Journal of Computational Physics 334 (2017) 327–348

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

# An interface-fitted mesh generator and virtual element methods for elliptic interface problems ☆

Long Chen [b,a], Huayi Wei [c,a,*], Min Wen [b]

[a] Beijing Institute for Scientific and Engineering Computing, Beijing University of Technology, Beijing, 100124, China
[b] Department of Mathematics, University of California at Irvine, Irvine, CA 92697, USA
[c] School of Mathematics and Computational Science, Xiangtan University, Xiangtan, Hunan, 411105, China

ARTICLE INFO

*Article history:*
Received 25 July 2016
Received in revised form 16 December 2016
Accepted 5 January 2017
Available online 10 January 2017

*Keywords:*
Elliptic interface problem
Interface-fitted mesh
Delaunay triangulation
Semi-structured
Virtual element method

ABSTRACT

A simple and efficient interface-fitted mesh generation algorithm which can produce a semi-structured interface-fitted mesh in two and three dimensions quickly is developed in this paper. Elements in such interface-fitted meshes are not restricted to simplices but can be polygons or polyhedra. Especially in 3D, the polyhedra instead of tetrahedra can avoid slivers. Virtual element methods are applied to solve elliptic interface problems with solutions and flux jump conditions. Algebraic multigrid solvers are used to solve the resulting linear algebraic system. Numerical results are presented to illustrate the effectiveness of our method.



## 1. Introduction

We consider finite element methods for solving elliptic interface problems which have a variety of applications in different research fields, including fluid dynamics, material science, and biological systems, etc. [20,47,55,62]. The importance of the coupling of the complex geometry of the interface with the numerical methods has been recognized and received rapidly increasing interest in recent years.

Let $\Omega$ be an open and bounded domain in $\mathbb{R}^d$ $(d = 2, 3)$, and $\Gamma$ be a continuous interface embedded in $\Omega$. The interface $\Gamma$ separates the domain $\Omega$ into disjoint regions $\Omega^+$ and $\Omega^-$, where $\Omega^+$ denotes the exterior domain and $\Omega^-$ is the interior domain enclosed by $\Gamma$. We consider numerical methods for solving the following elliptic interface problems:

$$-\nabla \cdot (\beta(x)\nabla u(x)) = f(x), \quad x \in \Omega \backslash \Gamma \tag{1}$$

with prescribed jump conditions across the interface $\Gamma$:

$$[u]_\Gamma = u^+ - u^- = q_0, \tag{2}$$

$$[\beta u_n]_\Gamma = \beta^+ u_n^+ - \beta^- u_n^- = q_1, \tag{3}$$

☆ The first author was supported by the National Science Foundation (NSF) DMS-1418934 and in part by the Sea Poly Project of Beijing Overseas Talents. The second author was supported by the NSFC, i.e., National Natural Science Foundation of China (Grant No. 11301449 and No. 91430213), and in part by the Specialized Research Fund for the Doctoral Program of Higher Education (Grant No. 20134301120003). The third author was supported by the National Science Foundation (NSF) DMS-1418934.

* Corresponding author at: School of Mathematics and Computational Science, Xiangtan University, Xiangtan, Hunan, 411105, China.
*E-mail addresses:* chenlong@math.uci.edu (L. Chen), weihuayi@xtu.edu.cn (H. Wei), mwen2@uci.edu (M. Wen).

http://dx.doi.org/10.1016/j.jcp.2017.01.004
0021-9991/

and boundary condition:

$$u = g \quad \text{on } \partial\Omega. \tag{4}$$

Here $u_n$ denotes the normal derivative $(\nabla u)\cdot n$ with $n$ being the unit norm direction of the interface $\Gamma$ pointing outward (from $\Omega^-$ to $\Omega^+$). The superscripts $+$ and $-$ stand for the restriction of a function on $\Omega^+$ and $\Omega^-$, respectively. The diffusion coefficient $\beta(x)$ is assumed to be uniformly positive and smooth on each subdomain, but may be discontinuous across the interface. Because of that, the solution $u$ is piecewise smooth but the global regularity is low [27,48,49].

Numerical methods for elliptic interface problems can be roughly classified into two categories by using either an interface-fitted (also known as body-fitted or interface conforming) mesh or an unfitted mesh (e.g. a uniform Cartesian mesh) in the discretization of the domain. In the unfitted mesh approach, a popular way to enforce the jump conditions is to modify the finite difference stencils or the finite element basis near the interface. A lot of numerical methods in this direction have been proposed such as the immersed boundary method [74], the immersed interface method [58,59], immersed finite element methods [37,44,54,63], ghost fluid methods [65], matched interface and boundary (MIB) methods [83,87,90], multiscale finite element methods [27], extended finite element methods (XFEM) [36,68,69], and many others [39,46,45,50,64,82]. The jump condition can be also imposed based on the Nitsche's method [72] by introducing penalty terms across interfaces, see, for example, the earlier work by Babuška [5], Barrett and Elliott [7], unfitted FEM by Hansbo and Hansbo [41], *hp*-discontinuous Galerkin method [67], CutFEM [17,43], and many others [8,17–19,41–43,53,78,79]. The most attractive feature of the unfitted mesh approach is the easiness of the mesh generation. Indeed, if the background mesh is Cartesian, there is no need of meshing which is very convenient, especially when the interface is moving in time.

On the other hand, using unfitted mesh approach, it is difficult to capture the complex geometry of the interface and to enforce jump conditions across the interface accurately, and the resulting linear system may not be always symmetric which could cause problems for fast solvers. Furthermore a rigorous error analysis is difficult. Recent progress on immersed finite element methods can be found in [40,88].

In this work, we focus on the interface-fitted mesh approach. Provided a mesh fitted to the interface, one can use conforming finite element methods and get a symmetric system which can be solved efficiently by fast solvers such as algebraic multigrid method. Rigorous error analysis is possible. Optimal a priori estimates of linear finite element are given in [86,14,24] and in [60] for high order finite elements. Recent work using hybridized discontinuous Galerkin (HDG) [51] and weak Galerkin (WG) [71] method is also based on a shape regular and body-fitted triangulation. The challenge of this approach is quickly generating an interface-fitted mesh, especially in three dimensions (3D), which is the topic of this study.

There is a lot of work on the unstructured interface-fitted mesh generation [66,73,89]. The unstructured mesh generator is, however, time consuming as it needs to modify the mesh for the whole domain, not just near the interface. For example, extensive and non-trivial computational effort is needed to generate a high quality 3D finite element mesh from biomedical image data or geological image data etc. [4,28].

We are interested in the semi-structured and body-fitted mesh generation methods [11,13,76] and will develop a simple and effective mesh generation algorithm. As an illustrative example, to generate an interface-fitted mesh in two dimensions (2D), we start from a uniform Cartesian mesh with $N$-mesh points, and apply three steps: 1) find all the intersection points, the mesh points near the interface, and add few auxiliary points; 2) generate a Delaunay triangulation of these points; 3) remove the unnecessary triangles and merge the regular meshes away from the interface. The resulting triangulations can preserve the interface and the maximal angle is bounded by 135°. Since the Delaunay triangulations are only constructed on a local region near the interface, the dominant cost is reduced to $\mathcal{O}(N^{1/2}\log N)$. Due to the semi-structured mesh and localization near the interface, some nice properties of structured mesh are still preserved such as superconvergence in the energy norm and fast convergence of algebraic multigrid methods [80].

The main restriction of this approach is the quality of the generated mesh especially in 3D. Most finite element methods require discretizing a domain into a set of shape regular tetrahedra in three dimensions. The accuracy of the simulations and the efficiency of the solvers could deteriorate by the presence of badly-shaped elements. The problematic tetrahedra are so-called slivers, which are a type of flat tetrahedra without small edges, but with nearly zero volume. Namely, four vertices of a sliver are almost coplanar. Due to the presence of slivers, three-dimensional mesh generation is much harder than the two-dimensional case, and removing slivers from a 3D tetrahedral mesh is one of the major tasks in the field of mesh generation [32,61,70].

We propose a new way to solve this difficulty. We choose polyhedral meshes rather than tetrahedral meshes. Then silvers will be merged into nearby polyhedra. The shape of the polyhedron or other tetrahedron could be still degenerate but the maximal angle is bounded uniformly away from $\pi$. Notice that finite element approximation retains accurate if the maximal angle condition [6] is satisfied. Namely tetrahedral with small volumes are allowed as long as the four vertices are non-planar [30,56]. Similar results can be established for polyhedral meshes and theoretical justification will be reported somewhere else.

Another difficulty is encountered in the implementation. Due to the large number of possible intersections between the fixed mesh and the interface, a variety of interface-cells are generated leading to an equally large number of treatments, which could result in complex coding logistics; see [75,76].

We propose an all-in-one solution. The connectedness of intersection points is obtained by the Delaunay algorithm which is a well developed algorithm in computational geometry and efficient implementation is available in many software packages. Our mesh generation algorithm in 3D is similar to the 2D case only different in step 3: post-processing. The

additional work is to merge tetrahedra into polyhedra. To facilitate the merging, the polyhedra are stored in the form of faces and the index of the elements to which the faces belong. The resulting mesh retains the following nice properties: the interface is approximately preserved, the maximal angle condition is satisfied, and cost-efficient. The Delaunay algorithm is only called for points near the interface and thus the dominated cost is reduced to $\mathcal{O}(N^{2/3})$ which is considerably smaller comparing with $\mathcal{O}(N)$ assembling and solving of the linear algebraic system. The quality and efficiency of our mesh generation algorithm are balanced and suitable for the finite element simulation. No additional mesh smoothing process is needed in our algorithm. Of course, adding such a mesh smoothing process will furthermore improve the quality of the mesh and probably improve the accuracy of the finite element approximation. However, it will destroy the structure of the mesh. In our mesh generator, the background mesh is fixed. The Delaunay algorithm can be called element by element and thus local modification is possible if only part of the interface is changed. These features are important for moving interface problems, which will be explored in our future work.

A similar mesh generation approach was introduced in [38], where the authors introduced the Voronoi diagrams and Delaunay triangulation of a point set of a surface and more focused on the surface mesh generation. Our algorithm seems simpler and more suitable for finite element simulation as we shall discuss below.

Since elements in such interface-fitted meshes are general polyhedra, we shall apply virtual element methods (VEM) [9,10], which can be considered as an extension of conforming finite element methods to polyhedral meshes. The resulting linear algebraic system is symmetric and positive definite and thus can be solved efficiently using algebraic multigrid solvers. Furthermore, according to our mesh generation algorithm, we will get polyhedra with triangular and square faces which will be much easier to assemble the matrices in VEM compared to the original approach in [10]. Optimal second order of convergence in the $L^2$ and $L^\infty$ norms and a superconvergence of energy norm is observed in several numerical examples.

An outline of the paper is as follows. In Sections 2 and 3, we present the mesh algorithm for the generation of interface-fitted meshes in two and three dimensions, respectively. In Section 4, we derive the weak formulation of the elliptic interface problems and discuss linear virtual element methods on all elements. In Section 5, we provide numerical results to show the effectiveness of our method. We end with several concluding remarks and future work.

## 2. Interface-fitted mesh generator: two dimensions

In this section, we introduce our interface-fitted mesh generator in 2D. We first describe the algorithm and then give two examples to illustrate the algorithm. In addition, we prove the generated mesh will preserve the interface approximatly and satisfy the maximal angle condition.

### 2.1. Algorithm

Let $\Gamma$ be an interface embedded in a rectangular domain $\Omega$. Assume $\Gamma$ can be represented by the zero-level set of a function $\phi(x)$, i.e., $\Gamma = \{x \in \Omega : \phi(x) = 0\}$. The interface $\Gamma$ separates $\Omega$ into subdomains $\Omega^+ := \{x \in \Omega : \phi(x) > 0\}$ and $\Omega^- := \{x \in \Omega : \phi(x) < 0\}$. Note that $\Omega^-$ could have multiple connected components when $\Gamma$ consists of two or more closed curves.

One can easily generate a uniform Cartesian mesh $\Omega_h$ of $\Omega$ with a given mesh size $h$. A vertex $p$ of $\Omega_h$ is said to be inside if $\phi(p) < 0$, outside if $\phi(p) > 0$, or on $\Gamma$ if $\phi(p) = 0$; an edge $(p_1, p_2)$ is called a cut edge if $\phi(p_1)\phi(p_2) < 0$; the point which the cut edge intersects with $\Gamma$ is called an intersection point; a square element $K$ of $\Omega_h$ which intersects with the interface $\Gamma$, i.e. $|\bar{K} \cap \Gamma| \neq \varnothing$, is called an interface element. We can find interface elements by using one of the following two rules:

(1) There exists at least two vertices $p$ and $q$ with opposite sign, i.e., $\phi(p)\phi(q) < 0$;
(2) There exists at least two vertices on the interface, namely the value of $\phi$ on these vertices is 0.

These two rules could detect all the interface elements in Fig. 1 except case (3), which could be avoided by choosing the initial mesh size $h$ small enough. For disconnected interfaces (cases (6)–(9)), we assume it is described by two level set functions (cf. Example 2.2), and the intersection points can be found by treating each level set function one by one. We remark that it is much more difficult to modify stencils or the basis for such cases. In general, the modified finite difference stencils or modified finite element basis near the interface is to introduce additional but local degrees of freedom near the interface and then use the jump conditions to eliminate these degree of freedom by solving a small linear system elementwise [2,12,35,52,69,81,84]. In almost all of these work, it is assumed the intersection meets the edges of an interface element at no more than two intersections and intersects at different edges for one element, cf. [44,63,64,88]. If this condition is violated, such as those cases (6)–(9) in Fig. 1, the local system will be much more involved since it depends on how the interface cuts the elements.

We define the *interface points* as the collection of intersection points, vertices of interface elements, and some auxiliary points explained below. When the intersection points are diagonal, we need to add the midpoints of corresponding elements, which are called auxiliary points.

Recall that a Delaunay triangulation for a set of points $P$ in a plane is a triangulation of the convex set of $P$ such that no point in $P$ is inside the circumcircle of any triangle in this triangulation [32,57].

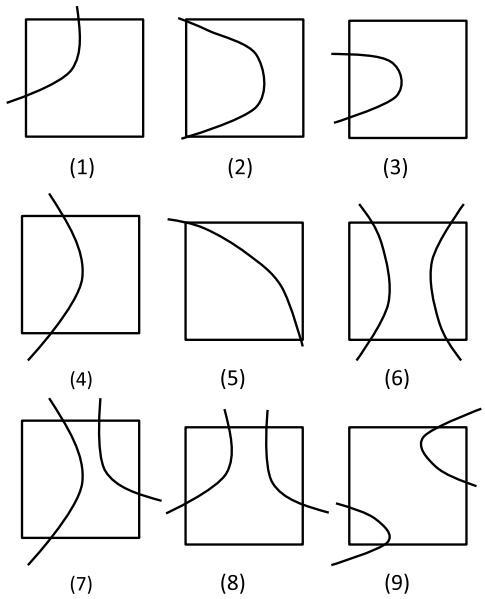



**Fig. 1.** Example of interface elements: (1)–(5) with one level set function and (6)–(9) with two level set functions.

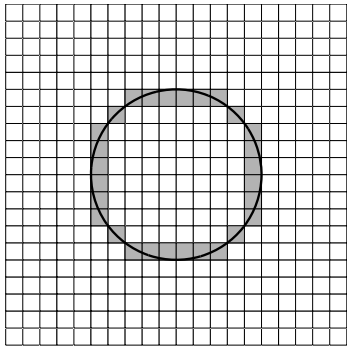


**Fig. 2.** Cartesian mesh $\Omega$ and a circle interface $\Gamma$. The grayed elements are interface elements.

Our 2D interface-fitted mesh generation algorithm is described as follows:

**Algorithm:** 2D interface-fitted mesh generation algorithm

**Input:** Mesh size, $h$, level set function, $\phi(x)$ and square domain, $\Omega$;
**Output:** An interface-fitted mesh of $\Omega$;
1. Find all the interface points.
2. Construct a Delaunay triangulation of these points.
3. Remove triangles not in the interface elements and merge all uncut elements.

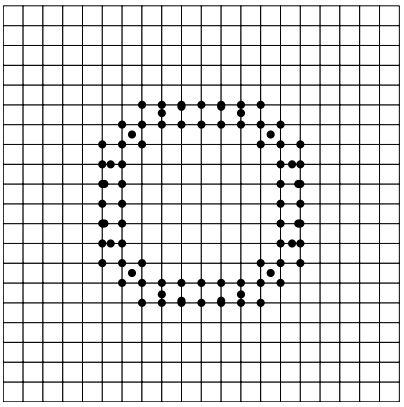
**Algorithm 1:** 2D mesh generator.

### 2.2. Examples

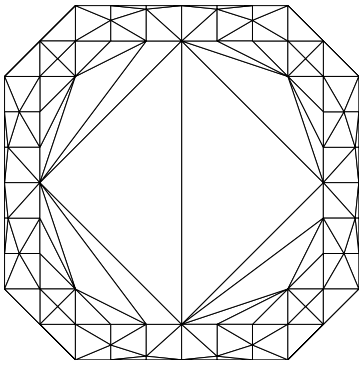
We give two examples to explain Algorithm 1 in detail. The first example shows the simple case when the interface is a circle. The second example illustrates a more complex case when the interface is unconnected and some interface elements are divided into three parts.

**Example 2.1** *(A circle).* Consider the domain $\Omega = (-1, 1)^2$ and a circle interface $\Gamma$ represented by the level set function $\phi(x, y) = x^2 + y^2 - r^2$, with $r = 0.5$. The interface elements are shown in Fig. 2.

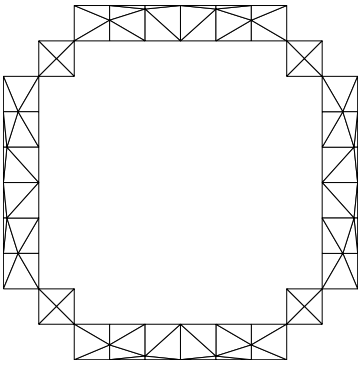
First, we construct a point set $\mathcal{P}$ which includes the intersection points between cut edges and $\Gamma$, the vertices of all interface elements, and some auxiliary points. See Fig. 3(a) for the illustration. Here we use the bisection method to compute the intersection points within the machine precision tolerance.



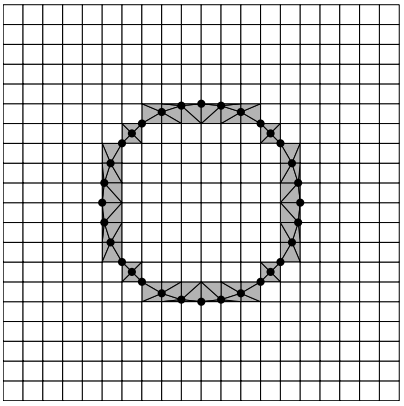
(a) Step 1: Find all the interface points.



(b) Step 2: a Delaunay triangulation of interface points.



(c) Step 3: Remove triangles not in the interface elements.



(d) Step 3: Merge all uncut elements to get an interface-fitted mesh.

**Fig. 3.** Three steps to generate an interface-fitted mesh.

Then we construct a Delaunay triangulation based on the point set $\mathcal{P}$. In MATLAB, we just call `DT = delaunay(x,y)` (see Fig. 3(b)).

In the last step, we keep the triangles in interface elements and merge the uncut elements to get the final interface-fitted semi-structured mesh in Fig. 3(c)–(d).

**Example 2.2** *(Two circles).* Consider the domain $\Omega = (-1, 1)^2$ and the unconnected interface $\Gamma$ represented by the level set function

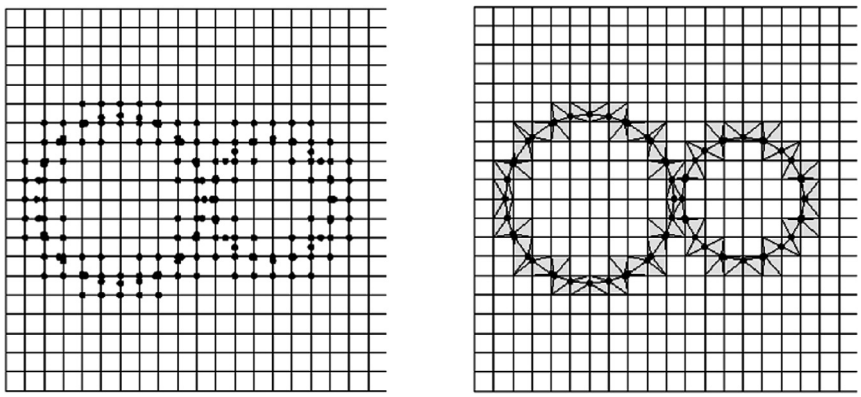
$$\phi(x, y) = \min\left\{(x+r)^2 + y^2 - (1.1r)^2,\ (x-r)^2 + y^2 - (0.8r)^2\right\},$$

with $r = 0.4$. We can apply the same algorithm and obtain the mesh in Fig. 4. The only difference is when computing intersection points, we compute them for each level set function separately. We use this example to show that our algorithm can handle unconnected interfaces.

### 2.3. Properties

We explore properties of the mesh obtained in Algorithm 1. A triangle is called an interior element when the barycenter of the triangle is inside. The interface $\Gamma$ could be approximated by the boundary of those interior elements and can be extracted easily. The obtained discrete interface is denoted by $\Gamma_h$.

**Proposition 2.3.** *The interface will be approximately recovered in the triangulation generated by Algorithm 1. More precisely, we have* $\mathrm{dist}(\Gamma_h, \Gamma) \lesssim h^2$ *provided* $\Gamma$ *is smooth enough and* $h$ *is small enough.*



(a) Interface points.

(b) The interface-fitted mesh when the interface is two disjoint circles.

**Fig. 4.** Interface points and interface-fitted meshes when the interface is unconnected.

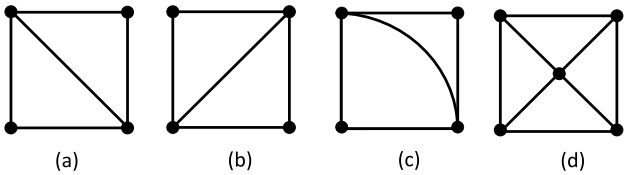



**Fig. 5.** Add one auxiliary point when two intersection points are diagonal.

**Proof.** We shall use another characterization of Delaunay triangulations: a Delaunay triangulation is the projection of the lower convex hull of points lifted to the paraboloid $f(\vec{x}) = \|\vec{x}\|^2$ [16,23,34].

The function values of $f(\vec{x})$ on the four vertices of a square will be on a plane. As the function $f$ is strictly convex, the function value of any intersection points which are different from the vertices of the square will be below this plane. Then the lower convex hull when lifted to $\mathbb{R}^3$ will always connect the intersection points. Thus, the interface will be recovered under this circumstance.

If there are two diagonal vertices of a square on the interface $\Gamma$ (see Fig. 5(c)), then the Delaunay triangulation on this square is not unique. Using either diagonal of the square is a valid Delaunay triangulation (see Fig. 5(a) and (b)). Therefore, we introduce the center of this square as an auxiliary point to make sure the interface is preserved (see Fig. 5(d)).

In both cases, $\Gamma_h$ contains a piecewise affine approximation of $\Gamma$ with nodes on the interface and thus the distance is in the order of $Ch^2$ with constant $C$ depends on the curvature of $\Gamma$. □

**Proposition 2.4.** *Assume the mesh size h is small enough such that the interior of each edge has at most one intersection point. Then the maximal angle of the triangulation generated by Algorithm 1 is bounded by* 135°.

**Proof.** Let $\mathcal{C}$ be a square with vertices $A, B, C, D$ which intersects with the interface, $\mathcal{S}$ the points set including the vertices of $\mathcal{C}$ and the intersection points, and $DT$ the Delaunay triangulation of $\mathcal{S}$.

The vertex of every angle in $DT$ can be a vertex of the square or an intersection point. The angle at a square vertex must be bounded by 90° as the two rays of the angle is inside the square. Next, let us prove that the angle at an intersection point must be bounded by 135°. Let $E$ be an intersection point on edge $AB$, $F$ and $G$ are the other two points of angle $\angle FEG$ and $G$ is on the right of $F$ (see Fig. 6). Here $F$ or $G$ can be an intersection point or a vertex of the square.

By our assumption, $F$ or $G$ cannot be in the interior of edge $AB$ and $F$ and $G$ cannot be in the interior of edge $CD$ simultaneously. So either $F$ is on the edge $AD$ or $G$ is on $BC$. Without loss of generality, we assume $G$ is on $BC$. Then the angle $\angle FCG \geq 45°$ since $F$ is on the left of the diagonal $AC$. Note that the triangle $\triangle FCG$ may not be in the $DT$. Nevertheless, if $\angle FEG > 135°$, then $\angle FEG + \angle FCG > 180°$ which means the circumcircle of $\Delta FEG$ must include vertices $C$ violating the Delaunay property. So $\angle FEG$ must be bounded by 135°. □

Let $N$ be the number of nodes. Since we restrict the Delaunay triangulation on a local region near the interface, the complexity of generating a Delaunay triangulation will be $\mathcal{O}(N^{1/2}\log N)$ in 2D which can be ignored compared with the $\mathcal{O}(N)$ complexity of assembling the matrix and solving the matrix equation. Such localization will make it possible to track the moving interface, which will be explored in our future work.

The overall complexity of our mesh generation algorithm is: $c_1 N + c_2 N^{1/2}\log N$ since we need to compute the sign of the level set function at $N$ vertices. In practice, however, the constant $c_1 \ll c_2$ and the time scales like $\mathcal{O}(N^{1/2})$.

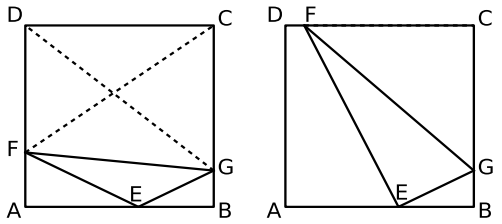



**Fig. 6.** The angle $\angle FEG$ at the intersection point $E$.

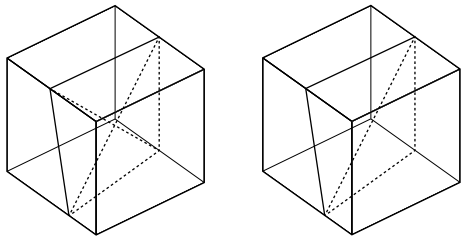


**Fig. 7.** Sliver exists (left) and is removed when the element is divided into polyhedra (right).

## 3. Interface-fitted mesh generator: three dimensions

In this section, we present a novel mesh generation algorithm to generate an interface-fitted mesh for a given smooth interface in three dimensions. We begin with a brief review on the difficulty of 3D mesh generation and then introduce our algorithm to overcome this difficulty.

### 3.1. Main difficulty

Tetrahedral meshes are frequently used in classical finite element methods. The size and shape of the tetrahedra influence the accuracy of finite element solutions [77]. The quality of the tetrahedron's shape can be measured by using the aspect ratio or the radius-edge ratio. The aspect ratio of a tetrahedron is usually defined as its circumradius divided by its inradius and the radius-edge ratio is the circumradius divided by the shortest edge length of the tetrahedron. The aspect ratio or radius-edge ratio of a mesh is the largest corresponding ratio of all of its tetrahedral elements. If the aspect ratio or radius-edge ratio of a mesh are small, we called the mesh well-shaped [32,61]. Ideally we expect each element in the mesh is shape regular. But violation is allowed as long as the so-called maximal angle condition is satisfied [1,6,15,31,56].

The difficulty of mesh generation in three dimensions is due to the existence of slivers. Slivers have small radius-edge ratio, but large aspect ratio, which are considered as bad-shaped tetrahedral elements. The results of the accuracy and convergence of finite element methods may not hold anymore in the existence of slivers which violates the maximal angle condition. A lot of methods have been developed to remove slivers; see e.g. [25,26,33]. Sliver removal methods, however, involve the addition and rearrangement of points and thus destroy the semi-structure of the mesh.

We shall introduce polyhedral meshes near the interface to remove slivers. When we get the interface elements (which is defined similarly to 2D and will be made clear later) and intersection points (the definition is the same as in 2D), we can divide interface elements into polyhedra instead of tetrahedra. Slivers will be eliminated and part of their faces will become the faces of polyhedral elements. For example, when the interface cuts across one element with four almost coplanar intersection points, if we divide the element into tetrahedra, then the four intersection points could form a sliver (see Fig. 7). If we use a polyhedral mesh, however, the two well-shaped triangles will become the boundary of two polyhedra.

### 3.2. Algorithm

We write down the algorithm and explain the details step by step.

**Algorithm:** 3D interface-fitted mesh generation algorithm

**Input:** Mesh size $h$, level set function $\phi(x)$, and a cubic domain $\Omega$;
**Output:** Interface-fitted mesh $\Omega$;
1. Find all the interface points.
2. Construct the Delaunay mesh $DT$ on these points.
3. Post processing: remove unnecessary tetrahedra in $DT$, merge tetrahedra into polyhedra, and merge with uncut elements.

**Algorithm 2:** 3D mesh generator.

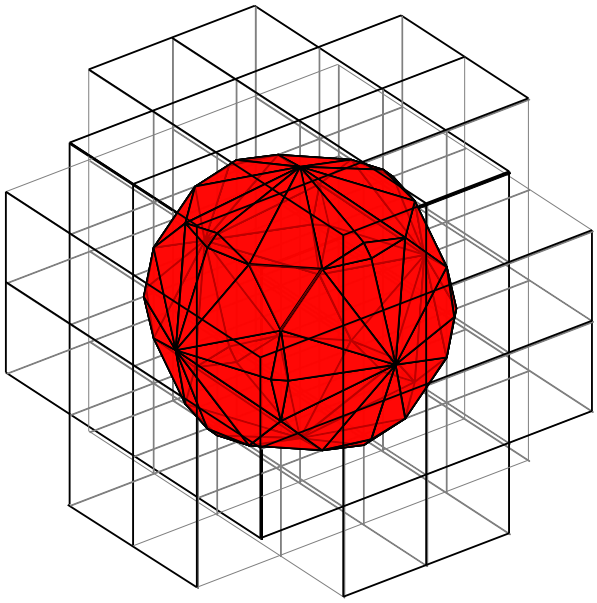


**Fig. 8.** The surface of the interface is embedded in the hexahedron.

Given a cubic domain $\Omega$ which includes the interface $\Gamma$ described as the zero level set of $\phi$, and a mesh size $h$, we first generate the uniform Cartesian mesh of $\Omega$ with size $h$. The cubes in the Cartesian mesh in domain $\Omega$ could be classified into exterior, interior, and interface elements by checking the sign of the centers of the cubes. We label them by 1, $-1$ and 0 respectively.

We define interface elements as elements satisfying one of the following rules:

(1) There exists at least two vertices $p$ and $q$ with opposite sign, namely $\phi(p)\phi(q) < 0$;
(2) There exists at least three vertices on the interface.

All interface elements will form a hexahedral mesh of a layer of the interface (see Fig. 8). All boundary faces of this hexahedral mesh are extracted and will be used as faces of the polyhedral mesh for the interface. Note that these boundary faces are square faces.

In step 1, similarly to two dimensions, we find cut edges and intersection points, and add auxiliary points if necessary. The criterion of adding auxiliary points is the same as 2D: if a square face contains two opposite vertices on the interface, we will add the center point of this square face as the auxiliary point.

In step 2, we generate a Delaunay mesh $DT$ of $\mathcal{P}$, the set of interface points, whose definition is the same as that in two dimensions.

In step 3, we post-process $DT$ to get a polyhedral mesh near the interface and merge all uncut cubic elements away from the interface.

Similar to the 2D case, we only keep tetrahedra inside the interface elements, which might contain slivers, and remove tetrahedra not in the interface elements which can be easily marked by checking the center of tetrahedra in $DT$.

Now we have a tetrahedral mesh of all the interface elements, and we still call this tetrahedral mesh as $DT$ for convenience. We could split the tetrahedron in $DT$ into two categories: exterior tetrahedral set $DT_E$ and interior tetrahedral set $DT_I$. For a tetrahedron in $DT$, if the minimum of the sign function of the $\phi$ value of the four vertex nodes is $-1$, we put it into $DT_I$, otherwise, we add it into $DT_E$. The interface $\Gamma$ could be extracted using the boundary faces of $DT_I$ and the normal direction of the extracted surface mesh points outside of the interface. Tetrahedra in each category will be merged into polyhedra element by element.

Instead of storing all vertices of a polyhedron, we shall store the polyhedral mesh by the data structure `face` and `face2elem`. The array `face` records indices of three (triangular face) or four (square face) vertices of all faces. The direction of all faces follows the right-hand side rule, that is, the normal direction of each face is outwards. The array `face2elem` records the index of the polyhedron to which the faces belong.

Fig. 9 is a simple example. Given a unit cube with three intersection points, it is divided into two polyhedra. Each polyhedron is stored by faces and elements to which they belong. The values of `face` and `face2elem` in Fig. 9 are shown in Table 1.

Notice that some face, e.g., `[3 7 8 4]` is stored as a square instead of two triangles since this face is shared by another cube which is not included as an interface element. Those square faces are boundary faces of the hexahedral mesh which consists of all interface elements. All such square faces have been extracted when we collect all interface elements.

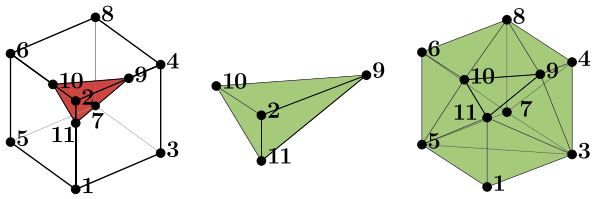



**Fig. 9.** An interface element is divided into two polyhedra.

**Table 1**
The `face` array (left) and `face2elem` (right) for two polyhedra in Fig. 9.

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 9 | 10 | |
| 2 | 2 | 11 | 9 | |
| 3 | 2 | 10 | 11 | |
| 4 | 11 | 10 | 9 | |
| 5 | 11 | 1 | 3 | |
| 6 | 5 | 1 | 11 | |
| 7 | 3 | 4 | 9 | |
| 8 | 3 | 9 | 11 | |
| 9 | 9 | 4 | 8 | |
| 10 | 5 | 10 | 6 | |
| 11 | 5 | 11 | 10 | |
| 12 | 10 | 8 | 6 | |
| 13 | 10 | 9 | 8 | |
| 14 | 11 | 9 | 10 | |
| 15 | 1 | 5 | 7 | 3 |
| 16 | 5 | 6 | 8 | 7 |
| 17 | 3 | 7 | 8 | 4 |
| | 1 | 2 | 3 | 4 |

| | |
|---|---|
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 4 | 1 |
| 5 | 2 |
| 6 | 2 |
| 7 | 2 |
| 8 | 2 |
| 9 | 2 |
| 10 | 2 |
| 11 | 2 |
| 12 | 2 |
| 13 | 2 |
| 14 | 2 |
| 15 | 2 |
| 16 | 2 |
| 17 | 2 |
| | 1 |

Every interface element is divided into several polyhedra according to our method. For each polyhedron, we need to assign a unique index. This index map from face to element, `face2elem`, can be generated in two stages. In most cases, the interface element is just divided into two polyhedra. In the first stage, for the interior part, we use the original interface element index $j$ and for the exterior part, we append a new index $j + N$, where $N$ is the number of elements in the initial Cartesian mesh. In some cases, however, one cube could be divided into three or more polyhedra (see the three cases in Fig. 10). In the second stage, we use Euler's formula to check the connectedness of the obtained polyhedral mesh. If a disconnected polyhedron is found, we group faces into different connected components which is equivalent to dividing the original polyhedron into more polyhedra. Thanks to our data structure, we only need to change `face2elem` when adding and storing the new polyhedra.

In a nutshell, we could get a polyhedral mesh near the interface by storing the triangular and square faces. The final interface-fitted mesh consists of polyhedra near the interface and uncut (cube) elements away from the interface.

### 3.3. Properties

The generated Delaunay triangulation will approximately recover the interface by the lifting method. Namely Proposition 2.3 also holds for the 3D case since the characterization of a Delaunay triangulation as the projection of the lower convex hull holds in general dimensions. We formally summarize below.

**Proposition 3.1.** *The interface will be approximately recovered in the triangulation generated by Algorithm 2. More precisely, we have* $\operatorname{dist}(\Gamma_h, \Gamma) \lesssim h^2$ *provided* $\Gamma$ *is smooth enough and* $h$ *is small enough.*

Next we shall show the maximum angle of the surface mesh is uniformly bounded by 144°. In [75], the author considers 12 types of subdivision of boundary cells (not necessarily satisfying the Delaunay property) in three dimensions and shows the same bound.

**Proposition 3.2.** *The maximal angle of the triangular faces of the polyhedral mesh is bounded by* 144°.

**Proof.** For simplicity, let $\mathcal{C}$ be a unit cube which intersects with the interface, and $\mathcal{S}$ the set of points including the eight cube vertices and the intersection points. Let $DT$ be the 3D Delaunay triangulation on $\mathcal{S}$.

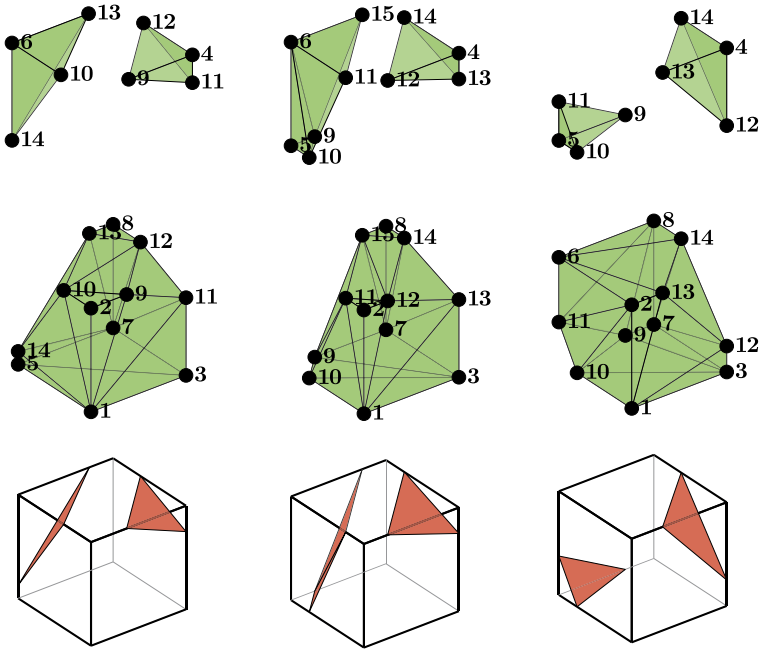


**Fig. 10.** A cube is divided into three parts.

For a 3D Delaunay triangulation, it satisfies the Delaunay empty sphere property such that no point in $\mathcal{S}$ is inside the circumsphere of any tetrahedron of $DT$. Given a tetrahedron $T$ in $DT$ which has a triangular face $\tau$ on one (denoted as $\mathcal{F}$) of the six square faces of $\mathcal{C}$. Since, on the plane spanned by $\mathcal{F}$, the circumcircle of $\tau$ is also on the circumsphere of $T$, then by Delaunay empty sphere property, there is no point of $S$ on $\mathcal{F}$ which is inside the circumcircle of $\tau$, namely, the boundary triangulation of $DT$ on $\mathcal{F}$ is also Delaunay, and thus the maximal angle of these triangles is bounded by 135° by Proposition 2.4.

Next we only need to consider the interface triangles with three vertices on the interface. For these interface triangles, their angles can be divided into 16 cases (see Fig. 11).

In case (1) to (15), one can find the upper bound of the angle by calculus analysis. Here we take the case (1) as an example to show how to find the upper bound, see Fig. 11 (1). Let $v_{LA}$ be the vector from point $L$ to point $A$ and $|v_{LA}|$ the length of $v_{LA}$. Similarly, we have vectors $v_{LH}, v_{LN}, v_{NA}, v_{NH}$, then

$$\begin{aligned}\cos\angle ALH &= \frac{v_{LA}\cdot v_{LH}}{|v_{LA}||v_{LH}|} = \frac{(v_{LN}+v_{NA})\cdot(v_{LN}+v_{NH})}{\sqrt{|v_{LN}|^2+|v_{NA}|^2}\sqrt{|v_{LN}|^2+|v_{NH}|^2}}\\ &= \frac{|v_{LN}|^2}{\sqrt{|v_{LN}|^2+|v_{NA}|^2}\sqrt{|v_{LN}|^2+|v_{NH}|^2}} \geq 0.\end{aligned}$$

When $|v_{LN}| = 0$, $\cos\angle ALH$ reaches the minimum value zero, namely, the maximum of $\angle ALH$ is 90°. By the similar method, one can get the upper bounds for other cases except case (16) in Fig. 11.

In case (16), provided $\triangle ALG$ is an interface triangle and $\angle ALG$ is the angle bigger than 144°. By Algorithm 2, there must exist a vertex of $\mathcal{C}$, for example, vertex $Q$ and $ALGQ$ is a tetrahedron in the Delaunay triangulation. Let $(1-h_1, 0, 0), (1, 0, h_2)$ and $(1, h_3, 1)$ be the coordinates of $A, L$ and $H$, respectively. Then one can get the circumcenter $O$ and circumradius $r$ of the circumsphere of $ALGQ$, then construct function $f(h_1, h_2, h_3) := r - |P - O|$. By the assumption $\angle ALG > 144°$ and the 2D Delaunay empty circle property on the boundary face of $\mathcal{C}$, one can show that $f(h_1, h_2, h_3) > 0$, namely $P$ is inside of the circumsphere of tetrahedron $ALGQ$, which contradicts with the Delaunay empty sphere property. □

**Remark 3.3.** For the proof of the 3D angle case (16), we use the *region_plot* function in SageMath [29] to show $f(h_1, h_2, h_3) > 0$ under the given assumptions.

**Remark 3.4.** We emphasize that the 16 cases plotted in Fig. 11 are used to prove the maximal angle condition. In the algorithm, we get the mesh by directly calling Delaunay algorithm with all interface points as input.

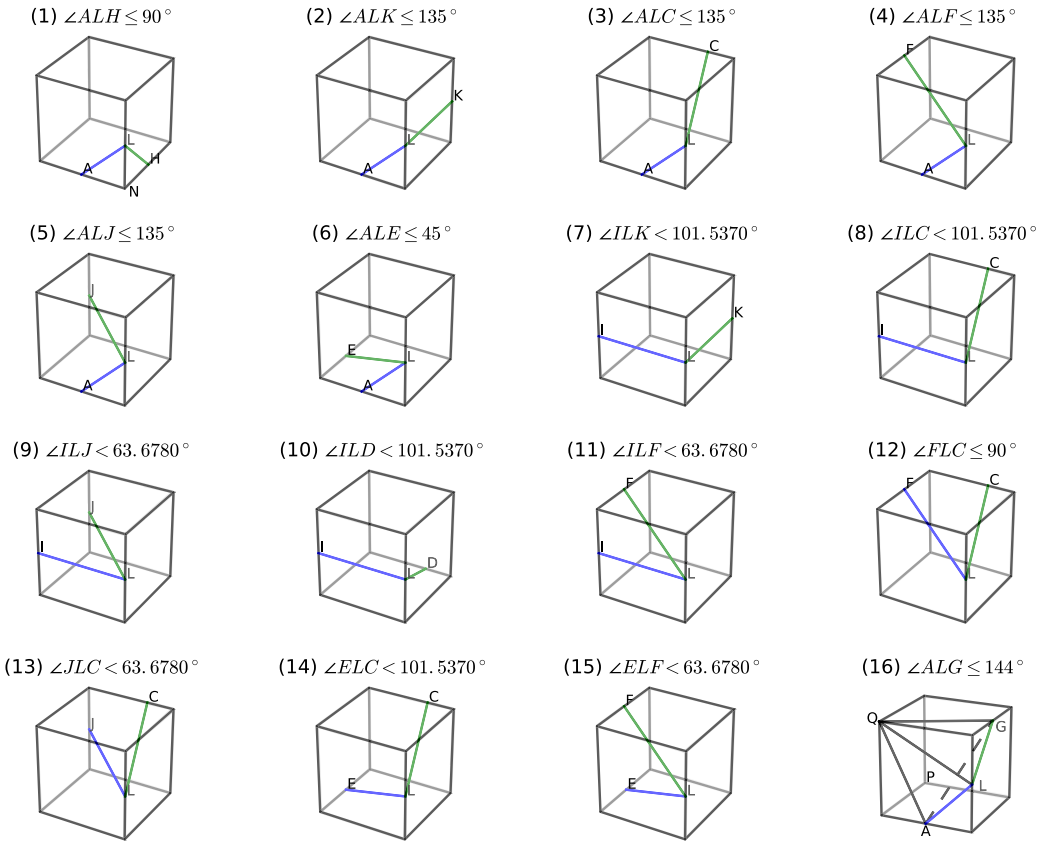



**Fig. 11.** Different angle cases in the interface triangles.

Again, we restrict the Delaunay triangulations on a local region near the interface. The overall complexity of our mesh generation algorithm is: $c_1 N + c_2 N^{2/3} \log N$ since we need to compute the sign of the level set function at $N$ vertices but $c_1 \ll c_2$. The meshing time scales like $\mathcal{O}(N^{2/3})$. See Section 5 for numerical results.

In summary, our mesh generator is simple and fast. The generated mesh is semi-structured. The interface is approximately recovered, and the maximum angle of the surface mesh is uniformly bounded.

## 4. Finite element methods for elliptic interface problems

We start with Sobolev spaces and the weak formulation of the elliptic interface problem (1)–(4). We then introduce the linear virtual element methods and discuss the implementation detail.

### 4.1. Sobolev spaces and weak formulation

Let $D$ denote a bounded and open set in $\mathbb{R}^d, d = 2, 3$ and $W^{m,p}(D)$ be the usual Sobolev space with standard norm $\|\cdot\|_{m,p,D}$ and semi-norm $|\cdot|_{m,p,D}$. In particular, for $p = 2$, we denote $H^m(D) = W^{m,p}(D)$ and the corresponding norm and semi-norm by $\|\cdot\|_{m,D} = \|\cdot\|_{m,p,D}$ and $|\cdot|_{m,D} = |\cdot|_{m,p,D}$, respectively. The space $H_0^1(D) = \{v \in H^1(D) : v|_{\partial D} = 0\}$ is the subspace of $H^1(D)$ with zero trace. Let $(\cdot,\cdot)_D$ and $\langle\cdot,\cdot\rangle_{\partial D}$ denote the standard $L^2$ inner products of $L^2(D)$ and $L^2(\partial D)$ respectively.

Domains are considered as open sets. Define $\tilde{\Omega} = \Omega^- \cup \Omega^+$ and notice that $\Omega = \Omega^- \cup \Gamma \cup \Omega^+ = \tilde{\Omega} \cup \Gamma$. For $v \in W^{m,p}(\tilde{\Omega})$, that is, $v|_{\Omega^-} \in W^{m,p}(\Omega^-)$ and $v|_{\Omega^+} \in W^{m,p}(\Omega^+)$, $v$ may not be in $W^{m,p}(\Omega)$ due to the jump across the interface $\Gamma$.

To derive the weak formulation of elliptic interface problems (1)–(4), we multiply (1) with a test function $v \in H_0^1(\Omega)$ and apply integration by parts. To address the jump of function values, we choose a $w^- \in H^1(\Omega^-)$ with $w^- = q_0$ on $\partial\Omega^-$. With a slight abuse of notation, the zero extension of $w^-$ to $H^1(\tilde{\Omega})$ is still denoted by $w^-$. The model (1)–(4) is equivalent to: find $p \in H_g^1(\Omega) = \{v \in H^1(\Omega) : v|_{\partial\Omega} = g\}$ such that

$$(\beta\nabla p, \nabla v)_\Omega = (f, v)_\Omega - \langle q_1, v\rangle_\Gamma + (\beta\nabla w^-, \nabla v)_{\Omega^-}, \quad \forall v \in H_0^1(\Omega), \tag{5}$$

and set $u = p - w^-$. It is easy to show that $u$ solves equations (1)–(4). Even though the choice of $w^-$ is not unique, the solution $u$ does not depend on the choice of $w^-$ by the maximal principle. The flux jump $[\beta u_n]_\Gamma = q_1$ is imposed in $H^{-1/2}(\Gamma)$ and the jump of function value is imposed in $H^{1/2}(\Gamma)$.

### 4.2. Finite element methods in 2D

In this subsection, we present the numerical analysis of the standard finite element methods on the two-dimensional interface-fitted mesh generated by Algorithm 1.

For simplicity of exposition, we assume the function value jump condition $[u]_\Gamma = 0$. Let $\mathcal{T}_h$ be an interface-fitted triangular mesh with maximal angles uniformly bounded away from $\pi$. For each $\tau \in \mathcal{T}_h$, let $h_\tau$ denote its diameter and $h = \max_{\tau\in\mathcal{T}_h} h_\tau$. The vertices on $\Gamma$ forms a polygon $\Gamma_h$ approximation of $\Gamma$. The polygon also splits $\Omega$ into two subdomains, $\Omega_h^+$ and $\Omega_h^-$, which are the approximations of $\Omega^+$ and $\Omega^-$, respectively. Each triangle $\tau \in \mathcal{T}_h$ is in either $\Omega_h^+$ or $\Omega_h^-$ and has at most two vertices on $\Gamma$.

Let $V_h$ be the linear finite element space on $\mathcal{T}_h$. The linear finite element approximation of (5) is as follows: find $u_h \in V_h \cap H_0^1(\Omega)$ such that:

$$(\beta_h \nabla u_h, \nabla v_h)_\Omega = (f, v_h)_\Omega - \langle \bar{q}_1, v\rangle_{\Gamma_h}, \quad \forall v_h \in V_h \cap H_0^1(\Omega), \tag{6}$$

where $\bar{q}_1 = q_1(\mathcal{P}_0(x))$ and $\mathcal{P}_0(x)$ is a well defined projection from $\Gamma_h$ to $\Gamma$ (cf. [80]).

We can get the nearly optimal $L^2$-norm and $H^1$-norm estimates as the results in [24,85].

**Theorem 4.1.** *Let $u$ be the solution of (5) and $u_h$ be the linear finite element approximation in (6) based on the two-dimensional interface-fitted mesh generated by Algorithm 1. We have*

$$\|\beta^{1/2}(\nabla u - \nabla u_h)\|_{0,\Omega} \lesssim h|\log h|^{1/2}(\|f\|_{0,\Omega} + \|q_1\|_{2,\Gamma}), \tag{7}$$

$$\|u - u_h\|_{0,\Omega} \lesssim h^2|\log h|(\|f\|_{0,\Omega} + \|q_1\|_{2,\Gamma}). \tag{8}$$

**Proof.** For finite element approximation, we have the Céa's lemma,

$$\|\beta^{1/2}(\nabla u - \nabla u_h)\|_{0,\Omega} \le \|\beta^{1/2}(\nabla u - \nabla u_I)\|_{0,\Omega}.$$

Then the energy error estimate is reduced to the interpolation error estimate. In [6], the authors proved that the local interpolation error estimate $\|(\nabla u - \nabla u_I)\|_{0,\tau} \lesssim h\|u\|_{2,\tau}$ provided the maximal angle condition is satisfied which has been verified for the interface-fitted mesh generated by Algorithm 1; see Proposition 2.4. Another difficulty is the mismatch of the curved interface and the discrete interface. Then following the proof in [24,85], and replacing the mesh regular condition there by the maximal condition, we obtain the desired results. □

A mesh is $\mathcal{O}(h^{2\sigma})$ irregular means the total area of all adjacent triangle pairs in $\mathcal{T}_h$ which do not form an $\mathcal{O}(h^2)$ approximate parallelogram is $\mathcal{O}(h^{2\sigma})$. For the interface-fitted mesh generated by Algorithm 1, only the adjacent triangle pairs near the interface is not $\mathcal{O}(h^2)$ approximate parallelogram and other adjacent triangle pairs away from the interface can exactly form a parallelogram. That is $\sigma = 0.5$ for the mesh generated by Algorithm 1.

Then following the proof procedure in [80], we can also prove the following superconvergence result.

**Theorem 4.2.** *If $u \in H^1(\Omega) \cap H^3(\tilde{\Omega}) \cap W^{2,\infty}$ and $\Gamma$ is of class $C^2$, then for all $v_h \in V_h$,*

$$\|\beta_h^{1/2}(\nabla u_h - \nabla u_I)\|_{0,\Omega} \lesssim h^{3/2}\left(\|u\|_{3,\tilde{\Omega}} + \|u\|_{2,\infty,\tilde{\Omega}} + \|u\|_{2,\infty,\tilde{\Omega}} + \|q_1\|_{0,\infty,\Gamma}\right). \tag{9}$$

Let $h_{\min}$ be the minimum element size of $\mathcal{T}_h$, by the discrete embedding result,

$$\|v_h\|_{0,\infty,\Omega} \lesssim |\log h_{\min}|^{1/2}|v_h|_{1,\Omega}, \text{ for all } v_h \in V_h \cap H_0^1(\Omega), \tag{10}$$

we have the error estimate for the maximal norm estimate.

**Corollary 4.3.** *Assume the same hypothesis in Theorem 4.2. Then*

$$\|\beta_h^{1/2}(\nabla u_h - \nabla u_I)\|_{0,\infty,\Omega} \lesssim |\log h_{\min}|^{1/2}\left[h^{3/2}(\|u\|_{3,\tilde{\Omega}} + \|u\|_{2,\infty,\tilde{\Omega}} + \|u\|_{2,\infty,\tilde{\Omega}} + \|q_1\|_{0,\infty,\Gamma})\right].$$

### 4.3. Virtual element methods in 3D

In this subsection, we focus on solving three-dimensional elliptic equations by the virtual element methods (VEM) developed by Brezzi's group [9,10].

Let $\mathcal{T}_h$ be the interface-fitted polyhedral mesh generated by the algorithm in Section 3. Recall that elements near the interface $\Gamma$ are polyhedra with triangular or square faces and a uniform cubic mesh away from the interface. We could not use the classical finite element methods which are not well-defined on polyhedra. Instead, we shall apply virtual element methods [9] which can be thought of as conforming finite element spaces defined on polyhedral meshes.

A local finite-dimensional vector space $V_h(E)$ for a polyhedron $E \in \mathcal{T}_h$ is defined as

$$V_h(E) := \{v \in H^1(E) : \Delta v|_E = 0,\ v|_{\partial E} \text{ is continuous and piecewise linear (on triangles) or bilinear (on squares)}\}.$$

As a piecewise linear or bilinear function will be uniquely determined by its value on vertices, $\dim V_h(E) = n_E^v$, where $n_E^v$ is the number of vertices of $E$.

We define the global virtual element space

$$V_h = \{v_h \in H^1(\Omega) : v_h|_E \in V_h(E) \ \text{ for all } E \in \Omega_h\}.$$

Let $\mathcal{N}(\mathcal{T}_h)$ be the set of vertices of mesh $\mathcal{T}_h$ and $N = |\mathcal{N}(\mathcal{T}_h)|$ be the number of vertices. We define the operator $\text{dof}_i$ from $V_h$ to $\mathbb{R}$ as $\text{dof}_i(v_h) = v_h(\boldsymbol{x}_i)$, for a vertex $\boldsymbol{x}_i \in \mathcal{N}(\mathcal{T}_h)$. The canonical basis $\{\phi_1, \cdots, \phi_N\} \subset V_h$ is chosen as $\text{dof}_i(\phi_j) = \delta_{ij}, i, j = 1, \cdots, N$. And the nodal interpolation $I_h : C(\bar{\Omega}) \to V_h$ is defined as $I_h u = \sum_{i=1}^N u(\boldsymbol{x}_i)\phi_i$ and denoted by $u_I = I_h u$. The basis does not need to be written explicitly which is the main difference between classical finite element methods and virtual element methods.

As mentioned before, we could extract an approximate surface $\Gamma_h$ which splits $\Omega$ into two subdomains: $\Omega_h^-$ and $\Omega_h^+$, which are the approximation of $\Omega^-$ and $\Omega^+$, respectively. Similarly, $\beta_h|_\tau = \beta^+$ for all $\tau \in \Omega_h^+$ and $\beta_h|_\tau = \beta^-$ for all $\tau \in \Omega_h^-$.

Let $w_h^-$ be the nodal interpolation of $w^-$ in $V_h$. A simple construction is one that: interpolates $q_0$ on $\Gamma_h$ and sets other coefficients to zero. The linear virtual element approximation of (5) is: finding $p_h \in V_h \cap H_g^1(\Omega)$ such that:

$$(\beta_h \nabla p_h, \nabla v_h)_\Omega = (f, v_h)_\Omega - \langle q_1, v_h\rangle_\Gamma + (\beta_h \nabla w_h^-, \nabla v_h)_{\Omega^-}, \quad \forall v_h \in V_h \cap H_0^1(\Omega)$$

and taking $u_h = p_h - w_h^-$. Suppose $p_h = \sum_{j=1}^N p_j\phi_j$, $w_h^- = \sum_{j=1}^N w_j\phi_j$, by linearity, we have for $i \in 1, \cdots, N$,

$$\sum_{j=1}^N (\beta_h \nabla\phi_j, \nabla\phi_i)_\Omega p_j = (f, \phi_i)_\Omega - \langle q_1, \phi_i\rangle_\Gamma + \sum_{j=1}^N (\beta_h \nabla\phi_j, \nabla\phi_i)_{\Omega^-} w_j. \tag{11}$$

We define the matrix $(\boldsymbol{A}_h^-)_{ij} = (\beta_h^- \nabla\phi_j, \nabla\phi_i)_{\Omega_h^-}$, $(\boldsymbol{A}_h^+)_{ij} = (\beta_h^+ \nabla\phi_j, \nabla\phi_i)_{\Omega_h^+}$ and $(\boldsymbol{A}_h)_{ij} = (\beta_h \nabla\phi_j, \nabla\phi_i)_{\Omega_h}$ in $\Omega_h$. Then $\boldsymbol{A}_h = \boldsymbol{A}_h^- + \boldsymbol{A}_h^+$. Define the vector $\boldsymbol{b} = (b_1, \cdots, b_N)^t$ by $b_i = (f, \phi_i)_\Omega - \langle q_1, \phi_i\rangle_\Gamma$. Equation (11) is written in the matrix form as

$$\boldsymbol{A}_h \boldsymbol{p}_h = \boldsymbol{b} + \boldsymbol{A}_h^- \boldsymbol{w}_h, \tag{12}$$

where $\boldsymbol{A}_h$ and $\boldsymbol{A}_h^-$ are $N \times N$ matrices, $\boldsymbol{p}_h = (p_h^1, \cdots, p_h^N)^t$ and $\boldsymbol{w}_h = (w_1, \cdots, w_N)^t$. Since the coefficient $\beta$ is a positive constant, the matrices $\boldsymbol{A}_h$ and $\boldsymbol{A}_h^-$ are symmetric and positive definite. The algebraic system (12) could be solved stably and efficiently by using algebraic multigrid methods.

For finite element methods, it suffices to compute the local stiffness matrix in each element and then, based on that, the matrices $\boldsymbol{A}_h^+, \boldsymbol{A}_h^-$ are assembled by summing the contribution from each element. Therefore, the major task is to compute $(\nabla\phi_j, \nabla\phi_i)_E$.

To do so, we introduce some projection operators at first. For each polyhedron $E$, the operator $\Pi^\nabla : V_h(E) \to \mathbb{P}_1(E)$ is defined as the $H^1$ projection to $\mathbb{P}_1(E)$ space, i.e.,

$$(\nabla p_k, \nabla\Pi^\nabla v_h)_E = (\nabla p_k, \nabla v_h)_E \quad \text{ for all } p_k \in \mathbb{P}_1(E),$$

where $\mathbb{P}_1(E)$ is the space of linear polynomials. It can be easily seen that the above condition defines $\Pi^\nabla v_h$ only up to a constant. This can be fixed by prescribing a projection operator onto constants $P_0 : V_h(E) \to \mathbb{P}_0(E)$ and requiring

$$P_0(\Pi^\nabla v_h - v_h) = 0.$$

One such choice is $P_0 v_h = \sum_{i=1}^{n_E^v} v_h(\boldsymbol{x}_i)/n_E^v = \sum_{i=1}^{n_E^v} \text{dof}_i(v_h)/n_E^v$.

Using the projection $\Pi^\nabla$, we write the basis function $\phi_i \in V_h(E)$ as $\Pi^\nabla\phi_i + (I - \Pi^\nabla)\phi_i$ and split the entry of the local stiffness matrix as

$$(\nabla\Pi^\nabla\phi_i, \nabla\Pi^\nabla\phi_j)_E + (\nabla(I - \Pi^\nabla)\phi_i, \nabla(I - \Pi^\nabla)\phi_j)_E.$$

Again the second term is not computable since the basis $\phi_i$ is not known point-wise. Instead we replace by a so-called stabilization term $S^E(\cdot,\cdot)$

$$(\nabla\Pi^\nabla\phi_i, \nabla\Pi^\nabla\phi_j)_E + S^E((I-\Pi^\nabla)\phi_i, (I-\Pi^\nabla)\phi_j).$$

Let $h_E$ be $|E|^{1/3}$, where $|E|$ means the volume of $E$. We use a scaled $l^2$ inner product in the stabilization term

$$S^E((I-\Pi^\nabla)\phi_i, (I-\Pi^\nabla)\phi_j) = h_E\sum_{r=1}^{n_E^v}\mathrm{dof}_r((I-\Pi^\nabla)\phi_i)\,\mathrm{dof}_r((I-\Pi^\nabla)\phi_j)$$

in order to satisfy the assumption of $S^E$

$$c_1(\nabla v,\nabla v)\le S^E(v,v)\le c_2(\nabla v,\nabla v),\quad \forall v\in V_h(E) \text{ and } \Pi^\nabla v = 0$$

for some positive constants $c_1$ and $c_2$ independent of $E$ and $h_E$. The explicit expression of the local stiffness matrix of the virtual element method is:

$$(\boldsymbol{K}_h^E)_{ij} := (\nabla\Pi^\nabla\phi_i, \nabla\Pi^\nabla\phi_j)_E + h_E\sum_{r=1}^{n_E^v}\mathrm{dof}_r((I-\Pi^\nabla)\phi_i)\,\mathrm{dof}_r((I-\Pi^\nabla)\phi_j).$$

We now give concrete formulae on the computation of the matrix representation of the operator $\Pi^\nabla$. Let $\boldsymbol{x}_E = (x_E, y_E, z_E)$ be the center of $E$, i.e. $\boldsymbol{x}_E = 1/n_E^v\sum_{i=1}^{n_E^v}\boldsymbol{x}_i$. We choose a scaled monomial basis of $\mathbb{P}_1(E)$ as $m_1 = 1, m_2 = (x-x_E)/h_E, m_3 = (y-y_E)/h_E, m_4 = (z-z_E)/h_E$.

Let $\boldsymbol{G}_{4\times4}$ be defined as

$$\boldsymbol{G} := \begin{pmatrix} P_0m_1 & P_0m_2 & \cdots & P_0m_4 \\ 0 & (\nabla m_2,\nabla m_2)_{0,E} & \cdots & (\nabla m_4,\nabla m_2)_{0,E} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & (\nabla m_2,\nabla m_4)_{0,E} & \cdots & (\nabla m_4,\nabla m_4)_{0,E} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & h_E\boldsymbol{I}_3 \end{pmatrix}$$

where $\boldsymbol{I}_3$ is a $3\times3$ identity matrix.

Let $\boldsymbol{B}_{4\times n_E^v}$ be a matrix defined as:

$$\boldsymbol{B} := \begin{pmatrix} P_0\phi_1 & \cdots & P_0\phi_{n_E^v} \\ (\nabla m_2,\nabla\phi_1)_E & \cdots & (\nabla m_2,\nabla\phi_{n_E^v})_E \\ \vdots & \ddots & \vdots \\ (\nabla m_4,\nabla\phi_1)_E & \cdots & (\nabla m_4,\nabla\phi_{n_E^v})_E \end{pmatrix}.$$

The formulae for the first row of $\boldsymbol{B}$ is $P_0\phi_1 = P_0\phi_2 = \ldots = P_0\phi_{n_E^v} = 1/n_E^v$. For the other components $(\nabla m_j,\nabla\phi_i)_E$, $j = 2,3,4$, we have $(\nabla m_j,\nabla\phi_i)_E = -\int_E\Delta m_j\phi_i + \int_{\partial E}\frac{\partial m_j}{\partial n}\phi_i$ by integration by parts. The first term is zero as $\Delta m_j = 0$ for linear polynomials. We only need to compute the second term. Due to our data structure, all the faces on the $\partial E$ are either triangles or squares. Then

$$\int_{\partial E}\frac{\partial m_j}{\partial n}\phi_i = \frac{\sum_{i\in\text{triangular face }f} n_f^j\,|f|}{3h_E} + \frac{\sum_{i\in\text{square face }f} n_f^j\,|f|}{4h_E}, \tag{13}$$

where $\boldsymbol{n_f} = (n_f^x, n_f^y, n_f^z) = (n_f^2, n_f^3, n_f^4)$ is an outward unit normal direction on each face $f$ and $|f|$ is the area for each face $f$.

**Remark 4.4.** Using our mesh generation algorithm in Section 3, we store the polyhedron in the form of either triangles or squares which leads to the simple formula (13). When the faces are general polygons, additional projection operators are needed in order to compute the integral $\int_f\frac{\partial m_j}{\partial n}\phi_i$ (see [9,10]). □

To compute the stabilization term, we need one more matrix $\boldsymbol{D}_{n_E^v\times4}$

$$\boldsymbol{D} := \left(\mathrm{dof}_i(m_j)\right) = h_E^{-1}\begin{pmatrix} h_E & x_1-x_E & y_1-y_E & z_1-z_E \\ h_E & x_2-x_E & y_2-y_E & y_2-z_E \\ \cdots & \cdots & \cdots & \cdots \\ h_E & x_{n_E^v}-x_E & y_{n_E^v}-y_E & z_{n_E^v}-z_E, \end{pmatrix}$$

where $(x_i, y_i, z_i), i = 1,\cdots,n_E^v$ are vertices in each polyhedron $E$.

By definition, $\Pi^\nabla v_h = \sum_{\alpha=1}^4 s^\alpha m_\alpha$ and the coefficients $(s^\alpha)$ are determined by the following linear systems

$$(\nabla m_\alpha, \nabla(\Pi^\nabla v_h - v_h))_E = 0 \quad \alpha = 1, \ldots, 4.$$

The matrix representation of $\Pi^\nabla : V_h(E) \to \mathbb{P}_1(E)$ relative to the basis $(m_\alpha)$ is $\boldsymbol{\Pi}^\nabla = \boldsymbol{G}^{-1}\boldsymbol{B}$.

We will also need the matrix representation of $\Pi^\nabla$ in the canonical basis $\{\phi_i\}$. Let $\Pi^\nabla \phi_i = \sum_{j=1}^{n_E^v} \mathrm{dof}_j(\Pi^\nabla \phi_i)\phi_j$, $i = 1, \cdots, n_E^v$, then the matrix representation $\boldsymbol{\Pi}_*^\nabla$ of the operator $\Pi^\nabla : V_h(E) \to V_h(E)$ in the canonical basis is given by $\boldsymbol{\Pi}_*^\nabla = \boldsymbol{D}\boldsymbol{G}^{-1}\boldsymbol{B} = \boldsymbol{D}\boldsymbol{\Pi}^\nabla$.

Finally the matrix formulation of $\boldsymbol{K}_h^E$ could be written as

$$\boldsymbol{K}_h^E = [\boldsymbol{\Pi}^\nabla]^T \tilde{\boldsymbol{G}} \boldsymbol{\Pi}^\nabla + h_E[\boldsymbol{I} - \boldsymbol{\Pi}_*^\nabla]^T[\boldsymbol{I} - \boldsymbol{\Pi}_*^\nabla],$$

where $\tilde{\boldsymbol{G}}$ is the same with $\boldsymbol{G}$ except that the elements in the first row are all zeros.

For the first term of $b_i$ in (12), we approximate $f$ by a piecewise constant and approximate

$$(f, \phi_i)_\Omega = \sum_{E \in \Omega_h} (f, \phi_i)_E \approx \sum_{E \in \Omega_h} |E| f(x_E, y_E, z_E)/n_E^v.$$

The second term of $b_i$ could be computed by Gauss quadrature on surface mesh $\Gamma_h$.

**Remark 4.5.** An abstract error estimate of VEM has been given in [9]. With a type of Céa's lemma, the convergence analysis is reduced to the interpolation error estimate $|u - u_I|_1$ and $|u - u_\pi|_{1,E}$, where $u_I$ is the nodal interpolation and $u_\pi$ is a local approximation of $u$. Notice that $u_I \in V_h$ is continuous but $u_\pi$ is most likely discontinuous. To obtain optimal order of the interpolation and approximation error, the authors in [9] further assume the shape-regular condition: there exists a $\gamma > 0$ such that each domain $E$ is star-shaped with respect to a ball of radius $\rho \geq \gamma h_E$, where $h_E = \mathrm{diam}(E)$. This shape regularity assumption will rule out elements generated by our algorithm.

As we mentioned before, for linear finite element space defined on triangles, a refined analysis shows that the optimal first order interpolation error estimate still holds if the maximum angle is uniformly bounded away from $\pi$ as $h \to 0$ [6]. Such angle condition is generalized to three dimensions, and to high order elements in [3,30,56]. Generalization to polyhedra, however, is unknown and under investigation. □

## 5. Numerical experiments

In this section, we present numerical results for the elliptic interface problems in three dimensions. We implement mesh generation and VEM based on the MATLAB® package *i*FEM [22]. We also solve the algebraic system by an algebraic multigrid (AMG) solver implemented in *i*FEM [22]. We start with a simple spherical interface and then consider more complex geometric shapes, including two spheres, an orthocircle shape and 12 intersecting spheres. We shall report the following errors:

$$\|u_I - u_h\|_A = \left( \|\beta_h^{1/2} \nabla(u_I^- - u_h^-)\|_{\Omega_h^-}^2 + \|\beta_h^{1/2} \nabla(u_I^+ - u_h^+)\|_{\Omega_h^+}^2 \right)^{1/2},$$

$$\|u_I - u_h\|_\infty = \max\left\{ \|u_I^- - u_h^-\|_{\infty,\Omega_h^-}, \|u_I^+ - u_h^+\|_{\infty,\Omega_h^+} \right\},$$

$$\|u_I - u_h\|_{0,h} = h^{3/2} \left( \sum_{\boldsymbol{x}_i \in \mathcal{N}(\Omega_h^-)} (u_I^-(\boldsymbol{x}_i) - u_h^-(\boldsymbol{x}_i))^2 + \sum_{\boldsymbol{x}_i \in \mathcal{N}(\Omega_h^+)} (u_I^+(\boldsymbol{x}_i) - u_h^+(\boldsymbol{x}_i))^2 \right)^{1/2},$$

where $u_h$ is the numerical solution obtained by the linear virtual element methods; $u_I^+$ and $u_I^-$ are the nodal interpolation of the exact solution $u$ in $\Omega_h^+$ and $\Omega_h^-$ respectively. Note that the squared energy norm $\|u_I - u_h\|_A^2$ can be computed by $(u_I^- - u_h^-)^T \boldsymbol{A}_h^-(u_I^- - u_h^-) + (u_I^+ - u_h^+)^T \boldsymbol{A}_h^+(u_I^+ - u_h^+)$ and $\|\cdot\|_{0,h}$ is a good approximation of $L^2$-norm. The rate is obtained by the least square fitting of the errors in the log log scale.

**Example 5.1** *(One sphere).* The domain $\Omega$ is $(-1, 1)^3$ and the interface is defined by $\phi(x, y, z) = x^2 + y^2 + z^2 - r^2$ with radius $r = 0.75$. The coefficient $\beta$ is piecewise constant. The analytic solution is given by $u^+ = 10(x + y + z)$ and $u^- = 5\exp(x^2 + y^2 + z^2) + 20$. In this case, the solution is discontinuous and the flux jump across the interface is also non-homogeneous.

Fig. 12 shows the surface mesh extracted from the volume mesh generated by our algorithm for the spherical interface. The maximal interior angle of triangular faces on the surface mesh is bounded by 112.8104°. Tables 2 and 3 show the error for $\beta^- = 1, \beta^+ = 10$ and $\beta^- = 1, \beta^+ = 100$, respectively. It can be seen that near second order accuracy is attained in both $\|\cdot\|_{0,h}$ and $\|\cdot\|_\infty$ norms. The convergence rate in the energy norm is near 1.5, which is consistent with the

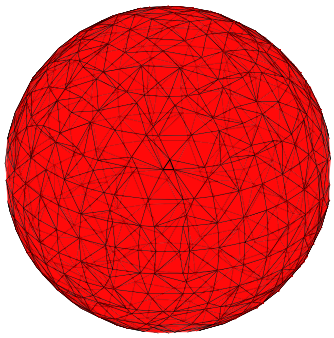


**Fig. 12.** An interface mesh with maximal angle 112.8104°.

**Table 2**
Errors for Example 5.1: $\beta^- = 1$ and $\beta^+ = 10$.

| #dof | $h$ | $\|u_I - u_h\|_A$ | $\|u_I - u_h\|_\infty$ | $\|u_I - u_h\|_0$ |
|---|---|---|---|---|
| 10,323 | 0.1 | 4.45798e−01 | 5.24227e−02 | 3.62347e−02 |
| 72,713 | 0.05 | 1.64215e−01 | 1.81762e−02 | 8.97415e−03 |
| 547,881 | 0.025 | 6.62120e−02 | 5.09725e−03 | 2.62269e−03 |
| 4,240,529 | 0.0125 | 2.43899e−02 | 1.37664e−03 | 7.05429e−04 |
| Rate | | 1.4 | 1.8 | 1.9 |

**Table 3**
Errors for Example 5.1: $\beta^- = 1$ and $\beta^+ = 100$.

| #dof | $h$ | $\|u_I - u_h\|_A$ | $\|u_I - u_h\|_\infty$ | $\|u_I - u_h\|_0$ |
|---|---|---|---|---|
| 10,323 | 0.1 | 6.65319e−01 | 5.14276e−02 | 3.40871e−02 |
| 72,713 | 0.05 | 2.41564e−01 | 1.84007e−02 | 8.42552e−03 |
| 547,881 | 0.025 | 8.98117e−02 | 5.21504e−03 | 2.50271e−03 |
| 4,240,529 | 0.0125 | 3.21041e−02 | 1.42298e−03 | 6.80724e−04 |
| Rate | | 1.5 | 1.7 | 1.9 |

**Table 4**
CPU time (in seconds) for Example 5.1: $\beta^- = 1$ and $\beta^+ = 10$.

| #dof | Assemble | Solve | Mesh |
|---|---|---|---|
| 10,323 | 0.228973 | 0.47 | 0.21165 |
| 72,713 | 0.970122 | 2.59 | 0.4854 |
| 547,881 | 6.99419 | 13.55 | 1.8981 |
| 4,240,529 | 62.1644 | 121.59 | 8.0172 |

**Table 5**
Example 5.1: Iteration steps of AMG with fixed $\beta^- = 1$ and various $\beta^+$.

| #dof | $\beta^+$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $10^{-3}$ | $10^{-2}$ | $10^{-1}$ | 1 | 10 | $10^2$ | $10^3$ | $10^4$ |
| 7,921 | 10 | 10 | 10 | 9 | 9 | 9 | 9 | 9 |
| 63,111 | 11 | 11 | 11 | 11 | 11 | 10 | 10 | 10 |
| 509,479 | 12 | 12 | 12 | 12 | 14 | 13 | 13 | 13 |
| 4,086,927 | 13 | 12 | 13 | 15 | 17 | 16 | 16 | 16 |

superconvergence result obtained in [21]. This superconvergence occurs due to the nice properties of our semi-structured mesh. It seems that the convergence rate is robust to the variation of $\beta$.

From Table 4, we can conclude that the runtimes of the mesh generation part can be ignored compared with the assembling and solving parts. In Table 5, we present the variation of iteration steps of the algebraic multigrid method with respect to the number of degrees of freedom (#dof) and to the variation of jump coefficients (fix $\beta^- = 1$ and change $\beta^+$). It indicates that the algebraic multigrid method is a robust and efficient solver: robust to the number of degrees of freedom and to the variation of jump coefficients.

**Example 5.2** *(Two spheres).* The domain $\Omega$ is $(-1, 1)^3$ and the interface is defined by

$$\phi(x, y, z) = \min\{\phi_1, \phi_2\}$$

where

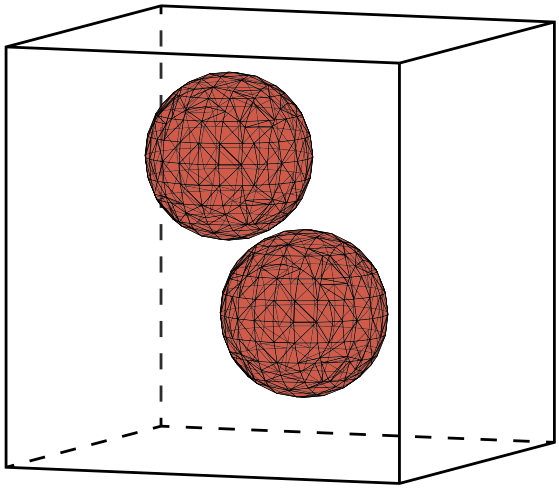


**Fig. 13.** Two balls are embedded in the unit cube. The maximal angle is 130.4665°.

**Table 6**
Errors for Example 5.2: $\beta^- = 1$ and $\beta^+ = 1$.

| #dof | $h$ | $\|u_I - u_h\|_A$ | $\|u_I - u_h\|_\infty$ | $\|u_I - u_h\|_0$ |
|---|---|---|---|---|
| 9,789 | 0.1 | 3.76723e−01 | 1.08148e−01 | 6.52076e−02 |
| 71,225 | 0.05 | 1.32276e−01 | 2.82699e−02 | 1.74225e−02 |
| 540,945 | 0.025 | 4.52335e−02 | 7.30380e−03 | 4.28321e−03 |
| 4,211,729 | 0.0125 | 1.60165e−02 | 1.92081e−03 | 1.11271e−03 |
| Rate | | 1.5 | 1.9 | 2 |

**Table 7**
Errors for Example 5.2: $\beta^- = 1$ and $\beta^+ = 100$.

| #dof | $h$ | $\|u_I - u_h\|_A$ | $\|u_I - u_h\|_\infty$ | $\|u_I - u_h\|_0$ |
|---|---|---|---|---|
| 9,789 | 0.1 | 3.67145e+00 | 1.15596e−01 | 5.54995e−02 |
| 71,225 | 0.05 | 1.35081e+00 | 3.08549e−02 | 1.51136e−02 |
| 540,945 | 0.025 | 4.78640e−01 | 8.22331e−03 | 3.74778e−03 |
| 4,211,729 | 0.0125 | 1.72934e−01 | 2.25395e−03 | 9.77991e−04 |
| Rate | | 1.5 | 1.9 | 2 |

**Table 8**
CPU time (in seconds) for Example 5.2: $\beta^- = 1$ and $\beta^+ = 1$.

| #dof | Assemble | Solve | Mesh |
|---|---|---|---|
| 9,789 | 0.160716 | 1.01 | 0.259938 |
| 71,225 | 0.706959 | 5.84 | 0.36018 |
| 540,945 | 6.32812 | 27.26 | 1.78691 |
| 4,211,729 | 52.7369 | 133.86 | 10.067 |

$$\phi_1 = (x + 0.5r)^2 + (y + 0.5r)^2 + (z + 0.75r)^2 - r^2,$$
$$\phi_2 = (x - 0.25r)^2 + (y - 0.75r)^2 + (z - r)^2 - r^2,$$

with radius $r = 0.4$. The coefficient $\beta$ is piecewise constant. The analytic solution is given by $u^+ = 10(x^2 + y^2 + z^2)$ and $u^- = 5\cos(x^2 + y^2 + z^2)$.

Fig. 13 shows that two spheres are embedded in the unit cube. When $h = 0.1$, i.e., the background Cartesian mesh is not fine enough, there exists an interface element which is divided into three parts by these two spheres. The maximal interior angle of triangular faces on the surface mesh is bounded by 130.4665°. Tables 6 and 7 show the numerical results for $\beta^- = 1, \beta^+ = 1$ and $\beta^- = 1, \beta^+ = 100$, respectively. Table 8 shows how the computational time grows with respect to the number of degrees of freedom. Table 9 shows the number of iterations taken by the algebraic multigrid method for various values of input parameters.

All results are consistent with our conclusion. It indicates that our algorithm works in a case when the interface is unconnected.

**Table 9**
Example 5.2: Iteration steps of AMG with fixed $\beta^- = 1$ and various $\beta^+$.

| #dof | $\beta^+$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $10^{-3}$ | $10^{-2}$ | $10^{-1}$ | 1 | 10 | $10^2$ | $10^3$ | $10^4$ |
| 7,387 | 10 | 11 | 10 | 9 | 9 | 9 | 9 | 9 |
| 61,623 | 11 | 11 | 11 | 10 | 10 | 10 | 10 | 10 |
| 502,543 | 12 | 12 | 12 | 12 | 12 | 11 | 12 | 11 |
| 4,058,127 | 13 | 13 | 13 | 12 | 13 | 13 | 13 | 13 |

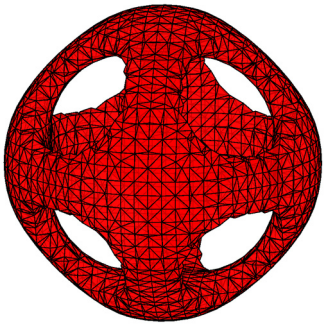


**Fig. 14.** The interface is an orthocircle with maximal angle 132.4673°.

**Table 10**
Errors for Example 5.3: $\beta^- = 1$ and $\beta^+ = 1$.

| #dof | $h$ | $\|u_I - u_h\|_A$ | $\|u_I - u_h\|_\infty$ | $\|u_I - u_h\|_0$ |
|---|---|---|---|---|
| 11,145 | 0.12 | 2.93834e−01 | 5.50055e−02 | 5.14120e−02 |
| 76,469 | 0.06 | 7.95795e−02 | 1.05921e−02 | 4.18525e−03 |
| 561,957 | 0.03 | 2.35627e−02 | 2.22961e−03 | 9.17810e−04 |
| 4,295,165 | 0.015 | 7.80143e−03 | 6.05301e−04 | 2.15878e−04 |
| Rate | | 1.7 | 2.1 | 2.1 |

**Table 11**
Errors for Example 5.3: $\beta^- = 1$ and $\beta^+ = 100$.

| #dof | $h$ | $\|u_I - u_h\|_A$ | $\|u_I - u_h\|_\infty$ | $\|u_I - u_h\|_0$ |
|---|---|---|---|---|
| 11,145 | 0.12 | 3.44042e+00 | 9.63266e−02 | 1.24556e−01 |
| 76,469 | 0.06 | 7.72426e−01 | 1.21513e−02 | 1.56542e−02 |
| 561,957 | 0.03 | 2.06488e−01 | 3.04186e−03 | 3.34684e−03 |
| 4,295,165 | 0.015 | 6.30787e−02 | 7.45803e−04 | 8.10506e−04 |
| Rate | | 1.8 | 2 | 2.1 |

**Table 12**
CPU time (in seconds) for Example 5.3: $\beta^- = 1$ and $\beta^+ = 1$.

| #dof | Assemble | Solve | Mesh |
|---|---|---|---|
| 11,145 | 0.493225 | 0.75 | 0.383038 |
| 76,469 | 1.41325 | 1.72 | 0.857616 |
| 561,957 | 8.1635 | 12.05 | 3.48605 |
| 4,295,165 | 62.6127 | 97.92 | 13.8495 |

**Example 5.3** *(An orthocircle).* The domain $\Omega$ is $(-1.2, 1.2)^3$ and the interface is defined by $\phi(x, y, z) = \left[(x^2 + y^2 - 1)^2 + z^2\right] \times \left[(x^2 + z^2 - 1)^2 + y^2\right]\left[(y^2 + z^2 - 1)^2 + x^2\right] - 0.075^2\left[1 + 3(x^2 + y^2 + z^2)\right]$. The coefficient $\beta$ is piecewise constant. The analytic solution is given by $u^+ = 1 - x^2 - y^2 - z^2$ and $u^- = \sin(\pi x)\sin(\pi y)\sin(\pi z)$.

Fig. 14 shows the interface-fitted mesh extracted as the boundary of $\Omega_h^-$. The maximal angle of triangular faces of the interface mesh is bounded by 132.4673°. Tables 10 and 11 show the numerical results for $\beta^- = 1, \beta^+ = 1$ and $\beta^- = 1, \beta^+ = 100$, respectively. Table 12 shows the computational time accordingly. The mesh part is still quick even the interface with complex geometry. Table 13 is the number of iterations of the algebraic multigrid solver.

These test results indicate that the proposed interface problem solver works well even for complex surfaces. Note that the maximal angles of the surface mesh in our examples are uniformly bounded by 144°, then there is no need to apply mesh smoothing as a post-processing step.

**Table 13**
Example 5.3: Iteration steps of AMG with fixed $\beta^- = 1$ and various $\beta^+$.

| #dof | $\beta^+$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $10^{-3}$ | $10^{-2}$ | $10^{-1}$ | 1 | 10 | $10^2$ | $10^3$ | $10^4$ |
| 8,743 | 15 | 14 | 15 | 13 | 11 | 10 | 10 | 10 |
| 66,867 | 13 | 12 | 11 | 11 | 11 | 11 | 11 | 11 |
| 523,555 | 13 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| 4,141,563 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 |

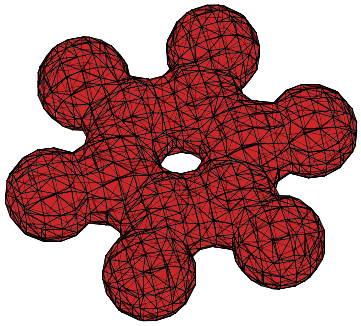


**Fig. 15.** Twelve balls are embedded in the cube. The maximal angle is 131.3925°.

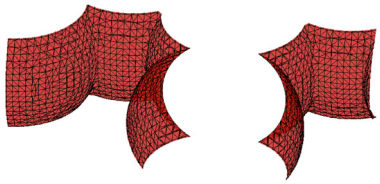


**Fig. 16.** Part of the interface-fitted mesh with sharp edges.

**Example 5.4** *(12 intersecting spheres).* In this example, we consider a more complicated interface formed by 12 intersecting spheres, which has many one-dimension sharp features. The domain $\Omega$ is $(-3, 3)^3$ and the interface is defined by

$$\phi(x, y, z) = \min\{\phi_1, \phi_2, \cdots, \phi_{12}\}$$

where $\phi_i, i = 1, \ldots, 12$ are equal radius balls with centers

$$(1.0, 0, 0), (-1.0, 0, 0), (0.5, 0.866025403784439, 0), (-0.5, 0.866025403784439, 0),$$
$$(0.5, -0.866025403784439, 0), (-0.5, -0.866025403784439, 0), (2.0, 0, 0), (-2.0, 0, 0),$$
$$(1.0, 1.73205080756888, 0), (-1.0, 1.73205080756888, 0), (-1.0, -1.73205080756888, 0),$$
$$(1.0, -1.73205080756888, 0)$$

and radius $r = 0.7$. The coefficient $\beta$ is a piecewise constant. The analytic solution is given by $u^+ = 10(x^2 + y^2 + z^2)$ and $u^- = 5\cos(x^2 + y^2 + z^2)$.

Fig. 15 shows the interface-fitted mesh extracted as the boundary of $\Omega_h^-$, which did not capture the sharp one-dimension features of the interface, and this will lead to a loss of the solution accuracy. From part of the interface-fitted mesh, there are geometric singularities with sharp edges at the intersection of balls. See Fig. 16 for an illustration. In order to capture the features of complicated interface and improve the solution accuracy, one need to use adaptive mesh near the geometric features, and this will be our future work. But the maximal angle condition is still satisfied as the maximal angle of triangular faces on the surface mesh is bounded by 131.3925°. Tables 14 and 15 show the numerical results for $\beta^- = 1, \beta^+ = 1$ and $\beta^- = 1, \beta^+ = 10$, respectively. It can be seen that the convergence rate in the energy norm is still near 1.5. But the convergence rates in the $\|\cdot\|_{0,h}$ and $\|\cdot\|_\infty$ norms are not second order due to geometric singularities. Table 16 shows how the computational time grows with respect to the number of degrees of freedom. Table 17 shows the number of iterations taken by the algebraic multigrid method for various values of input parameters.

**Table 14**
Errors for Example 5.4: $\beta^- = 1$ and $\beta^+ = 1$.

| #dof | $h$ | $\|u_I - u_h\|_A$ | $\|u_I - u_h\|_\infty$ | $\|u_I - u_h\|_0$ |
|---|---|---|---|---|
| 2,527 | 0.48 | 2.27225e+01 | 3.60464e+00 | 6.97070e+00 |
| 18,960 | 0.24 | 1.18700e+01 | 2.24845e+00 | 4.20002e+00 |
| 138,051 | 0.12 | 3.99601e+00 | 8.79537e−01 | 9.22094e−01 |
| 1,051,665 | 0.06 | 1.57695e+00 | 2.91154e−01 | 3.92484e−01 |
| Rate | | 1.4 | 1.2 | 1.6 |

**Table 15**
Errors for Example 5.4: $\beta^- = 1$ and $\beta^+ = 10$.

| #dof | $h$ | $\|u_I - u_h\|_A$ | $\|u_I - u_h\|_\infty$ | $\|u_I - u_h\|_0$ |
|---|---|---|---|---|
| 2,527 | 0.48 | 8.71247e+01 | 5.29705e+00 | 8.63843e+00 |
| 18,960 | 0.24 | 3.90973e+01 | 3.07134e+00 | 4.54532e+00 |
| 138,051 | 0.12 | 1.38105e+01 | 1.37945e+00 | 1.15474e+00 |
| 1,051,665 | 0.06 | 5.49027e+00 | 4.88589e−01 | 3.78441e−01 |
| Rate | | 1.3 | 1.1 | 1.6 |

**Table 16**
CPU time (in seconds) for Example 5.4: $\beta^- = 1$ and $\beta^+ = 1$.

| #dof | Assemble | Solve | Mesh |
|---|---|---|---|
| 2,527 | 0.213214 | 0.29 | 0.3674 |
| 18,960 | 0.431979 | 1.81 | 0.61863 |
| 138,051 | 2.22761 | 4.88 | 2.4452 |
| 1,051,665 | 29.9028 | 42.76 | 6.8316 |

**Table 17**
Example 5.4: Iteration steps of AMG with fixed $\beta^- = 1$ and various $\beta^+$.

| #dof | $\beta^+$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | $10^{-2}$ | $10^{-1}$ | 1 | 10 | $10^2$ | $10^3$ | $10^4$ |
| 1,611 | 11 | 10 | 9 | 8 | 9 | 8 | 8 |
| 15,208 | 12 | 11 | 10 | 10 | 10 | 10 | 10 |
| 123,049 | 12 | 12 | 11 | 11 | 11 | 11 | 11 |
| 991,663 | 14 | 12 | 12 | 12 | 12 | 12 | 12 |

## 6. Conclusion and future work

We have developed a simple interface-fitted mesh generator in both two and three dimensions. Near the interface, we generate a Delaunay triangulation and merge tetrahedra into polyhedra to avoid sliver tetrahedra. We then use virtual element methods as a substitution of classical finite element methods to solve the elliptic interface problems and use the algebraic multigrid solvers for the resulting linear algebraic system. Finally, we show some numerical results to confirm the effectiveness of our method.

Our interface-fitted mesh generator is based on a uniform Cartesian mesh. So it cannot capture the sharp features of complicated interfaces very well. In the future work, we will combine our algorithm and adaptive mesh refinement together. We will also present the convergence analysis in a forthcoming paper and explore high order virtual element methods with curved surfaces. Furthermore, we plan to apply our algorithm to solve moving interface problems for engineering and biological applications.

## Acknowledgements

We would like to thank the unknown referees for their valuable suggestions and careful reading which have helped us to improved the paper. The work is mainly done when the first two authors visit BISEC (Beijing Institute for Scientific and Engineering Computing, Beijing University of Technology) and we thank BISEC for the support and hospitality.

## References

[1] G. Acosta, R.G. Durán, The maximum angle condition for mixed and nonconforming elements: application to the stokes equations, SIAM J. Numer. Anal. 37 (1) (1999) 18–36.
[2] L. Adams, Z. Li, The immersed interface/multigrid methods for interface problems, SIAM J. Sci. Comput. 24 (2) (2002) 463–479.
[3] Al Shenk, Uniform error estimates for certain narrow Lagrangian finite elements, Math. Comput. 63 (207) (1994) 105–119.
[4] L. Antiga, J. Peiró, D.A. Steinman, From image data to computational domains, in: Cardiovascular Mathematics, Springer, 2009, pp. 123–175.
[5] I. Babuška, The finite element method for elliptic equations with discontinuous coefficients, Computing 5 (3) (1970) 207–213.
[6] I. Babuška, A.K. Aziz, On the angle condition in the finite element method, SIAM J. Numer. Anal. 13 (2) (1976) 214–226.

[7] J.W. Barrett, C.M. Elliott, Fitted and unfitted finite-element methods for elliptic equations with smooth interfaces, IMA J. Numer. Anal. 7 (1987) 283–300.
[8] R. Becker, E. Burman, P. Hansbo, A Nitsche extended finite element method for incompressible elasticity with discontinuous modulus of elasticity, Comput. Methods Appl. Mech. Eng. 198 (41–44) (2009) 3352–3360.
[9] L. Beirão da Veiga, F. Brezzi, A. Cangiani, G. Manzini, L. Marini, A. Russo, Basic principles of virtual element methods, Math. Models Methods Appl. Sci. 23 (01) (2013) 199–214.
[10] L. Beirao da Veiga, F. Brezzi, L. Marini, A. Russo, The hitchhiker's guide to the virtual element method, Math. Models Methods Appl. Sci. 24 (08) (2014) 1541–1573.
[11] B. Bejanov, J.-L. Guermond, P.D. Minev, A grid-alignment finite element technique for incompressible multicomponent flows, J. Comput. Phys. 227 (13) (2008) 6473–6489.
[12] P.A. Berthelsen, A decomposed immersed interface method for variable coefficient elliptic equations with non-smooth and discontinuous solutions, J. Comput. Phys. 197 (1) (2004) 364–386.
[13] C. Börgers, A triangulation algorithm for fast elliptic solvers based on domain imbedding, SIAM J. Numer. Anal. 27 (5) (1990) 1187–1196.
[14] J.H. Bramble, J.T. King, A finite element method for interface problems in domains with smooth boundaries and interfaces, Adv. Comput. Math. 6 (1) (1996) 109–138.
[15] J. Brandts, S. Korotov, M. Křížek, et al., A geometric toolbox for tetrahedral finite element partitions, in: Efficient Preconditioned Solution Methods for Elliptic Partial Differential Equations, 2011, pp. 103–122.
[16] K.Q. Brown, Voronoi diagrams from convex hulls, Inf. Process. Lett. 9 (5) (1979) 223–228.
[17] E. Burman, S. Claus, P. Hansbo, M.G. Larson, A. Massing, CutFEM: discretizing geometry and partial differential equations, Int. J. Numer. Methods Eng. 104 (2015) 472–501.
[18] E. Burman, P. Hansbo, Fictitious domain finite element methods using cut elements: I. A stabilized Lagrange multiplier method, Comput. Methods Appl. Mech. Eng. 199 (41–44) (2010) 2680–2686.
[19] E. Burman, P. Hansbo, Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method, Appl. Numer. Math. 62 (4) (2012) 328–341.
[20] D. Chen, Z. Chen, C. Chen, W. Geng, G.-W. Wei, MIBPB: a software package for electrostatic analysis, J. Comput. Chem. 32 (4) (2011) 756–770.
[21] L. Chen, Superconvergence of tetrahedral linear finite elements, Int. J. Numer. Anal. Model. 3 (3) (2006) 273–282.
[22] L. Chen, iFEM: An Integrated Finite Element Methods Package in Matlab, University of California at Irvine, 2009.
[23] L. Chen, J. Xu, Optimal Delaunay triangulations, J. Comput. Math. 22 (2) (2004) 299–308.
[24] Z. Chen, J. Zou, Finite element methods and their convergence for elliptic and parabolic interface problems, Numer. Math. 79 (2) (1998) 175–202.
[25] S.-W. Cheng, T.K. Dey, H. Edelsbrunner, M.A. Facello, S.-H. Teng, Silver exudation, J. ACM 47 (5) (Sept. 2000) 883–904.
[26] L.P. Chew, Guaranteed-quality Delaunay meshing in 3D (short version), in: Proceedings of the Thirteenth Annual Symposium on Computational Geometry, ACM, 1997, pp. 391–393.
[27] C.C. Chu, I.G. Graham, T.Y. Hou, A new multiscale finite element method for high-contrast elliptic interface problems, Math. Comput. 79 (2010) 1915–1955.
[28] F. Dassi, S. Perotto, L. Formaggia, P. Ruffo, Efficient geometric reconstruction of complex geological structures, Math. Comput. Simul. 106 (2014) 163–184.
[29] T.S. Developers, SageMath, the Sage Mathematics Software System (Version 7.2), 2016, http://www.sagemath.org.
[30] R.G. Duran, Error estimates for 3-D narrow finite elements, Math. Comput. 68 (225) (1999) 187–199.
[31] R.G. Durán, A.L. Lombardi, Error estimates for the Raviart–Thomas interpolation under the maximum angle condition, SIAM J. Numer. Anal. 46 (3) (2008) 1442–1453.
[32] H. Edelsbrunner, Triangulations and meshes in computational geometry, Acta Numer. 9 (2000) 133–213.
[33] H. Edelsbrunner, X.-Y. Li, G. Miller, A. Stathopoulos, D. Talmor, S.-H. Teng, A. Üngör, N. Walkington, Smoothing and cleaning up slivers, in: Proceedings of the 32nd Annual ACM Symposium on the Theory of Computing, ACM, 2000, pp. 273–277.
[34] H. Edelsbrunner, R. Seidel, Voronoi diagrams and arrangements, Discrete Comput. Geom. 1 (1) (1986) 25–44.
[35] T.-P. Fries, T. Belytschko, The intrinsic XFEM: a method for arbitrary discontinuities without additional unknowns, Int. J. Numer. Methods Eng. 68 (13) (2006) 1358–1385.
[36] T.-P. Fries, T. Belytschko, The extended/generalized finite element method: an overview of the method and its applications, Int. J. Numer. Methods Eng. 84 (April 2010) 253–304.
[37] Y. Gong, B. Li, Z. Li, Immersed-interface finite-element methods for elliptic interface problems with nonhomogeneous jump conditions, SIAM J. Numer. Anal. 46 (1) (2008) 472–495.
[38] S. Goswami, A. Gillette, C. Bajaj, Efficient Delaunay mesh generation from sampled scalar functions, in: Proceedings of the 16th International Meshing Roundtable, Springer, 2008, pp. 495–512.
[39] G. Guyomarc'h, C.-O. Lee, K. Jeon, A discontinuous Galerkin method for elliptic interface problems with application to electroporation, Commun. Numer. Methods Eng. 25 (10) (2009) 991–1008.
[40] J. Guzman, M. Sánchez, M. Sarkis, On the accuracy of finite element approximations to a class of interface problems, Math. Comp. 85 (2016) 2071–2098.
[41] A. Hansbo, P. Hansbo, An unfitted finite element method, based on Nitsche's method, for elliptic interface problems, Comput. Methods Appl. Mech. Eng. 191 (47–48) (2002) 5537–5552.
[42] A. Hansbo, P. Hansbo, A finite element method for the simulation of strong and weak discontinuities in solid mechanics, Comput. Methods Appl. Mech. Eng. 193 (2004) 3523–3540.
[43] P. Hansbo, M.G. Larson, S. Zahedi, A cut finite element method for a Stokes interface problem, Appl. Numer. Math. 85 (2014) 90–114.
[44] X. He, T. Lin, Y. Lin, Immersed finite element methods for elliptic interface problems with non-homogeneous jump conditions, Int. J. Numer. Anal. Model. 8 (2) (2011) 284–301.
[45] S. Hou, X.-D. Liu, A numerical method for solving variable coefficient elliptic equation with interfaces, J. Comput. Phys. 202 (2) (2005) 411–445.
[46] S. Hou, P. Song, L. Wang, H. Zhao, A weak formulation for solving elliptic interface problems without body fitted grid, J. Comput. Phys. 249 (2013) 80–95.
[47] T.Y. Hou, Z. Li, S. Osher, H. Zhao, A hybrid method for moving interface problems with application to the Hele–Shaw flow, J. Comput. Phys. 134 (2) (1997) 236–252.
[48] J. Huang, J. Zou, Some new a priori estimates for second-order elliptic and parabolic interface problems, J. Differ. Equ. 184 (2) (2002) 570–586.
[49] J. Huang, J. Zou, Uniform a priori estimates for elliptic and static Maxwell interface problems, Discrete Contin. Dyn. Syst., Ser. B 7 (1) (2007) 145–170.
[50] J.-S. Huh, J.A. Sethian, Exact subgrid interface correction schemes for elliptic interface problems, Proc. Natl. Acad. Sci. 105 (2008) 9874–9879.
[51] L.N.T. Huynh, N.C. Nguyen, J. Peraire, B.C. Khoo, A high-order hybridizable discontinuous Galerkin method for elliptic interface problems, Int. J. Numer. Methods Eng. 93 (2) (2013) 183–200.
[52] H. Ji, J. Dolbow, On strategies for enforcing interfacial constraints and evaluating jump conditions with the extended finite element method, Int. J. Numer. Methods Eng. 61 (14) (2004) 2508–2535.
[53] A. Johansson, M.G. Larson, A high order discontinuous Galerkin Nitsche method for elliptic problems with fictitious boundary, Numer. Math. 123 (2013) 607–628.

[54] R. Kafafy, T. Lin, Y. Lin, J. Wang, Three-dimensional immersed finite element methods for electric field simulation in composite materials, Int. J. Numer. Methods Eng. 64 (7) (2005) 940–972.
[55] B. Khoo, Z. Li, P. Lin, Interface Problems and Methods in Biological and Physical Flows, World Scientific, 2009.
[56] M. Krizek, On the maximum angle condition for linear tetrahedral elements, SIAM J. Numer. Anal. 29 (2) (1992) 513–520.
[57] D.-T. Lee, B.J. Schachter, Two algorithms for constructing a Delaunay triangulation, Int. J. Comput. Inf. Sci. 9 (3) (1980) 219–242.
[58] R.J. Leveque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (4) (1994) 1019–1044.
[59] R.J. Leveque, Z. Li, Immersed interface methods for Stokes flow with elastic boundaries or surface tension, SIAM J. Sci. Comput. 18 (3) (1997) 709–735.
[60] J. Li, J.M. Melenk, B. Wohlmuth, J. Zou, Optimal convergence of higher order finite element methods for elliptic interface problems, Appl. Numer. Math. 60 (2010) 19–37.
[61] X.-Y. Li, S.-H. Teng, Generating well-shaped Delaunay meshed in 3D, in: Proceedings of the Twelfth Annual ACM–SIAM Symposium on Discrete Algorithms, SODA '01, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2001, pp. 28–37.
[62] Z. Li, K. Ito, The Immersed Interface Method: Numerical Solutions of PDEs Involving Interfaces and Irregular Domains, vol. 33, SIAM, 2006.
[63] Z. Li, T. Lin, X. Wu, New Cartesian grid methods for interface problems using the finite element formulation, Numer. Math. 96 (1) (2003) 61–98.
[64] T. Lin, Y. Lin, X. Zhang, Partially penalized immersed finite element methods for elliptic interface problems, SIAM J. Numer. Anal. 53 (2) (2015) 1121–1144.
[65] X.-D. Liu, T.C. Sideris, Convergence of the ghost fluid method for elliptic equations with interfaces, Math. Comput. 72 (244) (2003) 1731–1746.
[66] R. Löhner, J.R. Cebral, F.E. Camelli, S. Appanaboyina, J.D. Baum, E.L. Mestreau, O.A. Soto, Adaptive embedded and immersed unstructured grid techniques, Comput. Methods Appl. Mech. Eng. 197 (25) (2008) 2173–2197.
[67] R. Massjung, An unfitted discontinuous Galerkin method applied to elliptic interface problems, SIAM J. Numer. Anal. 50 (6) (2012) 3134–3162.
[68] J.M. Melenk, I. Babuška, The partition of unity finite element method: basic theory and applications, Comput. Methods Appl. Mech. Eng. 139 (1) (1996) 289–314.
[69] N. Moës, J. Dolbow, T. Belytschko, A finite element method for crack growth without remeshing, Int. J. Numer. Methods Eng. 46 (1) (1999) 131–150.
[70] R. Moore, S. Saigal, Eliminating slivers in three-dimensional finite element models, Comput. Model. Eng. Sci. 7 (3) (2005) 283–291.
[71] L. Mu, J. Wang, X. Ye, S. Zhao, A new weak Galerkin finite element method for elliptic interface problems, J. Comput. Phys. 325 (2016) 157–173.
[72] J. Nitsche, Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind, Abh. Math. Semin. Univ. Hamb. 36 (2) (1971) 9–15.
[73] P.-O. Persson, G. Strang, A simple mesh generator in Matlab, SIAM Rev. 46 (2) (2004) 329–345.
[74] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.
[75] C. Pflaum, Subdivision of boundary cells in 3d, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.5035, 2000.
[76] C. Pflaum, Semi-unstructured grids, Computing 67 (2) (2001) 141–166.
[77] G. Strang, G.J. Fix, An Analysis of the Finite Element Method, vol. 212, Prentice-Hall, Englewood Cliffs, NJ, 1973.
[78] E. Wadbro, S. Zahedi, G. Kreiss, A uniformly well-conditioned, unfitted Nitsche method for interface problems, BIT Numer. Math. 53 (2013) 791–820.
[79] F. Wang, Y. Xiao, J. Xu, High-order extended finite element methods for solving interface problems, arXiv:1604.06171, 2016, pp. 1–25.
[80] H. Wei, L. Chen, Y. Huang, B. Zheng, Adaptive mesh refinement and superconvergence for two-dimensional interface problems, SIAM J. Sci. Comput. 36 (4) (2014) A1478–A1499.
[81] A. Wiegmann, K.P. Bube, The explicit-jump immersed interface method: finite difference methods for PDEs with piecewise smooth solutions, SIAM J. Numer. Anal. 37 (3) (2000) 827–862.
[82] H. Wu, Y. Xiao, An unfitted *hp*-interface penalty finite element method for elliptic interface problems, arXiv:1007.2893, 2010.
[83] K. Xia, M. Zhan, G.-W. Wei, MIB Galerkin method for elliptic interface problems, J. Comput. Appl. Math. 272 (7) (2014) 195–220.
[84] H. Xie, Z. Li, Z. Qiao, A finite element method for elasticity interface problems with locally modified triangulations, Int. J. Numer. Anal. Model. 8 (2) (2011) 189.
[85] J. Xu, Error estimates of the finite element method for the 2nd order elliptic equations with discontinuous coefficients, J. Xiangtan Univ. 1 (1982) 1–5.
[86] J. Xu, Estimate of the convergence rate of finite element solutions to elliptic equations of second order with discontinuous coefficients, Nat. Sci. J. Xiangtan Univ. 1 (1) (1982) 1–5.
[87] S. Yu, G.W. Wei, Three-dimensional matched interface and boundary (MIB) method for treating geometric singularities, J. Comput. Phys. 227 (1) (2007) 602–632.
[88] X. Zhang, Nonconforming Immersed Finite Element Methods for Interface Problems, PhD thesis, Virginia Polytechnic Institute and State University, 2013.
[89] X. Zheng, J. Lowengrub, An interface-fitted adaptive mesh method for elliptic problems and its application in free interface problems with surface tension, Adv. Comput. Math. (2016) 1–33.
[90] Y.C. Zhou, S. Zhao, M. Feig, G.W. Wei, High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources, J. Comput. Phys. 213 (1) (2006) 1–30.