Comput. Methods Appl. Mech. Engrg. 200 (2011) 774–796

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

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

Finite element analysis on implicitly defined domains: An accurate representation based on arbitrary parametric surfaces

Mohammed Moumnassi [a,b], Salim Belouettar [a,*], Éric Béchet [c], Stéphane P.A. Bordas [d], Didier Quoirin [e], Michel Potier-Ferry [b]

[a] Centre de Recherche Public Henri Tudor, 29, Avenue John F. Kennedy, L-1855 Luxembourg, G.D. of Luxembourg, Luxembourg
[b] Laboratoire d'Étude des Microstructures et de Mécanique des Matériaux, LEM3, CNRS, Université Paul Verlaine – Metz, Ile du Saulcy, 57045 Metz, France
[c] LTAS – Department of Aerospace and Mechanical Engineering, Université de Liège, Chemin des Chevreuils 1, 4000 Liège, Belgium
[d] Cardiff School of Engineering, Institute of Modelling & Simulation in Mechanics & Materials, Cardiff University, Queen's Buildings, The Parade, Cardiff CF24 3AA, Wales, UK
[e] Goodyear Innovation Center, L-7750 Colmar-Berg, Luxembourg

ARTICLE INFO

Article history:
Received 27 August 2010
Received in revised form 1 October 2010
Accepted 5 October 2010
Available online 27 October 2010

Keywords:
Implicit boundary representation
Level set method
Curved boundary and sharp edges
eXtended Finite Element Method
CAD-analysis
Dirichlet

ABSTRACT

In this paper, we present some novel results and ideas for robust and accurate implicit representation of geometric surfaces in finite element analysis. The novel contributions of this paper are threefold: (1) describe and validate a method to represent arbitrary parametric surfaces implicitly; (2) represent arbitrary solids implicitly, including sharp features using level sets and boolean operations; (3) impose arbitrary Dirichlet and Neumann boundary conditions on the resulting implicitly defined boundaries. The methods proposed do not require local refinement of the finite element mesh in regions of high curvature, ensure the independence of the domain's volume on the mesh, do not rely on boundary regularization, and are well suited to methods based on fixed grids such as the extended finite element method (XFEM). Numerical examples are presented to demonstrate the robustness and effectiveness of the proposed approach and show that it is possible to achieve optimal convergence rates using a fully implicit representation of object boundaries. This approach is one step in the desired direction of tying numerical simulations to computer aided design (CAD), similarly to the isogeometric analysis paradigm.

1. Introduction

Although mesh generation and regeneration is today a well researched area with robust and effective mesh generators, the generation of quality meshes for complex geometries remains a challenge [1]. When the geometry of the domain of interest is complex or when internal boundaries evolve in time, meshing and remeshing these surfaces is not straightforward.

Alternative methods have been proposed which aim at decoupling the representation of geometrical features from the discretization, *e.g.* the finite element mesh. Partition of unity enriched discretizations [2,3] serve, among others, this very purpose by enabling the simulation of moving boundaries independently of the background mesh.

In this paper, we are not concerned with modelling moving boundaries, but, rather, simplifying the description of the boundary of the solid, assumed here to be subjected to small displacements. A large number of researchers have investigated a variety of techniques aiming at solving partial differential equations (PDEs) on fixed grids which do not conform to the boundary. We provide here a non exhaustive list: immersed boundary [4], fictitious domain [5], embedded boundary [6], virtual boundary [7] and Cartesian grid [8] methods. All these methods hold are promising tools to reduce the difficulties associated with meshing complex domain boundaries [9] and remeshing [10].

Fixed grid methods (FGM) can be considered as "mesh-free" finite element methods where the computational mesh can be defined on a region of simple shape (say rectangular or hexahedral) which contains the domain over which the PDE is to be solved. The mesh need not conform to the potentially complex geometry of the boundary, but this flexibility is paid back by the necessity to determine which elements are inside, outside, or cut by the boundary and carry out numerical integration over each element type separately to ensure that void and full regions are accounted for correctly, impose boundary conditions on a surface which splits elements arbitrarily, etc. The step of identifying which elements are inside, outside, and cut by the boundary is known, in an extended finite element context, as "mesh-geometry interaction" [11,12].

Most methods relying on a fixed grid differ from each other on the following points:

* Corresponding author. Tel.: +352 54 55 80 530; fax: +352 42 59 91 333.
E-mail address: salim.belouettar@tudor.lu (S. Belouettar).

0045-7825/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2010.10.002

Construction of the domain on the background mesh grid. Two sets of techniques have been proposed in the literature to represent the object's boundary: explicit, or implicit. Explicit definitions comprise, for example, polygonal meshes. Examples of techniques to represent boundaries implicitly include

- The level set method [13].
- *R*-functions [14] or signed distance functions [15], which are particularly well suited to domains constructed with Boolean operations such as union, intersection, difference. Such Boolean operations are used in constructive solid geometry (CSG) [16], providing a straightforward way to create complex solid objects by combining simpler primitives.

A number of input data can be used to generate the implicit representation of a given domain for analysis on a structured fixed grid. The geometrical information can be, for example, provided by a polygonal mesh (typically obtained from STL (stereolithography) files of CAD models [17–20] or other CAD descriptions [21]). Medical image modalities [17,18,22] may also be used.

Numerical integration schemes. The most common method is subdivision into subtriangles [23] located on either side of the boundary and using standard Gaussian quadrature on each of the triangular subcells. Alternate techniques to integrate on arbitrary polygons [24,25] or transform domain integration into boundary integration [26] can also be used.

Imposition of essential (Dirichlet) boundary conditions. To enforce the Dirichlet boundary conditions along a boundary which does not necessarily contain nodes of the fixed grid, several methods are available in the literature. They are either based on the Lagrange multiplier method (LMM) [27–30], on a consistent penalty method, such as Nitsche's method [31–34]; or rely on modifying the basis functions to satisfy the constraints directly [35,17,36].

The concept of modelling geometrical features independently of the finite element mesh in the context of the extended finite element method (XFEM) [23] originated in Sukumar et al. [37]. Analytical level set functions (signed distance functions) are used to represent the location of internal surfaces and the authors show that imposing Neumann type boundary conditions is straightforward. This method was later used to model complex geometries with fixed grid avoiding elaborate meshing schemes [15].

Several other authors have also used an implicit surface definition for both external and internal boundaries using a structured grid, *e.g.* [9,36,17,20,21]. The location of the free surface is approximated on the background fixed mesh through piecewise linear interpolation of the grid data. In general, inside each element, the approximated surface is a straight line segment in two dimensions and a plane triangle in three dimensions. As a consequence, the accuracy with which a boundary can be approximated in this way depends on the refinement of the mesh. In order to alleviate this difficulty, a strategy for reducing the geometrical representation errors is described in Moës et al. [15]. The underlying mesh employed to construct surface geometry was obtained by an adaptive mesh refinement in order to carefully locate the curved regions and sharp features.

However, even when a fine mesh is used around the object surfaces, the approximation results in an object with rounded edges whose corners cannot be approximated exactly. Moreover, the approximability of curved boundaries is determined by the order of the model to be approximated. Moreover, when the design changes, high-order approximation of curved boundary is required to be improved with the underlying mesh grid (*e.g.* crack, topology optimization with implicit functions). In these circumstances, the background mesh grid is not allowed to be changed: to represent smoothly the boundary (high-order approximation of the object boundary inside the mesh grid) and during the computation when the geometry and topology changes.

To preserve such details, [36,38–40] proposed to use a finer mesh within the initial background grid/mesh to represent curved boundaries. This finer mesh was also used for Gaussian integration. The idea of this method is that a finer mesh is used to represent the implicit functions describing the boundary within each element of the fixed background grid, regardless of the finite elements used for the analysis process. Higher order finite element discretizations may also be used to define curved boundaries implicitly on structured grids as well as for the analysis process [41,42].

Explicit surface representations independent of the mesh for 3D crack growth with hp-GFEM and interfaces in fluid–structure interaction with higher-order XFEM have been presented in the works of Pereira et al. [43] and Mayer et al. [44], respectively.

In a nutshell, while representing the boundary of a computational domain without meshing it explicitly is attractive, it also poses a number of difficulties, namely:

- Representing the boundary implicitly from raw data obtained from CAD, medical images, etc.
- Performing the mesh-geometry interaction.
- Enforcing essential boundary conditions on the boundary.

In this paper, we will present a novel approach to addressing these three challenges. The salient features of this method are as follows:

- Representing an arbitrary object based on a fixed background grid, similarly to the method proposed by Belytschko et al. [9] in an XFEM framework.
- An automatic conversion from parametric surfaces to zero level sets for general unstructured meshes (triangular and tetrahedral).

There are two motivations for using parametric representations:

- The use of parametric information can control geometrical errors at the boundaries, which affect the convergence rate and solution accuracy, without changing the underlying fixed mesh for analysis.
- They are standard in computer aided design and manufacturing CAD/CAM systems.

The main idea is to use several zero level sets for defining accurate and controllable sharp features obtained from the parametric primitives which compose the object, and to use parametric information as a guide to generate the profile of the curved region inside a finer graded mesh, incorporated into the elements split by the boundary. This sub-mesh is only used for numerical integration purposes, not to increase the approximation power of the finite element discretization.

The proposed representation guarantees a priori the desired approximation of the original object and also provides efficient numerical integration where integrals over the volume and boundary surfaces are based on standard Gauss quadrature.

The Dirichlet boundary conditions are applied by building appropriate Lagrange multiplier space on the boundary using the sophisticated algorithms proposed by Moës et al. [27], Béchet et al. [29,28] and Géniaut et al. [28].

The remainder of this paper is organized as follows. The details on implicit representation of object boundaries with a single level set function and its limitations are presented in the next section. Section 3, describes the concept for building objects from parametric definitions of primitive shapes. The algorithm for automatic

conversion from a parametric surface into a zero level set defined on a narrow band of the mesh is described in Section 4. Section 5 describes how sharp features can be preserved, and how the finer graded mesh is constructed inside the split elements for integration purposes. We briefly discuss the problem of enforcing Dirichlet boundary conditions in Section 6. Section 7 provides several numerical examples that illustrate the convergence of the proposed technique. Finally, Section 8 provides conclusions and directions for future work.

2. Conventional level set modelling for an implicit representation of object boundaries

In this section, we define the basic notations and methodologies used in this paper. Let Gr denote the smallest bounding box that completely encloses the object $\Omega \subseteq Gr \subset \mathbb{R}^n$ ($n = 2$ or 3). The boundary $\Gamma = \partial\Omega$ of Ω is of dimension $(n-1)$. Gr is a mesh, either structured or unstructured (triangles in 2D and tetrahedra in 3D). Elements in this mesh are denoted generically by E and are of characteristic size h. Although we only discuss here the case where there are two phases (material and void), the idea can easily be extended to multi-phase material. In this article, all background meshes and figures are produced with the help of *GMSH* [45].

2.1. Level set modelling

The level set method introduced by Osher and Sethian [13] is an interface capturing method (as opposed to interface tracking) which is widely used to describe the evolution of surfaces without requiring their explicit discretization by mesh points. It is more particularly adapted to representing closed boundaries, although it has been used successfully to describe evolving 3D cracks using the XFEM [46].

This method defines the object implicitly using a scalar function $\phi(\underline{x})$ mapping the space where the interface is to be described to the real line, $\mathbb{R}$. The interface is then one of the isolines of function ϕ, *i.e.* it is obtained by cutting the graph of ϕ at different heights. A natural choice for ϕ is the signed distance function to the interface of interest, Γ. Using the signed distance function:

- the interior of the domain bounded by Γ is defined by the set $\{\underline{x} \in \mathbb{R}^n / \phi(\underline{x}) \leqslant 0\}$;
- the exterior is the set $\{\underline{x} \in \mathbb{R}^n / \phi(\underline{x}) \geqslant 0\}$;
- and the boundary Γ is defined by $\{\underline{x} \in \mathbb{R}^n / \phi(\underline{x}) = 0\}$ *i.e.* the zero level set.

In some very special cases, an analytical expression for the level set function of an object can be obtained easily, for example:

- for a plane $p(\underline{x}) = (\underline{x} - \underline{x}_0) \cdot \underline{n}$ where $\underline{x}_0$ is the projection of point $\underline{x}$ on the plane and $\underline{n}$ is the unit normal defining the plane;
- a sphere $s(\underline{x}) = \|\underline{x} - \underline{x}_0\| - r_0$, where r_0 is the radius of the cylinder and $\underline{x}_0$ is the center of the sphere;
- an infinite cylinder $c(\underline{x}) = \|\underline{x} - \underline{x}_\star\| - r_0$, where $\underline{x}_\star$ is the projection of $\underline{x}$ on the director of the cylinder of radius r_0.

In general, however, ϕ cannot be easily defined analytically, and its numerical value must then be evaluated at each node of the mesh. This process is known as initialization of the level set function. Then, the underlying shape functions associated with the mesh can be used to obtain the value of the level set function anywhere within the domain using the nodal values, through simple interpolation. This is known as the discretization of the level set. Of course, the smoothness of the resulting surface (3D) (or curve (2D)) is directly related to that of the shape functions constructed on the mesh.

Based on this representation $\phi(\underline{x}) = \pm dist(\underline{x}, \Gamma)$, the final object can be represented as Boolean operations (using min/max operators) on simpler half-spaces of level set functions ϕ^i, *e.g.*

$$\text{Intersection}: \quad \underline{x} \in \Omega^1 \cap \Omega^2 \Longleftrightarrow max\{\phi^1(\underline{x}), \phi^2(\underline{x})\}, \tag{1}$$

$$\text{Union}: \quad \underline{x} \in \Omega^1 \cup \Omega^2 \Longleftrightarrow min\{\phi^1(\underline{x}), \phi^2(\underline{x})\}, \tag{2}$$

$$\text{Difference}: \quad \underline{x} \in \Omega^1 / \Omega^2 \Longleftrightarrow max\{\phi^1(\underline{x}), -\phi^2(\underline{x})\}. \tag{3}$$

In the remainder of this paper, we shall use the convention that the level set function is negative inside domain Ω, positive outside and zero on the boundary $\partial\Omega$.

2.2. Element classification based on level set data

Using the sign of $\phi(\underline{x})$, we can categorise the elements E in the mesh Gr into three sets: interior elements E_I are those which are completely inside Ω, *i.e.* not intersected by $\partial\Omega$; exterior elements E_O which are completely outside Ω; boundary elements E_B which are split by $\partial\Omega$. Mathematically, this may be written:

$$\text{Interior}: \quad I = \{E_I \in Gr \text{ such that } E \subset \Omega \text{ and } E \cap \partial\Omega = \emptyset\}; \tag{4}$$

$$\text{Exterior}: \quad O = \{E_O \in Gr \text{ such that } E \subset \overline{\Omega}_h \text{ and } E \cap \Omega = \emptyset\}; \tag{5}$$

$$\text{Boundary}: \quad B = \{E_B \in Gr \text{ such that } E \cap \partial\Omega \neq \emptyset\}. \tag{6}$$

This classification may be usefully recast in terms of the level set function ϕ:

- Interior elements E_I are such that all their nodes lie within Ω. Hence they are the elements in the mesh such that the value of the level set function at all their nodes is negative.
- Exterior elements E_O, on the other hand are elements for which the level set function is positive at all of their nodes.
- Boundary elements E_B are such that the level set function is positive at some of their nodes, and negative at others.

2.3. Discretization of the level set and definition of sub-elements for numerical integration

Using the finite element shape functions to discretize the level set function ϕ, we obtain an approximation ϕ_h to the exact distance field ϕ and a corresponding boundary defined by $\partial\Omega_h$ which approximates $\partial\Omega$ (see Fig. 1).

The approximate zero level set $\phi_h = 0$ is obtained using the finite element shape functions associated with the mesh. This is useful, for example, to compute the intersection points of the zero level set with the element edges. To do this, it is sufficient to use the underlying shape functions to interpolate the level set function values along the element edges.

Once the intersection points between the zero level set and the element edges are known, the discretized (approximate) boundary $\partial\Omega_h$, may be obtained by constructing a line mesh obtained by joining the intersection points previously obtained. Assuming the FE shape functions are piecewise linear, this line mesh is a polyline in 2D and a triangulated surface in 3D.

The boundary elements $E_B \in B$ are further subdivided into sub-elements whose edges conform with the discretized boundary $\partial\Omega_h$. These can be separated into two sets, those located inside the discretized domain Ω_h which we designate by (IB), and those located outside, which we name exterior (OB) sub-elements. The original element $E_B \in B$ can thus be rewritten as the union of sub-elements E_Δ such that $E_B = \bigcup_{k=1}^{n} E_\Delta = \{E_{IB}\}_{n-m} \cup \{E_{OB}\}_m$ (see Fig. 2c). The set of sub-elements (IB) and (OB) are defined as

$$B = IB \cup OB = \{E_{IB} \textit{ sub-element} \in E_B : E_\Delta \subset \Omega_h\} \cup \{E_{OB} \textit{ sub-element} \in E_B : E_\Delta \subset \overline{\Omega}_h\}.$$

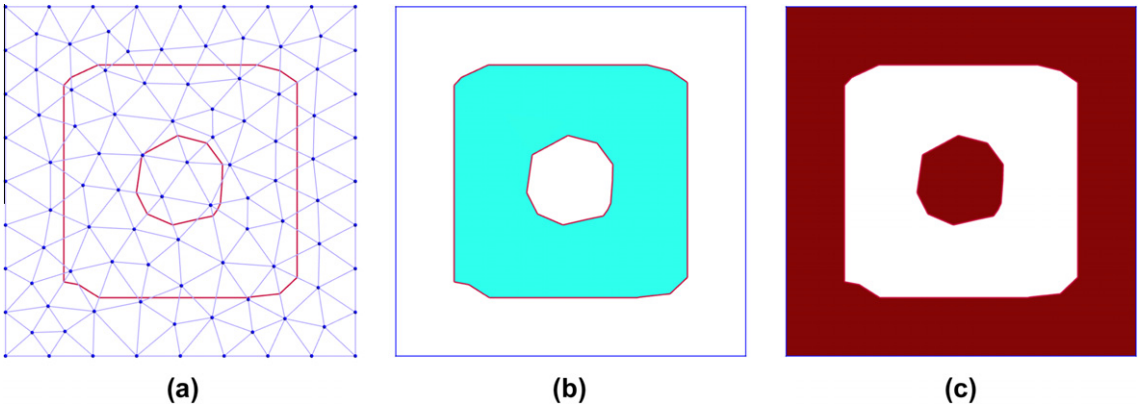

Fig. 1. Approximation of a square with a circular hole on a triangular mesh (a) shows the zero level set $\phi_h = 0$, resulting from the Boolean combination of half-spaces defined using analytical level set functions (four planes and cylinder). (b) Approximate domain Ω_h with mesh of size h. Second material part or void part shown in (c).

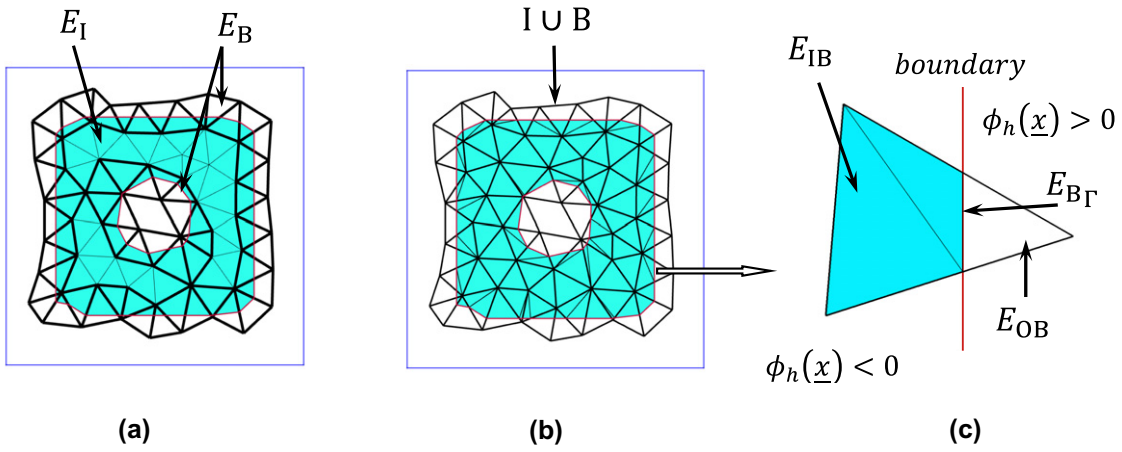

Fig. 2. (a) Definition of boundary elements E_B and interior elements E_I. (b) Part of mesh $Gr_{I\cup B}$ that fully covers the object. (c) Representation of material and void by partitioning E_B into interior sub-elements E_{IB} and exterior sub-elements E_{OB}. In this particular case of boundary element, $E_B = \bigcup_{k=1}^{3} E_\Delta = \{E_{IB}\}_2 \cup \{E_{OB}\}_1$ and $E_{B_\Gamma} = \bigcup_{k=1}^{1} E_{\Delta_\Gamma}$.

Because all sub-elements are known to be either inside or outside of the domain (they cannot be split by the interface), the value of sign (ϕ) at the centroid of a sub-element E_Δ is sufficient to determine its position relative to the interface. A negative value means that the sub-element is interior to the domain, and vice versa.

Sub-elements of an interior boundary element *IB* are located within the domain $E_\Delta = E_{IB}$ whereas the sub-elements of an exterior boundary element *OB* are located outside $E_\Delta = E_{OB}$.

We denote the part of the boundary $\partial\Omega_h$ inside a boundary element E_B by E_{B_Γ} such that $E_{B_\Gamma} \subset E_B$. We next show that in a general case, E_{B_Γ} can be rewritten as the union of sub-elements E_{Δ_Γ} such that $E_{B_\Gamma} = \bigcup_{k=1}^{n} E_{\Delta_\Gamma}$ (see Fig. 2c for a two-dimensional case and Fig. 9 for a three-dimensional case).

The interior of the object, to be considered for the analysis is then defined by the union of the interior elements (*I*) with the interior sub-elements (*IB*).

The level set description presented above is able to represent two-phase materials, *e.g.* $\phi > 0$ can represent voids, $\phi = 0$ the boundaries of the voids and $\phi < 0$ the region where actual material is present. The same method is obviously applicable for two-material systems. In the case where voids are modelled, all sub-elements located inside the void are discarded during the analysis since they do not contribute to the stiffness. In the case of multi-material systems, the integration points generated in interior sub-elements are attributed the material property associated with that of the "interior" material (see Fig. 2b).

2.4. Limitations of the single level set modelling

In the single level set framework defined above, a single level set function is used to define the whole domain as well as its boundary conditions, through Boolean combinations of simple half-spaces. It has been shown that a single level set model offers several advantages compared to the actual meshing of the domain boundary [15]. However, as illustrated in examples (see Figs. 1 and 3a), this approach has some drawbacks:

- It is not possible to represent sharp edges or corners exactly, as shown in Fig. 1. Corners will be "cut" because kinks are not allowed in the underlying finite element approximation used to discretize the level set function. As a consequence, it is not possible to ensure that the object is exactly reproduced by simple combinations of half-spaces.
- In order to reproduce the geometry accurately, significant mesh refinement is typically needed.
- Because the whole boundary is defined using one single function, it is not straightforward to locate and separate different regions on $\partial\Omega_h$ for attribution of appropriate boundary conditions.
- To efficiently approximate a curved domain, one generates a discrete approximation of the scalar distance field ϕ by evaluating the function on a sufficiently fine mesh, or by adaptive schemes like octree techniques to capture details of the domain

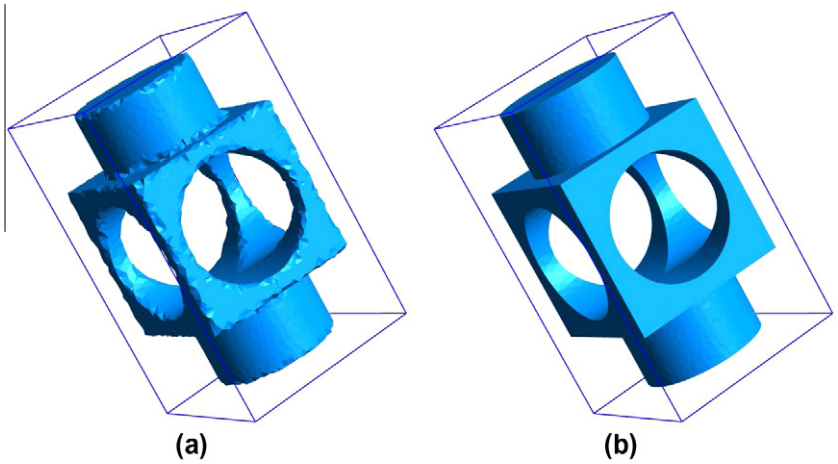

Fig. 3. Approximation of an object with convex and concave boundaries with the same background mesh, resulting from Boolean combinations of half-spaces defined using analytically defined level set functions (8-planes and 3-cylinders). (a) The object is constructed by a single level set resultant from Boolean operations (one scalar distance value is stored at each node) and (b) shows the approximation by our new approach that preserves sharp features (11 scalar distance values are stored at each node).

boundary $\partial\Omega_h$. However, linear interpolation of the mesh values to approximate the boundary is insufficient for higher order analysis.

In the following section, we present a new approach to represent arbitrary regions using level set functions, which alleviates the pitfalls of the "single-level-set-description".

3. Multiple level sets modelling for an implicit representation of object boundaries

3.1. Description of the procedure

As stated in the introduction, one of the goals of this paper is to present a new implicit representation that can ensure the accuracy of the approximated object defined over an unstructured mesh with minimal dependence on this background mesh. Our proposed framework is an adaptation of the single-level-set modelling technique described above, but to maximize flexibility, the new approach combines the strengths of this single-level-set description with those of popular parametric representations. The main steps involved in our implicit modelling scheme may be separated into three categories:

- Conversion of a smooth parametric function into a signed distance field.
- Subdivision of the mesh.
- Classification of the elements.

Let Ω be the solid under consideration and assume that $\partial\Omega$ can be partitioned into m distinct "faces." Consider now m parametric functions $\vec{S}_i(u,v):\mathbb{R}^2\rightarrow\mathbb{R}^3 (i=1,\ldots,m)$ describing each of these surfaces.

The basic procedure which we propose to follow to obtain a fully implicit representation of solid Ω is as follows:

Convert each parametric function $\vec{S}_i$ into a signed distance field ϕ_h^i in a selected narrow band ω_i [37] defined on the mesh as discussed in the following section. Using a narrow band allows to define the shortest distance information only where it is needed, in the vicinity of the boundaries, *i.e.* around the zero level set. There are m bands $\omega_i (i=1,\ldots,m)$ of elements, where the approximation of each distinct zero level set $\phi_h^i=\left\{\underline{x}\in\omega_i:\phi_h^i(\underline{x})\right\}$ comprises the entire domain boundary $\Gamma_h=\partial\Omega_h$. The nodal values of the signed distance fields ϕ_h^i (level sets) are interpolated using linear finite element shape functions N_j as $\phi_h^i=\sum_{j\in\omega_i}\phi_{ij}N_j$, where ϕ_{ij} is the signed distance for the jth node of ω_i to Γ_h^i.

Construct a polyline/triangulated surface mesh Γ_h^i from each narrow band. These are composed of straight line segments in two dimensions (called cut edge) and triangles in three dimensions (called cut triangle). These Γ_h^i are used to subdivide the elements belonging to each band ω_i for the purpose of numerical integration.

Classification of the elements into interior (I), boundary (B) and interior boundary (IB) sub-elements to define the approximated domain Ω_h and its boundary Γ_h.[1]

3.2. Illustration on a simple example

For the purpose of illustration, let us consider the same example as in (Fig. 1). In this case, we use five parametric functions as illustrated in (Fig. 4) corresponding to each of the distinct boundaries.

4. Conversion of an arbitrary parametric function into a level set (signed distance field)

4.1. A hybrid parametric/implicit representation

In geometric modelling, two methods coexist to represent surfaces: parametric and implicit representations. Each of them is particularly well suited for certain applications. Using parametric representations, it is easy to generate vertices on the surface that are required for volume meshing algorithms based on Delaunay triangulations or advancing front methods. On the other hand, implicit representations simplify mesh generation, since the latter need not conform to the boundary of the domain.[2]

Each of these two approaches, however, has its own disadvantages. The quality of meshes generated from parametric representations is constrained by the quality of the polygonal surface given

[1] Section 4 describes two different types of boundary elements E_B: those which are intersected by more than one level set and those elements that contribute to the representation of the local boundary curvature.

[2] The reader will have noted that sub-elements *do* need to conform to this boundary. However, there is *no* constraint posed on the quality of these elements, since they are only used to generate Gauss points to integrate the weak form, which differentiates them from finite elements.

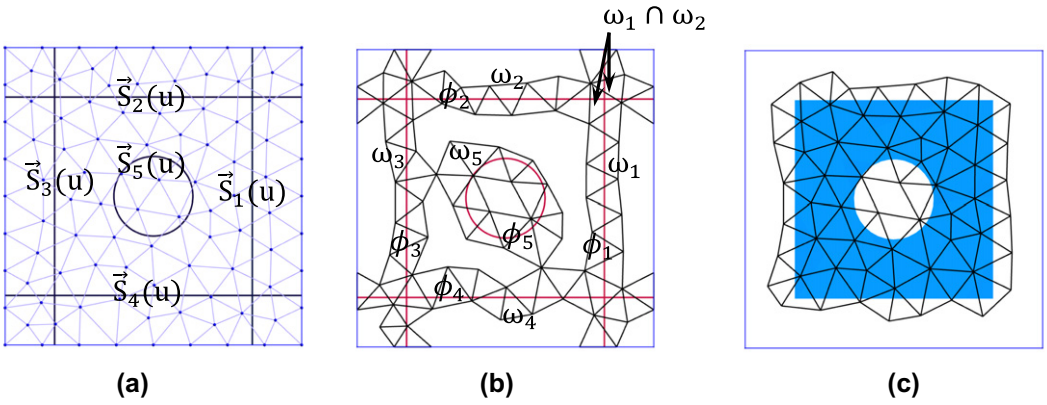

Fig. 4. (a) A parametric definition of a square with a circular hole converted into level sets which implicitly define the object and (b) shows the zero level sets $\phi_h^i = 0$ and their corresponding narrow bands ω_i. (c) Approximate domain Ω_h with the same mesh than in Fig. 1 and the second material part or void part.

as input. In the case of implicit representations, the quality of the approximated boundary is constrained by the quality of the background mesh used for its representation.

The method we propose here is a hybrid which exploits the advantages of the parametric and implicit representations. To control the quality of the boundary representation and of the resulting volume, we propose to work directly with the parametric surfaces and suppress the intermediate polygonisation step by directly converting this representation into an implicit form defined on the background mesh.

4.2. Generating the minimum distance map to an arbitrary parametric surface

The most widely used CAD systems are based on parametric surfaces. However, it is difficult to convert parametrically represented surfaces into an accurate implicit object in an unstructured mesh. This conversion requires computing the minimum distance values of each parametric surface to the mesh vertices. In general, it is necessary to minimize (for each vertex $V(\underline{x})$ to be projected onto the surface $S(u,v)$) the expression $d_{min} = \|S(u,v) - V(\underline{x})\|$, where the closest point $P(\underline{x})$ is on the surface and the normal vector $\vec{n}$ at $P(\underline{x})$ must satisfy $(P(\underline{x}) - V(\underline{x})) \cdot \vec{n} = 0$.

Newton-type techniques can be used with suitable start values in the parameter space to achieve convergence. In general, these initial values are hard to obtain for finding the closest point (foot point) on the parametric curve/surface and for computing the corresponding parameters $u/(u,v)$ of the projection (inversion).

This is why alternate algorithms have been proposed to solve the projection and inversion problems. Hartmann [47] proposed a first order derivative by using a normal form function to compute the foot point. Ma and Hewitt [48] proposed a method that provides a good initial value for the Newton–Raphson method to increase its stability. Since we are dealing with parametric surfaces and a background mesh, we can evaluate implicitly the minimum distance at the vertices of a band surrounding the parametric surface. Distance computation at a particular vertex in this band does not need to be computed by minimizing the quantity above. Instead of finding the footpoint on the parametric curve/surface and computing the corresponding parameter values of the projection, this minimization has been replaced by a technique for computing vertices lying exactly the intersection between a surface $S(u,v)$ and the edges of the (simplex) elements and evaluating the distance directly by using the analytical level set definition of a plane $p(\underline{x}) = (\underline{x} - \underline{x}_0) \cdot \underline{n}$ that passes through each intersection point. Given the intersection points in the mesh, the shortest distance from the vertices of the elements to the analytical definition of the plane can be evaluated.

It is well known that the pure algebraic Newton–Raphson iterative method is unsuitable for the computation of intersection points between a parametric surface and an edge, a good initial value must be very carefully selected to ensure convergence. For this purpose, we use adjacency relations between mesh entities (the relationship between the parametric coordinates on the surface of an intersection point and the corresponding edge) to provide a good initial value to compute the next intersection point. These adjacency relations are also used to define the narrow band.

4.3. Proposed algorithm

The goal of the algorithm we propose here is to enable the computation of the signed distance field at any point in the computational mesh without explicitly calculating the distance between the parametric surface and the vertices of the mesh. The algorithm can be viewed as a marching method taking as input the equation of the parametric surface and an unstructured background mesh and providing as output the signed-distance function defined on a narrow band (or, alternatively, on the whole mesh) corresponding to the parametric surface. The conversion from the parametric surface to the signed-distance function is made piecewise, starting from one or more seed elements which are intersected by the surface of interest.

Given a parametric surface and background mesh of characteristic size h, our goal is to generate a discrete signed distance field in a narrow band surrounding the input surface. Figs. 5–8 illustrate the different steps of the algorithm.

We summarize the algorithm as follows.

Input: Parametric surface, background mesh, start values (u_s, v_s), and a start vector $\underline{n}_s$ to define the sign of both half-spaces.
Output: Level set ϕ_h^i in a narrow band ω_i, and a polygonal mesh Γ_h^i.

Note: below, we use the abusive notation "edge" to refer both to the 2D and to the 3D case to simplify the explanation. In the 3D case, edge should be understood as "face." Similarly, we assume that the background mesh Gr is made up of simplex elements, *i.e.* triangles in 2D and tetrahedra in 3D.

1. Choose initial values of the parameters (u_s, v_s) and the corresponding point P_s on the surface. Search for the start element in the mesh, *i.e.* the element containing point P_s (see Fig. 5).
2. For each edge in the start element, compute the intersection point of the surface with the edge using the initial values (u_s, v_s), and establish the relationship between this edge and

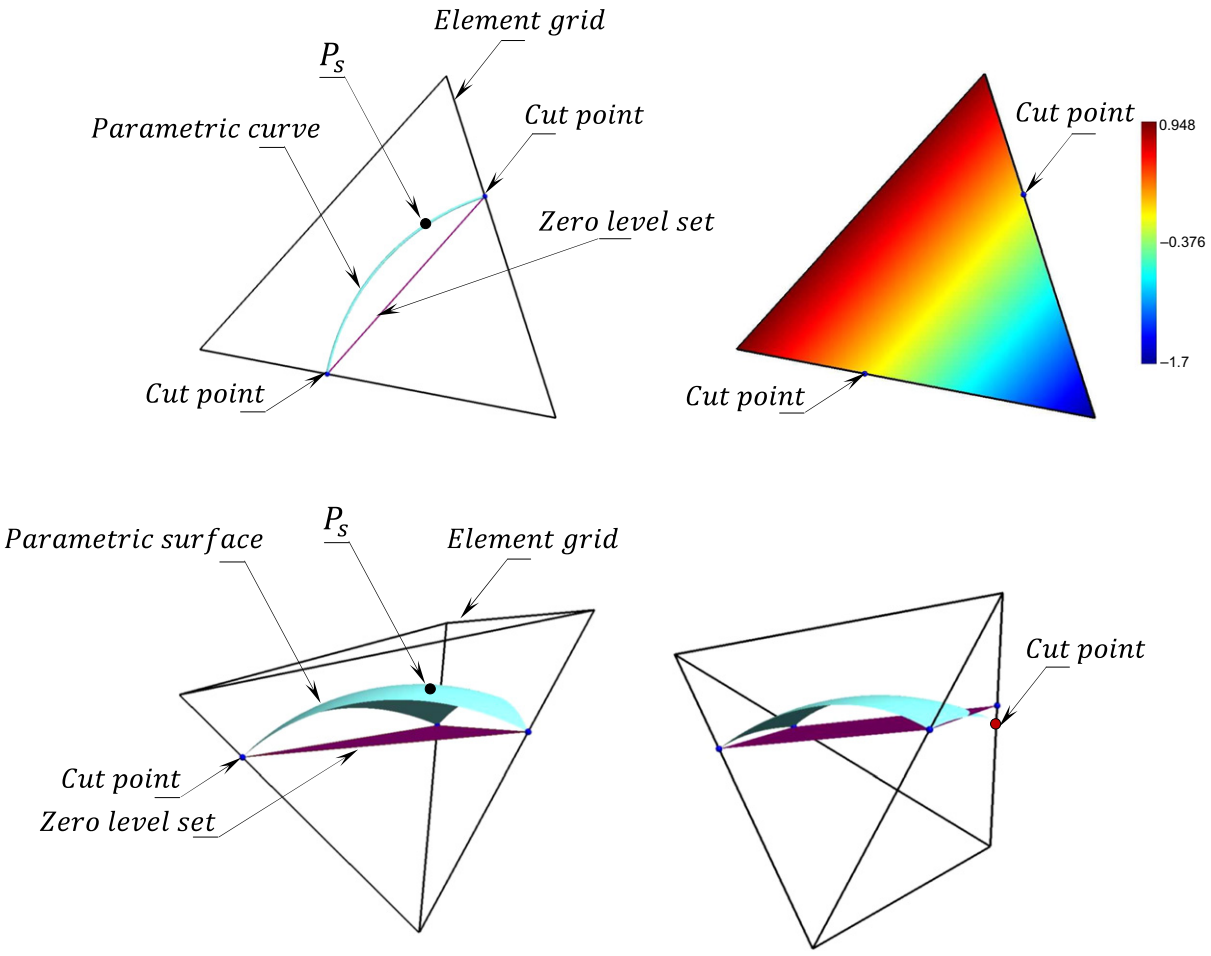

Fig. 5. Steps 1–3 of the algorithm. The start element in 2D and 3D. Evaluating the level set function $p(\underline{x})$ that passes through the intersection points to construct a local piece of the zero level set.

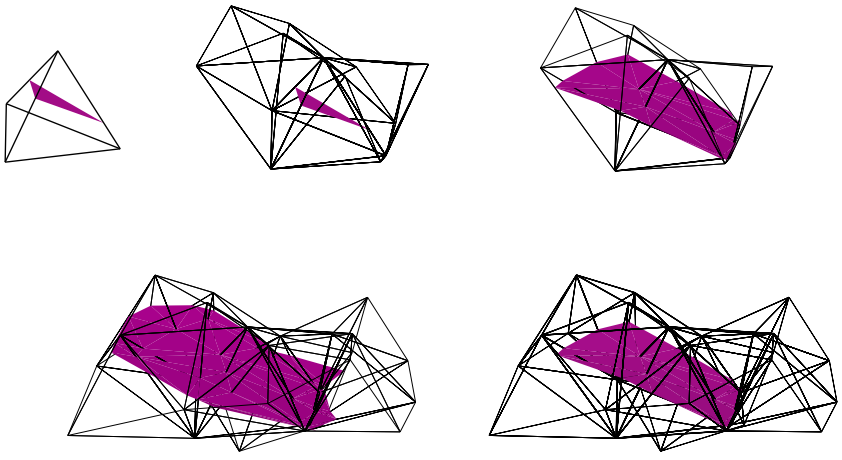

Fig. 6. The first steps of the algorithm: beginning from the start element.

the parameter values for the surface corresponding to this intersection point. Each parameter value is stored in an associative container[3] (called *Container*$_s$) such that they can be found from its corresponding edge. In the particular case, when the intersection point is located on a vertex of the start element, the parameter values can be found from all edges connected to this vertex (two edges in 2D and three edges in 3D).

3. Build the plane $p(\underline{x}) = (\underline{x} - \underline{x}_0) \cdot \underline{n}$ that passes through each intersection point and such that $\underline{n}$ verifies $\underline{n} \cdot \underline{n}_s > 0$ to define the sign

[3] In the associative container, the parameters values (u, v) on the parametric surface corresponding to the intersection points with each edge are accessed by the *key* = *edge* and are not necessarily arranged in any order.

Fig. 7. The steps of the marching method to convert a parametric function into a zero level set.

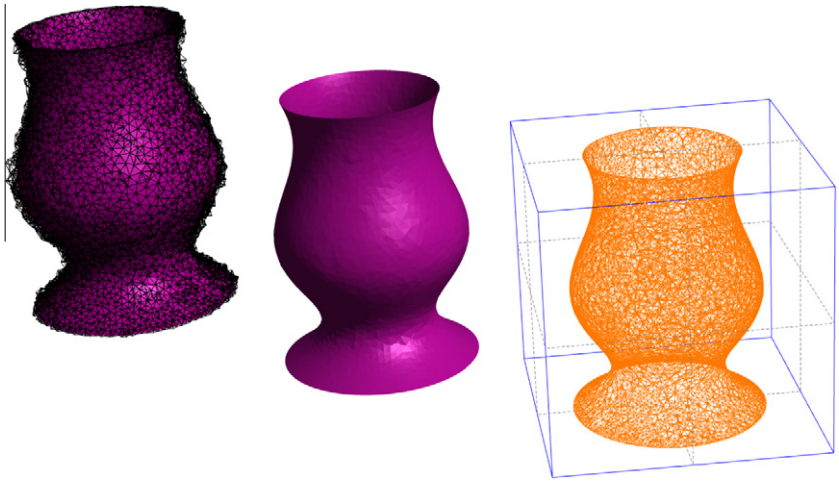

Fig. 8. Conversion of a parametric shape into a signed distance field on one narrow band ω (left). (center) Approximation of the corresponding zero level set $\phi_h = 0$ by linear interpolating the narrow band values. (right) Triangles are extracted to approximate the boundary Γ_h.

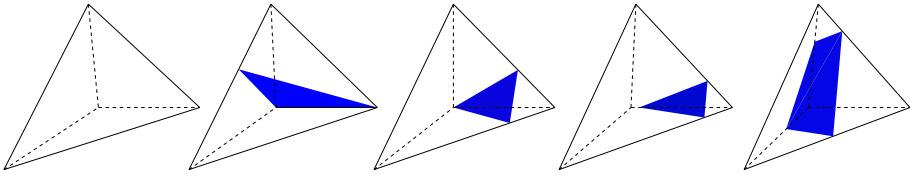

Fig. 9. The four possibilities for the subdivision of a boundary element E_B into sub-elements E_Δ (tetrahedra) by a portion of the zero level set (a triangle or a quadrilateral). In the case of a quadrilateral, the portion of the zero level set is split into two triangles (*i.e.* $E_{B_\Gamma} = \bigcup_{k=1}^{2} E_{\Delta_\Gamma}$) and then E_B is subdivided into six sub-elements (*i.e.* $E_B = \bigcup_{k=1}^{6} E_\Delta$).

of the distance field in the start element. Then evaluate and store the value of p at each vertex in the start element. $\underline{n}$ is stored in an associative container (called $Container_n$) such that it can be found from its corresponding element. The start element is stored in a list (called $list_B$) for building elements in the narrow band ω_i (see Fig. 5).

4. Obtain the location of elements surrounding the start element by using the adjacency relations between mesh entities (see Fig. 6). For each edge of the starting element in $list_B$ and already stored in $Container_s$, store elements adjacent to this edge in a list (called $list_{adj}$). In the particular case when the intersection point is located on a vertex of the start element, store elements around this vertex in $list_{adj}$.
5. Now, each element in $list_{adj}$ that has not been already visited becomes the new actual start element and step 2 is repeated with a new initial value (u_s, v_s). For each new start element, (u_s, v_s) is found through one of these edges that has already been stored in $Container_s$. In the particular case when none of these edges is in $Container_s$ the initial value (u_s, v_s) is obtained from one of the edges that share one of the vertices with the start element. This is the case for elements in $list_{adj}$ that have been obtained from the search process associated with the particular case of step 4. If more than one intersection point is located in the new start element (two intersection points in 2D and three/four in 3D), step 3 is repeated with a new vector $\underline{n}_s$. For each new start element, $\underline{n}_s$ is found through one of the elements around it that has already been stored in $Container_n$. To construct a local piece of the zero level set for this new start element, evaluate and store the value of p at each vertex that has not already been visited. Then for the rest of the vertices that have already been visited, store at each one the signed distance value such that verifies the minimum absolute value between the actual value of p and the value already stored at the previous steps.

6. Repeat step 5 until all elements stored in $list_{adj}$ have been visited. The algorithm then continues with an empty $list_{adj}$. Then repeat step 4 for each element in $list_B$ that has not already been processed to build $list_{adj}$.
7. If there is another non-empty $list_{adj}$, repeat steps 5, 6 (see Figs. 6 and 7).
8. If there are no more elements in $list_{adj}$, then the conversion is finished. Extract the zero level set ϕ_h^i from the narrow band ω_i of elements stored in $list_B$ by linearly interpolating the distance field inside this band. Then the corresponding polygonal surface mesh Γ_h^i is finally obtained (see Fig. 8).

4.4. Detailed algorithm

4.4.1. Determine the starting element given an arbitrary starting point on the parametric surface

The first step of the algorithm provides the start element containing the point $P_s = S(u_s, v_s)$ which can be chosen arbitrarily on the parametric surface. This is done by testing if an element contains this point using the maximum and minimum coordinates (x,y,z) of its vertices. The element which contains the point P_s is taken as the start element to begin the algorithm. This step is implemented through a loop over all elements, for simplicity. Note that this loop is only required to locate the first start element containing the seed point P_s. All other "touched" elements in the following steps are located by the adjacency relations between mesh entities and do not require a loop on all elements in the mesh.

4.4.2. Determine the intersection points of the parametric surface with the start element

In step 2, the algorithm first visits each edge of the start element found in step 1 to determine the intersection points with S (see Fig. 5). These intersection points can be determined by solving the following set of nonlinear equations:

$$\text{Find} \quad (u, v, t), \ \text{such that } F(u, v, t) = S(u, v) - E(t), \tag{7}$$

where $S(u,v)$ is a parameterization for the surface, and $E(t) = (1-t)E_0 + tE_1$ is a parameterization for the edge defined by its vertices E_0 and E_1. We find the approximate solution to this set of nonlinear equations by the pure algebraic Newton–Raphson iterative method. The initial value (u_s, v_s) corresponding to P_s is used to obtain the local solution for all edges of the start element, only a few iterations are needed to compute the intersection point when the edge crosses S. In practice, two iterations are sufficient to obtain the intersection point; this number of iterations is used as criterion to find the edges that intersect S between its vertices. The algorithm employs only the first order information of the curve/surface, this uses only geometric information that is common to all possible parameterizations.

For each edge/surface intersection, the solution (u,v,t) of the above set of equations is stored in $Container_s$ such that they can be found from its corresponding edge. This storage will be used in the next steps both to efficiently detect the next elements needed for our marching algorithm, and to provide a new good initial guess to achieve fast convergence for the Newton–Raphson method. This algorithm is written in C++ using the Standard Template Library (STL) containers, and benefits of associative containers for storing and accessing data with a good memory storage management and short execution times.

4.4.3. Compute the signed distance to the parametric surface in the start element

The third step makes use of all the intersection points obtained in step 2 and of the input vector $\underline{n}_s$ to compute the values of the signed-distance map at the vertices of the start element. The storage in $Container_n$ allows to keep in memory the sign of the distance field between elements stored in $list_B$ and its neighboring elements while the field is built. In the case of four intersection points, which may happen in 3D, we use only three arbitrary intersection points to build the plane $p(\underline{x})$. We show in the following section that the error to represent the local zero level set on the smooth parametric surface inside the boundary element E_B can be reduced by using a graded sub-mesh refinement in the split elements.

4.4.4. Populate the list of elements neighboring the start element

The fourth step requires mesh databases such as the Algorithm Oriented Mesh Database (AOMD) [49][4] to provide all possible relations between all adjacent mesh entities of the background mesh, *e.g.* vertex → edges, vertex → faces, vertex → elements (3D), edge → faces, edge → elements (3D), face → elements (3D). The marching algorithm stores new elements in $list_{adj}$ such that they can be searched by adjacency relations from the start element stored in $list_B$. This search process selects the set of elements around the start element which are likely to be themselves intersected by a portion of S (see Fig. 6). The narrow band around the surface of interest could be built in a piecewise manner by successively searching for the elements neighboring those which are intersected by the parametric surface. This is not what is done in practice, as shall become clear from the description of the next step.

4.4.5. Compute the signed distance function for the cluster of elements adjacent to the start element

Step 5 starts from an element stored in $list_{adj}$ that has not already been visited to construct the distance field, and stops when all elements stored in this list have been visited. This step selects elements from $list_{adj}$ to build a portion of the narrow band ω_i and computes the corresponding distance field. At this stage of the marching algorithm, two small portions of the surface S have been iteratively converted; the first portion corresponds to the first start element found in step 1, and the second portion corresponds to the cluster of elements belonging to the narrow band around the start element found by the adjacency relations (see Fig. 6).

4.4.6. Walking through adjacencies

In step 6, the algorithm starts with an empty $list_{adj}$, and propagates the search of a portion of elements by walking through adjacencies (see Fig. 6). Then stores this portion in the $list_{adj}$ which may intersect another portion of S.

4.4.7. Construction of the signed-distance field on the narrow band

The marching algorithm in step 7, builds the narrow band ω_i along with the corresponding distance field in a piecewise manner (see Fig. 7). Note that the algorithm works for both open and closed parametric curves/surfaces.

4.4.8. Construction of the zero level set on the narrow band and extraction of the polygonal mesh Γ_h^i

The end product of the algorithm as described above is the signed distance function at each node of the elements contained in the narrow band ω_i. The finite element shapes functions can then be used to calculate the value of the signed distance function anywhere within this narrow band by interpolating between the nodal values. Next, the extraction of explicit curve/surface information Γ_h^i for subdivision purposes can be performed (see Fig. 8).

[4] Or those used in [12,50] for damage tolerance of complex structures and the simulation of concrete mesostructure, respectively. The reader is directed to the discussion in [51] concerning the importance of including an algorithm-oriented mesh generator as part of extended finite element method implementations. The Ph.D. Thesis [52] provides more detail.

4.5. Remarks on the algorithm

It will be clear to the reader by now that the algorithm computes the signed distance map on a minimal narrow band around the parametric surface *only*. This band is made up of all elements adjacent to the elements split by the surface. We saw that using mesh databases such as AOMD is critical to the efficiency of this step. This narrow band is composed of the boundary elements B and produces two separate sets of elements, on either side of the surface.

It is then straightforward to generate a discretized version of the parametric surface, *i.e.* a polyline mesh in 2D and a triangulated surface in 3D. This discretized version Γ_h^i of the parametric surface (see Fig. 8) is generated for subdividing the boundary elements E_B into triangles in 2D and tetrahedra in 3D (see Fig. 9). To avoid extremely small solid parts inside a boundary element E_B, the nodal value of ϕ_h^i at nodes can be set to zero, when $\phi_h^i < 0$ and $|\phi_h^i| < \epsilon$. Where ϵ is a predefined (small) tolerance, the value of which can be based on an edge ratio as proposed by Moës et al. [46].

The signed distance field can be extended from the narrow band to the rest of the mesh using methods such as the fast marching method (FMM) [53]. Our approach uses the narrow band and the sign of ϕ_h^i to search for interior and exterior elements solely by walking through adjacencies, without needing to extend the distance field information to the whole mesh.

An advantage of this conversion from parametric into level set representation is that the closest point information generated by the intersection of the surface boundary and the edges of the boundary elements E_B can be used to represent the exact geometric model given by the NURBS (non-uniform rational B-splines) boundary representation within the isogeometric analysis framework proposed by Hughes et al. [54]. It will be presented in another paper.

5. Implicit geometrical models based on multiple parametric primitive representations

5.1. Introduction and motivation

In the prevenient section, we have detailed one possible algorithm to convert a parametric surface into an implicit signed distance/level set representation. In practice however, solids cannot be represented by a single parametric surface. Thus, it is necessary to have the capacity to combine a number of parametric (primitive) surface representations to build the final solid. Recall the example of the square plate with a hole of Fig. 4. The object of this section is to explain how the proposed algorithm can be used to define solids based on an arbitrary number of primitive surfaces.

As explained in Section 2, constructing a domain based on a fixed background mesh poses two major challenges:

Sharp features such as edges and corners cannot be reproduced accurately using usual constructive solid geometry (CSG) operations on level sets. We propose a cutting process to split boundary elements that are intersected by more than one zero level set.

Mesh independence is a requirement that aims at ensuring that minimal remeshing is performed in the definition of the solid. In general, the quality of the approximation of a curved domain is mainly governed by mesh size. To produce a mesh-independent approximation that may usefully be used for analysis, we have chosen to construct a finer graded mesh inside the narrow bands ω_i. This graded sub-element mesh has two roles.

- First, it is used for the conversion of the parametric functions to implicit signed distance representations.
- Second, it ensures that the inner and outer regions are accurately represented, even for severely curved boundaries.

5.2. Cutting method integration

For a given zero level set that encloses the analysis domain, the polygonal meshes Γ_h^i are extracted from the bands ω_i for the purpose of mesh subdivision. A cutting method is used to split the boundary elements B that contain triangular/tetrahedral elements E_B by the split edge/triangle that compose Γ_h^i. The subdivision level of a boundary element E_B depends on the number of zero level sets defined at its vertices. When more than one cut edge/triangle intersects an element (*i.e.* $\omega_k \cap \omega_l \neq \emptyset$, see Fig. 4b), the cutting method may be recursively used for each sub-element (triangle/tetrahedron) according to the number of cut edge/triangle.

This cutting method is developed as an extension of the standard subdivision used in XFEM (Moës et al. [23] in 2D and Sukumar et al. [55] in 3D). Each boundary element E_B (level of subdivision 0) crossed by the first zero level set ϕ_h^1 can be divided into sub-elements (triangles/tetrahedra, level of subdivision 1). If a second zero level set ϕ_h^2 is also defined over a boundary element E_B that has already been processed, then each sub-element (level of subdivision 1) crossed by the second zero level set can be subdivided in a similar manner as E_B, and a number of new sub-elements (level of subdivision 2) are created to replace the intersected sub-element (level of subdivision 1). In the general case, another zero level set ϕ_h^n defined over E_B is assigned to subdivision level n. Each sub-element created by the recursive cutting method crossed by a zero level set ϕ_h^n is subdivided and replaced by a new sub-element (level of subdivision n). Then, all these new sub-elements of different levels are stored for defining the domain boundary, performing numerical integration and to allow the specification of physical information for analysis. Note that, the sub-elements of level $n-1$ that are crossed by a zero level set ϕ_h^n, are replaced by the new sub-elements of level n, and all sub-elements E_Δ are triangles in 2D or tetrahedra in 3D.

5.3. Illustration of the ability of the cutting method to represent corners in two and three dimensions

Figs. 10–12 demonstrate the ability of our cutting method to represent corners in 2D and 3D. For the purpose of illustration, in the case of Fig. 10, the boundary element E_B is subdivided into three sub-elements that align with the first zero level set $\phi_h^1\left(E_B = \bigcup_{k=1}^{3} E_\Delta\right)$, then the second zero level set ϕ_h^2 subdivides each sub-element of level 1 into three sub-elements of level 2. Next, the new sub-elements of level 2 are created replacing the

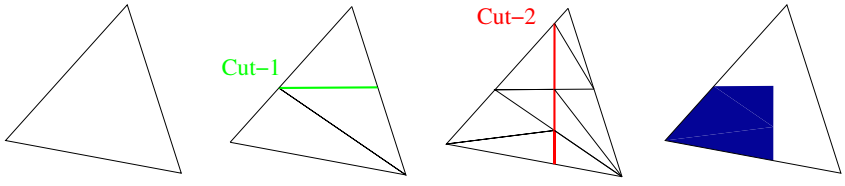

Fig. 10. The cutting process in 2D, with two intersections of a boundary element E_B.

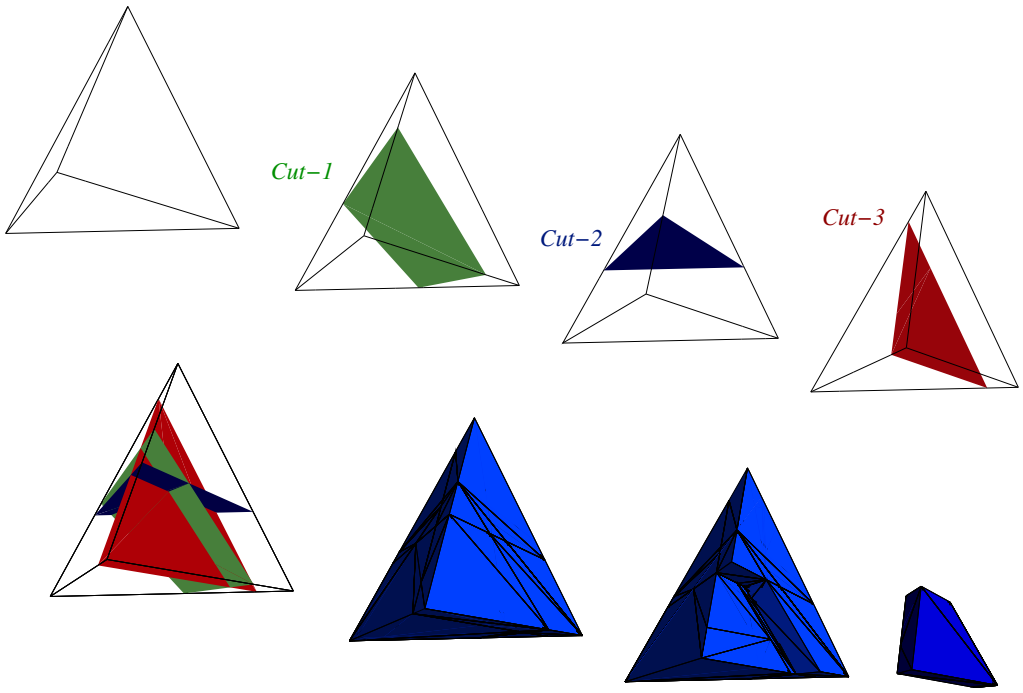

Fig. 11. The cutting process in 3D, with three intersections of a boundary element E_B.

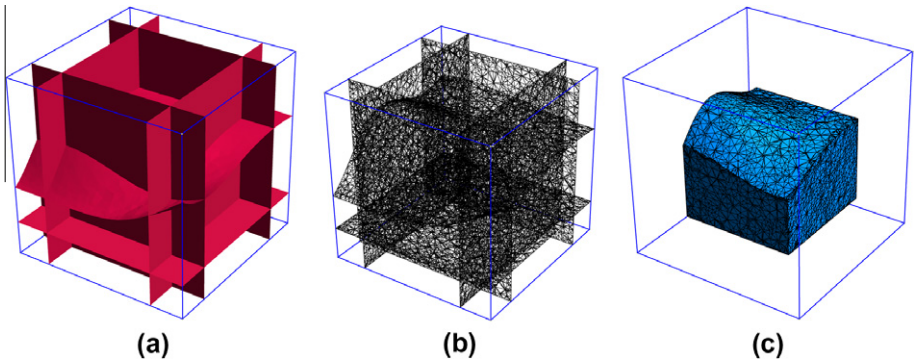

Fig. 12. (a) Conversion of four parametric functions into zero level sets. (b) Polygonal meshes extraction for the cutting method. (c) Approximated domain with sharp features.

sub-elements of level 1. Then, the sub-elements are categorised into interior E_{IB} and exterior sub-elements E_{OB} using a Boolean operator based on the value of the sign of ϕ_h^1 and ϕ_h^2 at the centroid of each sub-element E_Δ such that $E_B = \bigcup_{k=1}^{9} E_\Delta = \{E_{IB}\}_3 \cup \{E_{OB}\}_6$and $E_{B_\Gamma} = \bigcup_{k=1}^{3} E_{\Delta_\Gamma}$. This example illustrates the ability of the method to represent corners defined by the intersection of two zero level set functions in a particular configuration.

5.4. Adaptive sub-mesh refinement

We will now provide a technique that can control the accuracy of the implicit surfaces in the split (boundary) elements E_B without changing the underlying fixed mesh for analysis.

One strategy is to subdivide the boundary elements E_B using a finer mesh incorporated into the initial mesh [39] (*e.g.* a quadtree [40,36], or an octree [36]) such that the curved boundaries are approximated by a set of linear segments in 2D (triangles in 3D). To do so, a common method is to use volume (implicit) data representation that may be generated in a variety of ways (medical images, implicit functions, etc.).

A second strategy is to use the parametric surface to produce a polygonal mesh and then calculate a distance on the finer mesh to obtain the zero level set. This circumvents the inability to easily calculate signed distance to parametric surfaces by sampling the surface and using this discretized version of the surface to compute signed distances.

Our representation differs from previous efforts in that we work with the exact parametric representation of the surface inside the boundary elements E_B without polygonizing it.

To obtain an accurate geometry description for domains with curved boundaries, we present in the following section two different techniques: degenerated and graded sub-meshes which we shall name DSM and GSM, respectively.

5.4.1. Mesh refinement with degenerated sub-mesh (DSM)

We use the parametric information to generate the desired number of cut edges on the surface inside a boundary element E_B which are tangent to this parametric surface (see Fig. 13). These cut edges are created by the corresponding zero level sets such that they are generated by a succession of analytically known level set

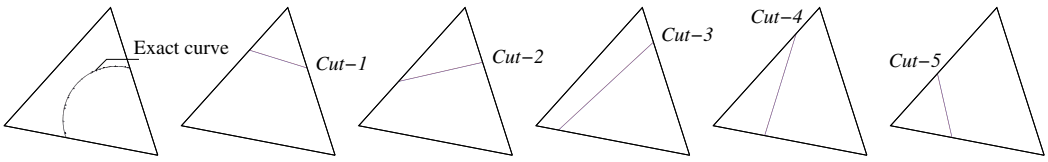

Fig. 13. Five cut edges are used to completely cover a smooth parametric curve to provide an accurate approximation of the boundary.

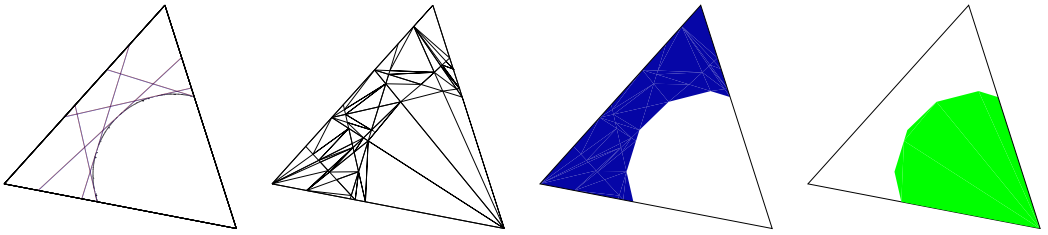

Fig. 14. The sub-elements E_Δ resulting from five cut edges inside a boundary element E_B (see Fig. 13). The sets $\{E_{IB}\}$ and $\{E_{OB}\}$ shown on the right represent the solid part and the second material or void part.

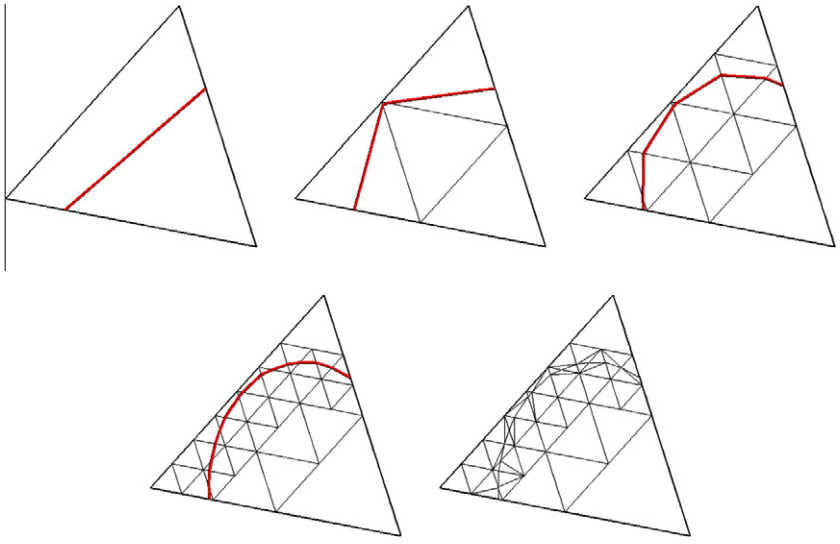

Fig. 15. Graded sub-mesh (GSM) refinement of level ($n = 3$) inside a boundary element E_B. The set of straight line segments refers to the linear approximation of the boundary for each level of refinement.

planes $p(\underline{x}) = (\underline{x} - \underline{x}_0) \cdot \underline{n}$ that pass through the point $\underline{x}_0$ on the surface and defined by the normal $\underline{n}$ at this point. Then we apply the cutting method to each boundary element E_B by using these zero level sets to create the sub-elements E_Δ. The next step is the classification of the sub-elements into the interior boundary *IB* and exterior boundary *OB* to define the part of the approximate domain Ω_h on the boundary *B* and the part of its boundary Γ_h (see Fig. 14).

5.4.2. Mesh refinement with graded sub-mesh (GSM)

The marching algorithm (*cf.* Section 4.3) benefits from a natural strategy to locate the narrow band from the all elements in the mesh, in which only the selected elements (*i.e.* ω_i) need to be used for refinement if desired. This is an attractive strategy to restrict local mesh refinement to boundary elements E_B. This strategy will be used locally in E_B and recursively within each level of refinement to obtain a finer graded mesh that encloses the portion of the curve/surface inside E_B, which internal graded sub-mesh does not conform to the boundary.

Fig. 15 shows the steps involved in the refinement in the 2D case for a boundary element E_B and the classification of sub-elements E_Δ contributing to the analysis. Each triangular boundary element E_B (level of refinement 0) is subdivided into four sub-triangles E_Δ (level of refinement 1) by connecting the midpoints of its three edges. The *set* of sub-triangles of level 1 (called *set-1*) is processed by the marching algorithm to locate the *sub-set* of sub-triangles of level 1 (called *sub-set-1*) which enclose the portion of the curve inside E_B.[5]

For those sub-elements E_Δ that do not contribute to the local representation of the boundary curvature (*i.e.* the sub-elements E_Δ in *set-1* but not in *sub-set-1*) would be categorised into interior E_{IB} and exterior E_{OB}. This classification is done by using the sign of the distance values stored in E_B (*i.e.* level 0) to evaluate the sign at the centroid of each sub-element E_Δ which does not contribute to the *sub-set-1*.[6]

[5] For the purpose of illustration, in the case of Fig. 15, the boundary element E_B is subdivided into four sub-elements E_Δ (*i.e. set-1*); the marching algorithm locates three sub-elements E_Δ (*i.e. sub-set-1*) to more accurately convert the portion of the parametric surface into another more accurate portion of zero level set.

[6] In the case of Fig. 15, one sub-element E_Δ in *set-1* does not contribute to building the approximated curve; three sub-element E_Δ in *sub-set-1* are located by the marching algorithm (*cf.* Section 4.3) which will be used later for another refinement if needed. Note that, the sub-elements E_Δ of *sub-set-1* store the scalar distance value on each of its vertices, which serve to define the zero level set on *sub-set-1*.

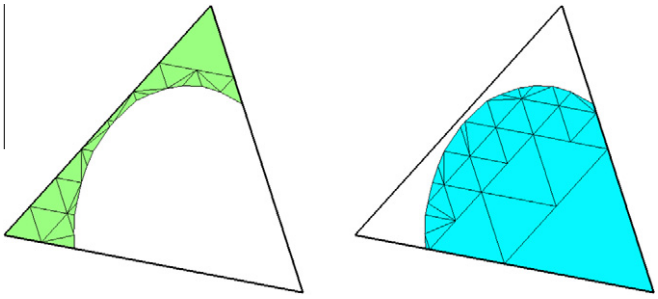

Fig. 16. The sub-elements E_Δ resulting from a graded sub-mesh refinement of level ($n = 3$) and from the cutting method applied to the sub-elements of level ($n = 3$) inside a boundary element E_B. (left and right) show an accurate representation of the domain boundary inside E_B.

If a second refinement is also needed, each sub-element E_Δ of *sub-set-1* can be subdivided in a similar manner as E_B, and new sub-elements (*set-2*; *i.e.* level of refinement 2) are created to replace the sub-elements of *sub-set-1*. Then, the sub-elements of *set-2* will be processed by the marching algorithm to locate the *sub-set-2* that encloses the portion of the curve inside the *sub-set-1*.

Those sub-elements E_Δ of level 2 that do not contribute to the local representation of the boundary curvature (*i.e.* the sub-elements E_Δ in *set-2* but not in *sub-set-2*) would be categorised into interior E_{IB} and exterior E_{OB}. This classification is done using the sign of the distance values stored in *sub-set-1* (*i.e.* level 1) to evaluate the sign at the centroid of each sub-element E_Δ which does not contribute to *sub-set-2*.

Now, if local refinement of level n is desired in E_B:

- Each sub-element E_Δ of the *sub-set-(n − 1)* can be subdivided in a similar manner as E_B, and new sub-elements (*set-n*; *i.e.* level of refinement n) are created to replace the sub-elements of *sub-set-(n − 1)*. Then, the sub-elements of *set-n* will be processed by the marching algorithm to locate the *sub-set-n* that encloses the portion of the curve inside the *sub-set-(n − 1)*.
- Those sub-elements E_Δ of level n that do not contribute to the local representation of the boundary curvature (*i.e.* the sub-elements E_Δ in *set-n* but not in *sub-set-n*) would be categorised into interior E_{IB} and exterior E_{OB}. This classification is done by using the sign of the distance values stored in *sub-set-(n − 1)* (*i.e.* level $n - 1$) to evaluate the sign at the centroid of each sub-element E_Δ which does not contribute to build the *sub-set-n*.
- We apply the cutting method to split each sub-element of *sub-set-n* by using the local zero level set defined on this *sub-set-n* to create sub-elements E_Δ, and then classify the sub-elements which were created by the cutting method into interior E_{IB} and exterior E_{OB}. This classification is done by using the sign of the distance values stored in *sub-set-n* to evaluate the sign at the centroid of each sub-element E_Δ which was created by the cutting method.

Remarks:

- The marching algorithm applied to the background mesh grid, propagates the search of a portion of elements by walking through adjacencies to construct the zero level set inside E_B of level 0. If local refinement is needed, another process of the marching algorithm is used locally and recursively at each step of refinement inside E_B, such that the inputs of the algorithm (*i.e.* the sub-set of level n, the start values (u_s, v_s), and the start vector $\underline{n}_s$) are obtained automatically.
- Each boundary element E_B is processed separately until the classification of all sub-elements E_Δ of level n is done, recursively, into an interior E_{IB} category or exterior E_{OB} category.

The graded sub-mesh refinement (GSM) in three-dimensions is similar to that previously described for two dimensions, but needs to deal with a few additional challenges: (1) a tetrahedral boundary element E_B will be divided into eight sub-elements E_Δ by connecting the midpoints of its six edges. (2) For each level of refinement needed, each sub-element E_Δ will be subdivided in a similar manner to E_B. (3) The marching algorithm (*cf.* Section 4.3) will be used globally in the three dimensional background mesh grid *Gr* and locally in E_B.

Examples shown in Figs. 16 and 17 are constructed to demonstrate the ability of our graded sub-mesh to construct the sub-elements E_Δ inside a boundary element E_B, and to represent a curved boundary by a set of linear segments E_{Δ_Γ} in 2D (triangles in 3D). These sub-elements E_Δ and E_{Δ_Γ} will be used for numerical integration of the local stiffness matrix and to apply boundary conditions (e.g. heat flux and traction in a mechanical problem), respectively. For this purpose, the sets $\{E_\Delta\}$ and $\{E_{\Delta_\Gamma}\}$ corresponding to a boundary element E_B are stored such that each E_Δ or E_{Δ_Γ} can be found by its corresponding E_B.

We discuss numerical integration in detail in the forthcoming section.

5.5. Numerical integration

In the upcoming developments, the sets of interior elements I and boundary elements B are assumed known.

Interior elements. Numerical integration in each interior element E_I is done using standard Gauss quadrature.

Boundary elements. For a boundary element E_B, *i.e.* an element split by the boundary of the domain, two different approaches are possible:

1. Subdividing E_B based on a linear (as in [23,55]) or higher order (as in [41,42]) description of the boundary.
2. Without subdividing E_B as proposed in Ventura [56] using equivalent polynomials. It is also possible to use the approach of Natarajan et al. [24,25] based on the Schwarz Christoffel (SC) mapping of the interior/exterior polygonal areas to the unit disk. Another alternative is strain smoothing where domain integration is transformed into boundary integration as in [26]. The advantage of the latter is that it has the potential to be amenable to three dimensional cases, whereas the SC mapping technique remains restricted to two-dimensional problems. To use the SC mapping in 3D, the interior and outer parts of a boundary element could be integrated using strain smoothing and the SC mapping subsequently used to integrate along the boundary of the interior and exterior subregions. Since each of those boundaries is composed of the union of polygons, the SC mapping (or any other method to integrate numerically on polygons) can be used to compute the integral on each polygon. Note that strain smoothing modifies the variational principle so that the resulting stiffness matrix is usually not as stiff as that of the original finite element. The interested reader can refer to [57,58] and, more recently [59–62] for details as well as [63] for upper and lower bounds.

However, both approaches, as they stand, can only deal with a unique linear/higher-order boundary surface of simple shape inside the boundary elements E_B. In our approach, we use linear sub-elements (finer mesh) incorporated into a boundary element E_B to control a priori the faithfulness of the boundary representation. This approach was also used with boundaries represented by an analytical level set in 2D [40] and explicit 3D polygonal mesh in [43,44].

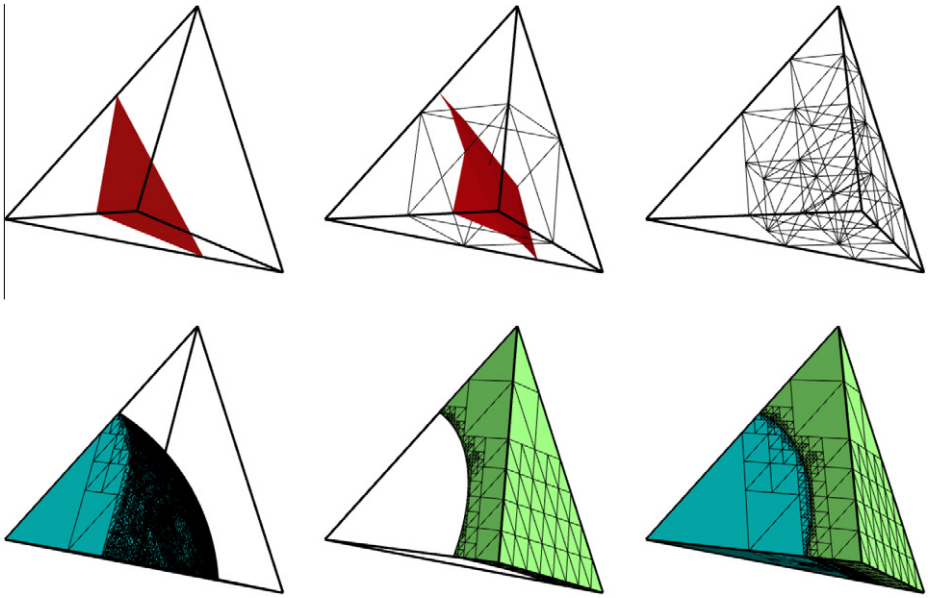

Fig. 17. A three-dimensional graded sub-mesh refinement of level ($n = 6$) inside a boundary element E_B.

Following ideas similar to the Heaviside step function approach of [9,36], we will define the indicator functions $\Lambda_{I\cup B}$ and Ξ_B to simplify the description of the domain Ω_h and its boundary Γ_h, respectively, as follows:

$$\Lambda_{I\cup B} = \begin{cases} 1 & \text{if} \quad (E_I, E_\Delta) \in \Omega_h, \\ 0 & \text{if} \quad E_\Delta \notin \Omega_h, \end{cases} \tag{8}$$

$$\Xi_B = \begin{cases} 1 & \text{if} \quad E_{\Delta_\Gamma} \in \Gamma_h, \\ 0 & \text{if} \quad E_{\Delta_\Gamma} \notin \Gamma_h. \end{cases} \tag{9}$$

These indicator functions are used to restrict the numerical integration procedure to those elements which should effectively contribute to the stiffness matrix and boundary conditions.

Domain integrals. All integrals in the set of interior elements I are computed using standard procedures. For each boundary element E_B numerical integration will be performed in each sub-element (triangles/tetrahedra) E_Δ that contribute to the approximated domain Ω_h. Therefore, the integral of a generic function f is then given by

$$\int_{Gr_{I\cup B}} \Lambda_{I\cup B} f \, d\Omega = \int_I f \, d\Omega + \int_B \Lambda_{I\cup B} f \, d\Omega, \tag{10}$$

where

$$\int_I f \, d\Omega = \sum_{E_I} \int_{E_I} f \, d\Omega$$

and

$$\int_B \Lambda_{I\cup B} f \, d\Omega = \sum_{E_B} \int_{E_B} \Lambda_{I\cup B} f \, d\Omega = \sum_{E_B} \sum_{E_\Delta} \int_{E_\Delta} \Lambda_{I\cup B} f \, d\Omega$$
$$= \sum_{E_B} \sum_{E_{IB}} \int_{E_{IB}} f \, d\Omega.$$

$Gr_{I\cup B}$ defines the sets of interior elements I and boundary elements B. E_Δ is one of sub-elements inside E_B. E_{IB} is one of the sub-elements that contribute to the approximated domain inside E_B, in this case $E_\Delta = E_{IB}$.

Boundary integrals. For each boundary element E_B, numerical integration along boundary will be performed on each linear segment/triangle E_{Δ_Γ} inside E_B, such that E_{Δ_Γ} contributes to the domain boundary Γ_h. Therefore, the integral of a generic function f on the domain boundary is given by

$$\int_{B(\Gamma)} \Xi_B f \, d\Gamma = \sum_{E_{B_\Gamma}} \int_{E_{B_\Gamma}} \Xi_B f \, d\Gamma = \sum_{E_{B_\Gamma}} \sum_{E_{\Delta_\Gamma}} \int_{E_{\Delta_\Gamma}} \Xi_B f \, d\Omega, \tag{11}$$

$B(\Gamma)$ defines the set of boundary elements B that enclose a part or all of the boundary Γ. E_{B_Γ} defines the part of the boundary which is inside E_B. E_{Δ_Γ} is one of the linear segments/triangles that contribute to Γ_h. In the particular case, when the domain is constructed without sharp features, (11) can be written as

$$\int_{B(\Gamma)} \Xi_B f \, d\Gamma = \sum_{E_{B_\Gamma}} \int_{E_{B_\Gamma}} f \, d\Gamma = \sum_{E_{B_\Gamma}} \sum_{E_{\Delta_\Gamma}} \int_{E_{\Delta_\Gamma}} f \, d\Gamma.$$

Fig. 18 shows some examples of boundary integrals in a particular geometric configuration.

5.6. Numerical experiments

As shown in Section 5.4, the degenerated finer sub-mesh (DSM) and graded sub-mesh (GSM) can be used as required to cover the space created by the curved boundaries inside a boundary element E_B. We show via numerical experiments that the degenerated sub-elements E_Δ (the extreme case) are very well suited to control geometric errors in the description of boundaries, and the large aspect ratio of E_Δ does not affect the accuracy of the numerical integration, as confirmed by other studies [43]. Also, we show that if these boundaries are not sufficiently well described, the advantage of using higher order finite element is indeed not clear, as errors in the boundary description lead to suboptimal convergence rates in the analysis.

5.6.1. Area of a circle

The first numerical example is concerned with calculating the relative error in the area of a parametrically defined circle $C(u) = (\cos(u), \sin(u))$ converted on an unstructured square grid meshed with triangular elements. The relative error in the area, defined as follows:

$$Re_a = \frac{|area^h - area^{ex}|}{area^{ex}} \tag{12}$$

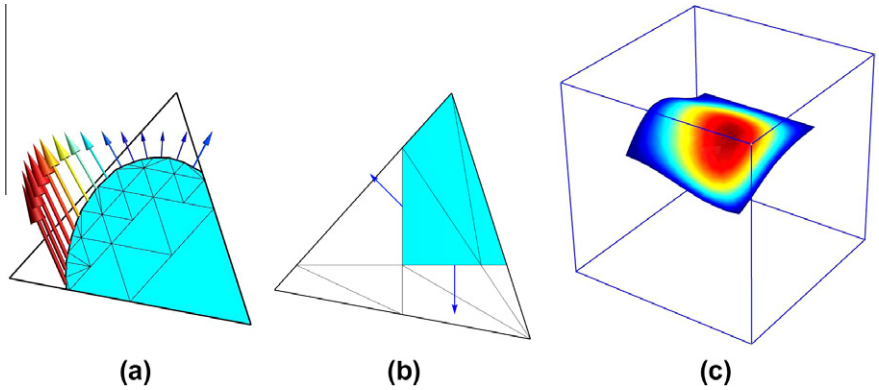

Fig. 18. A proper numerical quadrature of a vectorial function: (a) along the interface (on the linear approximation $\{E_{\Delta_l}\}_{15}$ of the boundary inside E_B); (b) on the bottom or left part of the boundary (restrict integrals by the indicator Ξ_B) and (c) shows the numerical quadrature of a scalar function restricted to the top part of the domain boundary (*cf.* Fig. 12).

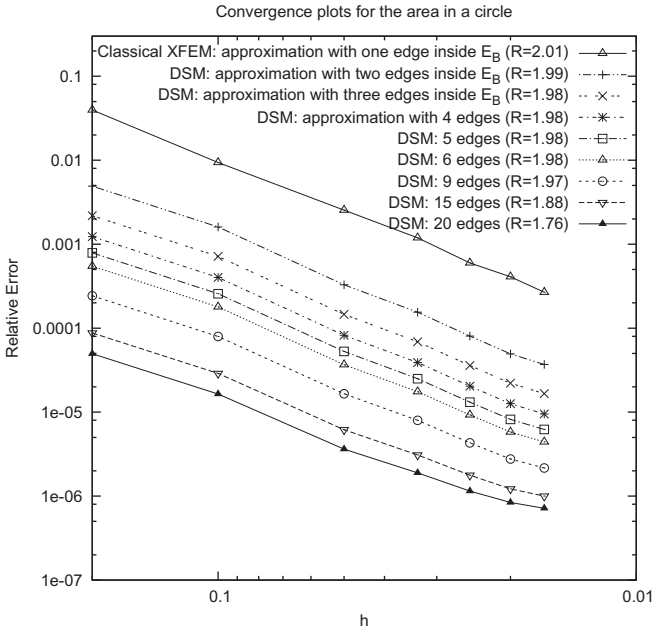

Fig. 19. Numerical experiment used to validate the numerical integration with degenerated sub-mesh (DSM).

with

$$area^h = \int_{Gr_{I\cup B}} \Lambda_{I\cup B} f \, d\Omega \quad \text{and} \quad f = 1,$$

computed with different mesh sizes h.

For representing a circle in the set of boundary elements B, we use the extreme case (DSM) to perform numerical integration over the degenerated sub-elements E_Δ. The area of a circle, defined using the proposed method is plotted against mesh refinement for various numbers of edges used to represent the circular shape within each split element. The results are given in Fig. 19 where it is clear that using additional edges to represent the curved boundary within each split boundary element leads to a very large reduction in the error, and the geometric approximation error decreases with the increase of additional edges as $O(h)^2$. Note that the error level in the approximation of the area of the circle is decreased by three orders of magnitude when using 20 edges within each split element, compared to using one single edge as is commonly done. One order of magnitude is already achieved by using two edges as opposed to one.

The convergence rate, however, seems to suffer, especially when more than 10 edges are used in each split element. The reason for this behavior is that the cutting method used for each additional edge inside E_B, process on more and more degenerated and finer sub-elements E_Δ which affects the precision of the cut process and hence of the integration. We recommend using 5 edges, which gives a good balance of computational efficiency versus accuracy.

5.6.2. Plate with a hole under tension

The second problem will be presented to illustrate the convergence rate for a 2D problem in classical linear elasticity with a curved boundary. We consider an infinite plate with a central hole of radius a loaded at infinity by a traction $\sigma = 1$ in the x-direction. The main objective of this problem is to study the effectiveness of GSM with higher order shape functions and study whether the use of a degenerated sub-mesh is be critical in the analysis. The exact solution of this problem is available as given in [64], and the displacements in the x- and y-direction as

$$\begin{aligned} u_x(r,\theta) &= \frac{a}{8\mu}\left[\frac{r}{a}(\kappa+1)\cos\theta + 2\frac{a}{r}((1+\kappa)\cos\theta + \cos 3\theta) - 2\frac{a^3}{r^3}\cos 3\theta\right], \\ u_y(r,\theta) &= \frac{a}{8\mu}\left[\frac{r}{a}(\kappa-3)\sin\theta + 2\frac{a}{r}((1-\kappa)\sin\theta + \sin 3\theta) - 2\frac{a^3}{r^3}\sin 3\theta\right], \end{aligned} \tag{13}$$

and the components of exact stress are:

$$\begin{aligned} \sigma_{11}(r,\theta) &= \sigma\left[1 - \frac{a^2}{r^2}\left(\frac{3}{2}\cos 2\theta + \cos 4\theta\right) + \frac{3}{2}\frac{a^4}{r^4}\cos 4\theta\right], \\ \sigma_{22}(r,\theta) &= \sigma\left[-\frac{a^2}{r^2}\left(\frac{1}{2}\cos 2\theta - \cos 4\theta\right) - \frac{3}{2}\frac{a^4}{r^4}\cos 4\theta\right], \\ \sigma_{12}(r,\theta) &= \sigma\left[-\frac{a^2}{r^2}\left(\frac{1}{2}\sin 2\theta + \sin 4\theta\right) + \frac{3}{2}\frac{a^4}{r^4}\sin 4\theta\right], \end{aligned} \tag{14}$$

where (r,θ) is a polar co-ordinate system with the origin at the center of the hole. A plane stress state $\kappa = 3 - 4\nu$ is assumed with dimensionless elastic modulus $E = 10^5$ and Poisson's ratio $\nu = 0.3$. μ is the shear modulus.

In the numerical model (see Fig. 20), we consider a square computational domain of size $L \times L$ with $L = 2$ and a hole radius $a = 0.4$ at its center. The exact tractions corresponding to the analytical solution (14) are prescribed on the boundary of the approximated domain Ω_h, with appropriate constraints added to remove the rigid body modes. In this example, the mesh does not conform to the circle but it does conform to the square boundary, in order to study the influence of the numerical quadrature only in the curved part. Convergence studies are carried out on different meshes using

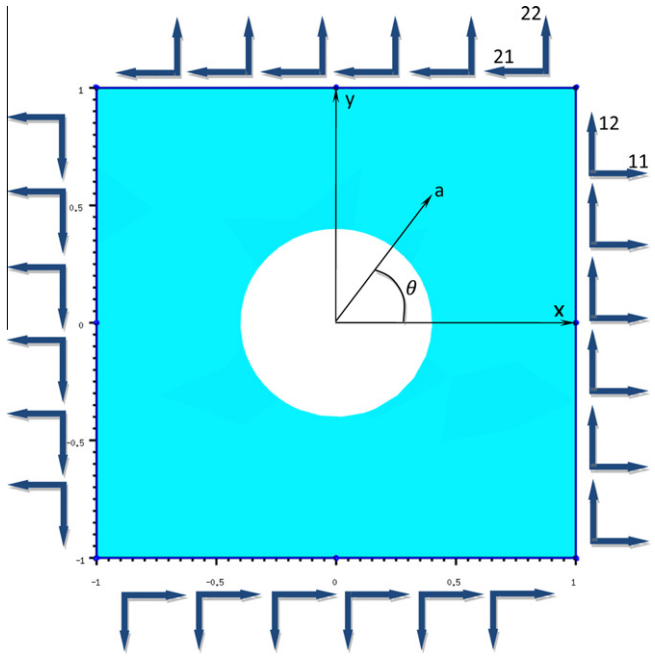

Fig. 20. Domain analysis for the infinite plate with a circular hole.

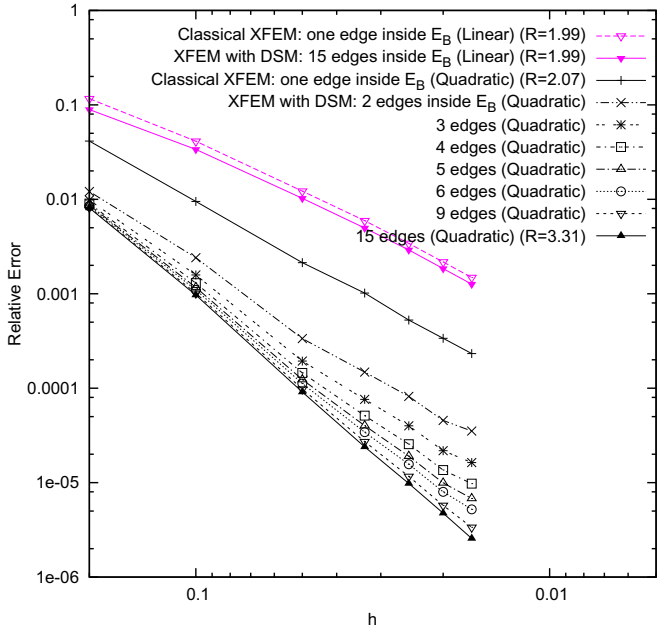

Fig. 21. $\mathscr{L}^2$ norm for 2D infinite plate with a hole problem. Linear and quadratic shape functions are used with different level of additional edge inside E_B and tested on unstructured square grid with different mesh size h.

linear and quadratic elements. For the purpose of error estimation, we use the displacement $\mathscr{L}^2(\Omega_h)$ norm

$$\mathscr{L}^2(\Omega_h) = \left(\frac{\int_{Gr_{I\cup B}} \Lambda_{I\cup B} (\underline{u}^h - \underline{u}^{ex})^T \cdot (\underline{u}^h - \underline{u}^{ex}) d\Omega}{\int_{Gr_{I\cup B}} \Lambda_{I\cup B} (\underline{u}^{ex})^T \cdot (\underline{u}^{ex}) d\Omega} \right)^{1/2} \tag{15}$$

such that the error decreases as $O(h)^{p+1}$, where p is the polynomial order of the shape functions, for this smooth problem.

The results are presented in Fig. 21 where it is clear that not only the accuracy, but also the convergence rates are increased by using the proposed technique as opposed to the classical XFEM approximation.

As expected, using 15 edges in each split element to represent the hole when using a linear approximation leads to very little improvement because the error is dominated by the inability of the finite element shape functions to represent the solution.

On the other hand, for quadratic approximations, the benefit of introducing additional points along the boundary of the hole is immediately apparent, even for only two edges inside each split element. This is also not surprising, because the advantage of using a quadratic approximation is lost by the inability to represent the curved boundary with a single edge in the split element.

Finally, the convergence rate in the $\mathscr{L}^2$ norm in the extreme case where 15 edges are used along the curved boundary inside E_B with a quadratic approximation is 3.31 which is better than the theoretical rate of $p+1$, versus the suboptimal convergence rate of 2.07 obtained for a quadratic XFEM approximation with a single edge in each split elements.

The results presented by the (DSM) technique show that the extreme case of degenerated sub-mesh can be used as required to cover the space created by the curved boundaries which decrease geometric errors inside the set of boundary elements B as needed by higher order FE methods. Moreover, the examples demonstrate that the optimal rate can be achieved even for large aspect ratios of the sub-elements E_Δ. However, one critical aspect of the technique based on the degenerated sub-mesh (DSM) is that the local computational geometry operations needed to subdivide the boundary element E_B is complex, especially in three-dimensions. We recommend the use of a graded sub-mesh (GSM) (*cf.* Section 5.4.2), which is fast, simple, and accurately controls the description of the object obtained by the converted surfaces on the background mesh.

6. Finite element analysis

In this section, we adapt the technique of XFEM for void-material interface to our proposed implicit representation. Our goal is to demonstrate that finite element analysis on a fully implicit representation of the domain whose boundaries (possibly defined by parametric functions initially) are independent of the mesh is possible within an XFEM framework and conserves the accuracy and convergence properties of the FEM even for domains with sharp features. This implicit representation is based on the signed distance field (level set) using a general unstructured grid and can be of interest for computational domains with free boundary problems:

- treatment of discontinuities (signed distance field can be used to construct enrichment functions within the XFEM framework),
- with possible evolving boundaries.

In this paper, only fixed boundaries with mixed Dirichlet and Neumann boundary conditions are considered. For simplicity we consider the Poisson problem, the generalization of the following developments to other PDEs or other boundary conditions is straightforward.

6.1. Governing equations

Consider an unstructured mesh grid Gr for analysis with regular mesh size h, that encloses an approximated object $\Omega_h \subset \mathbb{R}^n$, $(n = 2, 3)$, bounded by Γ_h. The domain divides Gr into disjoint parts, the sets of elements I, B and O. In our case, the void is not considered as an integral part of the object, thus the set O is of no further use in terms of the computation. Each element within the sets elements I and B are incorporated into the analysis, and let $Gr_{I\cup B}$ be a finite element mesh that completely encloses Ω_h from which the space $\mathscr{U}_h \subset H^1(Gr_{I\cup B})$ of order p is constructed.

Let us consider the following model problem in Ω:

$$\text{find } u : \Omega \mapsto \mathbb{R} \text{ such that}$$
$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = u_D & \text{on } \Gamma_D, \\ \nabla u \cdot \mathbf{n} = t_N & \text{on } \Gamma_N, \end{cases} \tag{16}$$

where f, u_D and t_N are given data, $\Gamma_D \cup \Gamma_N = \Gamma$, and $\mathbf{n}$ is the outward normal unit vector on Γ.

Finite element methods which employ a background mesh to model free objects can sidestep the mesh generation difficulties encountered with FE. On the other hand, new difficulties appear in the imposition of Dirichlet boundary conditions, which are imposed at nodes of the polygonal mesh Γ_h which do not necessarily coincide with any node of the background mesh. Therefore, the strong imposition of Dirichlet conditions inside boundary elements B leads to a phenomenon known as boundary looking which was already pointed out in the pioneering work of Babuška [65].

There are many techniques for the implementation of Dirichlet boundary conditions in such cases:

- Techniques that modify the basis functions to satisfy the constraints directly by using R-functions [66,18] or implicit equations [36].
- Techniques that enforce the conditions in a weak sense based on a modification of the weak form, such as the Lagrange multiplier method or the penalty method. When using Lagrange multipliers, only appropriate space that satisfy the discrete inf–sup condition [65,67] retain optimal convergence rates. Designing a proper space is needed to enforce approximately this condition in order to obtain an acceptable solution. When using a consistent penalty method such as Nitsche's method, an appropriate Nitsche parameter is required in order to get a stable formulation. The problem of this method is that the deduction of the weak form is not straightforward, and the choice of an appropriate parameter depends on the physics of the problem [34].

Among the different possible methods, the Lagrange multipliers method is particularly easily applicable to many classes of problems [34]. In fact, only a trivial modification of the weak form is needed to enforce the Dirichlet boundary conditions without requiring any user-defined parameters.

The weak form associated to (16) can be expressed as the following mixed formulation:

$$\text{find } (u, \lambda) \in \mathscr{U} \times \mathscr{L} \text{ such that}$$
$$\begin{aligned} &\int_\Omega \nabla v \cdot \nabla u \, d\Omega + \int_{\Gamma_D} v\lambda \, d\Gamma = \int_\Omega v f \, d\Omega + \int_{\Gamma_N} v t_N \, d\Gamma \\ &- \int_{\Gamma_D} \mu u \, d\Gamma = - \int_{\Gamma_D} \mu u_D \, d\Gamma \end{aligned} \tag{17}$$

for all $(v, \mu) \in \mathscr{U} \times \mathscr{L}$, where the Lagrange multiplier $\lambda = -\nabla u \cdot \mathbf{n}$ corresponds to the flux along the boundary and $\mathscr{L}$ denotes the space of Lagrange multipliers defined on the Dirichlet boundary.

Using the element shape functions N_i, the approximation of the field can be written as

$$u_h(x) = \sum_{i \in Gr_{I \cup B}} u_i N_i(x) \tag{18}$$

such that $N_i \in \mathscr{U}_h \subset H^1(Gr_{I \cup B})$.

The Lagrange multipliers are approximated by the classical shape functions such that these functions are constructed on a part of the polygonal surface mesh Γ_h^i containing the Dirichlet boundary. However, we show in the next section that only an appropriate finite element space of the Lagrange multipliers leads to stability of the above weak form as well as the optimal rate of convergence. The weak form for the discrete problem can be stated as:

$$\text{find } (u_h, \lambda_h) \in \mathscr{U}_h \times \mathscr{L}_h \text{ such that}$$
$$\begin{aligned} &\int_{Gr_{I \cup B}} \Lambda_{I \cup B} \nabla v_h \cdot \nabla u_h \, d\Omega + \int_{B(\Gamma_D)} \Xi_B v_h \lambda_h \, d\Gamma \\ &= \int_{Gr_{I \cup B}} \Lambda_{I \cup B} v_h f \, d\Omega + \int_{B(\Gamma_N)} \Xi_B v_h t_N \, d\Gamma \\ &- \int_{B(\Gamma_D)} \Xi_B \mu_h u_h \, d\Gamma = - \int_{B(\Gamma_D)} \Xi_B \mu_h u_D \, d\Gamma \end{aligned} \tag{19}$$

for all $(v_h, \mu_h) \in \mathscr{U}_h \times \mathscr{L}_h$, where $Gr_{I \cup B}$ is the union of sets I and B, $\Lambda_{I \cup B}$ and Ξ_B are the indicator functions (*cf.* Section 5.5), $B(\Gamma_N)$ and $B(\Gamma_D)$ define the two parts of boundary elements B which enclose the two parts of the boundaries Γ_N and Γ_D, respectively.

6.2. Imposition of Dirichlet boundary conditions

In the context of XFEM, [68,27] showed that using all Lagrange multipliers defined on the nodes of mesh Γ_h^i leads to instabilities, and only an optimal space will lead to optimal convergence. Moës et al. [27] proposed a strategy to reduce the Lagrange multiplier space by eliminating a number of constraints from the global matrix, thus relax the Dirichlet condition which suppresses oscillations in the multipliers. This strategy was extended to three-dimensions for imposing frictional contact within the XFEM framework in [28]. Recently, a theoretical analysis of the stability of this strategy in two-dimensions has been done in [29].

In the present work, we build the Lagrange multiplier space by employing the algorithm described in [28]. In conjunction with our framework, this algorithm imposes linear or equality relations among multipliers of the naïve space $\mathscr{L}_h$ to enforce the Dirichlet conditions in the weak sense. The algorithm consists in building a set of edges from the narrow band ω_i that are strictly cut by Γ_h^i. It is done by using the signed distance field defined in ω_i. Then a subset of these edges are selected with respect to the steps of the algorithm to construct vital edges and non-vital edges. Only the multipliers which are linked to vital edges will hold in the resolution of the discretized equation (19). Next, each multiplier which lies on a non-vital edge is imposed to be a linear combination or equal with the Lagrange multipliers of the vital edges around it. For more details about these steps, the reader is referred to [28].

We will show in the following section that this algorithm provides a suitable reduced Lagrange multiplier space for our framework.

7. Numerical examples

In this section numerical results will be presented to illustrate the analysis with fully implicit representation of the domains under mixed Dirichlet and Neumann boundary conditions. A Laplace problem in two and three dimensions with known analytical solutions will be analyzed to demonstrate the accuracy and convergence properties of the present method. Linear shape functions are used both to describe Γ_h and for analysis. The behavior of the proposed approach is studied on three academic examples with simple geometries. These geometries are not necessarily constructed by conversion from parametric functions into level sets, only analytical level sets suffice to approximate a proper domain with the cutting method (*cf.* Section 5.2). The exact Dirichlet and Neumann boundary conditions are prescribed on the boundaries of the approximate domain Ω_h, corresponding to the analytical solution of the potential u^{ex} and analytical normal flux, respectively. For the purpose of error estimation and convergence studies, the numerical stability of the formulation with respect to the inf–sup condition is verified by using the following relative error norms:

- energy norm over the entire approximate domain Ω_h

$$\mathscr{E}(\Omega_h, \nabla u) = \left(\frac{\int_{Gr_{\Gamma_{h\cup B}}} \Lambda_{I\cup B} \nabla(u^h - u^{ex}) \cdot \nabla(u^h - u^{ex}) d\Omega}{\int_{Gr_{\Gamma_{h\cup B}}} \Lambda_{I\cup B} \nabla u^{ex} \cdot \nabla u^{ex} d\Omega} \right)^{1/2}, \quad (20)$$

- $\mathscr{L}^2$ norm measuring the accuracy of the enforcement of the Dirichlet conditions on Γ_D

$$\mathscr{L}^2(\Gamma_h, u) = \left(\frac{\int_{B(\Gamma D)} \Xi_B (u^h - u^{ex})^2 d\Gamma}{\int_{B(\Gamma D)} \Xi_B (u^{ex})^2 d\Gamma} \right)^{1/2}, \quad (21)$$

- $\mathscr{L}^2$ norm measuring the accuracy of the Lagrange multipliers on Γ_D

$$\mathscr{L}^2(\Gamma_h, \lambda) = \left(\frac{\int_{B(\Gamma D)} \Xi_B (\lambda^h - \nabla_n u^{ex})^2 d\Gamma}{\int_{B(\Gamma D)} \Xi_B (\nabla_n u^{ex})^2 d\Gamma} \right)^{1/2}, \quad (22)$$

where u^{ex} and u^h are the exact and extended finite element solutions for the potential field, and $\nabla_n u^{ex}$ and λ^h correspond to the normal flux of the exact and extended finite solutions of the Lagrange multipliers.

7.1. Numerical validation for two-dimensional problems

Let us analyze the convergence in the particular example of the Laplace equation, already considered in [34] for mesh-free and [27,30,32,29] for XFEM approximation. The model problem is given by

$$\begin{cases} \Delta u = 0 & \text{in } (x,y) \in \Omega_h :=]0,1[\times]y^*,1[\text{ with } y^* \in [0,1[, \\ u = u^{ex}(x,y^*) & \text{at } y = y^*, \\ \nabla u \cdot \mathbf{n} = \nabla u^{ex} \cdot \mathbf{n}_o & \text{at } (x = 0,1; y^* < y < 1) \text{ and } y = 1 \end{cases} \quad (23)$$

with the analytical solution given by

$$u^{ex}(x,y) = [\cosh(\pi y) - \coth(\pi)\sinh(\pi y)]\sin(\pi x). \quad (24)$$

For this problem, we study two numerical examples: square domain ($y = y^* = 0$), and curved domain ($y = y^*$); see Fig. 22. We note that the Dirichlet boundary conditions are imposed both on the bottom part of the square domain and on the curved part of the curved domain.

The curved part of the domain is constructed with a circular arc centered at (0.5, 1), with radius $r = 2/3$. This curved domain has already been considered in [30] as a benchmark for enforcing Dirichlet boundary conditions on only the boundary curved part which is not fitted with the background mesh.

7.1.1. Implicit square domain

First, for the purpose of comparison, we present numerical tests with the standard mixed method (naive approach) without reducing the Lagrange multiplier space $\mathscr{L}_h$ for the square domain with conforming unstructured triangular mesh and with implicit computational domain on unstructured triangular mesh grid.

In the case of the implicit representation of the domain, a piecewise-linear Lagrange multiplier space $\mathscr{L}_h$ is constructed along the straight line segment of the bottom part of Γ_h with a degree of freedom at each vertex of the straight line segments inside the set of boundary elements B. Note that we are dealing with the two types of boundary conditions inside a boundary element E_B which contains a corner. Therefore, the integrals on the two parts of the boundary are discontinuous at the virtual corner point. These integrals are calculated using Gaussian quadrature and the indicator function Ξ_B (*cf.* Section 5.5).

Fig. 23 shows the dependence of various relative norms on mesh size h. From these results, the convergence rates for the case of classical FEM (see Fig. 23a) of the errors (20)–(22) for triangular linear elements are $O(h)^{0.95}$, $O(h)^2$ and $O(h)^{1.33}$, respectively. These indicate that the simulations reached a reasonable convergence of the approximated Dirichlet and flux on the bottom boundary of the square, and provide the rate of convergence in the energy norm slightly below the theoretical value of the linear interpolation, *i.e.* $O(h)^{p=1}$. However, extended finite element solutions (XFEM) with the standard mixed method for the domain defined implicity does not yield optimal convergence rates (see Fig. 23b). The suboptimal convergence rate in the energy norm is associated with significantly higher approximation of the Dirichlet boundary condition on the bottom boundary of the square. This gives rise to over-constraint of the primal variable space which yields oscillations of the Lagrange multipliers on the boundary. These oscillations are already observed in [68,27] within an XFEM framework, and can be alleviated by reducing the Lagrange multiplier space $\mathscr{L}_h$ as suggested by Moës et al. [27].

Fig. 24 shows the extended finite element solutions u^h of the problem (23) obtained on the implicit square domain with mixed Dirichlet and Neumann boundary conditions. For Dirichlet conditions, the reduced Lagrange multiplier space $\mathscr{L}_h^* \subset \mathscr{L}_h$, proposed for the treatment of frictional contact along an interface [28], is adopted.

The new results with the reduced Lagrange multiplier space $\mathscr{L}_h^*$ are presented in Fig. 25 where it is clear that analysis on a fully implicit computational domain produces very similar results to those obtained by a conforming mesh (see Fig. 23a). The convergence studies also show that analysis on a domain defined implicitly with

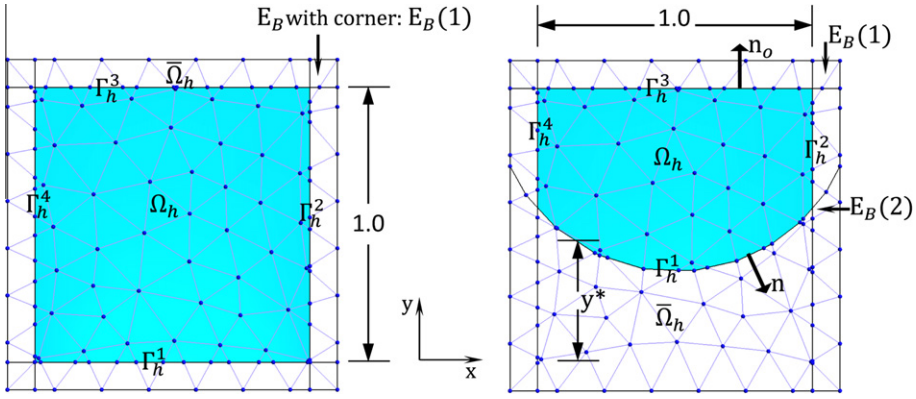

Fig. 22. Computational domains Ω_h for the free boundary problems. Approximation of each domain by four zero level sets ϕ_h^i. Extractions of the polyline meshes Γ_h^i those corresponding to ϕ_h^i for subdivision purpose. Only the parts of the polyline meshes Γ_h^i enclosed in the set of the boundary elements B will be used for imposing the boundary conditions. Boundary elements $E_B \in B$ those contain corners are separated into two categories: noted by $E_B(1)$ and $E_B(2)$. $E_B(1)$ and $E_B(2)$ are the boundary elements which include one type and a mix of boundary conditions, respectively.

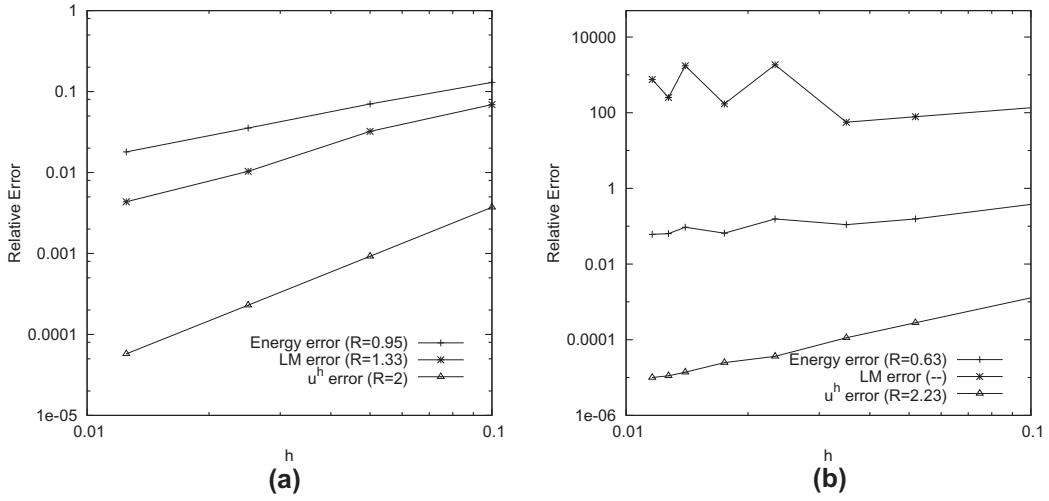

Fig. 23. (a) Convergence results (FEM) on an unstructured conforming triangular mesh and (b) convergence results (XFEM) for implicit computational domain on an unstructured triangular mesh using a full Lagrange multiplier space $\mathscr{L}_h$.

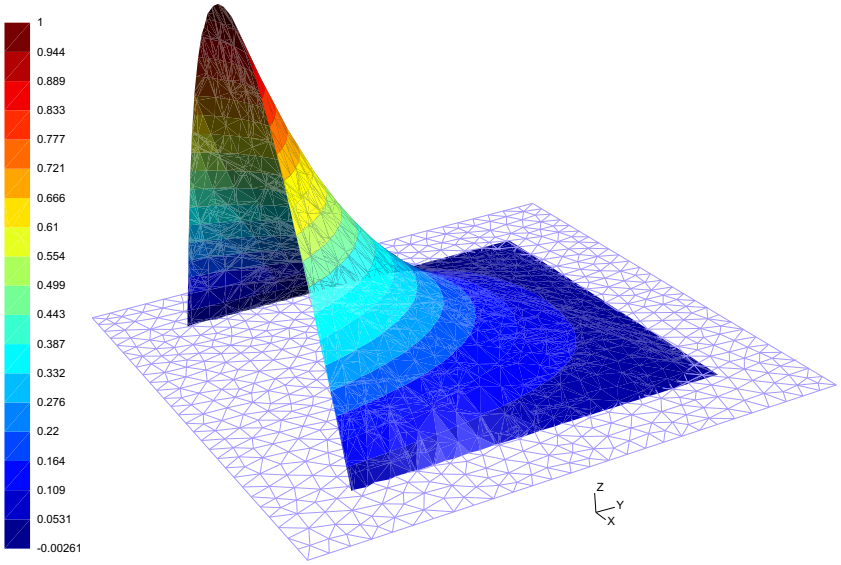

Fig. 24. Extended finite element solution for the 2D Laplace model problem using an implicit computational domain.

sharp features can be realized as in the case of a conforming mesh, the standard mixed method (naïve approach) was largely ameliorated by using a reduced Lagrange multiplier space and the present method possesses an excellent rate of convergence in the energy norm (20), the error on the enforcement of the Dirichlet constraint (21) as well as for the error on the Lagrange multipliers (22).

7.1.2. Implicit curved domain

An implicit curved domain problem is treated in order to analyze the enforcement of the Dirichlet constraint on the curved part of the boundary. Fig. 26 shows the extended finite element solutions u^h of the problem (23) obtained on the implicit curved domain with mixed Dirichlet and Neumann boundary conditions.

The Lagrange multiplier (flux) distribution along the curved part of the domain is shown in Fig. 27. As can be seen from this figure, the normal fluxes on the curved interface are acceptable and without oscillation.

The results of the convergence study are shown in Fig. 28 where the rates of convergence on the errors (20)–(22) for linear interpolation are $O(h)^{0.93}$, $O(h)^{1.99}$ and $O(h)^{1.18}$, respectively. The computed rate in the energy norm for the curved domain is slightly below the theoretical value of the linear interpolation. This level of accuracy is comparable to that obtained with conforming mesh and is superior to the standard mixed method (naive approach). A quadratic convergence is obtained on the Dirichlet conditions and an acceptable accuracy and rate of convergence are observed for the Lagrange multipliers λ^h.

The numerical studies show that the present method is of comparable accuracy and stability to conforming FEM even for sharp features and curved boundaries. Further numerical experiments which are not reported here confirm the importance of proper integration along boundaries with sharp features by using the indicator function Ξ_B (*cf.* Section 5.5), and therefore, suboptimal rates of convergence and a lower accuracy are obtained in comparison with Fig. 28. This is due to the fact that the error is concentrated in the boundary elements E_B containing corners. The need for a proper integration using the indicator function Ξ_B in the boundary elements E_B containing sharp features is clear in all the numerical

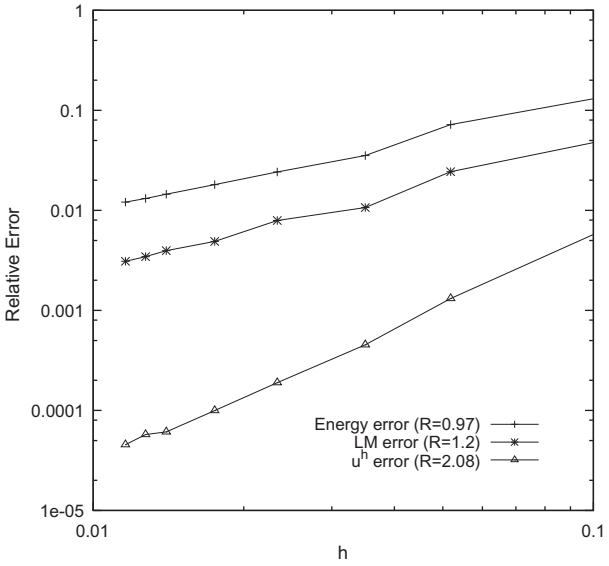

Fig. 25. Convergence results for an implicit computational domain on an unstructured triangular mesh using the reduced Lagrange multiplier space.

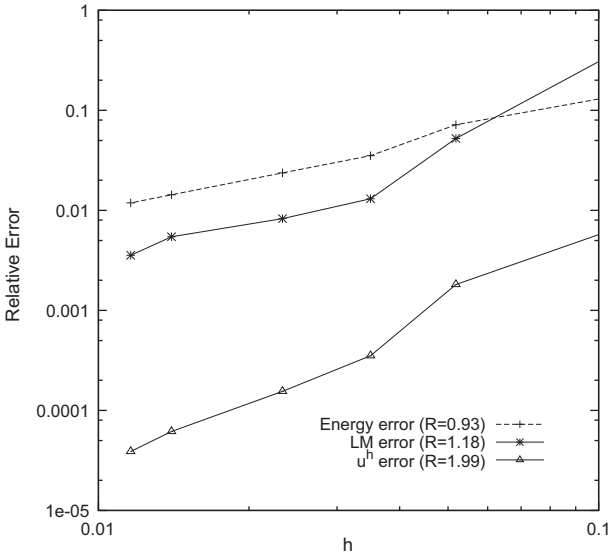

Fig. 28. Convergence results for implicit computational curved domain on unstructured triangular mesh using of the reduced Lagrange multiplier space.

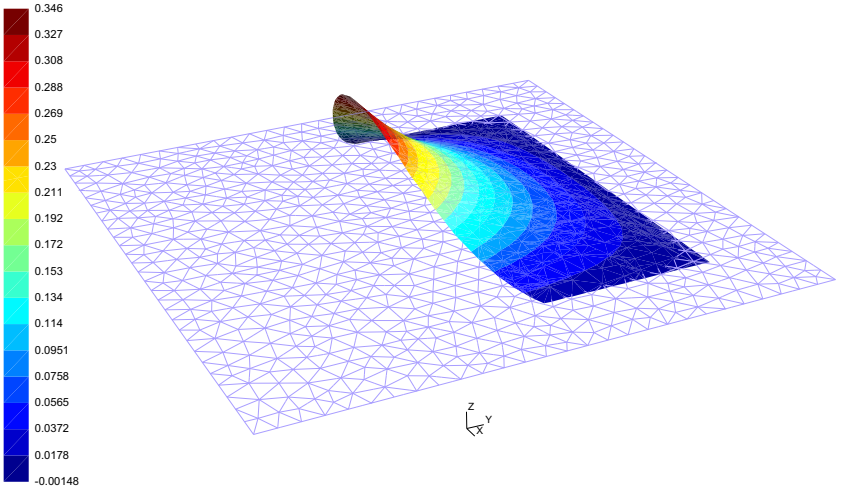

Fig. 26. Extended finite element solution for the 2D Laplace model problem using an implicit curved domain.

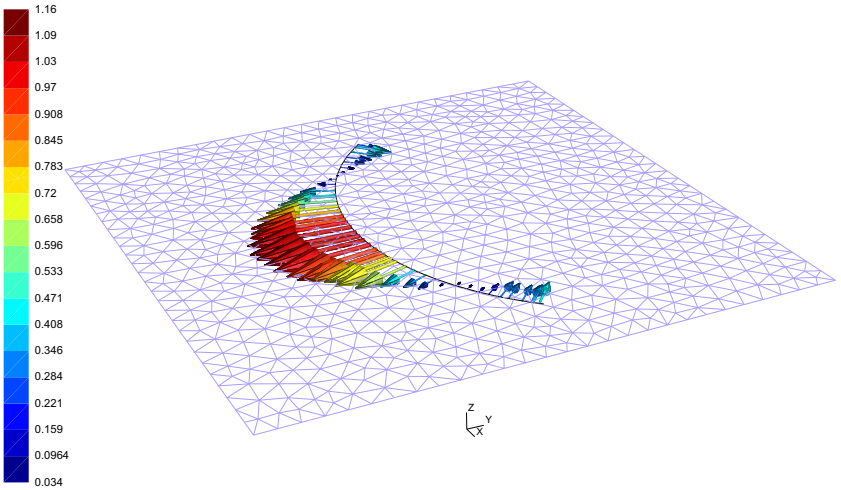

Fig. 27. Extended finite element solution of the normal flux λ^h on the circular boundary without oscillations.

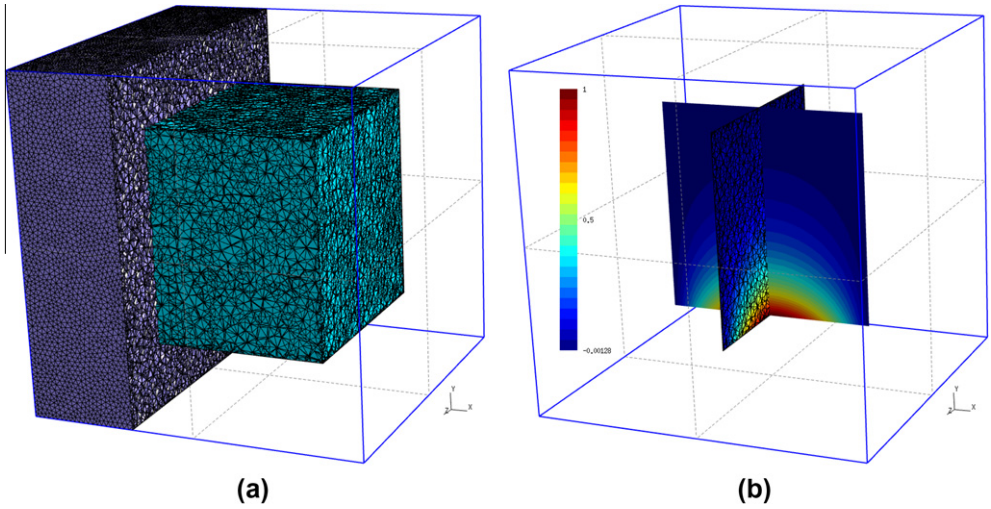

Fig. 29. Finite element solution of 3D Laplace model problem using implicit computational domain: (a) implicit representation of the domain with sharp features and (b) illustrate the cut view of the solution u^h.

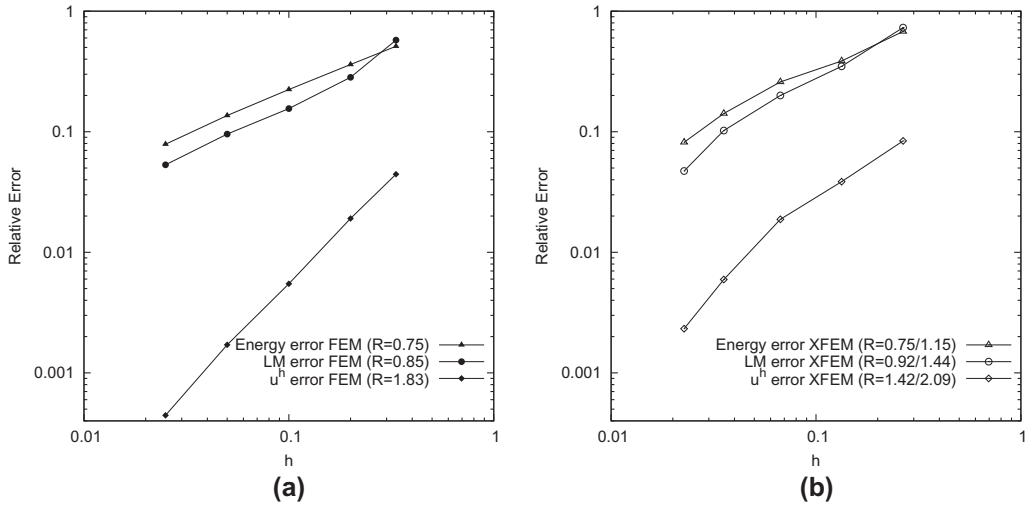

Fig. 30. Convergence study results for the mixed formulation on unstructured tetrahedral mesh: (a) analysis with a conforming mesh and FEM and (b) analysis with a non-conforming mesh and XFEM using the reduced Lagrange multiplier space.

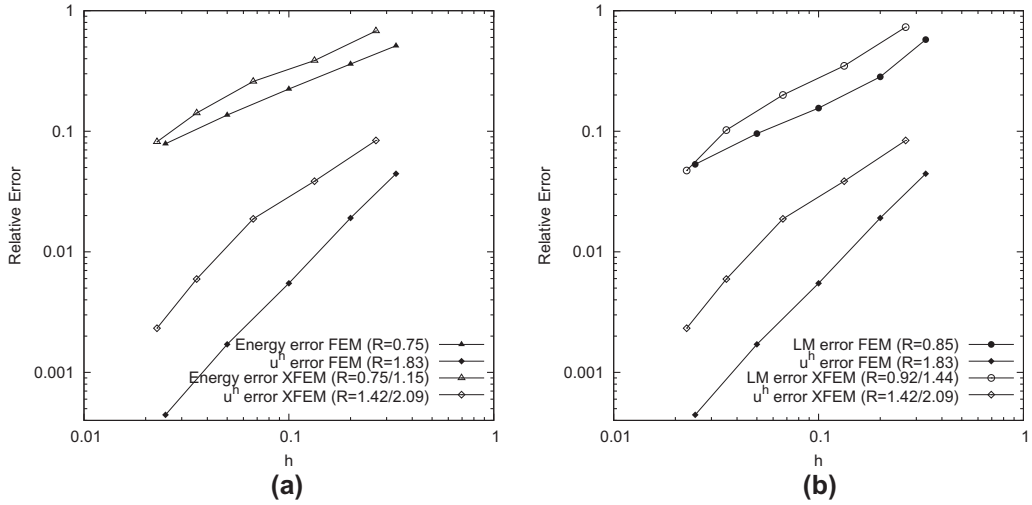

Fig. 31. Comparison study between analysis with conforming mesh (Fig. 30a) and non-conforming mesh (Fig. 30b).

tests presented with a fully implicit representation of object boundaries.

7.2. Numerical validation for a three-dimensional problem

We consider the same problem as studied above in three dimensions, where Ω_h is a unit cube $[0,1] \times [0,1] \times [0,1]$, such that the analytical solution is given by

$$u^{ex}(x,y,z) = \sin(\pi x)\sin(\pi z)\frac{\sinh[\pi\sqrt{2}(1-y)]}{\sinh(\pi\sqrt{2})}. \quad (25)$$

Dirichlet boundary conditions, u_D, corresponding to the analytical solution, are imposed on the lower side ($y = 0$) of the cube, and Neumann boundary conditions, also corresponding to the analytical normal flux, are imposed on the five other faces. The sharp features of the implicitly represented domain are exactly approximated on the background mesh using six zero level sets and the cutting method (*cf.* Section 5.2). Fig. 29 shows the domain defined over an unstructured mesh grid and the extended finite element solutions u^h of the Laplace problem obtained on this implicit representation with mixed Dirichlet and Neumann boundary conditions.

For comparison purposes, we present numerical tests with the same problem using a standard finite element method with a mixed formulation for the Laplace problem, where the mesh conforms to the domain boundary. In the case of the implicit representation, the reduced Lagrange multiplier space $\mathscr{L}_h^* \subset \mathscr{L}_h$ is constructed along the bottom boundary of the triangulated surface mesh. This reduced space is chosen using the set of vertices contained in the set of boundary elements B.

The evolution of relative errors norms (20)–(22) for the Laplace problem in the mesh size h is shown in Fig. 30a for the case of a conforming mesh with FEM and in Fig. 30b for the case of a non-conforming mesh with XFEM. Comparing the results of FEM (see Fig. 30a) to the theoretical values for a linear interpolation, *i.e.* $O(h)^{p=1}$ for the energy and $O(h)^{p+1=2}$ for the enforcement of the Dirichlet constraint, we note that the actual convergence rates are slightly below the optimal values. We therefore then compared the accuracy and the rates of convergence of the classical FEM and our implicit XFEM representation. These comparisons are shown in Fig. 31. As can be seen, the analysis with conforming and non-conforming mesh yield nearly the same accuracy and convergence rates in the approximated energy and Lagrange multipliers. As to the enforcement of the Dirichlet boundary conditions, the accuracy and convergence rate are governed by the choice of the Lagrange multiplier space $\mathscr{L}_h^*$. It is interesting to note that all these numerical results for the case of a non-conforming mesh are superior to the standard mixed method (naïve approach), which yields oscillations of the Lagrange multipliers on the boundary.

8. Conclusions

We presented and validated a general method to carry out finite element analysis on arbitrary implicitly defined domains obtained from parametric surfaces. The input to the algorithm is the parametric description of the boundary of the object which is converted automatically and efficiently into implicit level set representations. The computational domain is then obtained by Boolean operations on those level set functions. A special adaptive numerical integration technique which uses the parametric description to increase the geometrical faithfulness (thus decrease mesh dependence) was proposed. We showed that the resulting algorithm is adequate to describe objects with sharp features such as edges and corners.

The above paradigm required several contributions:

- an algorithm for fast conversion of a smooth parametric function into a discretized signed distance field on an arbitrary unstructured background mesh. For objects with sharp features, we use several zero level sets defined on narrow bands. This algorithm automatically converts parametric functions such as those used in computer aided design (CAD) into implicit level set representations. This implies that the framework can easily be coupled to existing CAD software;
- an adaptive integration algorithm to represent the geometry accurately where the parametric information can be used as a guide to generate the profile of the curved region inside an element as needed by higher order FE methods;
- a mixed formulation to weakly enforce Dirichlet conditions along boundaries so that the analysis can be performed without requiring any user-defined stabilization parameters.

We have shown through several numerical experiments that the proposed methodology yields very similar results to those obtained by a conforming mesh even for domains with sharp features and curved boundaries.

This method can be considered as an extension of the work done by Belytschko et al. [9] who first introduced the idea of using an XFEM framework within a fully immersed domain constructed from a single level set. The proposed method is an extension of this idea to domains described based on multiple level sets obtained from arbitrary parametric descriptions on general unstructured meshes.

Though the approach was only tested for fixed boundaries and no enrichment was used, there is no additional difficulty in applying this framework to more complex cases, such as the treatment of moving discontinuities. Indeed, our representation is based on a signed distance field and can be used to construct enrichment functions within the XFEM framework as well as possible evolving boundaries within a level set framework.

Acknowledgements

The authors would like to acknowledge the financial support of the Ministry of Research and High Education of the Grand Duchy of Luxembourg (AFR 07/070). The first author is grateful to Gaston Rauchs and Ahmed Makradi of the Modelling and Simulation (ModSi) Unit CRP Henri Tudor for helpful discussions. Stéphane Bordas would like to thank the financial support of the Royal Academy of Engineering and of the Leverhulme Trust for his Senior Research Fellowship entitled "Towards the next generation surgical simulators" as well as the support of EPSRC under Grant EP/G069352/1 Advanced Discretisation Strategies for "Atomistic" Nano CMOS Simulation and EP/G042705/1 Increased Reliability for Industrially Relevant Automatic Crack Growth Simulation with the eXtended Finite Element Method.

References

[1] S.P.A. Bordas, T. Rabczuk, J.-J. Rodenas, P. Kerfriden, M. Moumnassi, S. Belouettar, Recent advances towards reducing the meshing and re-meshing burden in computational sciences, Comput. Tech. Rev. 2 (2010) 51–82. doi:10.4203/ctr.2.3.

[2] I. Babuška, J.M. Melenk, The partition of unity method, Int. J. Numer. Methods Engrg. 40 (4) (1997) 727–758.

[3] T. Strouboulis, K. Copps, I. Babuška, The generalized finite element method, Comput. Methods Appl. Mech. Engrg. 190 (32–33) (2001) 4081–4193.

[4] R. Mittal, G. Iaccarino, Immersed boundary methods, Ann. Rev. Fluid Mech. 37 (1) (2005) 239–261.

[5] R. Glowinski, T.-W. Pan, J. Periaux, A fictitious domain method for Dirichlet problem and applications, Comput. Methods Appl. Mech. Engrg. 111 (3–4) (1994) 283–303.

[6] H. Johansen, P. Colella, A cartesian grid embedded boundary method for Poisson's equation on irregular domains, J. Comput. Phys. 147 (1) (1998) 60–85.

[7] E.M. Saiki, S. Biringen, Numerical simulation of a cylinder in uniform flow: application of a virtual boundary method, J. Comput. Phys. 123 (2) (1996) 450–465.
[8] T. Ye, R. Mittal, H.S. Udaykumar, W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries, J. Comput. Phys. 156 (2) (1999) 209–240.
[9] T. Belytschko, C. Parimi, N. Moës, N. Sukumar, S. Usui, Structured extended finite element methods for solids defined by implicit surfaces, Int. J. Numer. Methods Engrg. 56 (4) (2003) 609–635.
[10] T. Belytschko, S.P. Xiao, C. Parimi, Topology optimization with implicit functions and regularization, Int. J. Numer. Methods Engrg. 57 (8) (2003) 1177–1196.
[11] N. Sukumar, J.H. Prévost, Modeling quasi-static crack growth with the extended finite element method. Part i: Computer implementation, Int. J. Solids Struct. 40 (26) (2003) 7513–7537.
[12] S. Bordas, P.V. Nguyen, C. Dunant, A. Guidoum, H. Nguyen-Dang, An extended finite element library, Int. J. Numer. Methods Engrg. 71 (6) (2007) 703–732.
[13] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1) (1988) 12–49.
[14] V.L. Rvachev, T.I. Sheiko, V. Shapiro, I. Tsukanov, Transfinite interpolation over implicitly defined sets, Comput. Aided Geom. Des. 18 (3) (2001) 195–220.
[15] N. Moës, M. Cloirec, P. Cartraud, J.F. Remacle, A computational approach to handle complex microstructure geometries, Comput. Methods Appl. Mech. Engrg. 192 (28–30) (2003) 3163–3177.
[16] A. Rappoport, S. Spitz, Interactive boolean operations for conceptual design of 3-d solids, in: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press, Addison-Wesley Publishing Co., 1997, pp. 269–278.
[17] A. Düster, J. Parvizian, Z. Yang, E. Rank, The finite cell method for three-dimensional problems of solid mechanics, Comput. Methods Appl. Mech. Engrg. 197 (45–48) (2008) 3768–3782.
[18] J.J. Laguardia, E. Cueto, M. Doblaré, A natural neighbour Galerkin method with quadtree structure, Int. J. Numer. Methods Engrg. 63 (6) (2005) 789–812.
[19] K. Terada, M. Kurumatani, An integrated procedure for three-dimensional structural analysis with the finite cover method, Int. J. Numer. Methods Engrg. 63 (15) (2005) 2102–2123.
[20] L.P. Kobbelt, M. Botsch, U. Schwanecke, H.-P. Seidel, Feature sensitive surface extraction from volume data, in: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, ACM, 2001, pp. 57–66.
[21] S.F. Frisken, R.N. Perry, A.P. Rockwood, T.R. Jones, Adaptively sampled distance fields: a general representation of shape for computer graphics, in: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press, Addison-Wesley Publishing Co., 2000, pp. 249–254.
[22] F. Liehr, T. Preusser, M. Rumpf, S. Sauter, L.O. Schwen, Composite finite elements for 3d image based computing, Comput. Vis. Sci. 12 (4) (2009) 171–188.
[23] N. Moës, J. Dolbow, T. Belytschko, A finite element method for crack growth without remeshing, Int. J. Numer. Methods Engrg. 46 (1) (1999) 131–150.
[24] S. Natarajan, S. Bordas, D.R. Mahapatra, Numerical integration over arbitrary polygonal domains based on Schwarz–Christoffel conformal mapping, Int. J. Numer. Methods Engrg. 80 (1) (2009) 103–134.
[25] S. Natarajan, D.R. Mahapatra, S.P.A. Bordas, Integrating strong and weak discontinuities without integration subcells and example applications in an xfem/gfem framework, Int. J. Numer. Methods Engrg. 83 (3) (2010) 269–294.
[26] S.P.A. Bordas, T. Rabczuk, N.-X. Hung, V.P. Nguyen, S. Natarajan, T. Bog, D.M. Quan, N.V. Hiep, Strain smoothing in fem and xfem, Comput. Struct., in press, doi:10.1016/j.compstruc.2008.07.006.
[27] N. Moës, E. Béchet, M. Tourbier, Imposing Dirichlet boundary conditions in the extended finite element method, Int. J. Numer. Methods Engrg. 67 (12) (2006) 1641–1669.
[28] S. Géniaut, P. Massin, N. Moës, A stable 3d contact formulation using x-fem, Eur. J. Comput. Mech. 16 (2007) 259–275.
[29] É. Béchet, N. Moës, B. Wohlmuth, A stable Lagrange multiplier space for stiff interface conditions within the extended finite element method, Int. J. Numer. Methods Engrg. 78 (8) (2009) 931–954.
[30] H.M. Mourad, J. Dolbow, I. Harari, A bubble-stabilized finite element method for Dirichlet constraints on embedded interfaces, Int. J. Numer. Methods Engrg. 69 (4) (2007) 772–793.
[31] A. Hansbo, P. Hansbo, An unfitted finite element method, based on Nitsche's method, for elliptic interface problems, Comput. Methods Appl. Mech. Engrg. 191 (47–48) (2002) 5537–5552.
[32] J. Dolbow, I. Harari, An efficient finite element method for embedded interface problems, Int. J. Numer. Methods Engrg. 78 (2) (2009) 229–252.
[33] J. Dolbow, L. Franca, Residual-free bubbles for embedded Dirichlet problems, Comput. Methods Appl. Mech. Engrg. 197 (45–48) (2008) 3751–3759.
[34] S. Fernández-Méndez, A. Huerta, Imposing essential boundary conditions in mesh-free methods, Comput. Methods Appl. Mech. Engrg. 193 (12–14) (2004) 1257–1275.
[35] A.J. Lew, G.C. Buscaglia, A discontinuous-Galerkin-based immersed boundary method, Int. J. Numer. Methods Engrg. 76 (4) (2008) 427–454.
[36] A.V. Kumar, S. Padmanabhan, R. Burla, Implicit boundary method for finite element analysis using non-conforming mesh or grid, Int. J. Numer. Methods Engrg. 74 (9) (2008) 1421–1447.
[37] N. Sukumar, D.L. Chopp, N. Moës, T. Belytschko, Modeling holes and inclusions by level sets in the extended finite-element method, Comput. Methods Appl. Mech. Engrg. 190 (46–47) (2001) 6183–6200.
[38] R.K. Burla, A.V. Kumar, Implicit boundary method for analysis using uniform b-spline basis and structured grid, Int. J. Numer. Methods Engrg. 76 (13) (2008) 1993–2028.
[39] P. Bastian, C. Engwer, An unfitted finite element method using discontinuous galerkin, Int. J. Numer. Methods Engrg. 79 (12) (2009) 1557–1576.
[40] K. Dréau, N. Chevaugeon, N. Moës, Studied x-fem enrichment to handle material interfaces with higher order finite element, Comput. Methods Appl. Mech. Engrg. 199 (29–32) (2010) 1922–1936.
[41] A. Legay, H.W. Wang, T. Belytschko, Strong and weak arbitrary discontinuities in spectral finite elements, Int. J. Numer. Methods Engrg. 64 (8) (2005) 991–1008.
[42] K.W. Cheng, T.-P. Fries, Higher-order xfem for curved strong and weak discontinuities, Int. J. Numer. Methods Engrg. 82 (5) (2010) 564–590.
[43] J.P. Pereira, C.A. Duarte, D. Guoy, X. Jiao, hp-generalized fem and crack surface representation for non-planar 3-d cracks, Int. J. Numer. Methods Engrg. 77 (5) (2009) 601–633.
[44] U.M. Mayer, A. Gerstenberger, W.A. Wall, Interface handling for three-dimensional higher-order xfem-computations in fluid–structure interaction, Int. J. Numer. Methods Engrg. 79 (7) (2009) 846–869.
[45] C. Geuzaine, J.-F. Remacle, Gmsh: a 3-d finite element mesh generator with built-in pre- and post-processing facilities, Int. J. Numer. Methods Engrg. 79 (11) (2009) 1309–1331.
[46] N. Moës, A. Gravouil, T. Belytschko, Non-planar 3d crack growth by the extended finite element and level sets. Part i: Mechanical model, Int. J. Numer. Methods Engrg. 53 (11) (2002) 2549–2568.
[47] E. Hartmann, On the curvature of curves and surfaces defined by normalforms, Comput. Aided Geom. Des. 16 (5) (1999) 355–376.
[48] Y.L. Ma, W.T. Hewitt, Point inversion and projection for nurbs curve and surface: control polygon approach, Comput. Aided Geom. Des. 20 (2) (2003) 79–99.
[49] J.-F. Remacle, M.S. Shephard, An algorithm oriented mesh database, Int. J. Numer. Methods Engrg. 58 (2) (2003) 349–374.
[50] C. Dunant, K. Scrivener, Micro-mechanical modelling of alkali-silica-reaction-induced degradation using the AMIE framework, Cement Concrete Res. 40 (4) (2010) 517–525.
[51] C. Dunant, P. Vinh, M. Belgasmia, S. Bordas, A. Guidoum, Architecture tradeoffs of integrating a mesh generator to partition of unity enriched object-oriented finite element software, Revue Eur. Méc. Numér. 16 (2007) 237–258.
[52] C. Dunant, Experimental and Modelling Study of the Alkali-Silica-Reaction in Concrete, Ph.D. Thesis, École Polytechnique Fédérale de Lausanne, 2009.
[53] T.J. Barth, J.A. Sethian, Numerical schemes for the Hamilton–Jacobi and level set equations on triangulated domains, J. Comput. Phys. 145 (1) (1998) 1–40.
[54] T. Hughes, J. Cottrell, Y. Bazilevs, Isogeometric analysis: cad, finite elements, nurbs, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (39–41) (2005) 4135–4195.
[55] N. Sukumar, N. Moës, B. Moran, T. Belytschko, Extended finite element method for three-dimensional crack modelling, Int. J. Numer. Methods Engrg. 48 (11) (2000) 1549–1570.
[56] G. Ventura, On the elimination of quadrature subcells for discontinuous functions in the extended finite-element method, Int. J. Numer. Methods Engrg. 66 (5) (2006) 761–795.
[57] G. Liu, K. Dai, T. Nguyen, A smoothed finite element method for mechanics problems, Comput. Mech. 39 (2007) 859–877.
[58] G.R. Liu, T.T. Nguyen, K.Y. Dai, K.Y. Lam, Theoretical aspects of the smoothed finite element method (sfem), Int. J. Numer. Methods Engrg. 71 (8) (2007) 902–930.
[59] H. Nguyen-Xuan, S. Bordas, H. Nguyen-Dang, Smooth finite element methods: convergence, accuracy and properties, Inter. J. Numer. Methods Engrg. 74 (2) (2008) 175–208.
[60] H. Nguyen-Xuan, T. Rabczuk, S. Bordas, J. Debongnie, A smoothed finite element method for plate analysis, Comput. Methods Appl. Mech. Engrg. 197 (13–16) (2008) 1184–1203.
[61] N. Nguyen-Thanh, T. Rabczuk, H. Nguyen-Xuan, S.P.A. Bordas, A smoothed finite element method for shell analysis, Comput. Methods Appl. Mech. Engrg. 198 (2) (2008) 165–177.
[62] T. Rabczuk, G. Zi, S. Bordas, H. Nguyen-Xuan, A simple and robust three-dimensional cracking-particle method without enrichment, Comput. Methods Appl. Mech. Engrg. 199 (37–40) (2010) 2437–2455.
[63] Z.C. Xuan, T. Lassila, G. Rozza, A. Quarteroni, On computing upper and lower bounds on the outputs of linear elasticity problems approximated by the smoothed finite element method, Int. J. Numer. Methods Engrg. 83 (2) (2010) 174–195.
[64] B. Szabó, I. Babuska, Finite Element Analysis, Wiley, 1991.
[65] I. Babuška, The finite element method with Lagrangian multipliers, Numer. Math. 20 (3) (1973) 179–192.
[66] V.L. Rvachev, T.I. Sheiko, *R*-functions in boundary value problems in mechanics, Appl. Mech. Rev. 48 (4) (1995) 151–188.
[67] D. Chapelle, K. Bathe, The inf–sup test, Comput. Struct. 47 (4–5) (1993) 537–545.
[68] H. Ji, J.E. Dolbow, On strategies for enforcing interfacial constraints and evaluating jump conditions with the extended finite element method, Int. J. Numer. Methods Engrg. 61 (14) (2004) 2508–2535.