius control is obtained, as well as reduced aliasing effects, which are the main reasons for using the framework from [20] as opposed to the exponential kernel approach from Oeltze and Preim [48]. Finally, since the convolution surface function is a sum of independent segment functions, its implementation is straightforward to parallelize, with an example shown in Fig. 4.

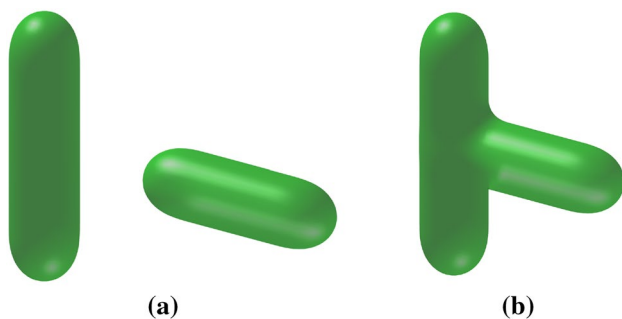


Fig. 4 Example of a convolution surface; firstly based on two separated segments, followed by two joined segments, to highlight the smooth blending effect

3 Block-structured higher-order mesh generation

Having described the convolution surfaces employed herein for the boundary of the domain, the mesh generation process is described, which is divided into four main steps:

1. Topological description of the domain using a block structure.
2. Mesh grading towards the boundary.
3. Generation of a higher-order surface mesh through an iterative procedure combined with the convolution surface level-set approach.
4. Volumetric mesh generation from the block structure and the surface mesh.

Volumetric mesh generation requires information about the domain topology and geometry. The first step therefore consists of generating a *block structure* [59, 60], which is a coarse subdivision of the domain into blocks, such that its topology matches the topology of the domain. An example of a block structure is shown in Fig. 1e. Grading information is then assigned to the block-structure, to induce mesh anisotropy refining towards the boundary. To properly account for geometry information, we then generate a higher-order *surface* mesh of the domain. Finally, the three are combined through the volumetric mesh generator, with the pipeline illustrated in Fig. 5, as well as in Fig. 1e–g. In this paper, we utilise hexahedra as blocks and quadrilaterals as surface mesh elements.

The continuity at the boundary shared by two or more blocks is C^0 . One has to distinguish the continuity of the geometry definition and the resulting (higher-order) hex-mesh. The geometry definition based on convolution surfaces may conceptionally be even C^∞ and a high level of continuity is an important asset for higher-order mesh generation, further justifying the choice of convolution surfaces in this work. On the other hand, no matter what the order of the resulting hex-mesh actually is, one can show that the resulting meshed geometry is only C^0 -continuous, due to the C^0 -continuous nature of the classical FEM shape functions. Of course, for higher orders, the kinks between element faces are extremely small. It is obvious that higher levels of *continuity* may not be achieved in a classical FEM framework but rather require novel approaches such as isogeometric analysis, coming with its own set of challenges.

3.1 Block structure generation

The generation of the block structure (Fig. 5b) is done automatically, based on a classification of the tubular

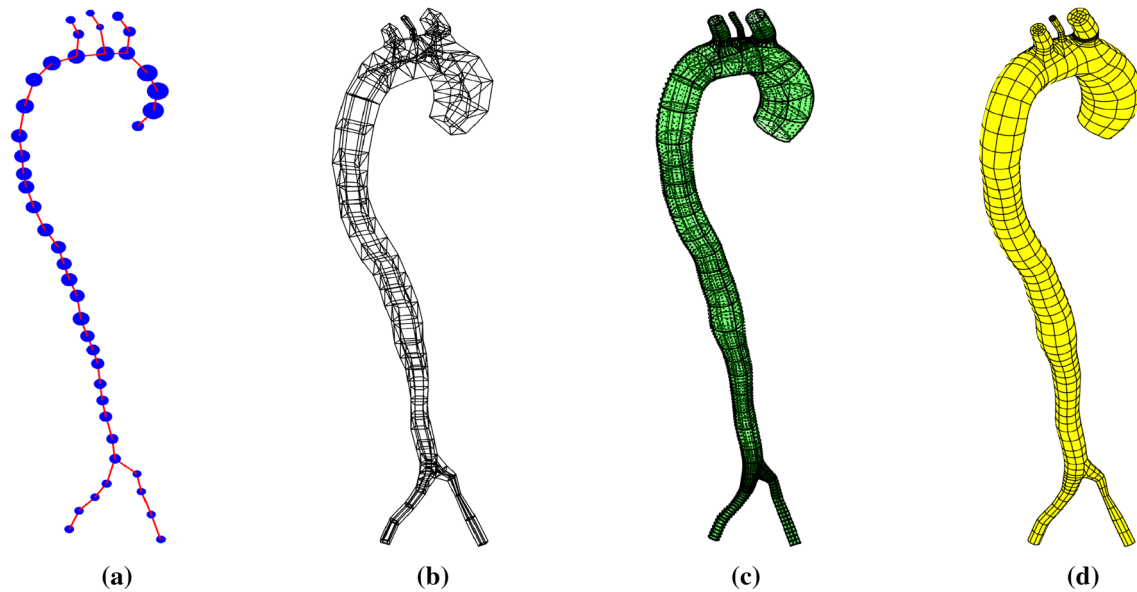


Fig. 5 Meshing pipeline demonstrated on the example of an aorta: **a** the skeleton where the point size indicates the relative radius at the given point, **b** the coarse block structure, **c** the higher-order surface mesh, and **d** the full volumetric mesh

structure into a few prototype cases, to be outlined below. Although we show concrete schemes for our designated application, the general approach is not limited to tubular structures. Different domain shapes would only require devising new schemes, without further changes in the pipeline. The process may be summarized as:

- Generation of blocks around junction points.
- Generation of blocks around non-junction points.
- Merging neighbouring junctions.
- Iterative repositioning of surface nodes.

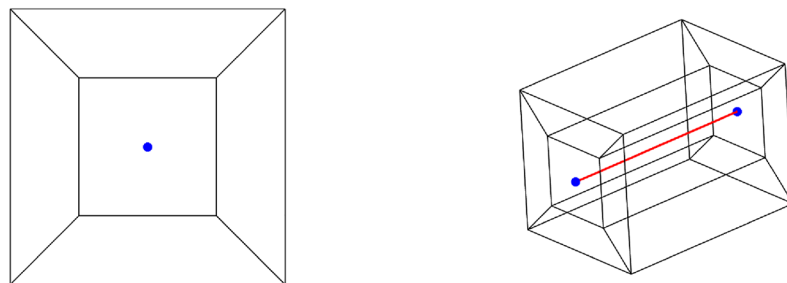
In the rest of this article, we refer to a skeletal point $X \in \Gamma$ as a *junction point* if it has 3 or more neighbours, otherwise we refer to it as a *non-junction point*. The latter are dealt with as follows: a cross-section is generated around each point, as

shown in Fig. 6a. They are then connected to form blocks, as shown in Fig. 6b. To determine the position of the cross-section around a skeleton point, three things are sufficient: the radius at the given point, the plane in which to place the cross-section, and its orientation inside that plane, i.e., torsion.

The radius of the cross-section is the input radius at the given skeleton point X . The plane is determined based on the neighbouring points of X . Assuming X is a non-junction point, it can have either one or two neighbours in the skeleton. For a point X with neighbours Y_1 and Y_2 , we denote the angle determined by those three points, with X being the central point, as $\angle Y_1XY_2$, i.e., the angle between $\overline{XY_1}$ and $\overline{XY_2}$. We choose the plane that divides the angle in half, meaning that for any point P in the plane it holds that

$$(P \neq X) \implies (\angle PXY_1 = \angle PXY_2),$$

Fig. 6 **a** An example of a cross-section formed around a single non-junction point of the skeleton, **b** five hexahedral blocks formed between two non-junction points (or two cross-sections) of the skeleton



(a) Cross-section around a skeleton point

(b) Blocks between two skeleton points

as illustrated in Fig. 7a, b. If X has only one neighbour Y , the plane is chosen to be perpendicular to the segment \overline{XY} , as shown in Fig. 7c. Finally, the torsion of the cross-section is chosen to minimize the torsion with its neighbours. In case of a domain with no junctions, one torsion is set randomly, but due to the tubular shape this does not negatively impact the final mesh. Otherwise, at least one junction scheme has already been generated, and the torsion is propagated from there onwards.

We now consider a junction point X of the skeleton, with neighbours Y_1, Y_2 and Y_3 . Instead of a cross-section, a structured cube is generated around X , as shown in Fig. 8a, consisting of seven hexahedral elements. The distance from X to the centers of the cube sides is the input radius at X . When connecting the cube with the cross-sections associated with non-junction points, the cube sides must be modified to yield conformal hexahedral blocks. The scheme is presented in Fig. 8b. It is worth noting, that subdividing the cube to accommodate a connection with a non-junction point affects only one of the original 7 hexahedra forming the cube, namely the hexahedron containing that very face. Thus, the junction cube refinement is not dependent on the order of the subdivisions. As each neighbouring point is assigned to one side of the cube, we seek to orient the cube based on an appropriate heuristic. It is oriented to maximize the angle formed by the segment connecting X to a given neighbour

Y_i , and the segment connecting X to that side of the cube, which we seek to connect with the cross-section around Y_i . Ideally, the branch cross-section is facing the side of the cube perpendicularly. Additional weights may be added to favour neighbours with a larger radius.

Due to the properties of convolution surfaces, we observe smooth blending between connecting objects, which fits well with the notion of anatomically motivated shapes [48]. Specifically, the smaller the angle between two branches at a junction, the stronger the effect from the smooth blending between them. To resolve this issue we subdivide the branch blocks into multiple parts, and connect them at the subdivision closest to the junction point, as shown in Fig. 9. The exact location is determined based on the convolution surface function.

In case of two junction cubes which are too close to each other or overlap, we merge them together and perform corrections to their inside schemes so as to account for the change caused by the merge. The overlap is shown in Fig. 10a, and the correction in Fig. 10b, c. Otherwise, blocks are formed between them in the same manner as between two cross-sections, see Fig. 6.

After all of the blocks have been generated, the surface nodes of the block structure are corrected using an iterative process. A node $\mathbf{x} = (x_1, x_2, x_3) \in \mathbb{R}^3$ is repositioned to the surface through an iterative procedure of

Fig. 7 Determining the plane for the cross-section depending on the number of neighbours of a non-junction skeleton point; **a** the plane determined by halving the angle between two segments, **b** the same situation in top-view and **c** the plane perpendicular to the segment

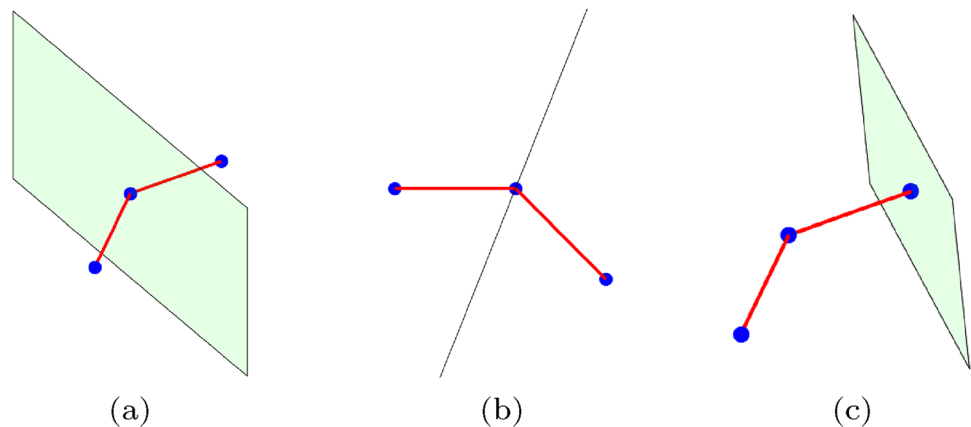


Fig. 8 **a** The scheme of a cube around a junction point, consisting of seven hexahedra. Each side that needs to be connected to a cross-section (see Fig. 6a) is equipped with a port, shown in **b** and **c**

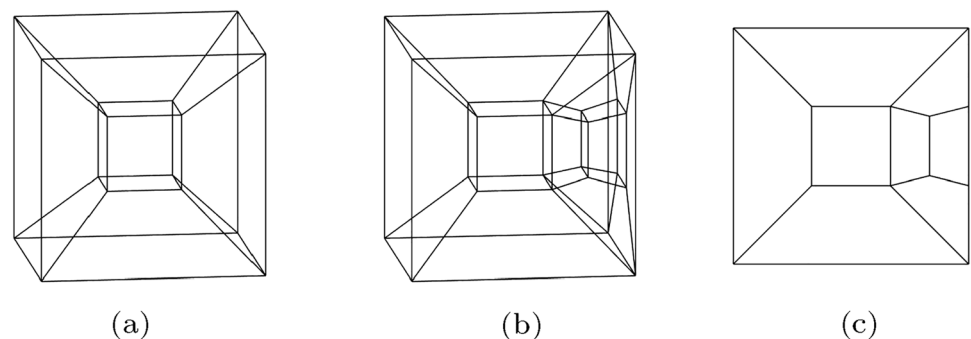


Fig. 9 The procedure of modifying the standard junction scheme shown in **a** to accommodate for a small angle between two branches; **b** the branches are subdivided depending on their length, and **c** connected depending on the location of the convolution surface blending

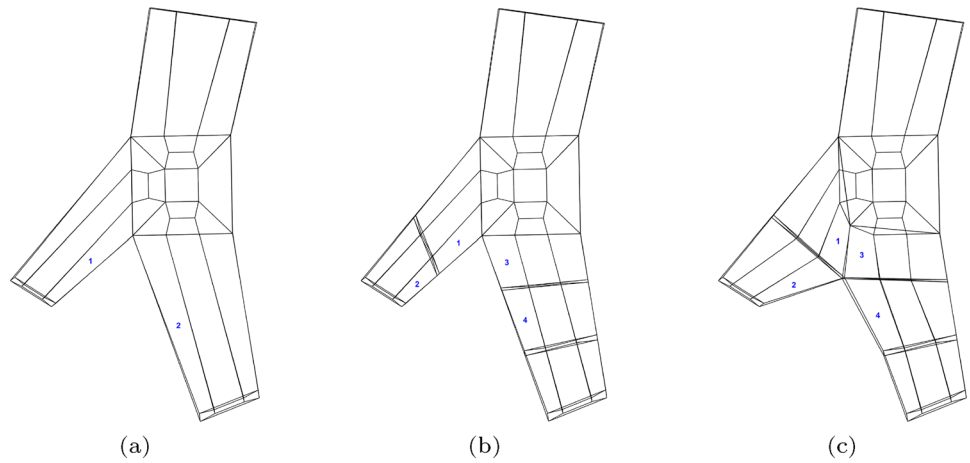
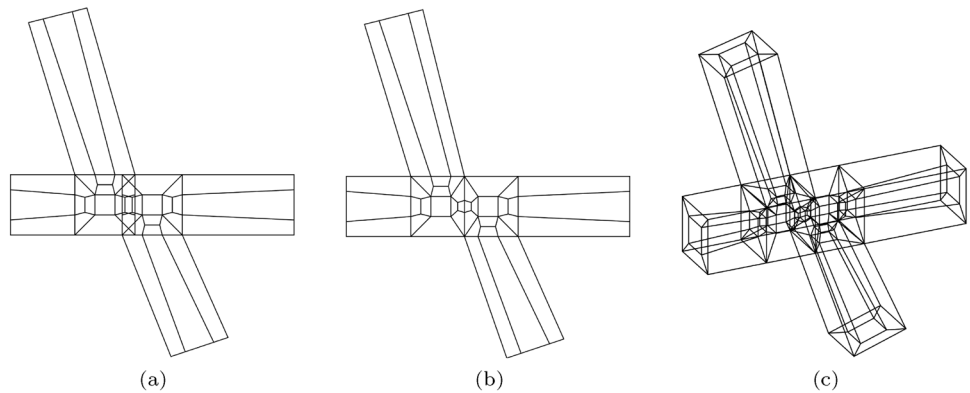


Fig. 10 The resolution of two junction cubes which are overlapping; **a** the overlap in side view, **b** the merge in side view, and **c** the merge in 3D view



finding a null-point, as described in Algorithm 1. The surface is represented as a level set $\{\mathbf{x} \in \mathbb{R}^3 : f(\mathbf{x}) = C\}$, where $f : \mathbb{R}^3 \mapsto \mathbb{R}$ is the convolution surface function and $C \in \mathbb{R}$ the chosen iso-value. However, within the presented framework, the block structure does not need to be aligned perfectly to the surface in the first step,

since the convolution surface function and Algorithm 1 are afterwards used to rectify initial positioning errors. Herein lies another advantage of the convolution surface: the fact that the function defining the surface is smooth allows utilising its gradient.

Algorithm 1 Iterative process of repositioning a point \mathbf{x} to the surface determined by $f(\mathbf{x}) = C$.

Input: $\mathbf{x} \in \mathbb{R}^3, f : \mathbb{R}^3 \rightarrow \mathbb{R}$, iso value $C \in \mathbb{R}$, tolerance $\varepsilon > 0$

$F(\mathbf{x}) = f(\mathbf{x}) - C$;

while $|F(\mathbf{x})| > \varepsilon$ **do**

$\mathbf{x} \leftarrow \mathbf{x} - \frac{F(\mathbf{x}) \cdot \nabla F(\mathbf{x})}{\|\nabla F(\mathbf{x})\|^2}$

3.2 Mesh grading

High-quality meshes for use in CFD must enable resolution of boundary layers, for which mesh grading is particularly useful [24, 38]. The goal is to obtain a tailored refinement of the mesh, with “thinner” elements near the boundary, as opposed to a standard uniform refinement. An example of a graded mesh compared to a uniformly refined mesh is shown in Fig. 2b. Most importantly, for the same number of elements, properly graded meshes lead to much better accuracy than uniform meshes.

An established approach to refining a hexahedron starts with subdividing each edge into n equal parts, to obtain uniform sub-hexahedra after refinement. To modify the uniform subdivision, a different *scaling* of the edges is introduced, thus resulting in a non-uniform refinement of the hexahedron. Here we use two types of scaling functions: quadratic and cubic. Assume that every edge is locally equipped with a coordinate t which linearly varies between -1 and 1 along the edge. For an input scaling parameter $m \in \langle -1, 1 \rangle$ the quadratic scaling transforms a value $x \in \mathbb{R}$ with the function

$$f_m(t) = m \frac{t^2}{2} + t + \frac{m}{2}, \tag{10}$$

and the cubic scaling with the function

$$f_m(t) = t - m \frac{t^3 - t}{2}. \tag{11}$$

We demonstrate the effects of the scaling using a simple discretisation of the interval $[-1, 1]$ into 20 subintervals, with the values of the input parameter $m \in \{-0.9, -0.8, \dots, 0.8, 0.9\}$, as well as the two extreme values close to -1 and 1 , respectively, as shown in Fig. 11. For $m = 0$ no scaling occurs, in both aforementioned cases. Quadratic scaling moves the points closer to the left or the right edge of the interval, depending on whether the input parameter m is positive or negative. Cubic scaling is symmetric, where negative values of m imply a denser subdivision towards the edges of the interval, and positive values of m imply a denser subdivision towards the middle. Figure 12b shows which edges in the block structure are selected for grading, as well as an example of an ungraded and a graded mesh, in Fig. 12c, d.

3.3 Higher-order surface mesh generation

Next, the convolution surface defined in Sect. 2, is converted (discretised) to a coarse, (very) high-order surface mesh,

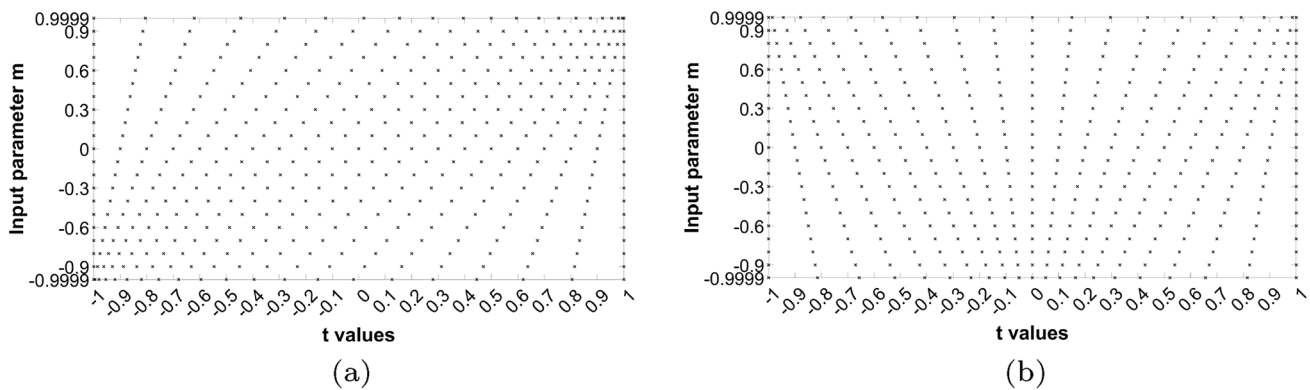


Fig. 11 The two examples of scaling the interval $[-1, 1]$ uniformly discretised into 20 subintervals, by **a** quadratic and **b** cubic scaling. The uniform discretisation is achieved for $m = 0$, in both scaling cases

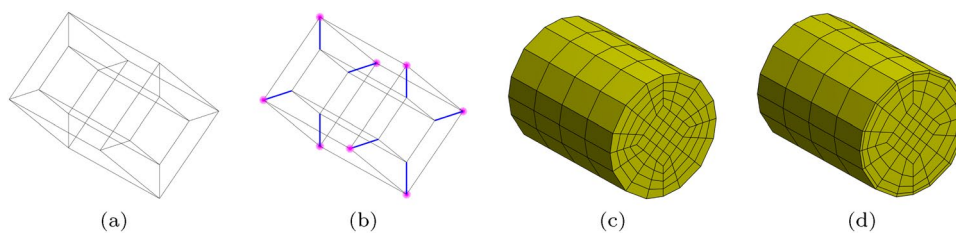


Fig. 12 a Shows the block structure, and **b** highlights its edges which are selected for grading (blue), with indicated edge endings (magenta). As grading is not necessarily symmetrical, edges for grad-

ing need to be properly oriented. The ungraded mesh is shown in **c**, and a quadratically graded mesh, with $m = 0.9$, in **d** (colour figure online)

Fig. 13 The procedure of generating higher-order edges on the surface of the block structure: **a** a segment lying on the surface is treated as a 1D finite element, **b** then converted to a sixth order element followed by **c** iterative inner node repositioning to the convolution surface. **d** The final result after all edges have been processed is shown

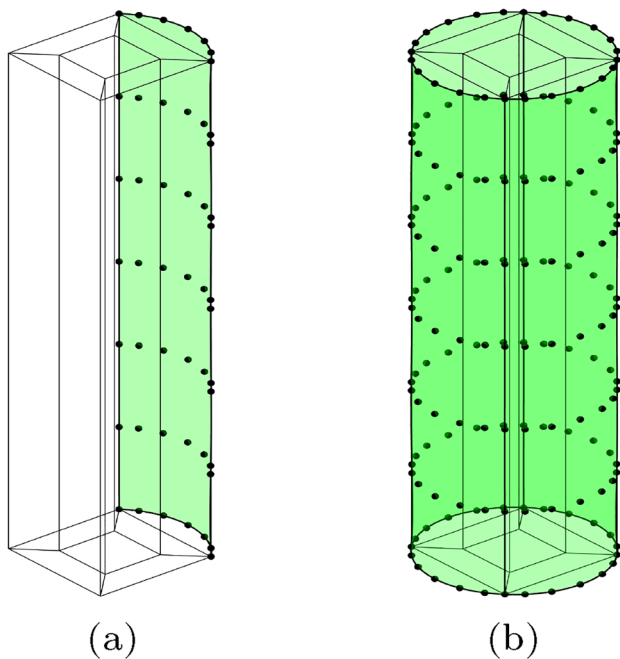
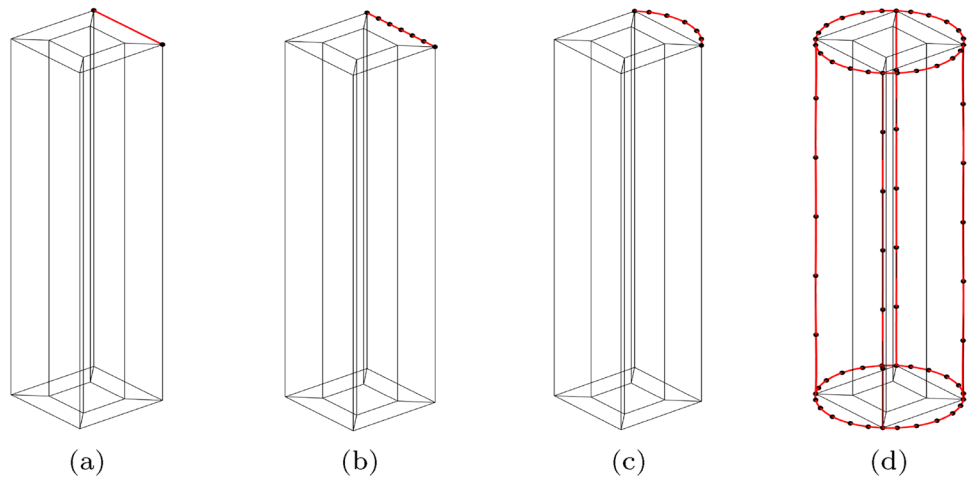


Fig. 14 The procedure of generating higher-order faces on the surface of the block structure: **a** a surface face is treated as a 2D finite element, formed based on the four edges; in case the inner nodes are not perfectly aligned on the surface they get repositioned. **b** The final surface after all of the nodes have been repositioned

which is then used to drive the resulting volumetric mesh of any desired order. This is illustrated in Fig. 1f. Each element of the surface mesh is associated to the outer faces of the building blocks. The pipeline of the higher-order surface mesh generation consists of the following steps:

- Generation of higher-order surface edges.
- Repositioning the inner edge nodes to the surface.
- Generation of higher-order surface faces from the edges.
- Repositioning the inner face nodes to the surface.

A surface edge, i.e., a segment \overline{XY} lying on the surface of the block structure, may be viewed as a linear 1D finite element embedded in a three-dimensional space. As such, it can be converted to a higher-order 1D element, as illustrated in Fig. 13b. The end points of the higher-order edge are the same as the end points of the original segment it was obtained from, i.e., the end points are already on the domain surface. Therefore, only the inner nodes of the edge need to be repositioned, whilst keeping the edge a valid element. This is again performed using Algorithm 1, and illustrated step-by-step in Fig. 13.

From the newly generated edges we obtain the faces through so-called *transfinite mappings*, which will be outlined in detail in Sect. 3.4. A face on the surface of the block structure may analogously be viewed as a two-dimensional element embedded in a three-dimensional space. The end points of the face are the nodes of the edges that form it, thus they already lie on the surface. Consequently, we reposition the inner nodes only, applying Algorithm 1 once again as illustrated in Fig. 14.

As individual edges and, afterwards, individual faces get processed independently, the accompanying computations are performed in parallel. After we have the topology information in the form of a coarse block structure, mesh grading assignment, and the geometry information embedded in the higher-order surface mesh, we turn to the final volumetric meshing step.

3.4 Volumetric mesh generation

The main goal is now to generate a volumetric mesh based on the topology, geometry and grading information outlined above, as illustrated in Fig. 1g. The desired number of elements per building block n_{el} and their order p are user-defined. This specifies the number of nodes on every edge of the building block structure $n_{edge} = n_{el} \cdot p + 1$ and,

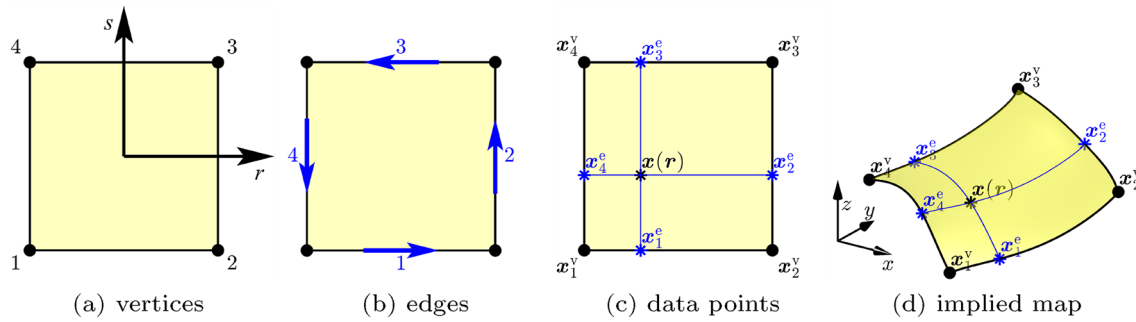


Fig. 15 Transfinite maps in quads: **a** local coordinate system (r, s) and vertex numbering, **b** edge numbering, **c** data points required to generate $\mathbf{x}(r)$, **d** interpreting the data in a Cartesian coordinate system (x, y, z)

thereby, also on every face, $n_{\text{face}} = n_{\text{edge}}^2$, and block interior, $n_{\text{block}} = n_{\text{edge}}^3$. To place the n_{block} nodes inside every block, we need the concept of so-called *transfinite maps*.

3.4.1 Transfinite maps in quads

The rationale of transfinite maps is first outlined based on generic quadrilateral faces, later for hexahedral blocks. The approach chosen here is well-documented in [61–63] and is often called *blending-function method*. The starting point is a quadrilateral reference element as seen in Fig. 15a with associated (r, s) -coordinate system and vertex numbering; (b) shows the numbering of the edges. The corresponding (bi-linear) shape functions $N_i(\mathbf{r})$ of this reference element are

$$N_1(\mathbf{r}) = \frac{1}{4}(1 - r) \cdot (1 - s), \quad N_2(\mathbf{r}) = \frac{1}{4}(1 + r) \cdot (1 - s),$$

$$N_3(\mathbf{r}) = \frac{1}{4}(1 + r) \cdot (1 + s), \quad N_4(\mathbf{r}) = \frac{1}{4}(1 - r) \cdot (1 + s).$$

One may then equip every edge with a ramp function $R_i(\mathbf{r})$ being 1 along its edge

$$R_1 = N_1 + N_2, \quad R_2 = N_2 + N_3,$$

$$R_3 = N_3 + N_4, \quad R_4 = N_4 + N_1.$$

Next, let there be some point $\mathbf{r} = (r, s)^T$ naturally related to each edge as seen in Fig. 15c. At these edge positions and the vertices, coordinate data $\mathbf{x}_i^e \in \mathbb{R}^3$ and $\mathbf{x}_i^v \in \mathbb{R}^3$ with $i = 1, \dots, 4$ is given, respectively. These coordinates may, of course, also be interpreted in a Cartesian (x, y, z) -coordinate system as seen in Fig. 15d. The transfinite map according to the blending-function method defines the position $\mathbf{x}(\mathbf{r})$ as follows:

$$\mathbf{x}(\mathbf{r}) = \sum_{i=1}^4 R_i(\mathbf{r}) \cdot \mathbf{x}_i^e - \sum_{i=1}^4 N_i(\mathbf{r}) \cdot \mathbf{x}_i^v. \quad (12)$$

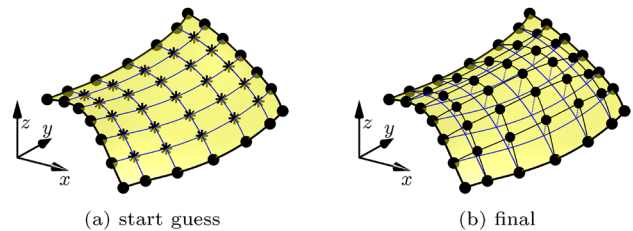


Fig. 16 **a** Transfinite maps based on 4 higher-order edge elements ($p = 6$) yield the black stars as start guesses for the inner face nodes. **b** These are then moved to the exact geometry via Algorithm 1 to generate the final higher-order surface element for the geometry definition

Interpreting this map $\mathbf{x}(\mathbf{r})$ for any point \mathbf{r} reveals that this equation defines the smooth surface between 4 edge curves forming a closed contour in the three-dimensional space \mathbb{R}^3 , see the black contour line in Fig. 15d and the resulting yellow surface.

Although this assessment mostly serves the purpose to pave the road to transfinite maps in hexahedra, it is also noted that the map in Eq. (12) is concretely used within the presented framework to generate start guesses for the face nodes of higher-order surface elements for the geometry definition as discussed in Sect. 3.3, see also Fig. 16.

3.4.2 Transfinite maps in hexas

We follow a similar outline as above and start with a hexahedral reference element as shown in Fig. 17a with local (r, s, t) -coordinate system and vertex numbering, (b) shows the numbering of the edges and faces. The (tri-linear) shape functions $N_i(\mathbf{r})$ in the hexahedron are

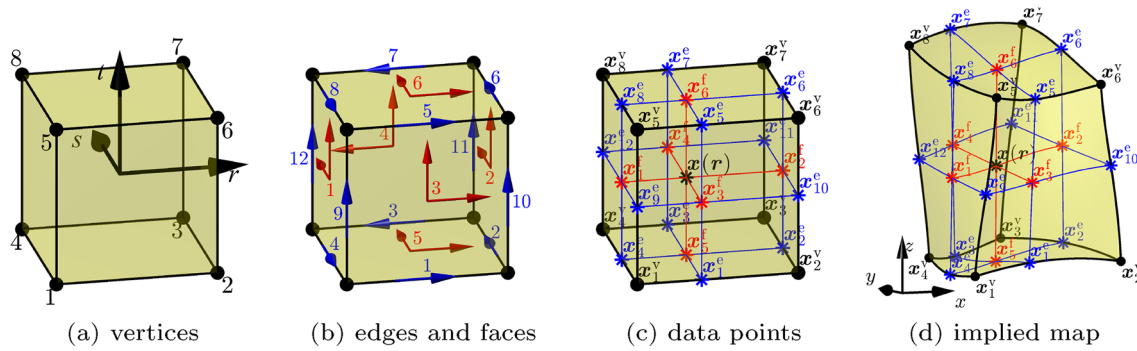


Fig. 17 Transfinite maps in hexes: **a** local coordinate system (r, s, t) and vertex numbering, **b** edge and face numbering, **c** data points required to generate $x(r)$, **d** interpreting the data in a Cartesian coordinate system (x, y, z)

$$\begin{aligned}
 N_1(r) &= \frac{1}{4}(1-r) \cdot (1-s) \cdot (1-t), \\
 N_2(r) &= \frac{1}{4}(1+r) \cdot (1-s) \cdot (1-t), \\
 N_3(r) &= \frac{1}{4}(1+r) \cdot (1+s) \cdot (1-t), \\
 N_4(r) &= \frac{1}{4}(1-r) \cdot (1+s) \cdot (1-t), \\
 N_5(r) &= \frac{1}{4}(1-r) \cdot (1-s) \cdot (1+t), \\
 N_6(r) &= \frac{1}{4}(1+r) \cdot (1-s) \cdot (1+t), \\
 N_7(r) &= \frac{1}{4}(1+r) \cdot (1+s) \cdot (1+t), \\
 N_8(r) &= \frac{1}{4}(1-r) \cdot (1+s) \cdot (1+t).
 \end{aligned}$$

Every edge is associated with a ramp function $R_i^e(r)$

$$\begin{aligned}
 R_1^e &= N_1 + N_2, & R_2^e &= N_2 + N_3, \\
 R_3^e &= N_3 + N_4, & R_4^e &= N_4 + N_1, \\
 R_5^e &= N_5 + N_6, & R_6^e &= N_6 + N_7, \\
 R_7^e &= N_7 + N_8, & R_8^e &= N_8 + N_5, \\
 R_9^e &= N_1 + N_5, & R_{10}^e &= N_2 + N_6, \\
 R_{11}^e &= N_3 + N_7, & R_{12}^e &= N_4 + N_8,
 \end{aligned}$$

and every face with a ramp function $R_i^f(r)$

$$\begin{aligned}
 R_1^f &= N_1 + N_4 + N_5 + N_8, & R_2^f &= N_2 + N_3 + N_6 + N_7, \\
 R_3^f &= N_1 + N_2 + N_5 + N_6, & R_4^f &= N_3 + N_4 + N_7 + N_8, \\
 R_5^f &= N_1 + N_2 + N_3 + N_4, & R_6^f &= N_5 + N_6 + N_7 + N_8.
 \end{aligned}$$

It is easily verified that edge ramp functions $R_i^e(r)$ are unity along their corresponding edges and face ramp functions $R_i^f(r)$ on their corresponding faces.

For some point $r = (r, s, t)^T$, there is a related position on every edge and face according to Fig. 17c. At these face and edge positions and the vertices, coordinate data $x_i^f \in \mathbb{R}^3$,

$i = 1, \dots, 6, x_j^e \in \mathbb{R}^3, j = 1, \dots, 12$, and $x_k^v \in \mathbb{R}^3, k = 1, \dots, 8$, is given, respectively. The interpretation of these coordinates in a Cartesian (x, y, z) -coordinate system is shown in Fig. 15d. The transfinite map according to the blending-function method for hexahedra defines the position $x(r)$ as follows:

$$x(r) = \sum_{i=1}^6 R_i^f(r) \cdot x_i^f - \sum_{i=1}^{12} R_i^e(r) \cdot x_i^e + \sum_{i=1}^8 N_i(r) \cdot x_i^v. \quad (13)$$

3.4.3 Mesh generation in a building block

With the transfinite maps defined above, the generation of a sub-mesh in every hexahedral building block may now be described as follows. Recall, that some of the faces of the building block may be equipped with a (very) high-order surface element defining the true geometry. Otherwise, the face is merely the bi-linear surface element from the building block itself. In any case, it is easily possible to map $n_{\text{face}} = n_{\text{edge}}^2$ nodes to every surface element resulting in the coordinates x_i^f . Simple compatibility requirements between the faces automatically yield the edge coordinates x_i^e and vertex coordinates x_i^v . Based on the transfinite map (13), every node inside the building block may now be generated, so that finally the coordinates of all $n_{\text{block}} = n_{\text{edge}}^3$ nodes are specified.

The situation is exemplified in Fig. 18: (a) shows six higher-order surface elements with order $p = 4$. The aim is to generate $5 \times 5 \times 5 = 125$ elements with order $p = 2$ in this building block. This gives $n_{\text{edge}} = 5 \cdot 2 + 1 = 11$ nodes per building block edge, $n_{\text{face}} = 11^2 = 121$ nodes per face and $n_{\text{block}} = 11^3 = 1331$ nodes for the whole sub-mesh in this building block. Figure 18b shows the n_{face} nodes per face (blue crosses) resulting from a simple finite element map based on the higher-order faces (black dots). The

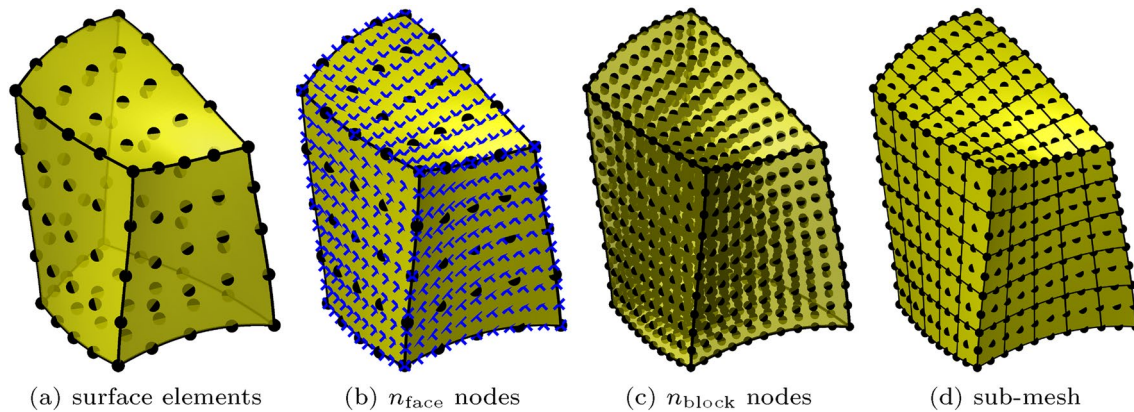


Fig. 18 The generation of a higher-order sub-mesh in a building block: **a** the higher-order surface elements ($p = 4$) for the geometry definition, **b** placing n_{face} nodes on each face, **c** placing n_{block} nodes

inside the building block via transfinite maps, **d** the resulting sub-mesh ($5 \times 5 \times 5$ elements with order $p = 2$) (colour figure online)

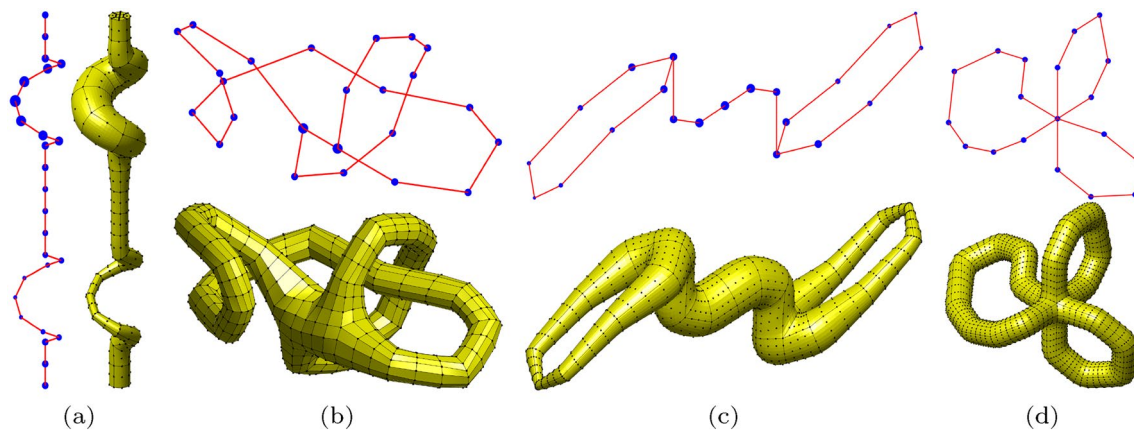


Fig. 19 Four examples of synthetic meshes, together with their skeletons, where the (relative) radius information is encoded in the size of the points. Further information relating to the meshes may be found in Table 1

position of the final n_{block} nodes of this building block is plotted in Fig. 18c.

Finally, the n_{block} generated nodes must be associated with elements which is achieved by setting up a proper connectivity matrix. This, however, is a standard task in the finite element method [10, 61, 64] and not further outlined herein. Continuing the example from before, the

final resulting sub-mesh is seen in Fig. 18d. Connecting all sub-meshes from the individual building blocks yields the desired volumetric mesh of the whole (tubular) geometry, to be used for FEM simulations and visualisations.

4 Experiments and results

4.1 Mesh examples

Here we demonstrate several meshes obtained with our method, followed by examples of simulations performed on them. First, we showcase several synthetic examples, i.e., meshes generated based on manually constructed skeletons. The radius information of the skeleton points is shown in the form of relative point sizes, as in Fig. 5. Meshes are generated with different refinement levels and mesh orders.

Table 1 Information about the skeletons and meshes shown in Fig. 19

Mesh example	(a)	(b)	(c)	(d)
Skeleton info				
Num. of points	26	24	21	19
Num. of segments	25	25	22	21
Mesh info				
Mesh order	2	1	4	3
Num. of nodes	1275	4988	10,735	32,338
Num. of elements	125	4428	154	1136

Particular situations of interest that occur in the synthetic meshes are neighbouring bifurcations in Fig. 19b, non-planar bifurcations in Fig. 19c and a six-furcation in Fig. 19d. The skeletons in the synthetic examples contain loops. Regarding this, the junction cubes are generated first, and not globally subdivided. More specifically, independent parts of the cube (each corresponding to one outer face of the cube) may be subdivided if a branch is coming into the aforementioned face, without affecting the other parts of the cube, as shown in Fig. 8b. Thus, the order of these subdivisions has no effect on the final configuration—the junction cube is the same in the end. Adding to this, the cross-sections around non-junction points are fully symmetric. Hence, no particular issues are encountered when the skeleton contains loops. Though they are

not of immediate interest for this application, unless one wants to model the entire arterial and venal system, we encountered no issues in modeling them.

Furthermore, we highlight two patient-specific examples; an aorta and a coronary artery. To ease reproducibility, we performed our experiments on a publicly available CTA imaging and segmentation of the human aortic tree from the AVT dataset [65] as a representative tubular structure. The binary segmentation is first triangulated using Marching Cubes and the 1D vascular skeletons are generated by mesh contraction [54]. To limit the number of skeletal points, they are resampled at a minimum distance and radial information is computed as the distance between the skeletal point and the surface mesh.

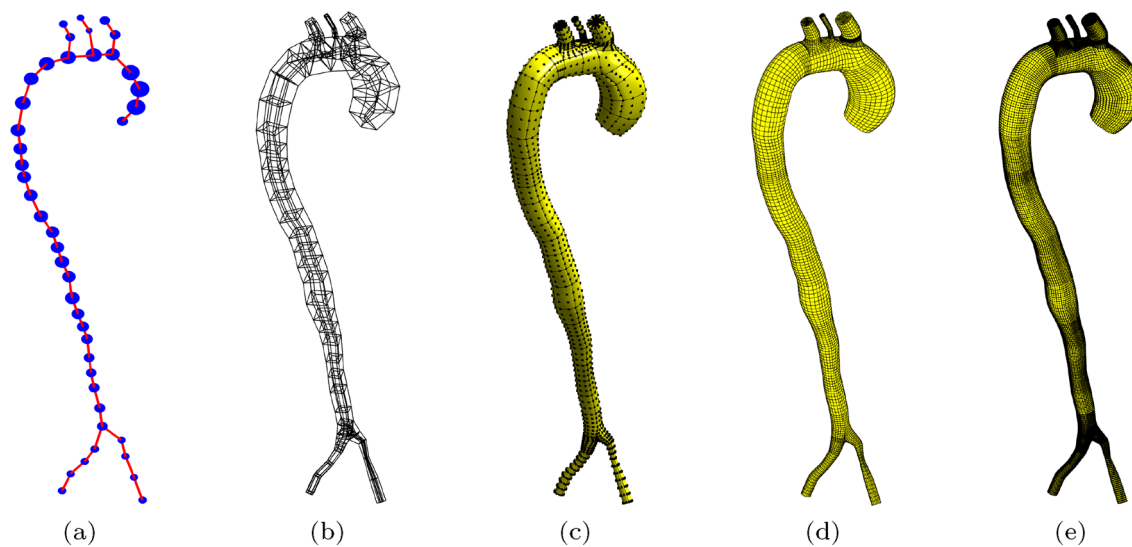


Fig. 20 An example of a patient-specific aorta: **a** the skeleton with radius information, **b** the block structure (320 blocks), **c** a coarse third-order mesh with 320 elements, **d** a refined first-order (linear) mesh with 69,120 elements, and **e** a refined second-order mesh with 69,120 elements

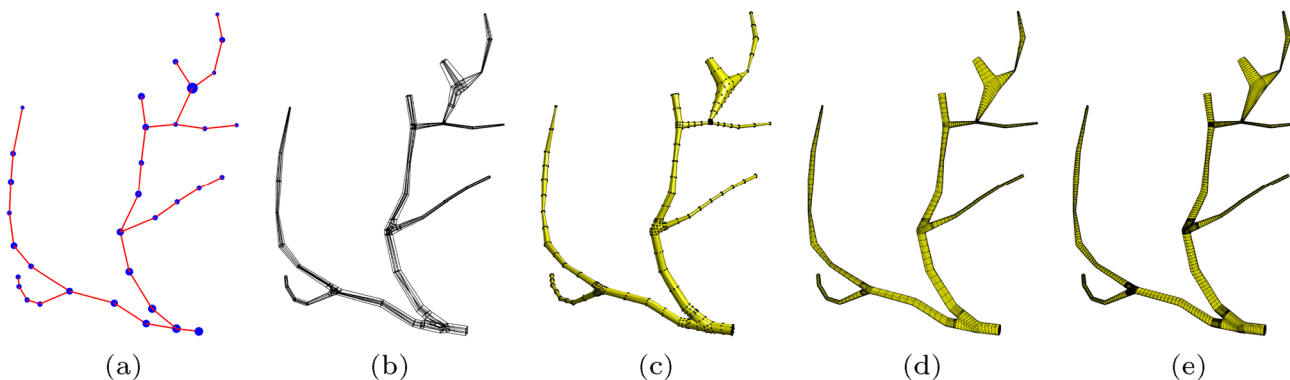


Fig. 21 An example of a patient-specific coronary artery: **a** the skeleton with radius information, **b** the block structure (432 blocks), **c** a coarse second-order mesh with 432 elements, **d** a refined first-

order (linear) mesh with 93,312 elements, and **e** a refined second-order mesh with 93,312 elements

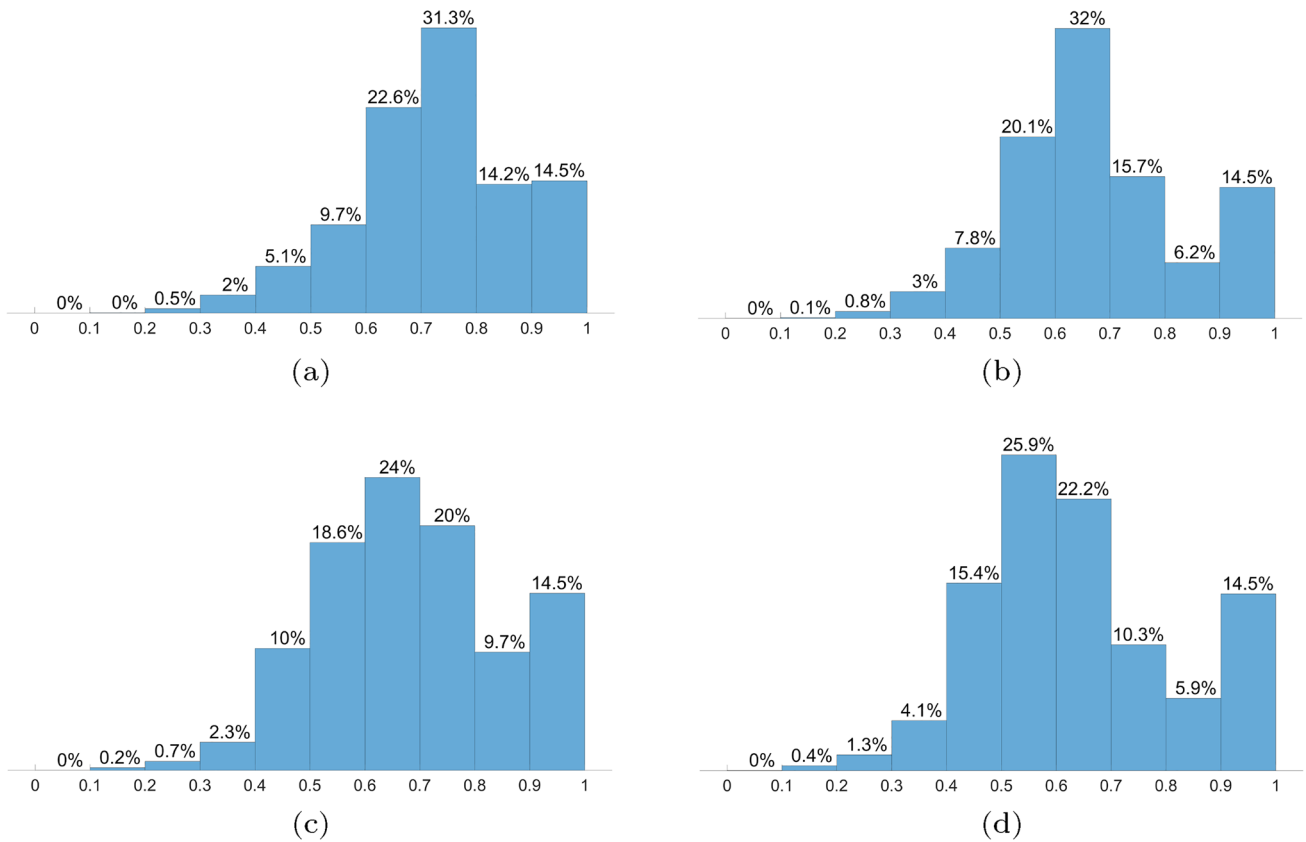


Fig. 22 Histograms of the values of the scaled Jacobian metric for the meshes of the aorta in Fig. 20d, e, followed by equivalent histograms for the meshes of the coronary artery given in Fig. 21d, e. Negative values indicate an invalid element, and a value of 1 indicates an ideal element

We show the skeleton and the block structure, together with several meshes, for an aorta in Fig. 20, and a coronary artery in Fig. 21. Additionally, we show histograms of element-wise values of the *scaled Jacobian* [16] metric in Fig. 22. The values may be in the range $(-\infty, 1]$, where a negative value indicates an invalid element (hence an unusable mesh), and a value of 1 indicates an ideal element. The scaled Jacobian is computed on the meshes with

increased refinement levels, as they are more suitable candidates for simulations.

Aside from the scaled Jacobian metric, another commonly utilised mesh quality metric is the *equiangular skewness quality* [16], defined as (for an element in the mesh)

$$S_{\text{elem}} = \max \left\{ \frac{\theta_{\max} - \theta_{\text{ideal}}}{180^\circ - \theta_{\text{ideal}}}, \frac{\theta_{\text{ideal}} - \theta_{\min}}{\theta_{\text{ideal}}} \right\}. \tag{14}$$

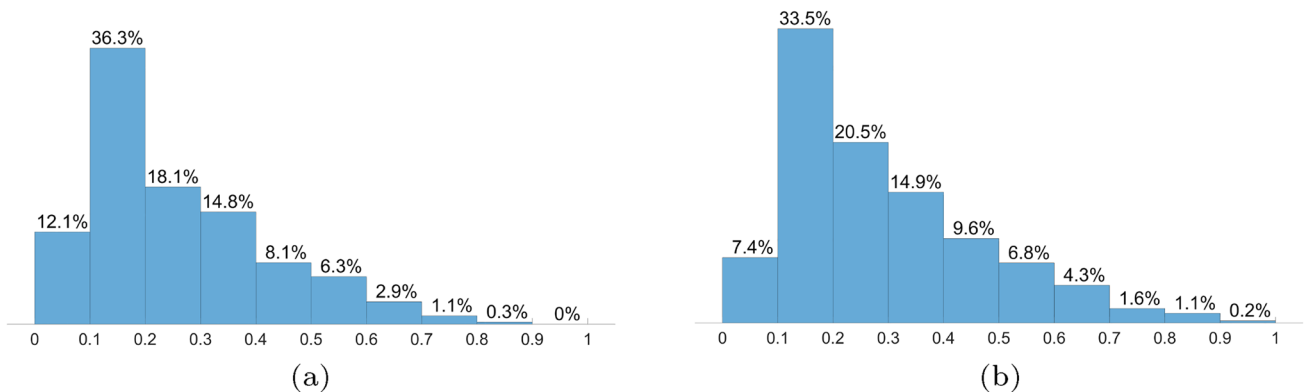


Fig. 23 Histograms of the values of the equiangular skewness quality metric for the linear mesh of the aorta in Fig. 20d, and the linear mesh of the coronary artery given in Fig. 21d. Values closer to zero are desired

Here, θ_{\max} and θ_{\min} denote the maximum and the minimum angle formed by edges of the element, respectively. It measures how far each element is from an ideal element, in the sense of angles between the edges of the element. In this regard, an ideal hexahedron is the one in which all of the aforementioned angles are exactly 90° , as it is the case in standard cubes or bricks. This is why $\theta_{\text{ideal}} = 90^\circ$. Unlike the Scaled Jacobian metric, values closer to zero are desirable, since an ideal element would satisfy $\theta_{\max} = \theta_{\min} = \theta_{\text{ideal}}$ and thus $S_{\text{elem}} = 0$. The values of the equiangular skewness are shown in Fig. 23.

4.2 Benchmarking higher-order accuracy

To demonstrate the higher-order accuracy of the proposed meshing technique and to reinforce the potential advantages of higher-order finite elements, we solve a Poisson equation with Dirichlet boundary conditions

$$-\Delta u = f \quad \text{in } \Omega, \tag{15}$$

$$u = 0 \quad \text{on } \partial\Omega, \tag{16}$$

on a domain Ω . The skeleton with the radius information is constructed manually, leading to a convolution surface description of the domain. We then generate meshes ranging from order 1 to 5. The corresponding approximation errors are measured with respect to the energy norm,

$$E := \int_{\Omega} \nabla u \cdot \nabla u \, dx. \tag{17}$$

The exact solution, and by extension the exact energy, are unknown. To evaluate the error with respect to energy, a reference value is thus required. Hence, we compute an overkill

solution, i.e., we allow for significantly more degrees of freedom in a single simulation to diminish approximation errors. Performing a convergence study, we then proceed in computing energy norms for orders 1 to 5 under grid refinement, computing approximation errors taking the overkill solution's value as reference. The mesh, the reference solution of the Poisson problem (15)–(16), and the convergence study in the energy error are shown in Fig. 24.

4.3 Practical application in aortic blood flow

In a final example, we showcase the applicability of the proposed scheme in the context of cardiovascular flow, that is, pulsatile blood flow through the reconstructed arterial tree. Here, we choose a computational domain starting from the aortic root down to the common iliac arteries involving the aortic arch with the brachiocephalic trunk, left common carotid and left subclavian arteries as depicted in Fig. 5. Having such a spatial discretisation at hand, we can directly apply solvers for incompressible flow of generalised Newtonian fluids [66, 67]. Considering five cardiac cycles, we prescribe the volumetric flow rate at the aortic root adapting the temporal scale from [1], and recover realistic outlet pressures via lumped parameter models. All involved parameters and models are chosen in the physiological range, but are omitted here for the sake of brevity, referring the interested reader to [3, 68].

As the snapshots of the fluid velocity and pressure at the fifth cycle's peak systole shown in Fig. 25 suggest, the hexahedral mesh constructed from medical image data via the proposed method is indeed capable of resolving the flow field properly. Here, we use a total of 40,000 hexahedral (trilinear) elements to accurately capture circulatory flow fields while showcasing applicability in the lower-order case, which is of high practical relevance as well. Note, however,

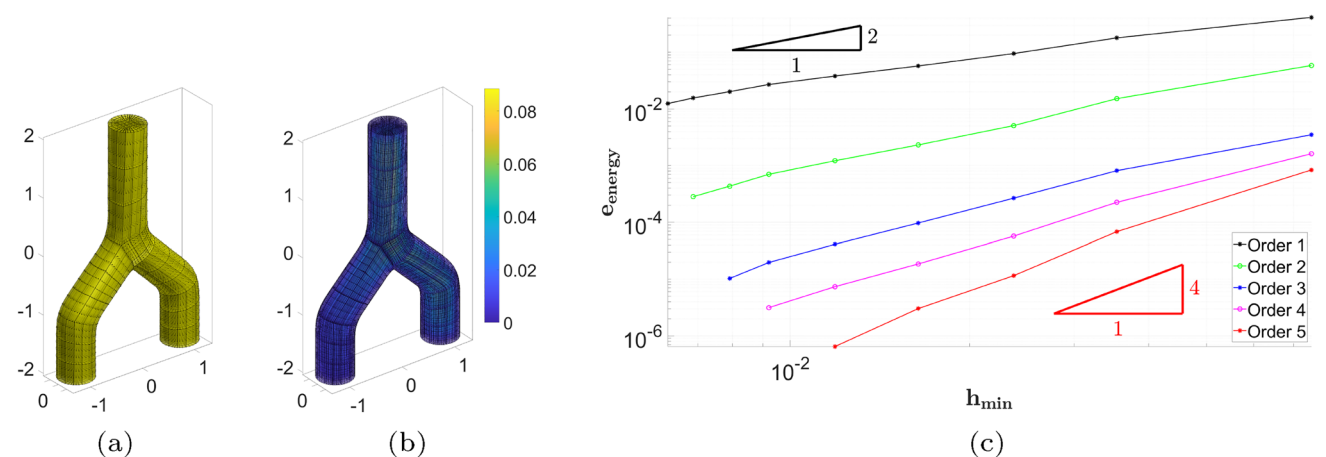


Fig. 24 **a** The mesh of the domain, followed by **b** the values of the solution. **c** The convergence study for different p-FEM orders and refinement levels, where the error is measured with respect to the energy defined in Eq. 17

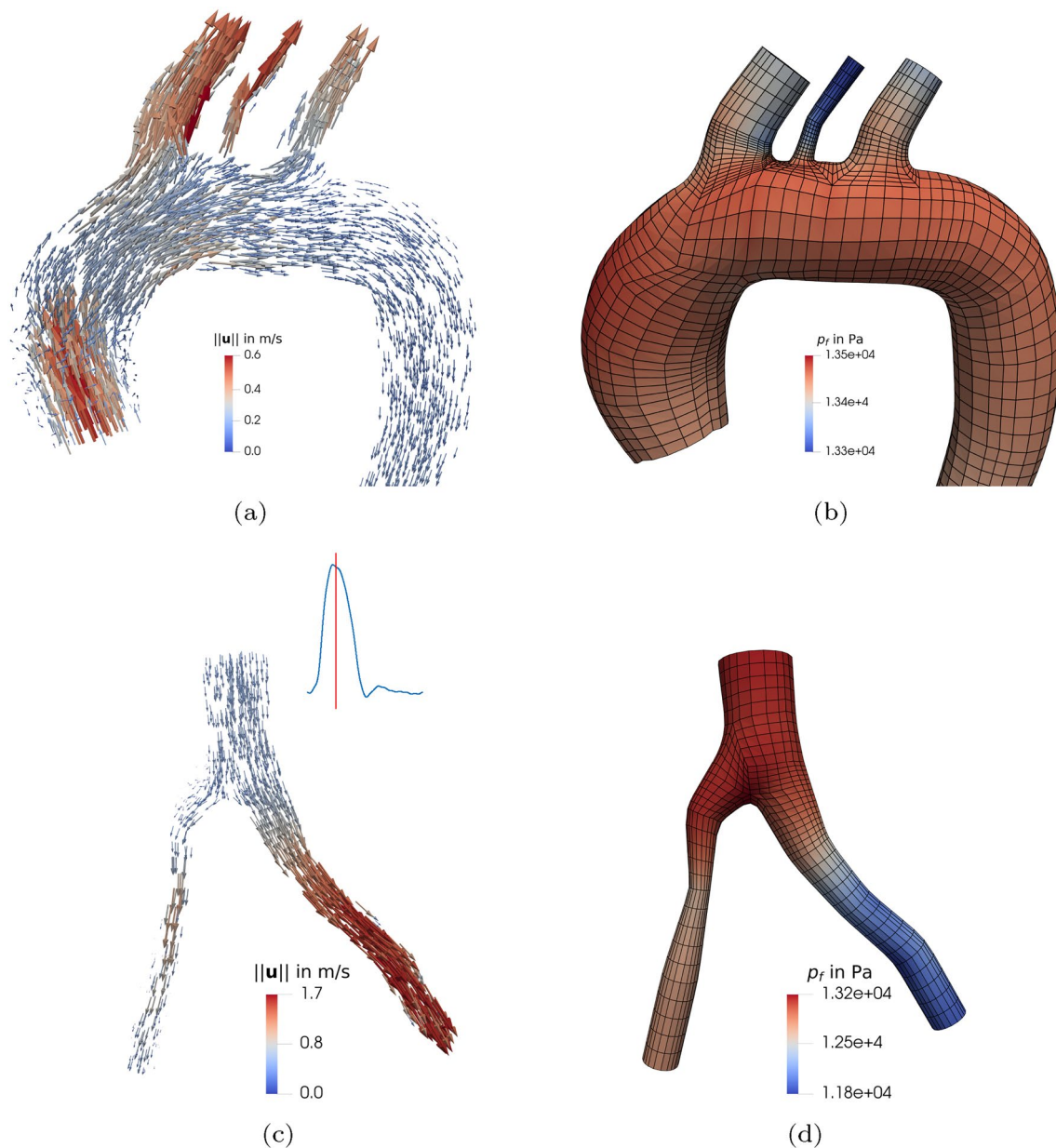


Fig. 25 Physiological blood flow through the aorta using 40,000 trilinear elements: velocity vectors (left) and pressure fields including element edges (right) in aortic arch region (top) and at the iliac

bifurcation (bottom) at peak systole as indicated by vertical line in the temporal inflow scale

that the discretisation order and selecting the number of elements is left entirely to the user, whom we provide with maximum freedom since coarse meshes involving merely 320 elements still resolving the current complex topology (but not the physical solution) can be constructed. Thus, one may use rather fine, but tailored pure hexahedral meshes for lower-order methods, or employ higher-order discretisations with much lower element counts, whereas the desired geometry is resolved accurately in both scenarios.

5 Conclusions and future work

Within this work, we presented a pipeline for block-structured generation of higher-order hexahedral meshes of tubular structures, starting from patient-specific CT scans, for use in fluid simulations. The focus of the paper lies on the generation of CFD-compliant higher-order meshes, resembling the domain with a spatial resolution as coarse or fine

as required by the user. Future and ongoing work is centered around devising new block structure generation schemes and improving mesh quality, as well as resolving solid domains. These adaptations would pave the way for the application of the constructed meshes with an FSI framework, such as [3, 68].

The geometry is described by convolution surfaces, whose capabilities far encompass tubular structures. As the examples shown in the paper cover the healthy aorta, a next step could then involve pathological cases such as aneurysms, coarctations or aortic dissection. As mentioned, with modifications of the block structure, the entire pipeline could be straightforwardly expanded, e.g., to networks found in the respiratory system or in capillaries. Hence, a natural extension of this work consists of meshing structures that have no tubular-prior assumptions.

Acknowledgements The authors gratefully acknowledge Graz University of Technology, Austria for the financial support of the LEAD-project: Mechanics, Modelling and Simulation of Aortic Dissection. Moreover, the article was supported by the TU Graz Open Access Publishing Fund.

Funding Open access funding provided by Graz University of Technology.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Bäumler K, Vedula V, Sailer AM, Seo J, Chiu P, Mistelbauer G, Chan FP, Fischbein MP, Marsden AL, Fleischmann D (2020) Fluid–structure interaction simulations of patient-specific aortic dissection. *Biomech Model Mechanobiol* 19(5):1607–1628
- Schussnig R, Rolf-Pissarczyk M, Holzapfel G, Fries TP (2021) Fluid–structure interaction simulations of aortic dissection. *Proc Appl Math Mech* 20:e202000125
- Schussnig R, Pacheco DRQ, Kaltenbacher M, Fries T-P (2022) Semi-implicit fluid–structure interaction in biomedical applications. *Comput Methods Appl Mech Eng* 400:115489
- Owen S (2000) A survey of unstructured mesh generation technology. In: *Proceedings of international meshing roundtable conference 3*
- Si H (2015) TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Trans Math Softw* 41(2):11
- Tu J, Yeoh GH, Liu C (2012) *Computational fluid dynamics: a practical approach*. Elsevier, Amsterdam
- Herbert T (1988) Secondary instability of boundary layers. *Annu Rev Fluid Mech* 20(1):487–526
- Carey G, Dinh H (1985) Grading functions and mesh redistribution. *SIAM J Numer Anal* 22(5):1028–1040
- Ray N, Sokolov D, Reberol M, Ledoux F, Lévy B (2018) Hex-dominant meshing: mind the gap! *Comput Aided Des* 102:94–103. <https://doi.org/10.1016/j.cad.2018.04.012>
- Zienkiewicz OC, Taylor RL (2000) *The finite element method: the basis, vol 1*. Butterworth-Heinemann, Oxford
- Usai F, Livesu M, Puppo E, Tarini M, Scateni R (2016) Extraction of the quad layout of a triangle mesh guided by its curve skeleton. *ACM Trans Graph* 35(1):6
- Livesu M, Muntoni A, Puppo E, Scateni R (2016) Skeleton-driven adaptive hexahedral meshing of tubular shapes. *Comput Graph Forum* 35(7):237–246
- Lin H, Jin S, Liao H, Jian Q (2015) Quality guaranteed all-hex mesh generation by a constrained volume iterative fitting algorithm. *Comput Aided Des* 67–68:107–117
- Livesu M, Attene M, Patané G, Spagnuolo M (2017) Explicit cylindrical maps for general tubular shapes. *Comput Aided Des* 90:27–36
- Pandey K, Bærentzen JA, Singh K (2022) Face extrusion quad meshes. In: *ACM SIGGRAPH 2022 conference proceedings*. SIGGRAPH'22 10:1–9
- De Santis G, De Beule M, Van Canneyt K, Segers P, Verdonck P, Verheghe B (2011) Full-hexahedral structured meshing for image-based computational vascular modeling. *Med Eng Phys* 33:1318–1325. <https://doi.org/10.1016/j.medengphy.2011.06.007>
- Ghaffari M, Tangen K, Alaraj A, Du X, Charbel F, Linninger A (2017) Large-scale subject-specific cerebral arterial tree modeling using automated parametric mesh generation for blood flow simulation. *Comput Biol Med*. <https://doi.org/10.1016/j.combiomed.2017.10.028>
- Pietroni N, Campen M, Sheffer A, Cherchi G, Bommès D, Gao X, Scateni R, Ledoux F, Remacle J, Livesu M (2022) Hex-mesh generation and processing: a survey. *ACM Trans Graph* 42(2):16
- Fuentes Suárez A, Hubert E (2018) Scaffolding skeletons using spherical Voronoi diagrams: feasibility, regularity and symmetry. *Comput Aided Des* 102:83–93. *Proceeding of SPM 2018 Symposium*
- Fuentes Suárez AJ, Hubert E, Zanni C (2019) Anisotropic convolution surfaces. *Comput Graph* 82:106–116. <https://doi.org/10.1016/j.cag.2019.05.018>
- Panotopoulou A, Ross E, Welker K, Hubert E, Morin G (2018) Scaffolding a skeleton, *Research in Shape Analysis*. Association for Women in Mathematics Series, vol 12, pp 17–35
- Grosland N, Shivanna K, Magnotta V, Kallemeyn N, DeVries N, Tadepalli S, Lisle C (2009) IA-FEMesh: an open-source, interactive, multiblock approach to anatomic finite element model development. *Comput Methods Programs Biomed* 94(1):96–107
- Dubey A et al (2014) A survey of high level frameworks in block-structured adaptive mesh refinement packages. *J Parallel Distrib Comput* 74(12):3217–3227
- Ali Z, Dhanasekaran PC, Tucker PG, Watson R, Shahpar S (2017) Optimal multi-block mesh generation for CFD. *Int J Comput Fluid Dyn*. <https://doi.org/10.1080/10618562.2017.1339351>

25. Wesseling P, Oosterlee CW (2001) Geometric multigrid with applications to computational fluid dynamics. *J Comput Appl Math* 128(1):311–334
26. Babuska I, Suri M (1994) The p and h-p versions of the finite element method, basic principles and properties. *SIAM Rev* 36(4):578–632
27. Taylor C, Hood P (1973) A numerical solution of the Navier–Stokes equations using the finite element technique. *Comput Fluids* 1:73–100. [https://doi.org/10.1016/0045-7930\(73\)90027-3](https://doi.org/10.1016/0045-7930(73)90027-3)
28. Gresho PM, Sani RL (2000) Incompressible flow and the finite element method, vol 1+2. Wiley, Chichester
29. Arndt D, Fehn N, Kanschat G, Kronbichler KKM, Munch P, Wall W, Witte J (2020) ExaDG: high-order discontinuous Galerkin for the exascale. In: *Software for exascale computing—SPPEXA 2016–2019*. Springer, Cham, pp 189–224
30. Bazilevs Y, Gohean JR, Hughes TJR, Moser RD, Zhang Y (2009) Patient-specific isogeometric fluid–structure interaction analysis of thoracic aortic blood flow due to implantation of the Jarvik 2000 left ventricular assist device. *Comput Methods Appl Mech Eng* 198(45):3534–3550
31. Bazilevs Y, Hsu M, Zhang Y, Wang W, Kvamsdal T, Hentschel S, Isaksen J (2010) Computational vascular fluid–structure interaction: methodology and application to cerebral aneurysms. *Bio-mech Model Mechanobiol* 9:481–498
32. Zhang Y, Bazilevs Y, Goswami S, Bajaj C, Hughes TJR (2007) Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. *Comput Methods Appl Mech Eng* 196:2943–2959. https://doi.org/10.1007/978-3-540-34958-7_5
33. Wei X, Zhang Y, Liu L, Hughes T (2017) Truncated t-splines: fundamentals and methods. *Comput Methods Appl Mech Eng* 316:349–372
34. Yu Y, Zhang Y, Takizawa K, Tezduyar T, Sasaki T (2020) 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
35. Urlick B, Sanders T, Hossain S, Zhang Y, Hughes T (2017) Review of patient-specific vascular modeling: template-based isogeometric framework and the case for CAD. *Arch Comput Methods Eng* 26:1–24
36. Zhang Y (2016) Geometric modeling and mesh generation from scanned images, Chapman & Hall/CRC Mathematical and Computational Imaging Sciences Series. CRC Press, Taylor & Francis Group, pp 1–340
37. Zhang Y (2013) Challenges and advances in image-based geometric modeling and mesh generation, Springer Publisher, pp 1–10
38. Turner M (2017) High-order mesh generation for CFD solvers. Dissertation. Imperial College London. <https://doi.org/10.25560/57956>
39. Bloomenthal J, Bajaj C, Blinn J, Cani-Gascuel MP, Rockwood A, Wyvill B, Wyvill G (1997) Introduction to implicit surfaces. Morgan Kaufmann, San Francisco
40. Pepe A, Li J, Rolf-Pissarczyk M, Gsaxner C, Chen X, Holzapfel G, Egger J (2020) Detection, segmentation, simulation and visualization of aortic dissections: a review. *Med Image Anal* 65:101773
41. Moccia S, De Momi E, El Hadji S, Mattos L (2018) Blood vessel segmentation algorithms—review of methods, datasets and evaluation metrics. *Comput Methods Programs Biomed* 158:71–91
42. Li J, Pimentel P, Szengel A, Ehlke M, Lamecker H, Zachow S, Estacio L, Doenitz C, Ramm H, Shi H et al (2021) Autoimplant 2020-first MICCAI challenge on automatic cranial implant design. *IEEE Trans Med Imaging* 40(9):2329–2342
43. Hong Q, Li Q, Wang B, Liu K, Qi Q (2019) High precision implicitly modeling for patient-specific coronary arteries. *IEEE Access* 7:72020–72029
44. Shen T, Gao J, Yin K, Liu M-Y, Fidler S (2021) Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Adv NeurIPS* 34:6087–6101
45. Lamata P, Niederer S, Nordsletten D, Barber DC, Roy I, Hose DR, Smith N (2011) An accurate, fast and robust method to generate patient-specific cubic Hermite meshes. *Med Image Anal* 15(6):801–813
46. Mistelbauer G, Rössl C, Bäumler K, Preim B, Fleischmann D (2021) Implicit modeling of patient-specific aortic dissections with elliptic Fourier descriptors. In: *Computer graphics forum*, vol 40. Wiley Online Library, pp 423–434
47. Vukicevic A et al (2018) Three-dimensional reconstruction and NURBS-based structured meshing of coronary arteries from the conventional X-ray angiography projection images. *Sci Rep* 8:1711. <https://doi.org/10.1038/s41598-018-19440-9>
48. Oeltze S, Preim B (2005) Visualization of vasculature with convolution surfaces: method, validation and evaluation. *IEEE Trans Med Imaging* 24(4):540–548
49. Sethian JA (1996) A fast marching level set method for monotonically advancing fronts. *Proc Natl Acad Sci* 93:1591–1595. <https://doi.org/10.1073/pnas.93.4.1591>
50. Osher S, Fedkiw RP (2001) Level set methods: an overview and some recent results. *J Comput Phys* 169:463–502. <https://doi.org/10.1006/jcph.2000.6636>
51. Osher S, Fedkiw RP (2003) Level set methods and dynamic implicit surfaces. Springer, Berlin. <https://doi.org/10.1007/b98879>
52. Lin C, Liu L, Li C, Kobbelt L, Wang B, Xin S, Wang W (2020) Seg-mat: 3d shape segmentation using medial axis transform. *IEEE Trans Vis Comput Graph* 28:2430–2444
53. Tagliasacchi A, Delame T, Spagnuolo M, Amenta N, Telea A (2016) 3d skeletons: a state-of-the-art report. *Comput Graph Forum* 35:573–597
54. Au O, Tai C, Chu H, Cohen-Or D, Lee T (2008) Skeleton extraction by mesh contraction. *ACM Trans Graph* 27(3):1–10
55. Rebain D, Angles B, Valentin J, Vining N, Peethambaran J, Izadi S, Tagliasacchi A (2019) LSMAT least squares medial axis transform. *Comput Graph Forum* 38:5–18
56. Petrelli L, Pepe A, Disanto A, Gsaxner C, Li J, Jin Y, Buongiorno D, Brunetti A, Bevilacqua V, Egger J (2022) Geometric modeling of aortic dissections through convolution surfaces. In: *Medical imaging 2022: imaging informatics for healthcare, research, and applications*, vol 12037. SPIE, pp 198–206
57. Kanamori Y, Szego Z, Nishita T (2008) GPU-based fast ray casting for a large number of metaballs. *Comput Graph Forum* 27(2):351–360
58. Clenshaw CW, Curtis AR (1960) A method for numerical integration on an automatic computer. *Numer Math* 2:197–205
59. Ali Z, Tyacke J, Tucker PG, Shahpar S (2016) Block topology generation for structured multi-block meshing with hierarchical geometry handling. *Procedia Eng* 163:212–224. <https://doi.org/10.1016/j.proeng.2016.11.050>
60. Armstrong CG, Fogg HJ, Tierney CM, Robinson TT (2015) Common themes in multi-block structured quad/hex mesh generation. *Procedia Eng* 124:70–82. <https://doi.org/10.1016/j.proeng.2015.10.123>
61. Šolín P, Segeth K, Doležel I (2003) Higher-order finite element methods. CRC Press, Boca Raton. <https://doi.org/10.1201/9780203488041-7>
62. Gordon WJ, Hall CA (1973) Transfinite element methods: blending function interpolation over arbitrary curved element domains. *Numer Math* 21:109–129. <https://doi.org/10.1007/BF01436298>
63. Coons SA (1967) Surfaces for computer-aided design of space forms. MAC-TR-41, MIT, Cambridge
64. Hughes TJR (1987) The finite element method: linear static and dynamic finite element analysis. Prentice-Hall, Englewood Cliffs

65. Radl L, Jin Y, Pepe A, Li J, Gsaxner C, Zhao F, Egger J (2022) AVT: multicenter aortic vessel tree CTA dataset collection with ground truth segmentation masks. *Data Br* 40:107801
66. Schussnig R, Pacheco DRQ, Fries TP (2021) Robust stabilised finite element solvers for generalised Newtonian fluid flows. *J Comput Phys* 442:110436
67. Pacheco DRQ, Schussnig R, Fries TP (2021) An efficient split-step framework for non-Newtonian incompressible flow problems with consistent pressure boundary conditions. *Comput Methods Appl Mech Eng* 382:113888
68. Schussnig R, Pacheco DRQ, Fries TP (2022) Efficient split-step schemes for fluid-structure interaction involving incompressible generalised Newtonian flows. *Comput Struct* 260:106718

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.