Computer-Aided Design 173 (2024) 103730

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Research Paper

A Three-Grid High-Order Immersed Finite Element Method for the Analysis of CAD Models

Eky Febrianto [a,*], Jakub Šístek [b], Pavel Kůs [b], Matija Kecman [c], Fehmi Cirak [d]

[a] Glasgow Computational Engineering Centre, University of Glasgow, University Avenue, Glasgow G12 8QQ, UK
[b] Institute of Mathematics of the Czech Academy of Sciences, Žitná 25, 115 67 Prague, Czech Republic
[c] Epic Games, Inc., 620 Crossroads Blvd, Cary NC 27518, USA
[d] Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK

ARTICLE INFO

Keywords:
Immersed finite elements
High-order
Implicit geometry
Parallel computing
Domain decomposition

ABSTRACT

The automated finite element analysis of complex CAD models using boundary-fitted meshes is rife with difficulties. Immersed finite element methods are intrinsically more robust but usually less accurate. In this work, we introduce an efficient, robust, high-order immersed finite element method for complex CAD models. Our approach relies on three adaptive structured grids: a geometry grid for representing the implicit geometry, a finite element grid for discretising physical fields and a quadrature grid for evaluating the finite element integrals. The geometry grid is a sparse VDB (Volumetric Dynamic B+ tree) grid that is highly refined close to physical domain boundaries. The finite element grid consists of a forest of octree grids distributed over several processors, and the quadrature grid in each finite element cell is an octree grid constructed in a bottom-up fashion. The resolution of the quadrature grid ensures that finite element integrals are evaluated with sufficient accuracy and that any sub-grid geometric features, like small holes or corners, are resolved up to a desired resolution. The conceptual simplicity and modularity of our approach make it possible to reuse open-source libraries, i.e. openVDB and p4est for implementing the geometry and finite element grids, respectively, and BDDCML for iteratively solving the discrete systems of equations in parallel using domain decomposition. We demonstrate the efficiency and robustness of the proposed approach by solving the Poisson equation on domains described by complex CAD models and discretised with tens of millions of degrees of freedom. The solution field is discretised using linear and quadratic Lagrange basis functions.

1. Introduction

1.1. Motivation and overview

Immersed finite elements, also called embedded, unfitted, extended, cut-cell or shifted-boundary finite element methods, have unique advantages when applied to complex three-dimensional geometries [1–12]. By large, they can sidestep the challenges in automating the boundary-fitted finite element mesh generation from CAD models. The prevailing parametric CAD models based on BRep data structures and NURBS surfaces are generally rife with gaps, overlaps and self-intersections, which need to be repaired before mesh generation [13–15]. In addition, CAD models contain small geometric details, like fillets, holes, etc., consideration of which would lead to unnecessarily large meshes so that they are typically manually removed, or defeatured [13]. Most of these challenges can be circumvented by present low-order immersed finite elements, but not so when high-order accuracy is desired.

In immersed finite elements, the CAD model is discretised by embedding the domain in a structured background mesh, or grid, usually consisting of rectangular prisms, or cells. The boundary conditions are enforced approximately and the weak form of the governing equations is integrated only in the part of the cut-cells inside the domain. A polygonal approximation of the boundary surface will always lead to a low-order method irrespective of the polynomial order of the basis functions. For instance, according to standard error estimates for second-order elliptic boundary value problems to obtain an approximation of order $p+1$ using basis functions of degree p, the boundary surface must be approximated with polynomials of degree p [16,17]. That is, the cut-element faces adjacent to the boundary must be curved and be at least of degree p. However, for complex geometries, generating geometrically and topologically valid curved elements is not trivial because of the mentioned problems with parametric CAD models.

Alternatively, as pursued in this paper, the accuracy of the finite element solution can be increased by approximating the boundary

* Corresponding author.
E-mail address: eky.febrianto@glasgow.ac.uk (E. Febrianto).

https://doi.org/10.1016/j.cad.2024.103730
Received 14 January 2024; Received in revised form 17 April 2024; Accepted 15 May 2024
Available online 23 May 2024

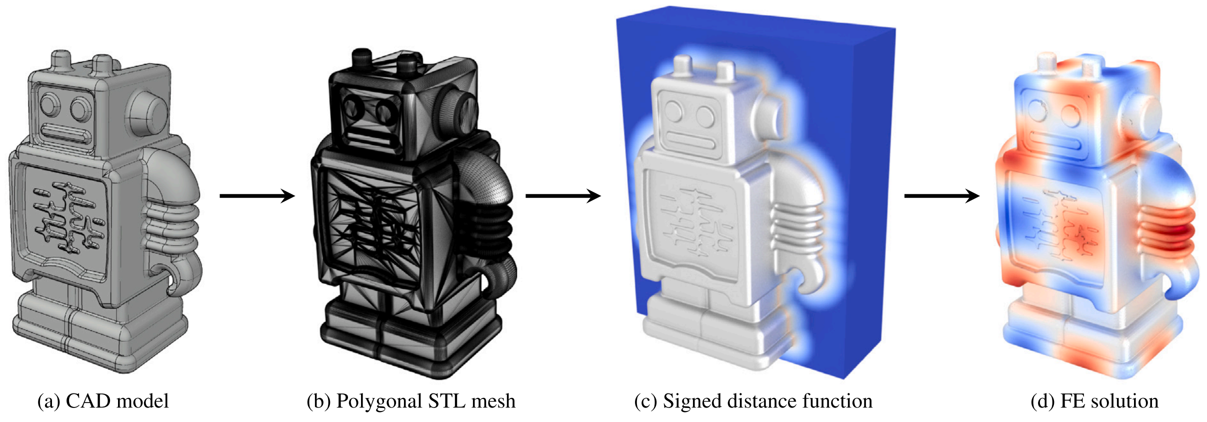


Fig. 1. Robust high-order accurate immersed finite element analysis of a CAD model. The BRep CAD model consisting of NURBS surface patches (a) is first exported from a CAD system as a sufficiently fine facetted STL mesh (b). The respective implicit signed distance function (c) is obtained by sampling the STL mesh. The implicit signed distance function is stored on a grid restricted to a tight narrow band around the boundary. The finite element analysis is performed on a much coarser non-boundary-fitted grid using high-order basis functions. The restriction of the finite element solution on the grid (d) represents the sought solution.

surface as a polygonal mesh with a characteristic edge length much smaller than the cell size of the finite element grid. To this end, we introduce, in addition to the immersed *finite element grid*, a *geometry grid* for piecewise-linear implicit geometry representation and a *quadrature grid* for evaluating the element integrals in the cut-cells. The three independent grids are all adaptive and have different resolutions.

The conceptual simplicity of the proposed approach makes it easy to embed it within a parallel domain decomposition context and analyse large complex CAD models efficiently. Efficient solvers and parallel domain decomposition are essential for CAD geometries from engineering because their discretisation easily leads to systems of equations with several tens of thousands of elements and the associated computing times are below a few minutes.

1.2. Related research

In an implicit geometry description, the domain is represented as a scalar-valued signed distance, or level set, function. The level set function is zero at the boundary, positive inside and negative outside of the domain. Well-known advantages of implicit geometry representations include ease of robust Boolean operations and shape interrogation, including the computation of ray-surface intersection, see e.g. [18–22]. Although there are algebraic implicitisation techniques to determine the level set function of a single or a few NURBS patches [23–25], such methods are presently not scalable to complex CAD models. Alternatively, the level set function can be obtained by sampling the CAD model and storing the determined distances on a structured geometry grid. However, sampling directly the CAD model requires distance computations or ray-surface intersections, leading to expensive and unstable nonlinear root-finding problems which would annihilate any advantages of an implicit geometry representation.

In contrast, the sampling of polygonal surface meshes can be performed exceedingly robustly. Polygonal mesh models are prevalent in manufacturing and virtually all CAD systems can approximate a CAD model with an intersection-free polygonal mesh, that is an STL mesh, with a prescribed accuracy. An accurate representation of the level set function with a resolution comparable to the element size of the polygonal STL mesh is only required in the immediate vicinity of the boundary. Although octree data structures have been used to this end, they generally tend to have a large memory footprint and slow access times. An adaptive grid with approximately constant access time and a memory footprint that scales quadratically with the number of elements in the polygonal surface mesh is provided by the VDB (Volumetric Dynamic B+ tree) data structure and its open source implementation OpenVDB [26,27]. In VDB the difference in the refinement level between two neighbouring cells is much higher than two, which is combined with other algorithmic features, making it ideal for processing and storing level set functions. OpenVDB also provides algorithms for efficient computation of the level set function by solving the Eikonal equation using the fast sweeping method [28,29].

The purpose of the finite element discretisation grid is to approximate the physical solution field and usually has a different resolution requirement from the geometry grid for representing the level set function. As per classical a-priori estimates, the ideal finite element cell size distribution depends on the smoothness properties of the solution field and the polynomial order of the basis functions used. A uniform cell size distribution is seldom ideal so that grid adaptivity is crucial. Furthermore, different from the geometry grid, abrupt changes in the finite element grid size distribution must be avoided because it is usually associated with artefacts in the approximate solution field. Hence, balanced octree grids are suitable as finite element discretisation grids.

On the non-boundary-fitted octree finite element grid the boundary conditions are enforced by modifying either the variational weak form or the basis functions. According to historical work by Kantorovich and Krylov [30] in Ritz-like methods homogeneous Dirichlet boundary conditions can be enforced by multiplying the global basis functions with a zero weight function at the boundary. Building on this idea, several approaches have been introduced for enforcing the boundary conditions by altering the basis functions [1,3,6,31,32]. The required weight function is typically obtained from the signed distance function. Alternatively, the boundary conditions can be enforced by modifying the variational weak form, mainly using the Nitsche method [33] and its variations [34–38]. The finite element integrals in the cut-cells traversed by the boundary are evaluated only inside the domain. Because finite element basis functions are locally supported, some basis functions close to cut-cells may have a negligible physically active support domain, leading to ill-conditioned system matrices. Several approaches have been proposed for cut-cell stabilisation, including basis function extension [1,39,40] and ghost-penalty method [41], see also the recent review [42].

The evaluation of the finite element integrals in the cut-cells requires special care in high-order accurate immersed methods. Its accuracy depends on the chosen quadrature scheme, number of quadrature points, and the domain boundary approximation in the cut-cells. The number of quadrature points affects the overall efficiency of the immersed finite element method and must be kept low. The cut-cells are usually identified by evaluating the signed distance function at the cell vertices. This approach is sufficient when the geometry and finite element grids have the same resolution but can miss some of the cut-cells when the signed distance function is given in algebraic form or via a finer geometry grid. In both cases supersampling the signed distance function on the finite element grid can improve the correct classification of cells as cut-cells [43]. After identification, the cut-cells are decomposed into easy-to-integrate primitive shapes using marching tetrahedra partitioning [39,44], octree partitioning [4] or a combination thereof [45]. For a discussion on the mentioned and other cut-cell integration approaches see [46]. The collection of the primitive shapes, that is the integration cells, yields the integration grid.

The quadrature approaches described so far are generally low order. In case of recursive octree partitioning high order accuracy can, in principle, be achieved by successively refining the integration cells traversed by the domain boundary. This leads however to a vast number of quadrature points and exceptionally inefficient approach. An alternative approach to improve the boundary approximation is to introduce additional nodes on the edges and faces and to push those to the boundary surface [45,47]. Unfortunately, such methods are very brittle because of the involved floating point operations, the presence of subgrid features, like small holes, and sharp corners and edges. They require a defeaturing step, like in boundary-fitted mesh generation, for engineering CAD models, annihilating one of the main advantages of immersed methods. It is worth bearing in mind, that the exceptional robustness of low order immersed methods can be attributed to the automatic defeaturing, or geometry filtering, by representing a CAD geometry on a coarse grid and the discarding of subgrid geometry details [3,6].

Parallel computation and grid adaptivity are essential for three-dimensional finite element analysis of large CAD models. The required computing memory and time for the analysis become quickly unwieldy, particularly when high-order basis functions are used. Adaptive refinement and coarsening of the finite element grid using a distributed octree data structure provides a mean to spread the grid and the computation over several processors and to choose a grid size distribution that best approximates the physical solution field [48]. Compared to equivalent parallel boundary-fitted finite element implementations, an octree-based immersed finite element implementation leads to highly scalable domain partitioning and allows for easy dynamic load balancing, especially when adhering to balanced octree structures. There are efficient distributed octree implementations, most notably p4est [48] for distributing the finite element grid over many processors. In such a distributed setting, the corresponding linear system of equations is typically solved using iterative Krylov subspace methods in combination with parallel preconditioners. The system matrix for the entire problem is never assembled. As parallel preconditioners, both algebraic multigrid [49] and domain decomposition [50,51] have been applied in immersed methods. In [51] a standard single-level additive preconditioner and in [50] a two-level balancing domain decomposition based on constraints [52] are considered. The possible ill-conditioning of the system matrices in immersed methods presents a challenge in applying preconditioners. However, this problem can be alleviated by using the mentioned cut-cell stabilisation techniques.

1.3. Proposed approach

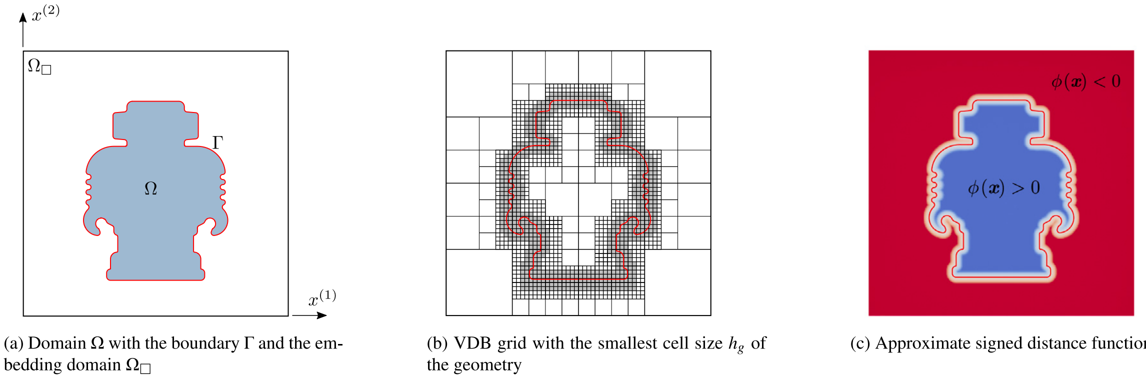
The robustness, accuracy and efficiency of immersed methods effectively depend on the description of the domain geometry, the treatment of the cut-cells, the resolution of the discretisation grid and the solution of the resulting linear systems of equations. In the case of large CAD models with geometric and topological faults and small geometric features, an inevitable trade-off is essential in satisfying the three competing objectives of robustness, accuracy and efficiency. To achieve this, we introduce three different adaptive grids for representing the geometry, finite element discretisation and integration of element integrals. The resolution of each of the grids can be chosen in dependence of the required accuracy and available computing budget. As an additional benefit, the use of three different grids leads to a modular software architecture and makes it possible to reuse available open-source components, specifically openVDB [26], p4est [48] and BDDCML [53]. The accuracy of the geometry representation is determined by the polygonal STL mesh with a user-prescribed precision exported from a CAD system. The respective signed distance function is computed at the nodes of the adaptive geometry grid with the minimum resolution length h_g. The signed distance value within the cells is obtained by linearly interpolating from the nodes. The length h_g is chosen in dependence of the smallest feature size appearing in the polygonal surface mesh. Any geometric details smaller than $\approx h_g$ are automatically discarded by switching from the polygonal mesh to the implicit signed distance function. We use the openVDB library for computing and storing the implicit signed distance function in a narrow band of width δ along the boundary surface. The narrow band size is chosen as a multiple of the minimum geometry resolution length h_g. Inside the narrow band, the geometry cells have the length h_g and outside they are much coarser.

The adaptive finite element grid is constructed from a coarse base grid enveloping the polygonal surface mesh. The coarse base grid is distributed over the available processors and is refined until a desired cell size distribution is obtained while maintaining a refinement level difference of one or less between two neighbouring cells. The smallest resolution length h_f is usually larger than the geometry grid resolution h_g because the memory and computing requirements are dominated by the solution of the discretised equations. This is especially true when the domain is three-dimensional and high-order basis functions are used. In the included examples, we use Lagrange basis functions of polynomial orders $p = 1, 2$, and all are three-dimensional. Furthermore, all cut-cells are refined up to the finest resolution h_f. We identify the cut-cells via supersampling the signed distance function with a resolution h_g, within each finite element cell. This is necessary to detect small geometric features and sharp edges/corners that are easily missed by evaluating the signed distance function at the nodes of the finite element grid. We use the open source p4est (forest-of-trees) to refine and distribute the octree grid over all processors. Furthermore, we solve the respective distributed linear system of equations using the adaptive-multilevel BDDC (balancing domain decomposition based on constraints library) BDDCML.

The quadrature grid is constructed only in the cut-cells and has a resolution $h_q \geq h_g$. It ensures that the finite element integrals are evaluated correctly in the presence of curved boundary surfaces and geometric features much smaller than the finite element cell size. We adopt a bottom-up octree strategy by starting from a coarse quadrature grid and successively merging cells that are not cut by the domain boundary. We implement fast cell traversal via the Morton code, or Z-curve. This bottom-up construction is crucial for robust and accurate recovery of quadrature cells cut by the boundary. The leaf cells of the quadrature grid are then tessellated using the marching tetrahedra algorithm.

1.4. Overview of the paper

This paper is divided into five sections. In Section 2, we begin by providing a brief review of the implicit description of geometry and its discrete sampling over the geometry grid. We then introduce the FE grid and discuss its construction and classification in relation to geometrical and physical features. In Section 3, we present the immersed finite element scheme which utilises the three grids. We specifically address the treatment of cut-cells, including the construction of the bottom-up quadrature grid and tetrahedralisation. In Section 4 we delve into the numerical strategies that enable the proposed immersed FE analysis to be computed on parallel processors. Finally, in Section 5, we demonstrate the optimal convergence of the developed approach and its strong and weak scalability. We demonstrate the robustness of our approach through several examples, showing its ability to analyse various complex geometries.



(a) Domain Ω with the boundary Γ and the embedding domain $\Omega_\square$ (b) VDB grid with the smallest cell size h_g of the geometry (c) Approximate signed distance function $\phi(\boldsymbol{x})$

Fig. 2. Representative geometry setup.

2. Geometry and finite element grids

2.1. Geometry grid

We assume that the domain $\Omega \in \mathbb{R}^3$ with the boundary surface Γ is given as a scalar valued implicit signed distance function $\phi(\boldsymbol{x}) : \Omega \to \mathbb{R}$ such that

$$\phi(\boldsymbol{x}) = \begin{cases} \operatorname{dist}(\boldsymbol{x}, \Gamma) & \text{if } \boldsymbol{x} \in \Omega \\ 0 & \text{if } \boldsymbol{x} \in \Gamma \\ -\operatorname{dist}(\boldsymbol{x}, \Gamma) & \text{otherwise}, \end{cases} \tag{1}$$

where $\operatorname{dist}(\boldsymbol{x}, \Gamma) = \min_{\boldsymbol{y} \in \Gamma} |\boldsymbol{x} - \boldsymbol{y}|$ denotes the shortest distance between the point $\boldsymbol{x}$ and the surface Γ. By definition, the signed distance function $\phi(\boldsymbol{x})$ is positive inside the domain, negative outside and the zeroth isosurface $\phi^{-1}(0)$ corresponds to the boundary Γ.

It is difficult to obtain the closed-form signed distance function $\phi(\boldsymbol{x})$ for complex CAD geometries especially when they consist of freeform splines, see for example Fig. 1(a). In that case, it is often expedient to consider a finely discretised polygonal surface mesh, i.e., STL mesh, approximating the exact CAD surface, see as an example Fig. 1(b) or its two-dimensional depiction in Fig. 2(a). Practically, all CAD systems are able to export STL mesh irrespective of the internal representation they use.

To compute the approximate signed distance function $\phi(\boldsymbol{x})$, the respective domain Ω is first embedded, or immersed, in a sufficiently large cuboid $\Omega_\square \supset \Omega$. We discretise the embedding domain $\Omega_\square$ with an adaptive grid consisting of cuboidal cells with minimum edge length h_g. The signed distance function $\phi(\boldsymbol{x})$ is determined by first computing the distance of the grid nodes to the surface Γ and then linearly interpolating within the cells. The tree hierarchy of the geometry grid is established using the highly efficient VDB data structure as implemented in the open-source openVDB library [26,54]. The resulting geometry grid is highly refined within a narrow band of thickness 2δ around the boundary, i.e., $\{\boldsymbol{x} \in \mathbb{R}^3 \mid \operatorname{dist}(\boldsymbol{x}, \Gamma) < \delta\}$, see for example Fig. 2(b), and has a very small memory footprint. In our computations, we choose $\delta = 3\ h_g$. The VDB data structure provides constant-time random access, insertion, and deletion, allowing fast scan conversion and evaluation of the signed distance function at any point in space.

2.2. Finite element grid

The finite element grid must ensure that the features of the physical solution field, for instance, singularities and internal layers, are sufficiently captured. The construction of the FE grid starts with a very coarse base grid, see Fig. 3(a). The base grid is usually obtained by uniformly subdividing the bounding box $\Omega_\square$. To obtain the FE grid, we increase the resolution of the base grid by repeated octree refinement of selected cells, i.e. by subdividing cells into eight cells. The maximum and minimum cell size are often prescribed as constraints of the refinement.

In boundary value problems it is usually necessary to refine the grid towards the domain boundaries because of the practical importance of the solution close to the boundaries. As an example, we illustrate in Fig. 3(b) the refinement of the base grid towards the boundary utilising the FE grid cell predicates that will be introduced in Section 2.3. When a cell is identified as a cut-cell, it is subdivided into eight sub-cells. The octree data structure representing the finite element grid is managed using the open-source library p4est [48]. As will be discussed in Section 4.1, p4est also manages the partitioning of the octree into subprocesses and distributes them across several processors. Different from the VDB data structure for the geometry, we constrain the refinement level between adjacent cells to have a 2:1 ratio, i.e., the refinement level between neighbouring cells may differ at most by one.

The resulting finite element grid comprises of the non-overlapping leaf cells $C = \{\omega_i\}$ that decompose the bounding box, i.e.,

$$\Omega_\square = \bigcup_i \omega_i \,. \tag{2}$$

Each cell ω_i represents a finite element, and the physical field is discretised using the basis functions associated with the cells.

2.3. Classification of finite element cells

In this section we describe the classification of FE grid cells pertinent to the adaptive refinement of the cells. It is assumed that the discretised signed distance function $\phi(\boldsymbol{x})$ can be evaluated at any point $\boldsymbol{x} \in \Omega_\square$. The finite element cells C are split into three disjoint sets

$$C^a = \{\omega_i \in C \mid \min_{\boldsymbol{x} \in \omega_i} \phi(\boldsymbol{x}) > 0\}\,, \tag{3a}$$

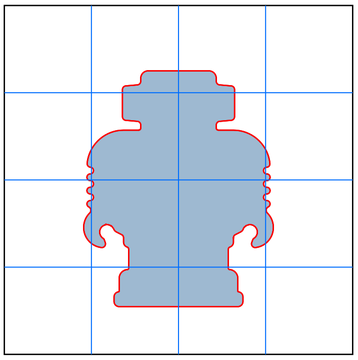
$$C^i = \{\omega_i \in C \mid \max_{\boldsymbol{x} \in \omega_i} \phi(\boldsymbol{x}) < 0\}\,, \tag{3b}$$

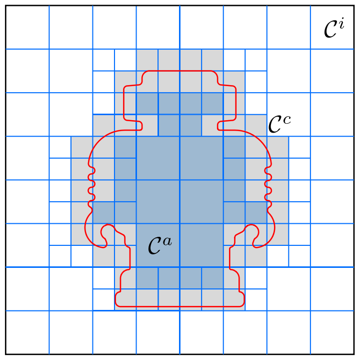
$$C^c = C \setminus (C^a \cup C^i)\,. \tag{3c}$$

The cells in the sets C^a, C^i and C^c are referred to as the active, inactive and cut cells, see Fig. 3(b).

Although it is tempting to classify the finite element cells by evaluating the signed distance function $\phi(\boldsymbol{x})$ only at their corners, it usually leads to a crude finite element approximation of the domain Ω. As depicted in Figs. 5(a) and 5(b), such an approach may miss physically important small geometric details. A more accurate finite element approximation can be obtained by supersampling the signed distance function $\phi(\boldsymbol{x})$ within the cells. The chosen sampling distance represents a low-pass filter for the geometry and implies a resolution length for geometric details. The sampling distance can be equal or larger than the edge length h_g of the geometry grid.

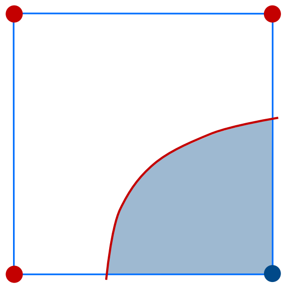
Industrial CAD geometries usually contain sharp features in the form of creases and corners that are often physically essential, see Fig. 5(c). They can be identified by considering the change of the normal to the

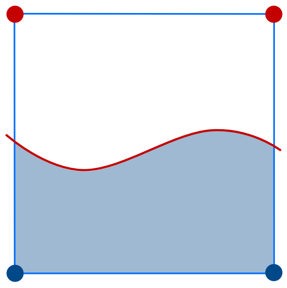

(a) Base finite element grid



(b) Finite element grid with smallest cell size h_f

Fig. 3. A finite element grid obtained by octree refinement towards the domain boundary starting from a base grid. The domain boundary is given by the zeroth level set $\phi^{-1}(0)$ of the signed distance function. The grid elements are categorised as active $\mathcal{C}^a$, cut $\mathcal{C}^c$ or inactive $\mathcal{C}^i$.


(a)


(b)

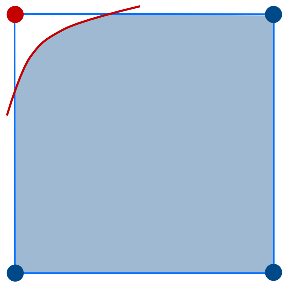
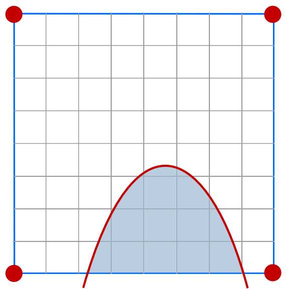
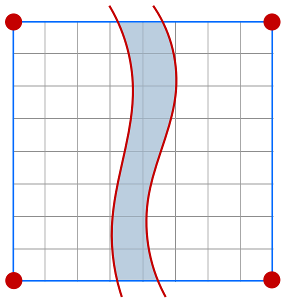
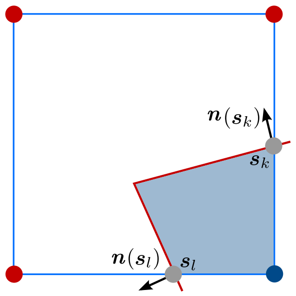

(c)

Fig. 4. Identification of the ordinary cut finite element cells $\omega_i \in \mathcal{C}^{c_o} \subset \mathcal{C}$. Blue dots correspond to $\phi(\boldsymbol{x}) > 0$ and red dots to $\phi(\boldsymbol{x}) < 0$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)


(a)


(b)



(c)

Fig. 5. Identification of the extraordinary cut finite element cells $\omega_i \in \mathcal{C}^{c_e} \subset \mathcal{C}$. Blue dots correspond to $\phi(\boldsymbol{x}) > 0$ and red dots to $\phi(\boldsymbol{x}) < 0$. Supersampling of selected cells to detect intersections with small geometric details (left and centre) and cells with sharp geometric features (right). In (a) and (b) $h_f = 8h_g$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

signed distance function [55]. The respective unit normal $\boldsymbol{n}(\boldsymbol{x})$ is given by

$$\boldsymbol{n}(\boldsymbol{x}) = \frac{\nabla\phi(\boldsymbol{x})}{\|\nabla\phi(\boldsymbol{x})\|}\,, \tag{4}$$

where ∇ denotes the gradient operator.

Consider a set of points in a cell ω_i denoted as $\mathcal{S}_i$, where we can evaluate ϕ and $\nabla\phi$. In our implementation the set $\mathcal{S}_i$ corresponds to grid points of a local uniform grid. If the spacing of this grid is the cell size itself, $\mathcal{S}_i$ contains just the vertices of ω_i. We can define an indicator of sharp features by

$$\min_{s_k\in\mathcal{S}_i,\, s_l\in\mathcal{S}_i} \boldsymbol{n}(s_k)\cdot\boldsymbol{n}(s_l) < \cos(\theta)\,, \tag{5}$$

where $\cos(\theta)$ is a user prescribed parameter chosen as $\cos(\theta) = 0.3$ in the presented computations.

In light of the mentioned observations, we employ the following algorithm to classify the finite element cells.

S1. Define $\mathcal{S}_i$ as the vertices of the finite element cell ω_i, and sample and store the signed distance values $\phi(\boldsymbol{s}_j)$ and the normal vectors $\boldsymbol{n}(\boldsymbol{s}_j)$ for $\boldsymbol{s}_j \in \mathcal{S}_i$.

S2. Identify all the cut-cells $\mathcal{C}^c$ such that

$$\mathcal{C}^c = \{\omega_i \mid \min_{s_j\in\mathcal{S}_i} \phi(\boldsymbol{s}_j)\cdot \max_{s_k\in\mathcal{S}_i}\phi(\boldsymbol{s}_k) < 0\}\,, \tag{6}$$

identify the subset of cut-cells $\mathcal{C}^{c_e} \subset \mathcal{C}^c$ that contain a sharp feature using criterion (5), and define the subset of *ordinary* cut cells $\mathcal{C}^{c_o} := \mathcal{C}^c \backslash \mathcal{C}^{c_e}$, as depicted in Fig. 4. Categorise the remaining cells into active $\mathcal{C}^a$ and inactive $\mathcal{C}^i$ considering the signed distance values $\phi(\boldsymbol{s}_j)$.

S3. Collect the set of cells $\mathcal{P} = \bigcup_i \mathcal{P}_i$, where $\mathcal{P}_i$ indicates the face neighbour of a cut-cell $\omega_i \in \mathcal{C}^c$.
S4. For cells in $\mathcal{P} \setminus \mathcal{C}^c$, redefine $\mathcal{S}_i$ by supersampling, see for example Fig. 5, and repeat S1 and S2. Cells identified as cut after supersampling are added to the set of extraordinary cells $\mathcal{C}^{c_e}$.

The final set of the cut cells is given as $\mathcal{C}^c = \mathcal{C}^{c_o} \cup \mathcal{C}^{c_e}$.

3. Immersed finite element method

3.1. Governing equations

As a representative partial differential equation, we consider on a domain $\Omega \in \mathbb{R}^3$ with the boundary $\Gamma = \Gamma^D \cup \Gamma^N$ the Poisson equation

$$\begin{aligned} -\nabla \cdot \nabla u &= f \quad \text{in } \Omega\,, \\ u &= \overline{u} \quad \text{on } \Gamma^D\,, \\ \boldsymbol{n} \cdot \nabla u &= \overline{g} \quad \text{on } \Gamma^N\,, \end{aligned} \tag{7}$$

where $u, f : \Omega \to \mathbb{R}$ are the unknown solution and the prescribed forcing, $\overline{u} : \Gamma^D \to \mathbb{R}$ is the prescribed Dirichlet data, $\overline{g} : \Gamma^N \to \mathbb{R}$ is the prescribed Neumann data, and $\boldsymbol{n} : \Gamma \to \mathbb{R}^3$ is the unit boundary normal. The weak formulation of the Poisson equation can be stated according to Nitsche [33] as follows: find $u \in H^1(\Omega)$ such that

$$\begin{aligned} \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega + \gamma \int_{\Gamma_D} (u - \overline{u}) v \, \mathrm{d}\Gamma \\ = \int_\Omega f v \, \mathrm{d}\Omega + \int_{\Gamma_N} \overline{g} v \, \mathrm{d}\Gamma \\ + \int_{\Gamma_D} \big((u - \overline{u}) \boldsymbol{n} \cdot \nabla v + (\boldsymbol{n} \cdot \nabla u) v \big) \, \mathrm{d}\Gamma \end{aligned} \tag{8}$$

for all $v \in H^1(\Omega)$. The space $H^1(\Omega)$ is the standard Sobolev space such that the test functions v do not have to be zero on Γ_D and the Dirichlet boundary conditions are satisfied only weakly by the solution u. The stability parameter $\gamma > 0$ can be chosen, for instance, according to [34,40] as γ_0/h_f, where h_f is the local finite element size. This choice of γ requires that the adverse effects of small cut cells is eliminated by other means, e.g., by basis function extrapolation as will be introduced in Section 3.5.

We discretise the solution field u and the test function v with Lagrange basis functions $N_i(\boldsymbol{x})$ that are defined on the non-boundary fitted finite element grid defined on the domain $\Omega_\square \supset \Omega$. The approximation of the solution u and the test function v over the finite element grid is given by

$$u^h = \sum_i N_i(\boldsymbol{x}) u_i\,, \quad v^h = \sum_i N_i(\boldsymbol{x}) v_i\,. \tag{9}$$

After introducing both into the weak form (8) all integrals are evaluated by iterating over the cells in the finite element grid. Only the active cells in $\mathcal{C}^a$ and the cut cells in $\mathcal{C}^c$ have a non-zero contribution and are considered in finite element analysis. The integrals over active cells can be evaluated using standard tensor-product Gauss quadrature. The evaluation of the integrals over cut cells requires some care. As mentioned, according to standard a-priori error estimates, optimal convergence of the finite element solution is only guaranteed when the boundary geometry is approximated with polynomials of the same degree as the basis functions $N_i(\boldsymbol{x})$.

Unfortunately, trying to reconstruct the boundary geometry from the zeroth isocontour of the discretised signed distance function, i.e. $\phi^{-1}(0)$, with higher than linear polynomials is as challenging as creating a conforming 3D finite element mesh. Therefore we use in the proposed approach only a piecewise linear reconstruction of the boundary on the cut cells in $\mathcal{C}^c$, as will be detailed in Section 3.3. To increase the integration accuracy, we introduce a fine quadrature grid obtained by octree refinement of the finite element cut cells, which will be detailed in Section 3.2. The discretisation of the weak formulation (8) yields after integration using the quadrature grid the discrete system of equations

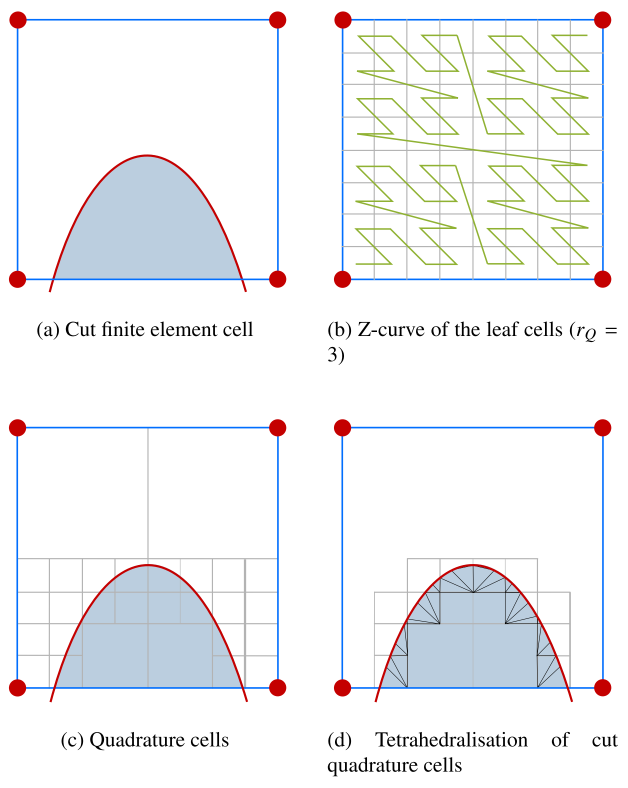

(a) Cut finite element cell
(b) Z-curve of the leaf cells (r_Q = 3)
(c) Quadrature cells
(d) Tetrahedralisation of cut quadrature cells

Fig. 6. Quadrature grid and bottom-up cut cell refinement of a cut finite element cell in $\mathcal{C}^c$ for quadrature and the tetrahedralisation of the cut quadrature cells. The red dots correspond to $\phi(\boldsymbol{x}) < 0$.

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}\,, \tag{10}$$

where $\boldsymbol{A}$ is the symmetric positive definite system matrix, $\boldsymbol{u}$ is the solution vector and $\boldsymbol{f}$ is the forcing vector.

3.2. Quadrature grid and bottom-up cut cell refinement

The quadrature grid in the cut finite element cells in $\mathcal{C}^c$ is obtained by r_Q steps of octree refinement. This implies that $r_Q = h_f/h_q$, where h_q is the smallest quadrature cell size. In our computations, we usually choose $0 \le r_Q \le 4$. The refinement process yields for each cut cell 8^{r_Q} leaf quadrature cells over which the finite element integrals have to be evaluated. As exemplified in Fig. 6(a), not all of the leaf quadrature cells are intersected by the domain boundary represented by $\phi^{-1}(0)$. Considering that the evaluation of the finite element integrals is computationally expensive (more so for high-order basis functions), it is desirable to have as few integration cells as possible. Hence, it is expedient to merge all the leaf cells not intersected by $\phi^{-1}(0)$ into larger cuboidal cells. This does not harm the finite element convergence rate because the resulting larger cells are integrated with the same tensor product quadrature rule like the active elements in $\mathcal{C}^a$.

We use a bottom-up octree construction to merge the leaf cells of the quadrature octree grid into larger integration cells. To this end, first, all the leaf quadrature cells are classified into active, inactive and cut by evaluating the signed distance value $\phi_g(\boldsymbol{x})$ at their corners. Subsequently, the bottom-up octree is constructed using the Morton code, or Z-curve [56], of the leaf quadrature cells, see Fig. 6(b). As the Z-curve is locality preserving, we can efficiently traverse it and apply a simple coarsening to build up the octree from the bottom up. The so obtained few larger cells can be integrated much more efficiently,

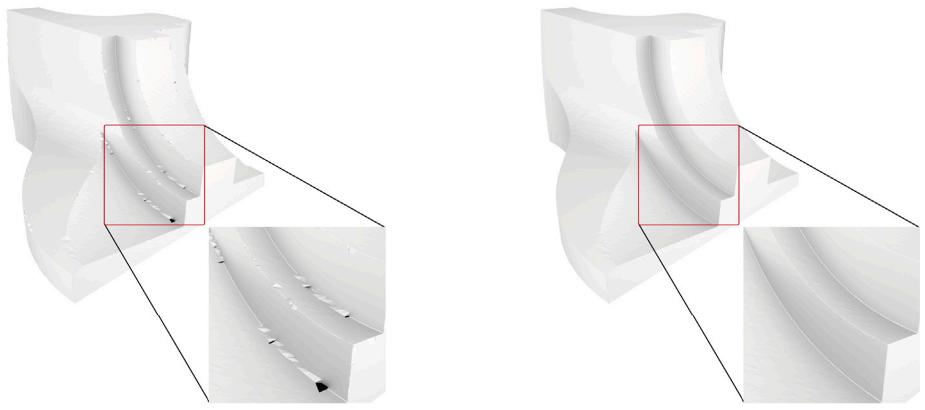

(a) Octree refinement level $r_Q = 0$
(b) Octree refinement level $r_Q = 3$

Fig. 7. Fan disk geometry reconstructed from the signed distance function $\phi(\boldsymbol{x})$ using bottom-up cut cell refinement and subsequent tetrahedralisation of cut quadrature cells.

see Fig. 6(c). For future reference, we denote the subcells of the finite element cut cell $\omega_i \in C^c$ with $\omega^Q_{i,k}$ such that

$$\omega_i = \bigcup_k \omega^Q_{i,k} \,. \tag{11}$$

In the following, the cells $\omega^Q_{i,k}$ are referred to as quadrature cells. Active quadrature cells are integrated using the tensor-product Gauss quadrature in a straightforward way. On the other hand, the cut quadrature cells first undergo the tetrahedralisation algorithm followed by integration introduced in Sections 3.3 and 3.4.

For nontrivial industrial CAD geometries, the bottom-up octree refinement of the cut cells leads to a significant improvement in the representation of the domain boundaries. The representation of the boundary of a fan disk with and without cut cell refinement are illustrated In Fig. 7. In case of no refinement ($r_Q = 0$), the surface has topological inconsistencies, and the sharp edges are poorly represented. In contrast, the boundary is faithfully represented with refinement ($r_Q = 3$).

3.3. Cut quadrature cell tetrahedralisation

We consider the decomposition of the quadrature cells $\omega^Q_{i,k}$ that are cut by the domain boundary into tetrahedra. This is necessary to increase the accuracy of the evaluation of the finite element integrals. Whether $\omega^Q_{i,k}$ is cut is determined by evaluating the signed distance $\phi(\boldsymbol{x})$ at its corners. The part of a cut quadrature cell $\omega^Q_{i,k}$ lying within the domain with $\phi(\boldsymbol{x}) \geq 0$ is partitioned into several simplices such that

$$\omega^Q_{i,k}\Big|_{\phi(\boldsymbol{x})\geq 0} \approx \bigcup_l \tau_l \,, \tag{12}$$

see Fig. 6(d). The finite element integrals are evaluated over the simplices $\{\tau_l\}$. Each cut quadrature cell $\omega^Q_{i,k}$ has its own simplices. We did not make this dependence explicit in order not to clutter further the notation.

To obtain the simplices $\{\tau_l\}$, first, each cut cuboidal quadrature cell $\omega^Q_{i,k}$ is split into six simplices and later the marching tetrahedra algorithm is used to decompose the part of the cell inside the domain [57]. The specific splitting pattern chosen in the first step is optimised so that the marching tetrahedra algorithm leads to as few simplices as possible. Specifically, as illustrated in Fig. 8, the cell is split into six simplices using one of the three depicted splitting patterns [58]. The domain boundary may cut all or some of the six resulting simplices. The tetrahedralisation of the parts of the cut simplices inside the domain is obtained with marching tetrahedra. Depending on the sign of $\phi(\boldsymbol{x})$ at the corners of the cut simplices, there are three different possible splitting patterns in marching tetrahedra [39]. In the tetrahedralisation of the cut simplices the intersection of their edges with the signed distance function $\phi(\boldsymbol{x}) = 0$ is required. To this end, we use a simple bisection algorithm. Note that the tetrahedralisation procedure also provides an approximation of the domain boundary using triangles within the FE cells, which are subsequently used for surface integrals.

3.4. Integration of cut quadrature cells

We introduce now the evaluation of the finite element integrals over the cut quadrature cells $\omega^Q_{i,k} \subset \omega_i \in C^c$ in (11) using the set of simplices $\{\tau_l \subset \omega^Q_{i,k}\}$ in (12). Fig. 9 shows a typical setup with a cut quadrature cell and the associated integration simplices. The quadrature rules are given for a reference simplex τ which is mapped to τ_l using the affine mapping $\boldsymbol{\varphi}_l : \boldsymbol{\eta} \in \tau \mapsto \boldsymbol{x} \in \tau_l$. Focusing, for instance, on the stiffness integral, its quadrature is given by

$$\begin{aligned}\int_{\omega^Q_{i,k}} \nabla u^h \cdot \nabla v^h \,\mathrm{d}\omega &\approx \sum_l \int_{\tau_l} \nabla u^h \cdot \nabla v^h \,\mathrm{d}\tau \\ &= \sum_l \sum_g \left(\nabla u^h(\boldsymbol{x}_g) \cdot \nabla v^h(\boldsymbol{x}_g)\right) \left|\nabla_{\boldsymbol{\eta}} \boldsymbol{\varphi}_l\right| w_g \,,\end{aligned} \tag{13}$$

where $\boldsymbol{x}_g$ are the quadrature points mapped to the physical domain, w_g are the weights, and $|\nabla_{\boldsymbol{\eta}} \boldsymbol{\varphi}_l|$ is the absolute value of the determinant of the Jacobi matrix of the affine mapping. The solution u^h and the test function v^h are according to (9) given in terms of the shape functions $N_i(\boldsymbol{x})$ of the finite element cell ω_i. Hence, the quadrature points $\boldsymbol{\eta}_g$ must be mapped to the respective points $\boldsymbol{x}_g$. This involves the mapping $\boldsymbol{\varphi}_l$ and the mapping implied by the bottom-up refinement of the finite element cell ω_i into $\omega^Q_{i,k}$.

3.5. Cut cell stabilisation

The cut finite element cells in C^c may have a small overlap with the physical domain Ω, and some of the associated basis functions may have a very small contribution to the system matrix $\boldsymbol{A}$ in (10) resulting in an ill-conditioned matrix [42,59]. One approach to improving the conditioning is an elimination of their respective coefficients from the system matrix. Simply discarding some of the coefficients and basis functions would harm the finite element convergence rates. Therefore, we eliminate the critical coefficients by extrapolating the solution field from the nearby nodal coefficients of the cells inside the domain. This approach is inspired by the extended B-splines by Höllig et al. [1], which has also been applied in other immersed discretisation techniques [39,40,60].

A basis function $N_i(\boldsymbol{x})$ is critical when the intersection of its support with the physical domain Ω lies below a threshold. Formally, this is expressed as

$$\frac{\left|\operatorname{supp} N_i(\boldsymbol{x}) \cap \left\{\boldsymbol{x} \in \mathbb{R}^3 \mid \phi(\boldsymbol{x}) \geq 0\right\}\right|}{|\operatorname{supp} N_i(\boldsymbol{x})|} < \epsilon \,, \tag{14}$$

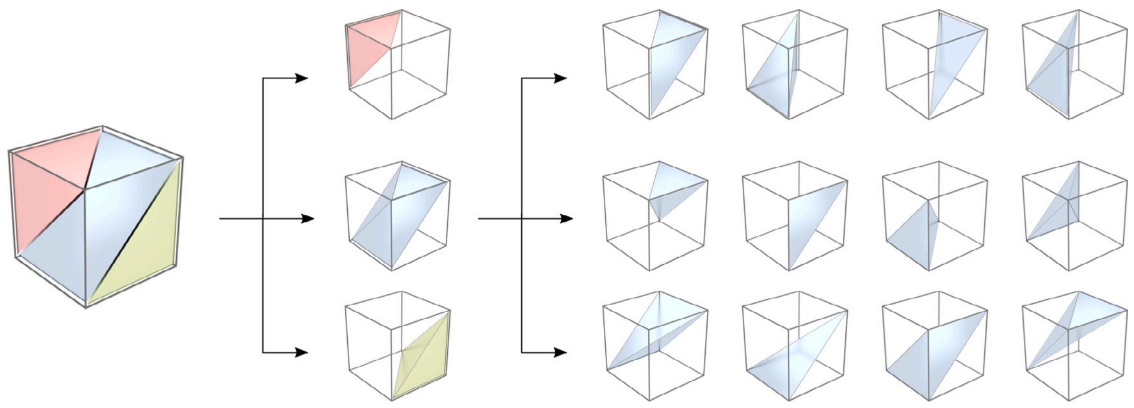


Fig. 8. Splitting patterns for subdividing a cut quadrature cell $\omega^Q_{i,k}$ into six simplices prior to the application of the marching tetrahedra algorithm.

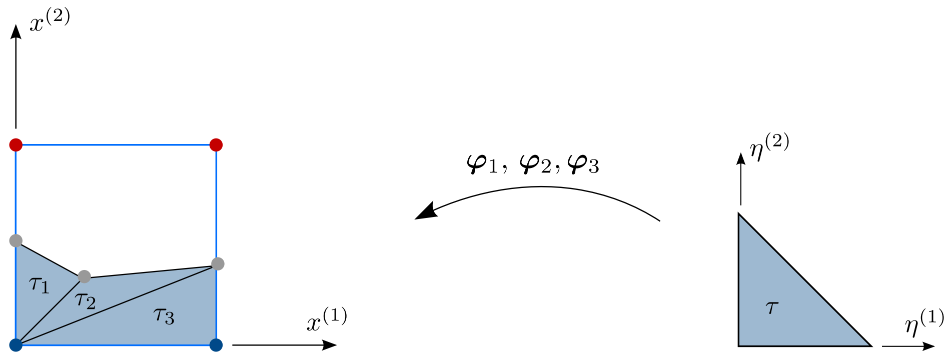


Fig. 9. 2D illustration of the integration of a quadrature cut cell $\omega^Q_{i,k}$ corresponding to a finite element cut cell $\omega_i \in C^c$. A reference simplex (right) is affinely mapped to the three simplices τ_1, τ_2 and τ_3 obtained with marching tetrahedra.

where $|\cdot|$ is the volume of the respective set, and the threshold is chosen as $\epsilon = 1/8$ in our numerical computations, see Fig. 10. To determine the numerator and denominator of (14) for each basis function $N_i(\boldsymbol{x})$, we perform an assembly-like procedure, in which each FE cell contributes its active and total volume to the corresponding basis functions.

The coefficients u_i of a critical basis function $N_i(\boldsymbol{x})$ are extrapolated from the nodal coefficients of a cell $\widetilde{\omega}_j$ which contains only non-critical basis functions. To identify $\widetilde{\omega}_j$ we first collect all the active cells in the two neighbourhood of the node $\boldsymbol{x}_i$ corresponding to $N_i(\boldsymbol{x})$. Subsequently, we select from the candidate cells the cell $\widetilde{\omega}_j$ with the centroid closest to node i. Denoting the set of nodal indices of $\widetilde{\omega}_j$ with $\mathbb{J}$ the critical coefficient u_i is obtained by simply evaluating the basis functions $N_j(\boldsymbol{x})$ with $j \in \mathbb{J}$ at $\boldsymbol{x}_i$, that is,

$$u_i = \sum_{j \in \mathbb{J}} N_j(\boldsymbol{x}_i) u_j \, . \tag{15}$$

After extrapolating the coefficients of all the critical basis functions present in the grid, the original nodal coefficients can be expressed abstractly as

$$\boldsymbol{u} = \boldsymbol{E} \widetilde{\boldsymbol{u}} \, . \tag{16}$$

Introducing this relation in the discrete system of Eqs. (10) we obtain

$$\widetilde{\boldsymbol{A}} \widetilde{\boldsymbol{u}} = \widetilde{\boldsymbol{f}} \, , \tag{17}$$

where $\widetilde{\boldsymbol{A}} = \boldsymbol{E}^{\mathsf{T}} \boldsymbol{A} \boldsymbol{E}$ and $\widetilde{\boldsymbol{f}} = \boldsymbol{E}^{\mathsf{T}} \boldsymbol{f}$. The matrix $\widetilde{\boldsymbol{A}}$ is well conditioned so that (17) can be robustly solved.

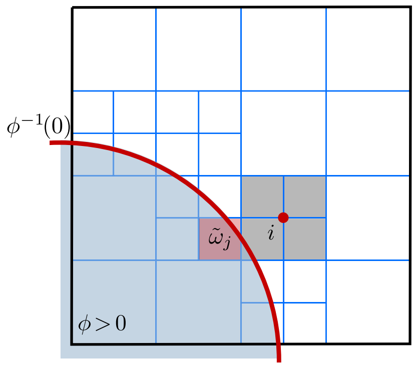


Fig. 10. The support of the critical basis function $\operatorname{supp} N_i(\boldsymbol{x})$ (in grey) corresponding to node i (in red) has a small overlap with the domain with $\phi_h(\boldsymbol{x}) \geq 0$ (in blue). The respective coefficient u_i is expressed as the linear combination of the coefficients of the basis functions of the nodes of the active cell $\widetilde{\omega}_j$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4. Domain partitioning and parallel solution

4.1. Finite element grid partitioning

In this section, we describe the decomposition of the FE grid into a set of subdomains assigned to individual processors of a parallel computer. As mentioned in Section 2.2, the FE grid is generated by refining the base grid through selective octree refinements. We can either perform refinements towards the geometric boundary, i.e., by identifying the cut cells C^c, or towards the solution features. The latter

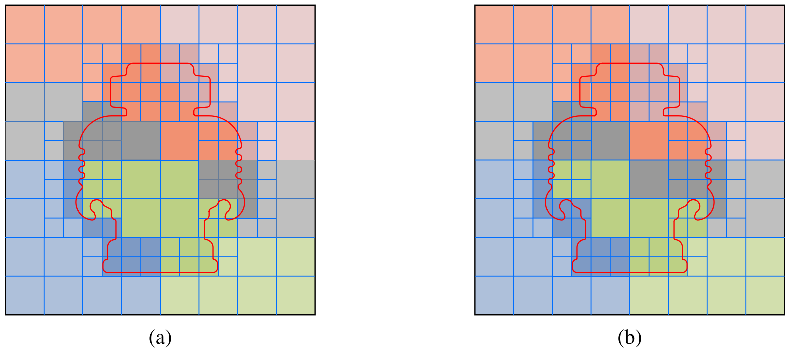


Fig. 11. Domain decomposition of $\Omega_\square$. Darker shades indicate that the cell is either active or cut, i.e. it belongs to C^d. (a) Domain decomposition based solely on equal partition of the Z-curve. The total number of cells of different colours are roughly equal but the active and cut cells differ for each colour. (b) A more balanced domain decomposition is achieved by applying larger weights for active and cut cells. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

is more demanding since the problem has to be solved repeatedly, or otherwise requires a-priori estimates of the solution.

We consider a division of the FE grid in $\Omega_\square$ into N_S nonoverlapping subdomains, $\Omega_{\square i}$, $i = 1, \dots, N_S$, which are assigned to individual processor cores and further processed in parallel. This is achieved by first generating the Z-curve of the FE grid, and subsequently subdividing the Z-curve into partitions of approximately equal length. It is important to note that each FE cell has predicate active, cut, or inactive, where only the first two kinds contribute significantly to the computation. Considering these characteristics in the partitioning leads to a more balanced load distribution in each processor. Specific to the processing of the cut-cell identification algorithm, we restrict the face-neighbour search within each subdomain.

In particular, we consider applying different weights on FE grid cells in the partitioning of the Z-curve. Let us denote the union of the active cells and cut cells as $C^d = C^a \cup C^c$. We apply for the set C^d a larger weight of 100 than those for inactive cells C^i of weight 1. Then, the division into subdomains is inherited from the division of $\Omega_\square$ as intersections of C^d with subdomains $\Omega_{\square i}$, i.e. $\Omega_i = C^d \cap \Omega_{\square i}$, $i = 1, \dots, N_S$. The weighted subdivision yields a decomposition of the domain with a balanced number of cut and active cells per processor, as illustrated in Fig. 11.

Depending on the computing memory requirements of the application, the geometry grid can be replicated on each partition or partitioned using the level set fracturing functionality of openVDB according to the bounding box of each partition [54]. The quadrature grid is generated in the cut cells belonging to each partition according to Section 3.2 by sampling the level set values.

4.2. Hanging nodes

Octree refinement in constructing the FE grid lead to the presence of hanging nodes. These are nodes created at the faces and edges between two or more elements by refining only one of them, see Fig. 12. For preserving the continuity of the FE solution, we require that the solution at the hanging node is determined by the values in the nodes of the larger element.

Consider the constraining element ω_L and the constrained element ω_R, as exemplified in Fig. 12. We express the solution at the hanging node at $\boldsymbol{x}_j$ as

$$u_h(\boldsymbol{x}_j) = \sum_{k=1}^{n_k} H_{j,k} u_k \,, \tag{18}$$

where $H_{j,k}$ is the value of the k−th shape function associated with the element ω_L at $\boldsymbol{x}_j$, i.e., $H_{j,k} = N_k(\boldsymbol{x}_j)$. Given that the only nonzero shape functions are those associated to the nodes at the common edge, the solution at the hanging node is constrained by the nodal values of ω_L at the edge, i.e., at the degrees of freedom g_1 and g_4. In our example

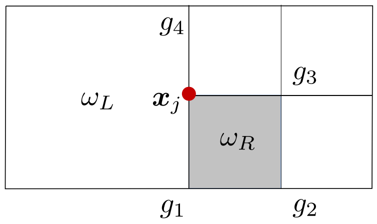


Fig. 12. Example of a hanging node. Large constraining element ω_L determines the value in the hanging node (red dot) of the constrained element ω_R (gray). Hanging node does not have a degree of freedom in the global system, its value is dictated by values of degrees of freedom g_1 and g_4. Algorithmically the constraint can be realised by applying the change-of-basis matrix $\boldsymbol{T}_R$ to the element matrix of ω_R and assembling the transformed local matrix to degrees of freedom g_1, g_2, g_3, and g_4.

with the bilinear shape functions, (18) simplifies to $u_h = \frac{1}{2}u_1 + \frac{1}{2}u_4$, where u_1 and u_4 are the values of the degrees of freedom g_1 and g_4, respectively. With respect to the implementation, the procedure for eliminating hanging nodes is equivalent to the one for eliminating the critical nodes through extrapolation in (15).

As a final remark, we note that it is possible for a hanging node to be constrained by a node which is subject to extrapolation. In the extreme case, the node from which we extrapolate can also be a hanging node and constrained by another regular node. This situation is in fact handled naturally by nesting the assembly lists related to the hanging nodes and those related to the extrapolation. At the end of this nesting, the element contributes its local matrix to regular degrees of freedom potentially through a relatively long assembly list. A more formal description of the potential chaining of constraints due to hanging nodes and the extrapolation has been recently given in [61].

4.3. Iterative substructuring

We employ iterative substructuring method to solve the linear system arising from the immersed finite element system (17). We consider that the global stiffness matrix $\widetilde{\boldsymbol{A}}$ and the right-hand side vector $\widetilde{\boldsymbol{f}}$ appearing in (17) can be assembled from the local contributions from each subdomain $\Omega_{\square i}$, that is, $\widetilde{\boldsymbol{A}} = \sum_{i=1}^{N_S} \boldsymbol{R}_i^\top \boldsymbol{A}_i \boldsymbol{R}_i$ and $\sum_{i=1}^{N_S} \widetilde{\boldsymbol{f}} = \boldsymbol{R}_i^\top \boldsymbol{f}_i$. The Boolean restriction matrix $\boldsymbol{R}_i$ containing a single 1 in each column selects the local from the global degrees of freedom.

For each local subdomain, we separate the degrees of freedom belonging to interior $\boldsymbol{u}_i^I$ and the interface $\boldsymbol{u}_i^\Sigma$, which leads to a 2×2 blocking of the local linear system,

$$\begin{bmatrix} \boldsymbol{A}_i^{II} & \boldsymbol{A}_i^{I\Sigma} \\ \boldsymbol{A}_i^{\Sigma I} & \boldsymbol{A}_i^{\Sigma\Sigma} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_i^I \\ \boldsymbol{u}_i^\Sigma \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_i^I \\ \boldsymbol{f}_i^\Sigma \end{bmatrix} . \tag{19}$$

Here the local interface degree of freedom $\boldsymbol{u}_i^\Sigma$ can be assembled into a global set of unknowns at the interface of all subdomains Σ utilising an interface restriction matrix $\boldsymbol{R}_i^\Sigma : \Sigma \to \Sigma_i$, where $\boldsymbol{u}^\Sigma = \sum_{i=1}^{N_S} \boldsymbol{R}_i^{\Sigma\top} \boldsymbol{u}_i^\Sigma$. Interface Σ is formed by degrees of freedom shared by several subdomains.

The iterative substructuring seeks the solution of the global interface problem

$$\boldsymbol{S}\boldsymbol{u}^\Sigma = \boldsymbol{h}\,, \tag{20}$$

using, for instance, the preconditioned conjugate gradient (PCG) method. The global Schur complement matrix $\boldsymbol{S}$ comprises of the subdomain contribution

$$\boldsymbol{S} = \sum_{i=1}^{N_S} \boldsymbol{R}_i^{\Sigma\top} \boldsymbol{S}_i \boldsymbol{R}_i^\Sigma\,, \tag{21}$$

where the local Schur complement with respect to Σ_i is defined as

$$\boldsymbol{S}_i = \boldsymbol{A}_i^{\Sigma\Sigma} - \boldsymbol{A}_i^{\Sigma I} \left(\boldsymbol{A}_i^{II}\right)^{-1} \boldsymbol{A}_i^{I\Sigma}\,. \tag{22}$$

Similarly, the right-hand side is assembled from the subdomains

$$\boldsymbol{h} = \sum_{i=1}^{N_S} \boldsymbol{R}_i^{\Sigma\top} \boldsymbol{h}_i\,, \tag{23}$$

where

$$\boldsymbol{h}_i = \boldsymbol{f}_i^\Sigma - \boldsymbol{A}_i^{\Sigma I} \left(\boldsymbol{A}_i^{II}\right)^{-1} \boldsymbol{f}_i^I\,. \tag{24}$$

Once we know the local solution at the interface $\boldsymbol{u}_i^\Sigma$, the solution in the interior of each subdomain $\boldsymbol{u}_i^I$ is recovered from the first row of (19). Note that neither the global matrix $\boldsymbol{S}$ nor the local matrices $\boldsymbol{S}_i$ are explicitly constructed in the iterative substructuring. Only multiplications of vectors with $\boldsymbol{S}_i$ are needed at each PCG iteration.

4.4. BDDC preconditioner

Next, we briefly describe the balancing domain decomposition based on constraints (BDDC) preconditioner in solving the interface problem (20). An action of the BDDC preconditioner $\boldsymbol{M}_{BDDC}^{-1}$ produces a preconditioned residual $\boldsymbol{z}^\Sigma$ from the residual in the kth iteration $\boldsymbol{r}^\Sigma = \boldsymbol{S}\boldsymbol{u}_{(k)}^\Sigma - \boldsymbol{h}$ by implicitly solving the system $\boldsymbol{M}_{BDDC}\boldsymbol{z}^\Sigma = \boldsymbol{r}^\Sigma$. Specifically, BDDC considers a set of coarse degrees of freedom which will be continuous across subdomains such that the preconditioner is invertible yet inexpensive to invert. In this work we consider degrees of freedom at selected interface nodes (corners) and arithmetic averages across subdomain faces and edges as the coarse degrees of freedom. This gives rise to a global coarse problem with the unknowns $\boldsymbol{u}_C$ and local subdomain problems with independent degrees of freedom $\boldsymbol{u}_i$ which are parallelisable.

BDDC gives an approximate solution which combines the global coarse and local subdomain components, i.e.,

$$\boldsymbol{z}^\Sigma = \sum_{i=1}^{N_S} \boldsymbol{R}_i^{\Sigma\top} \boldsymbol{W}_i \boldsymbol{R}_{Bi} \left(\boldsymbol{u}_i + \boldsymbol{\Phi}_i \boldsymbol{R}_{Ci} \boldsymbol{u}_C\right). \tag{25}$$

In particular, $\boldsymbol{u}_C$ and $\boldsymbol{u}_i$ are obtained in each iteration by solving

$$\boldsymbol{S}_C \boldsymbol{u}_C = \sum_{i=1}^{N_S} \boldsymbol{R}_{Ci}^\top \boldsymbol{\Phi}_i^\top \boldsymbol{R}_{Bi}^\top \boldsymbol{W}_i \boldsymbol{R}_i^\Sigma \boldsymbol{r}^\Sigma, \tag{26}$$

$$\begin{bmatrix} \boldsymbol{A}_i & \boldsymbol{C}_i^\top \\ \boldsymbol{C}_i & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_i \\ \boldsymbol{\mu}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{Bi}^\top \boldsymbol{W}_i \boldsymbol{R}_i^\Sigma \boldsymbol{r}^\Sigma \\ \boldsymbol{0} \end{bmatrix}, \; i = 1, \dots, N_S, \tag{27}$$

where $\boldsymbol{S}_C$ is the stiffness matrix of the global coarse problem, matrix $\boldsymbol{A}_i$ is assembled from elements in the ith subdomain, and $\boldsymbol{C}_i$ is a constraint matrix enforcing zero values of the local coarse degrees of freedom. The diagonal matrix $\boldsymbol{W}_i$ applies weights to satisfy the partition of unity, and it corresponds to a simple arithmetic averaging in this work. The Boolean restriction matrix $\boldsymbol{R}_{Bi}$ selects the local interface unknowns from those at the whole subdomain, the columns of $\boldsymbol{\Phi}_i$ contain the local coarse basis functions, and $\boldsymbol{R}_{Ci}$ is the restriction matrix of the global vector of coarse unknowns to those present at the ith subdomain. Our implementation of the BDDC preconditioner is detailed in [62]. For the interested readers, we refer to [63–65] for a thorough description of the BDDC method and its multilevel variants [50,66,67].

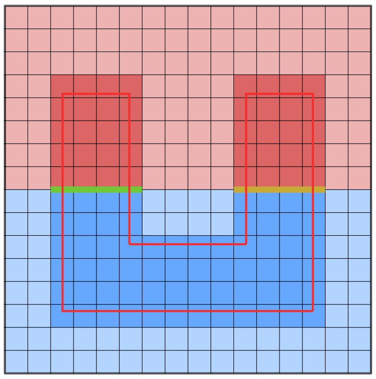


Fig. 13. Example of fragmenting of subdomains due to the extraction of the active and cut cells C^d. By analysis of the dual graph of each subdomain, the two components of the red subdomain are detected, and the two faces between red and blue subdomains are identified. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

In the context of the domain decomposition of our FE grid, we might encounter the issue of subdomain fragmentation. One typical cause is the partitioning of the Z-curve which does not guarantee connected subdomains, see for example the gray subdomain showcased in Fig. 11(a). Another possible cause of the fragmenting is due to the subdomain extraction $\Omega_i = C^d \cap \Omega_{\square i}$. A simple case of this effect is shown in Fig. 13.

As proposed in [62], we remedy this problem by analysing the *dual graph* of the FE grid of each subdomain followed by generation of inter-subdomain faces and corresponding constraints independently for each component. Within the dual graph, FE grid cells correspond to graph vertices, and a graph edge is introduced between two vertices whenever the corresponding cells share at least four degrees of freedom. Consequently, the local saddle point problem (27) will become solvable. In our implementation, we rely on the open-source *BDDCML* solver [53] for the implementation of the multilevel BDDC method, and the solver is equipped with a component analysis based on the dual graph of each subdomain grid.

5. Examples

We introduce several examples of increasing complexity to demonstrate the convergence of the proposed immersed finite element scheme and its robustness for complex 3D CAD geometries. We focus the analysis on the Poisson problem, e.g., modelling heat transfer, where the solution is a scalar field. Throughout this section, we emphasise the use of three grids characterised by their resolution, namely h_g for geometry grid, h_f for FE grid and h_q for quadrature grid.

5.1. Interplay between geometry, quadrature and FE grid sizes

As a first example we consider the Poisson–Dirichlet problem on a unit sphere with a prescribed solution $u = \cos(x_3)$ as illustrated in Fig. 14.

In this example, we first study the effect of the geometry grid resolution h_g and quadrature grid resolution h_q on the volume and surface integration. The sphere geometry is described using an analytical signed distance function $\phi(\boldsymbol{x}) = 1 - (x_1^2 + x_2^2 + x_3^2)^{1/2}$. To start with, we sample the signed distance function over the nodes of the geometry grid with a resolution $h_g = \{1/2, 1/8, 1/32, 1/128\}$. For each value of h_g, we

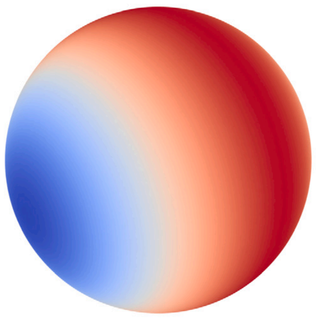


Fig. 14. Solution of the Poisson problem in a sphere, $u = \cos(x_3)$.

use quadrature grid with resolution $h_q = \{0.6, 0.3, 0.15, 0.07, 0.035\}$ to linearly reconstruct the volume and the surface by means of marching tetrahedra. Fig. 15 indicates the reconstructed sphere for various h_q for the geometry grid resolution $h_g = 1/128$. We compare the volume and surface area of the reconstructed representation with the analytical value in Fig. 16. For each h_g it is evident that the errors reach plateau when $h_q < h_g$, indicating that the geometric error is bounded by the geometry grid resolution h_g.

Next, we verify the numerical convergence of the relative error for the Poisson–Dirichlet problem both in L_2-norm, i.e. $\|u-u_h\|_{L_2}/\|u_h\|_{L_2}$, and H^1-seminorm, i.e. $|u-u_h|_{H^1}/|u_h|_{H^1}$. We take a fine STL mesh of the sphere to obtain the signed distance function $\phi(\boldsymbol{x})$. The resolution of the geometry grid is twice smaller than the finest resolution of the quadrature grid, i.e., $h_g = h_q/2$, where the finest quadrature grid resolution is $h_q = h_f/8$. The convergence is established against the resolution of the uniform FE grid h_f. Note that FE grid is used for the field discretisation using Lagrange basis functions. For integration we use a quadrature grid of size $h_q = \{h_f, h_f/2, h_f/4, h_f/8\}$. As an example, $h_q = h_f/4$ indicates that quadrature grid is obtained by twice subrefining the cut FE cells. We also include the case $h_q \equiv h_f$, i.e., h_q is refined whenever h_f is refined. Fig. 17 shows the convergence in L_2-norm and H^1-seminorm for both linear ($p = 1$) and quadratic ($p = 2$) basis functions. It is evident from Fig. 17 that for the linear case optimal convergence rate is achieved for all h_q. However, for the quadratic case the convergence rate is optimal only for finer quadrature resolution and degrades for coarser quadrature resolution. This result emphasises the importance of subrefining the cut FE cells into sufficiently fine quadrature grid to improve the approximation of the curved boundary.

5.2. *Adaptive refinement of the FE grid*

As the second example we consider an internal layer problem in a unit cube $\Omega = [0,1]^3$ with the prescribed solution $u = \arctan(60(r - \pi/3))$. Here $r = (x_1^2 + x_2^2 + x_3^2)^{1/2}$ is the distance from the origin. Note that the cube domain Ω is embedded in a larger bounding box $\Omega_\square$ which generally is not aligned with the unit cube. In this example we test a sequence of uniform and error-driven adaptive FE grid refinements. The coarsest FE grid shown in Fig. 18 is obtained using 4 uniform refinements of the bounding domain $\Omega_\square$ which is adaptively refined once and twice, see Fig. 18. It is also emphasised that in this example we perform 3 octree refinements of the extraordinary cut FE cells detected using the sharp feature indicator (5), i.e., $h_q = h_f/8$ for FE cells containing the corners and edges of Ω. The computation was performed using 1024 cores of the *Salomon* supercomputer at the IT4Innovations National Supercomputing Centre in Ostrava, Czech Republic. Its computational nodes are equipped with two 12-core Intel Xeon E5-2680v3, 2.5 GHz processors, and 128 GB RAM.

The dependence of the solution error measured as the relative L_2-norm and the relative H^1-seminorm with respect to the FE grid resolution h_f for uniform grid refinements is presented in Fig. 19. Here we can see that for this domain with straight faces, we are able to achieve the optimal convergence rates for linear and quadratic basis functions even without using the improved quadrature, i.e., with $h_q = h_f$. Fig. 20 shows the relative L_2-norm and the relative H^1-seminorm errors with respect to the number of degrees of freedom for both uniform and adaptive refinement. In the adaptive, or error-driven, approach, we refine the elements with the largest H^1-seminorm error within each adaptive step such that approximately 15% of elements of the whole box are refined. A detailed description of a parallel implementation of this refinement strategy is provided in [62]. It can be observed from Fig. 20 that the adaptive refinement achieves optimal convergence rate for both linear and quadratic basis functions with lower relative errors than the uniform refinement. In other words it requires more than ten times less degrees of freedom to achieve the same precision as the uniform grid.

5.3. *Strong scalability*

We assess the strong parallel scalability of our solver by solving the Poisson–Dirichlet problem with a prescribed solution $u = \cos(x_3)$. We consider in this example a handle block geometry as shown in Fig. 21. In strong scaling the finite element size is fixed and the number of processors is continuously increased. The signed distance function of the handle block is obtained using constructive solid geometry (CSG) [18] from three cylinders and a box. In this example we use a base grid obtained by refining the bounding box $\Omega_\square$ uniformly six times, where grid cells far from the boundary are coarsened by default. We obtain from the base grid an FE grid through five levels of refinements towards the boundary with one additional refinement level for quadrature of extraordinary cells. The resulting FE grid contains 12 million trilinear elements and 11 million degrees of freedom.

The strong scaling test was performed on the *Salomon* supercomputer at the IT4Innovations National Supercomputing Centre in Ostrava, Czech Republic. Its computational nodes are equipped with two 12-core Intel Xeon E5-2680v3, 2.5 GHz processors, and 128 GB RAM. The number of subdomains, i.e., the number of processes in the parallel computation, ranges from 128 to 1024.

The results of the strong scaling test are presented in Fig. 22(a), where the runtimes of the important components of the solver are analysed separately. These include assembly of the matrices and time for the BDDC solution. The latter is further divided into the time spent in the BDDC setup and in PCG iterations. In addition, we report the total time necessary for the whole simulation. Optimal strong scalability corresponds to halving the computational time every time the number of subdomains is doubled. In the logarithmic scale, this corresponds to a straight line, marked as 'optimal' in Fig. 22(a). It is evident from the figure that the scalability of assembly and BDDC setup is optimal. However, the PCG iterations within BDDC do not scale optimally, which leads to a suboptimal scaling of the overall simulation. The suboptimal scaling in the PCG can be attributed to the growing number of iterations as identified in Fig. 22(b). Consequently, we also plot the time for one PCG iteration showing the favourable strong scalability of our implementation.

5.4. *Application to complex engineering CAD models*

In this section we first assess the robustness of the proposed method for solving Poisson–Dirichlet problems on various complex geometries from computer graphics and engineering, see Fig. 23. The engineering models are given in STEP format and are converted into sufficiently fine STL meshes using FreeCAD. The computer graphics models are given as STL meshes. The STL meshes are immersed in the geometry grid with resolution smaller than the resolution of the quadrature grid, i.e., $h_g \leq h_q$. For all geometries, the base grid is obtained by five uniform refinements of $\Omega_\square$. The FE grid is obtained by six refinements of the base grid towards the boundary. The quadrature grid is obtained by

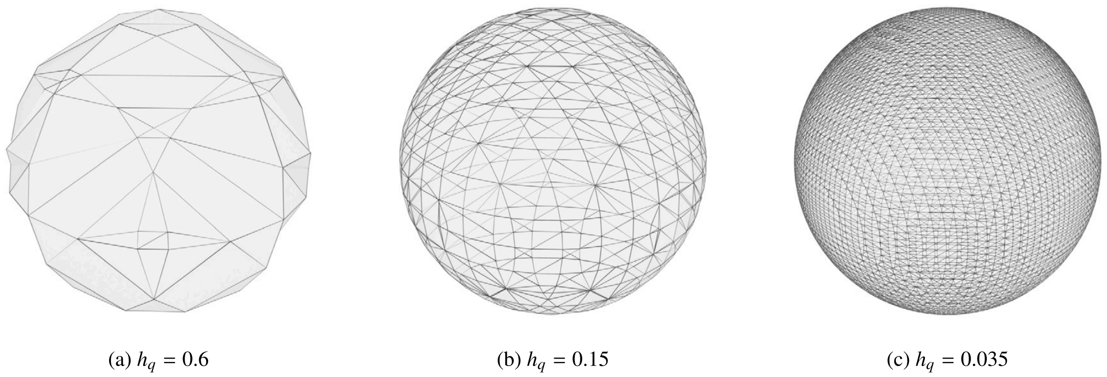


Fig. 15. Sphere reconstructed from the quadrature grid using the marching tetrahedra algorithm for a fixed geometry grid resolution $h_g = 1/128$ and different quadrature grid resolutions h_q.

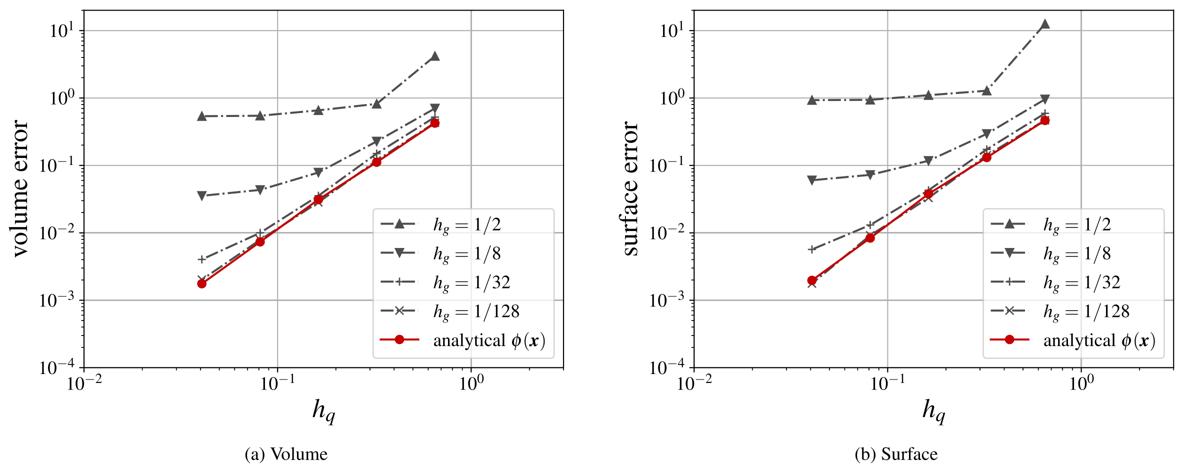


Fig. 16. Dependence of the error in volume and surface computation for a fixed quadrature grid resolution $h_q = 0.035$ and different geometry grid resolutions h_g. The red lines indicate the surface and volume errors when analytical signed distance function is used.

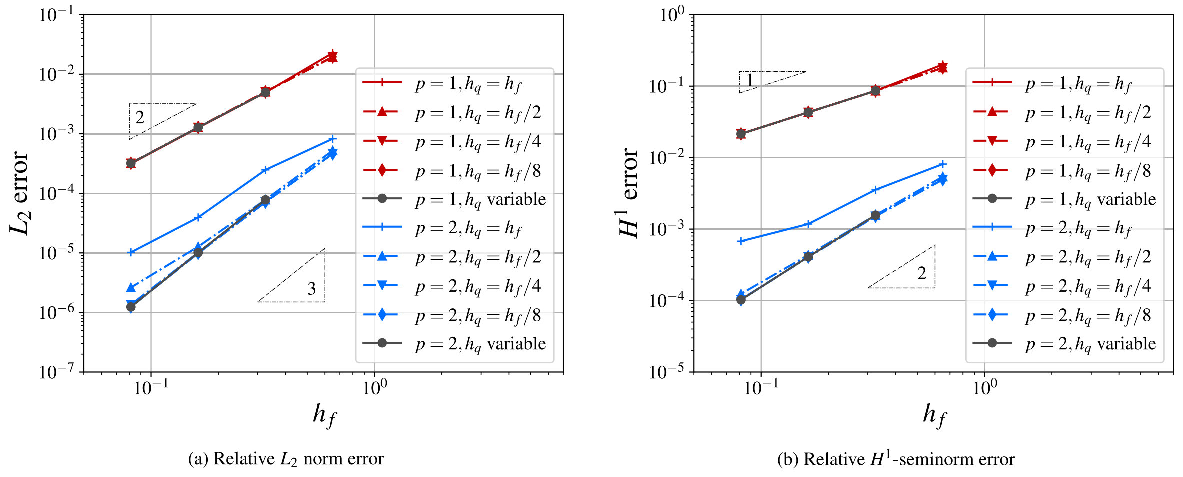
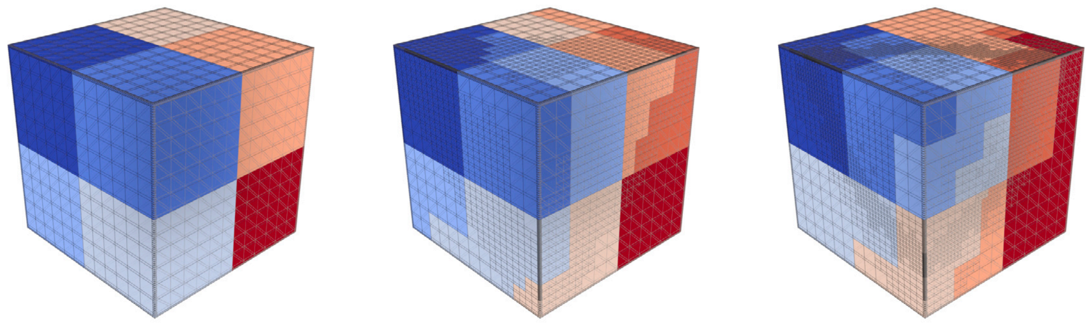


Fig. 17. Dependence of the relative finite element errors for linear ($p = 1$) and quadratic basis ($p = 2$) functions and for different finite element and quadrature grid resolutions h_f and h_q.

refining the extraordinary cut FE cells once. We consider the prescribed solution $u = \cos(x_3)$, trilinear Lagrange polynomials on FE cells as the basis, and three levels in the BDDC method.

The computations are performed on the *Karolina* supercomputer at the IT4Innovations National Supercomputing Centre in Ostrava, Czech Republic. The computational nodes are equipped with two 64-core AMD 7H12 2.6 GHz processors, and 256 GB RAM.

We decompose the FE grid into 256 subdomains assigned to the same number of computer cores. Table 1 shows the number of degrees of freedom, number of elements, number of iterations to reach the


(a) Decomposition of the FE grid into eight subdomains

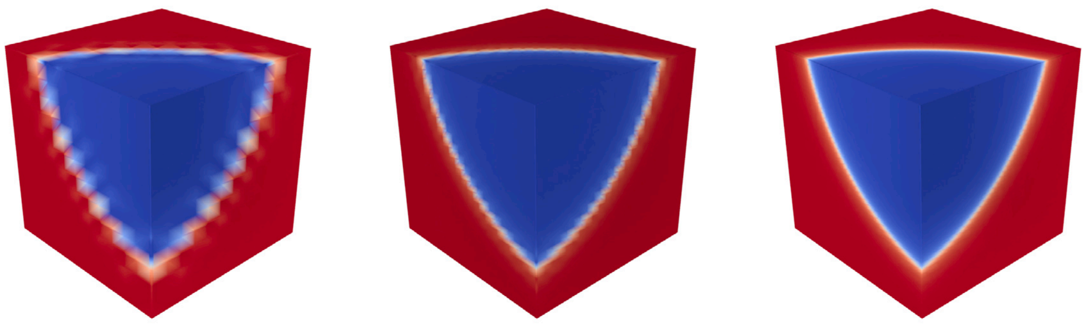

(b) FE solution

Fig. 18. Adaptively refined grids for the internal layer problem. In each column the FE grid and the respective solution are shown.

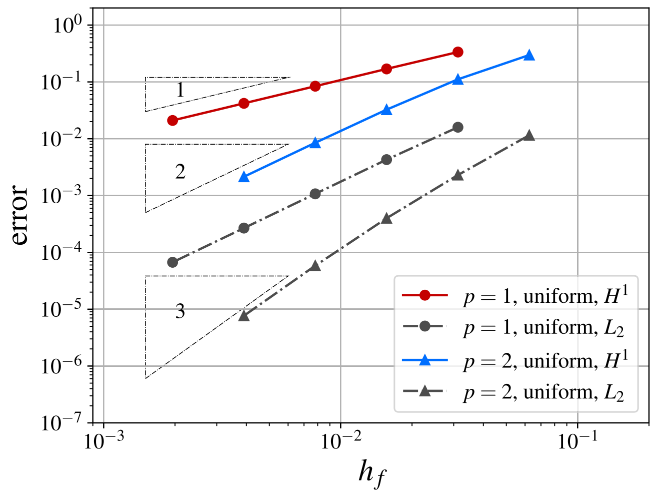


Fig. 19. Dependence of the solution error on h_f for linear ($p = 1$) and quadratic ($p = 2$) elements for the internal layer problem.

relative residual precision of 10^{-6}, and computational runtimes for each geometry. In addition, we compare the runtimes of the domain decomposition solver with the parallel sparse direct solver MUMPS [68]. It is evident from Table 1 that the BDDC solver is reasonably robust and able to converge for all these geometries requiring from 257 to 913 PCG iterations. While MUMPS is also able to provide the solutions for all geometries when provided enough memory, it is consistently slower than the BDDC solver by a factor ranging from 1.2 to 6.3 depending on the problem.

5.5. *Weak scalability studies*

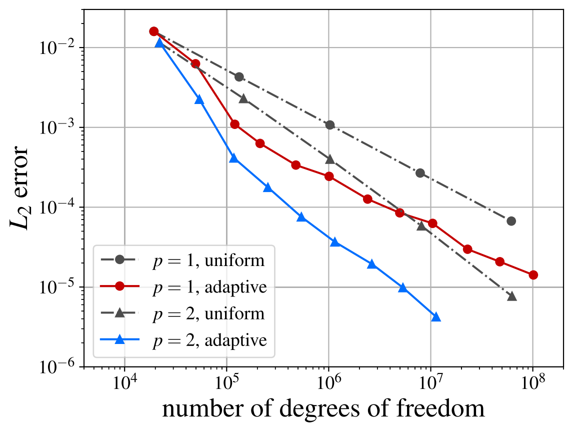
Finally, we test the weak scalability of the proposed approach first using the *turbo* and *cover* geometries shown in Fig. 23. In the left part of Fig. 24, we present computational time for an increasing number of processes, and we compare the three-level BDDC method with using a distributed sparse Cholesky factorisation by MUMPS. During the test, the number of processes increases from 16 to 1 024, and we perform one refinement of elements towards boundary within each step. The problem size grows approximately four times with each of these refinements, starting at 660 thousand unknowns and finishing with 46 million unknowns in the global system for the *cover* problem, and from 290 thousand to 20 million unknowns for the *turbo* problem. For both problems, the local problem size is kept approximately fixed, around 45 thousand unknowns per subdomain for the cover problem, and around 19 thousand for the turbo problem.

We can clearly observe the growing time of the MUMPS solver corresponding to the increasing complexity of the sparse Cholesky factorisation of the distributed matrix. On the other hand, the three-level BDDC method requires significantly less time to reach the desired precision, although the solution time grows as well. Most of this growth can be attributed to the increasing number of iterations, which is plotted separately in the right part of Fig. 24. For the largest presented problem, the three-level BDDC method is 38 times faster than the MUMPS solver.

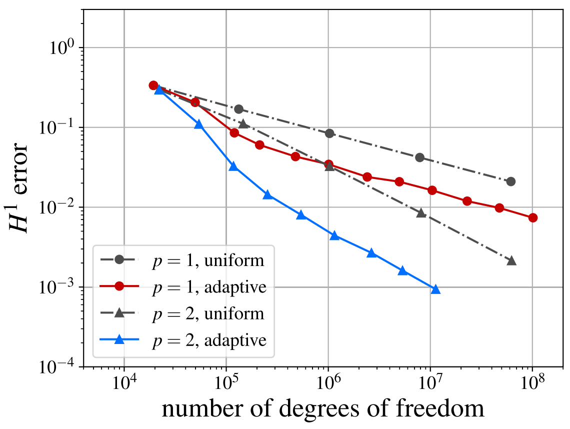
Furthermore, we consider the unit cube geometry introduced in Section 5.2 to evaluate the weak scalability of our approach for a more structured problem. We can see from Fig. 24 that the weak scalability for this problem is almost optimal with the number of iterations not growing beyond 10.

6. Conclusions

We introduced an immersed finite method that uses three non-boundary conforming background grids to solve the Poisson problem

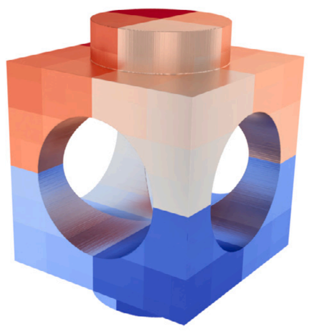


(a) Relative L_2 norm error

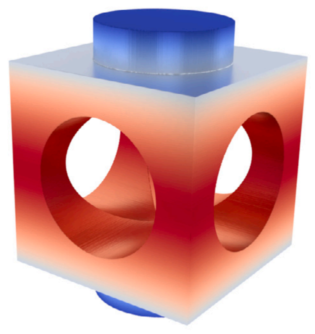


(b) Relative H^1-seminorm error

Fig. 20. Dependence of the relative finite element errors on the number of degrees of freedom for uniform and adaptive refinements for the internal layer problem.

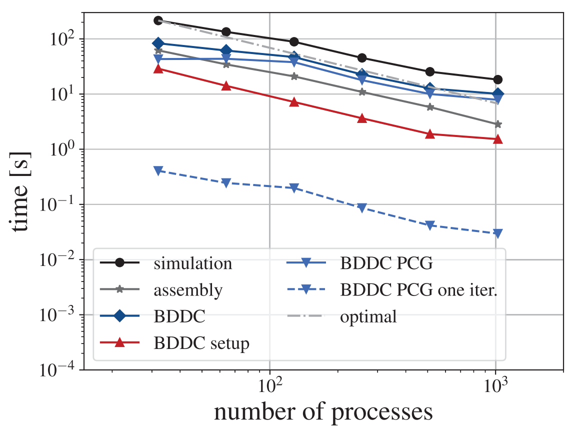


(a) Decomposition of the FE grid

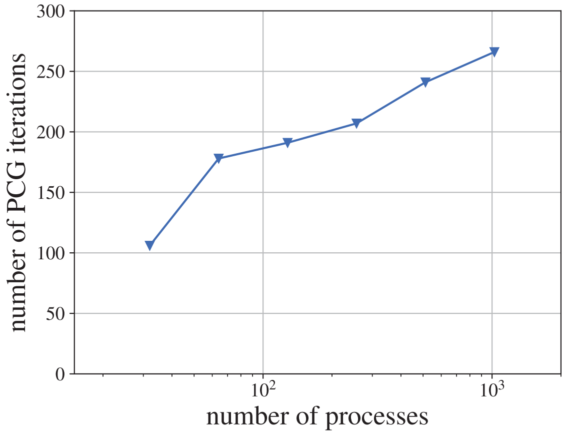


(b) FE solution

Fig. 21. The handle block geometry. The decomposition of the FE grid into 128 subdomains and the FE solution.



(a)



(b)

Fig. 22. Strong scaling test on the handle block geometry, 6 uniform refinements, 3 unrefinements, 5 refinements towards boundary, 1 octree refinement of extraordinary cells, finite element order 1, 3-level BDDC. Problem with 12 million elements and 11 million DOFs. Timing of the important parts of the solver (a) and number of PCG iterations (b).

over complex CAD models. The three grids, namely the geometry, finite element, and quadrature grids, are each constructed to represent the geometry, discretise the physical field, and for integration, respectively. In our examples, we use an implicit description of the geometries in the form of a scalar-valued signed distance function. This is obtained by evaluating the distance from the grid points to the sufficiently fine triangle surface mesh (STL mesh) approximating the exact CAD surface. To capture geometrical features up to a user-prescribed resolution, the geometry grid is refined within a narrow band from the boundary. In our computations, we choose the geometry grid to have the highest

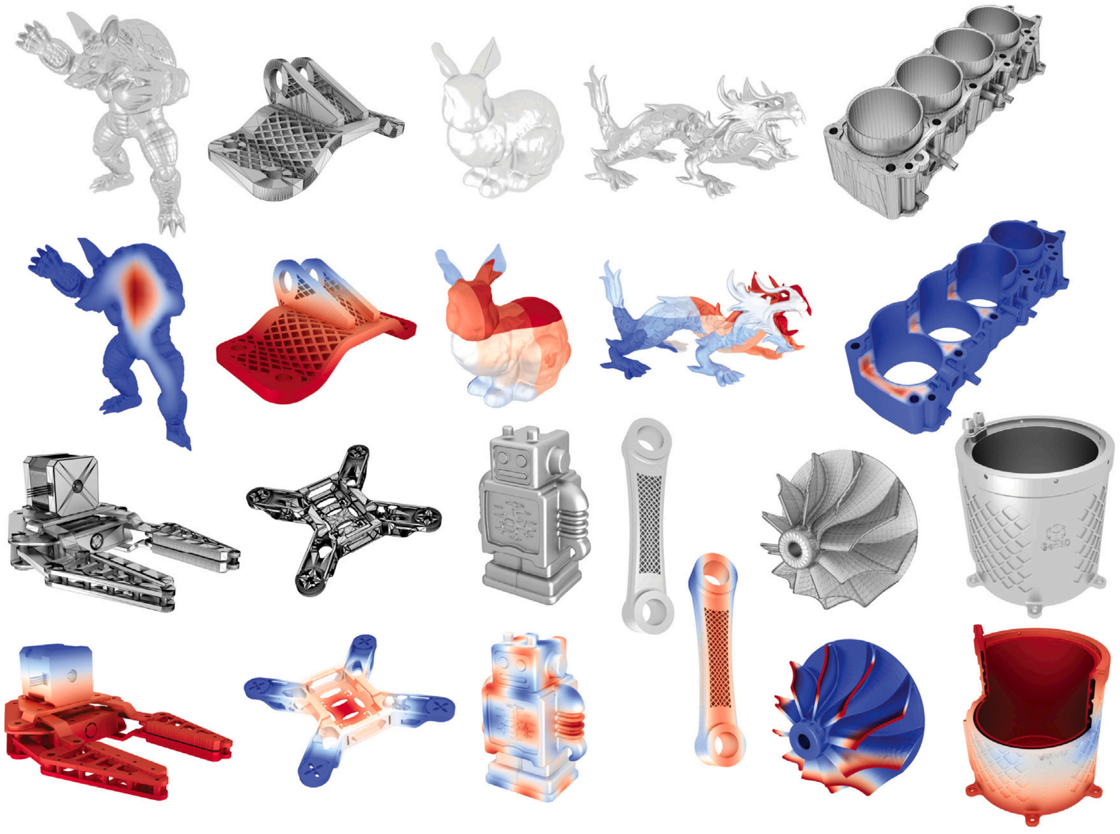


Fig. 23. Complex geometries used for the robustness test (from top left to bottom right): *armadillo*, *bracket*, *bunny*, *dragon*, *engine*, *gripper*, *quadcopter*, *robo*, *spanner*, *turbo*, and *cover*.

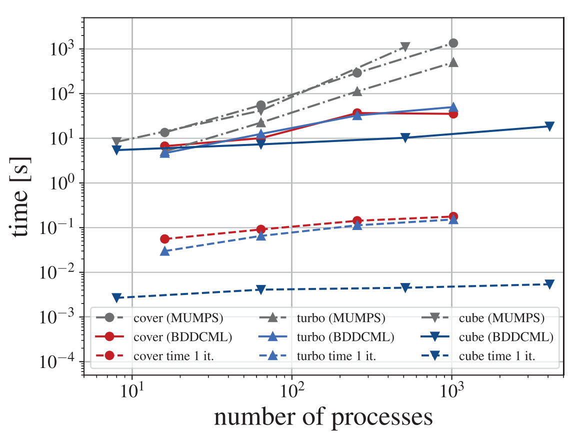
Table 1
Robustness test for different geometries, 5 uniform refinements, 3 unrefinements, 6 refinements towards boundary, 1 refinement for quadrature. In the legend, 'Interf.' is *interface*, 'Elems.' is *number of elements*, 'Iters.' is *number of iterations*, 'Setup' is *BDDC setup*, 'Solve BDDC' is the overall *BDDC solver time* and 'Solve MUMPS' is the overall time using the MUMPS distributed sparse direct solver.

Problem	DOFs				Time [s]				
	Global	Interf.	Elems.	Iters.	Assembly	Setup	PCG	Solve BDDC	Solve MUMPS
Armadillo	9.2M	384k	11.9M	360	20.1	2.9	72.1	80.0	238.5
Bracket	11.3M	523k	14.4M	343	28.9	3.3	77.0	87.9	275.8
Bunny	12.8M	463k	16.7M	450	29.5	4.3	110.5	122.4	359.8
Dragon	5.2M	280k	6.8M	375	11.2	1.7	79.3	83.6	125.3
Engine	11.6M	533k	14.9M	436	32.7	4.0	89.3	105.4	290.9
Gripper	10.1M	443k	12.7M	913	25.9	3.2	160.1	173.9	230.9
Quadcopter	4.5M	277k	5.6M	264	12.0	1.5	59.5	64.0	95.9
Robo	11.2M	378k	14.6M	257	27.5	3.9	65.2	77.6	260.2
Spanner	5.2M	338k	6.7M	503	12.4	1.9	97.2	102.1	124.0
Turbo	2.0M	620k	2.5M	270	50.6	6.1	58.5	79.0	498.2
Cover	11.6M	424k	15.0M	209	35.1	10.1	33.6	45.0	459.7

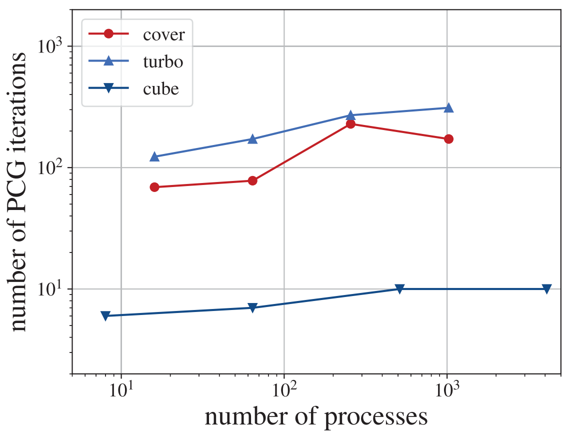
resolution among the three grids. Our examples have shown that the resolution of the geometry grid plays a crucial role in determining the boundary approximation error as well as limiting the integration error.

The finite element grid for discretisation is constructed by considering the features of the physical field. In cases where the solution features coincide with the boundary, we have presented a technique to correctly identify FE grid cells that are cut by the domain boundary, i.e., cut cells, including those containing sharp and other small geometrical features. In the integration step, such cut cells are integrated by first constructing the quadrature grid using a bottom-up strategy. The leaf cells of the quadrature grid are subsequently tessellated using marching tetrahedra. We have demonstrated with an internal layer example that an optimal convergence rate can be obtained for both linear and high-order polynomial basis functions using two levels of refinement for the quadrature grid. When FE basis functions cover only a tiny part of the physical domain, we use a cut-cell stabilisation technique that extends the basis support over the nearest active cells.

There are several promising extensions of the proposed three-grid immersed finite element method. One potential future direction of this research is to use a neural network for representing the implicit signed distance function, as demonstrated in [69] for point cloud input data. The use of point clouds as geometric input is gaining interest due to its direct connection with inspection and monitoring of engineering products, as shown in [70]. The application of the current approach beyond the Poisson equation to other second-order elliptic equations, like linear elasticity, is straightforward, as the only significant change involves the integrands in the weak form. However, the consideration of other kinds of linear and nonlinear equations, like elastodynamics, requires further analysis. Additionally, the proposed cut-cell identification algorithm can be modified to detect long-sliver cracks and other features if they are considered necessary in some applications. One possible strategy



(a)



(b)

Fig. 24. Weak scaling of the BDDC and the direct solver (MUMPS) for the *cover*, the *turbo*, and the *unit cube* problems (left), and the corresponding numbers of iterations required by the BDDC method to reach relative residual precision of 10^{-6} (right).

is to use bottom-up adaptive mesh refinement approach for the FE grid, starting from the finest resolution of the geometry grid. Lastly, another promising extension is the refinement of the finite element grid according to an a posteriori error indicator instead of relying on a priori knowledge of the solution features, as done in the current work.

CRediT authorship contribution statement

Eky Febrianto: Writing – review & editing, Writing – original draft, Software, Methodology, Data curation, Conceptualization. **Jakub Šístek:** Writing – review & editing, Writing – original draft, Software, Methodology, Data curation, Conceptualization. **Pavel Kůs:** Writing – original draft, Software, Methodology. **Matija Kecman:** Writing – review & editing, Methodology. **Fehmi Cirak:** Writing – review & editing, Writing – original draft, Supervision, Software, Methodology, Data curation, Conceptualization.

Declaration of competing 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.

Data availability

Data will be made available on request.

Acknowledgements

JS has been supported by the Czech Science Foundation through GAČR 23-06159S, and by the Czech Academy of Sciences through RVO:67985840. Computational time on the Salomon and Karolina supercomputers has been provided thanks to the support of the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90140). EF has been supported by the Royal Society Research Grants through RGS\R2\222318.

References

[1] Höllig K, Reif U, Wipper J. Weighted extended B-spline approximation of Dirichlet problems. SIAM J Numer Anal 2001;39:442–62.
[2] Hansbo A, Hansbo P. An unfitted finite element method, based on Nitsche's method, for elliptic interface problems. Comput Methods Appl Mech Engrg 2002;191:5537–52.
[3] Freytag M, Shapiro V, Tsukanov I. Field modeling with sampled distances. Comput Aided Des 2006;38:87–100.
[4] Parvizian J, Düster A, Rank E. Finite cell method. Comput Mech 2007;41:121–33.
[5] Rüberg T, Cirak F. An immersed finite element method with integral equation correction. Internat J Numer Methods Engrg 2011;86:93–114.
[6] Sanches RAK, Bornemann PB, Cirak F. Immersed b-spline (i-spline) finite element method for geometrically complex domains. Comput Methods Appl Mech Engrg 2011;200:1432–45.
[7] Burman E, Claus S, Hansbo P, Larson MG, Massing A. CutFEM: Discretizing geometry and partial differential equations. Internat J Numer Methods Engrg 2015;104:472–501.
[8] Kamensky D, Hsu MC, Schillinger D, Evans JA, Aggarwal A, Bazilevs Y, et al. An immersogeometric variational framework for fluid-structure interaction: Application to bioprosthetic heart valves. Comput Methods Appl Mech Engrg 2015;284:1005–53.
[9] Main A, Scovazzi G. The shifted boundary method for embedded domain computations. Part I: Poisson and Stokes problems. J Comput Phys 2018;372:972–95.
[10] Noël L, Schmidt M, Doble K, Evans JA, Maute K. XIGA: An extended IsoGeometric analysis approach for multi-material problems. Comput Mech 2022;70(6):1281–308.
[11] Schmidt M, Noël L, Doble K, Evans JA, Maute K. Extended isogeometric analysis of multi-material and multi-physics problems using hierarchical B-splines. Comput Mech 2023;71(6):1179–203.
[12] Li K, Michopoulos JG, Iliopoulos A, Steuben JC, Scovazzi G. Complex-geometry simulations of transient thermoelasticity with the Shifted Boundary Method. Comput Methods Appl Mech Engrg 2024;418:116461.
[13] Shapiro V, Tsukanov I, Grishin A. Geometric issues in computer aided design/computer aided engineering integration. J Comput Inf Sci Eng 2011;021005:1—021005:13.
[14] Marussig B, Hughes TJR. A review of trimming in isogeometric analysis: Challenges, data exchange and simulation aspects. Arch Comput Methods Eng 2017;25:1059–127.
[15] Xiao X, Alliez P, Busé L, Rineau L. Delaunay meshing and repairing of NURBS models. Comput Graph Forum 2021;40:125–42.
[16] Ciarlet PG, Raviart PA. Interpolation theory over curved elements, with applications to finite element methods. Comput Methods Appl Mech Engrg 1972;1:217–49.
[17] Strang G, Fix G. An analysis of the finite element method. 2nd ed. Wellesley-Cambridge Press; 2008.
[18] Ricci A. A constructive geometry for computer graphics. Comput J 1973;16:157–60.
[19] Farin GE, Hoschek J, Kim M-S, editors. Handbook of computer aided geometric design. Elsevier; 2002.
[20] Museth K, Breen DE, Whitaker RT, Barr AH. Level set surface editing operators. In: Proceedings of the 29th annual conference on computer graphics and interactive techniques. 2002, p. 330–8.
[21] Patrikalakis NM, Maekawa T. Shape interrogation for computer aided design and manufacturing. Springer; 2009.
[22] Kambampati S, Jauregui C, Museth K, Kim HA. Geometry design using function representation on a sparse hierarchical data structure. Comput Aided Des 2021;133:102989:1–102989:14.
[23] Upreti K, Song T, Tambat A, Subbarayan G. Algebraic distance estimations for enriched isogeometric analysis. Comput Methods Appl Mech Engrg 2014;280:28–56.
[24] Vaitheeswaran P, Subbarayan G. Improved Dixon resultant for generating signed algebraic level sets and algebraic boolean operations on closed parametric surfaces. Comput Aided Des 2021;135:103004.

[25] Xiao X, Sabin M, Cirak F. Interrogation of spline surfaces with application to isogeometric design and analysis of lattice-skin structures. Comput Methods Appl Mech Engrg 2019;351:928–50.
[26] Museth K. VDB: High-resolution sparse volumes with dynamic topology. ACM Trans Graph 2013;32:27.
[27] Museth K, Lait J, Johanson J, Budsberg J, Henderson R, Alden M, et al. OpenVDB: an open-source data structure and toolkit for high-resolution volumes. In: ACM SIGGRApH 2013 courses. 2013, p. 1–19.
[28] Zhao H. Parallel implementations of the fast sweeping method. J Comput Math 2007;421–9.
[29] Zhao H. A fast sweeping method for eikonal equations. Math. Comput. 2005;74:603–27.
[30] Kantorovich LV, I. KV. Approximate methods of higher analysis. Interscience Publishers; 1964.
[31] Rvachev VL, Sheiko TI. R-Functions in boundary value problems in mechanics. Appl Mech Rev 1995;48:151–88.
[32] Rvachev VL, Sheiko TI, Shapiro V, Tsukanov I. On completeness of RFM solution structures. Comput Mech 2000;25:305–17.
[33] Nitsche JA. Über ein variationsprinzip zur lösung von Dirichlet-problemen bei verwendung von teilräumen, die keinen randbedingungen unterworfen sind. Abh Math Semin Univ Hambg 1971;36:9–15.
[34] Embar A, Dolbow J, Harari I. Imposing Dirichlet boundary conditions with Nitsche's method and spline-based finite elements. Internat J Numer Methods Engrg 2010;83:877–98.
[35] Schillinger D, Harari I, M-C H, Kamensky D, Stoter SKF, Yu Y, et al. The non-symmetric Nitsche method for the parameter-free imposition of weak boundary and coupling conditions in immersed finite elements. Comput Methods Appl Mech Engrg 2016;309:625–52.
[36] Saye R. Implicit mesh discontinuous Galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid–structure interaction, and free surface flow: Part I. J Comput Phys 2017;344:647–82.
[37] Saye R. Implicit mesh discontinuous Galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid–structure interaction, and free surface flow: Part II. J Comput Phys 2017;344:683–723.
[38] Gulizzi V, Saye R. Modeling wave propagation in elastic solids via high-order accurate implicit-mesh discontinuous Galerkin methods. Comput Methods Appl Mech Engrg 2022;395:114971.
[39] Rüberg T, Cirak F. Subdivision-stabilised immersed b-spline finite elements for moving boundary flows. Comput Methods Appl Mech Engrg 2012;209:266–83.
[40] Rüberg T, Cirak F, García Aznar JM. An unstructured immersed finite element method for nonlinear solid mechanics. Adv Model Simul Eng Sci 2016;3:1–28.
[41] Burman E, Hansbo P. Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method. Appl Numer Math 2012;62:328–41.
[42] de Prenter F, Verhoosel CV, van Brummelen EH, Larson MG, Badia S. Stability and conditioning of immersed finite element methods: analysis and remedies. Arch Comput Methods Eng 2023;30:3617–56.
[43] Luft B, Shapiro I. Geometrically adaptive numerical integration. In: Proceedings of the 2008 ACM symposium on solid and physical modeling. 2008, p. 147–57.
[44] Bandara K, Rüberg T, Cirak F. Shape optimisation with multiresolution subdivision surfaces and immersed finite elements. Comput Methods Appl Mech Engrg 2016;300:510–39.
[45] Kudela L, Zander N, Kollmannsberger S, Rank E. Smart octrees: Accurately integrating discontinuous functions in 3D. Comput Methods Appl Mech Engrg 2016;306:406–26.
[46] Divi SC, Verhoosel CV, Auricchio F, Reali A, Van Brummelen EH. Error-estimate-based adaptive integration for immersed isogeometric analysis. Comput Math Appl 2020;80(11):2481–516.
[47] Fries T-P, Omerović S, Schöllhammer D, Steidl J. Higher-order meshing of implicit geometries—Part I: integration and interpolation in cut elements. Comput Methods Appl Mech Engrg 2017;313:759–84.
[48] Burstedde C, Wilcox L, Ghattas O. p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. SIAM J Sci Comput 2011;33:1103–33.
[49] Verdugo F, Martín AF, Badia S. Distributed-memory parallelization of the aggregated unfitted finite element method. Comput Methods Appl Mech Engrg 2019;357:112583:1–32.
[50] Badia S, Verdugo F. Robust and scalable domain decomposition solvers for unfitted finite element methods. J Comput Appl Math 2018;344:740–59.
[51] Jomo JN, de Prenter F, Elhaddad M, D'Angella D, Verhoosel CV, Kollmannsberger S, et al. Robust and parallel scalable iterative solutions for large-scale finite cell analyses. Finite Elem Anal Des 2019;163:14–30.
[52] Mandel J, Dohrmann CR. Convergence of a balancing domain decomposition by constraints and energy minimization. Numer Linear Algebra Appl 2003;10:639–59.
[53] Sousedík B, Šístek J, Mandel J. Adaptive-multilevel BDDC and its parallel implementation. Computing 2013;95:1087–119.
[54] OpenVDB webpage. 2024, https://www.openvdb.org. [Accessed May 2024].
[55] Kobbelt LP, Botsch M, Schwanecke U, Seidel HP. Feature sensitive surface extraction from volume data. In: SIGGRApH 2001 conference proceedings. 2001, p. 57–66.
[56] Morton GM. A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing. Technical Report, International Business Machines Company New York; 1966.
[57] Gueziec A, Hummel R. Exploiting triangulated surface extraction using tetrahedral decomposition. IEEE Trans Vis Comput Graphics 1995;1:328–42.
[58] Nielson GM, Hamann B. The asymptotic decider: resolving the ambiguity in marching cubes. In: Proceedings of the 2nd conference on visualization'91. IEEE Computer Society Press; 1991, p. 83–91.
[59] de Prenter F, Verhoosel C, van Zwieten G, van Brummelen E. Condition number analysis and preconditioning of the finite cell method. Comput Methods Appl Mech Engrg 2017;316:297–327, Special Issue on Isogeometric Analysis: Progress and Challenges.
[60] Xiao H, Febrianto E, Zhang Q, Cirak F. An immersed discontinuous Galerkin method for compressible Navier-Stokes equations on unstructured meshes. Internat J Numer Methods Fluids 2019;91:487–508.
[61] Badia S, Martín AF, Neiva E, Verdugo F. The aggregated unfitted finite element method on parallel tree-based adaptive meshes. SIAM J Sci Comput 2021;43:C203–34.
[62] Kůs P, Šístek J. Coupling parallel adaptive mesh refinement with a nonoverlapping domain decomposition solver. Adv Eng Softw 2017;110:34–54.
[63] Dohrmann CR. A preconditioner for substructuring based on constrained energy minimization. SIAM J Sci Comput 2003;25:246–58.
[64] Fragakis Y, Papadrakakis M. The mosaic of high performance domain Decomposition Methods for Structural Mechanics: Formulation, interrelation and numerical efficiency of primal and dual methods. Comput Methods Appl Mech Engrg 2003;192(35):3799–830.
[65] Toselli A, Widlund OB. Domain Decomposition Methods—Algorithms and Theory. Springer, vol. 34, Springer-Verlag; 2005.
[66] Tu X. Three-level BDDC in three dimensions. SIAM J Sci Comput 2007;29:1759–80.
[67] Mandel J, Sousedík B, Dohrmann CR. Multispace and multilevel BDDC. Computing 2008;83:55–85.
[68] Amestoy PR, Buttari A, L'Excellent J-Y, Mary TA. Bridging the gap between flat and hierarchical low-rank matrix formats: The multilevel block low-rank format. SIAM J Scient Comput 2019;41(3):A1414–42.
[69] Park JJ, Florence P, Straub J, Newcombe R, Lovegrove S. Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 165–74.
[70] Balu A, Rajanna MR, Khristy J, Xu F, Krishnamurthy A, Hsu M-C. Direct immersogeometric fluid flow and heat transfer analysis of objects represented by point clouds. Comput Methods Appl Mech Engrg 2023;404:115742.