Semi-implicit representation of sharp features with level sets

H. Asadi Kalameh[a,b], O. Pierard[a], C. Friebel[a], E. Béchet[b]

[a]*Morfeo team, Cenaero*
Rue des frères Wright, 29, B-6041 Gosselies, Belgium
[b]*Department of Aerospace and Mechanical Engineering, Universityé de Liège,*
Allée de la Découverte 9, 4000 Liège, Belgium

Abstract

The present contribution enriches the nowadays "classical" level set implicit representation of geometries with topological information in order to correctly represent sharp features. For this, sharp features are classified according to their positions within elements of the level set support. Based on this additional information, sub-elements and interface-mesh used in a finite element context for integration and application of boundary conditions are modified to match exactly to the sharp features. In order to analyze evolving geometries, Boolean operations on these semi-implicit representations are derived so that the minimal additional information to represent correctly the new geometry is stored. This approach has been successfully applied to complex two-dimensional geometries. It computes in a robust way numerous Boolean operations and guarantees the precision and the convergence rate of the numerical simulations.

Keywords: level set method, sharp feature, implicit representation, Boolean operation, finite elements

1. Introduction

The level set method (LSM) was originally introduced by Osher and Sethian (1; 2; 3) as a robust technique to represent implicitly the evolution of interfaces which have a smooth geometry in two or three dimensions. The representation of the interface, or more generally any boundary, is obtained by the iso-zero of the level set function, classically a distance function to the boundary. This function is defined on a grid so that it is suitable for using it within a finite element context. A major advantage is that the simulation mesh does not need to match the boundary anymore. In case of a moving interface in the normal direction, a Hamilton-Jacobi equation - also called the level set equation - is solved to track the interface.

This method has been successfully employed in a wide variety of applications such as the solidification process (4), crystal growth (5), crack representation (6), image processing (7) or multi-phases flows (8; 9). Another key topic for which implicit representation with level set is helpful is topology optimization (10; 11; 12).

Implicit representation of smooth interfaces is particularly efficient with the level set method. However, when the boundary has small curvature radius, sharp features like corners or small items with respect to the characteristic length of the grid, a smoothing effect is observed. In some cases, as for the implicit representation of a CAD model, this might be unacceptable. Several improvements have been proposed over the years in order to circumvent these limitations. A first approach is to use higher-order level sets instead of the classical first-order interpolation (13). Accurate integration requires a particular attention (14). Such an approach has been successfully used in several applications, including magneto-mechanical problems (15). Another approach is to dissociate the computation mesh from the grid on which the level set is defined as adopted by Legrain *et al.* (16). Typically, a finer grid might be used in regions where sharp geometrical features are located. This latter approach has been used by Legrain *et al.* (17) to represent implicitly CAD thin structures. Even if these two approaches improve the implicit representation by reducing the geometrical error, both are unable to represent exactly sharp features.

Using several level sets to represent accurately sharp features as corners or edges has been set up by Moumnassi *et al.* (18). Typically, each level set represents one basic geometric feature like a plane and Boolean operations are performed between them to capture an intersecting edge. This approach is also coupled to level set definition on a sub-grid to improve representation of curvatures. In Tran *et al.* (19), several level sets are also used for the representation of complex microstructures, each one representing an inclusion.

In the present paper, it is proposed to use a single level set for the implicit representation of the structure. For a correct representation of sharp features, information from the geometry is added to the level set so that the representation is not purely implicit anymore but semi-implicit.

The paper is structured as follows. In section 2, the definition of level set to represent implicitly a boundary is recalled as well as the way to use it in a finite element context. The effect of smoothing corners is highlighted. In section 3, the concept of *level set plus* is introduced. It enriches the classical level set with geometrical information to correctly represent the sharp features. In section 4, Boolean operations on level set plus are examined. Finally, in section 5, numerical validations are presented for semi-implicit geometry representation, Boolean operations and finite element computations. In what

follows “corner” is used along with of “sharp feature” whit the same meaning.

2. Interface representation with level set

In this section, basic notations and methodologies used for representation of interfaces with classical level set in the context of a finite element problem are recalled. Problems related to the capture of corners are highlighted.

2.1. The classical level set method

The primary concept of the level set technique is to implicitly describe an interface Γ by a function (3). The level set function Φ is a signed distance function with respect to Γ such that the following sign convention applies:

$$\begin{cases} \Phi(x) < 0 \Leftrightarrow & x \in \Omega^- \\ \Phi(x) = 0 \Leftrightarrow & x \in \Gamma \\ \Phi(x) > 0 \Leftrightarrow & x \in \Omega^+ \end{cases} \quad (1)$$

where x is a point in the level set support Ω containing the interface. The level set support can be defined on an unstructured or structured non-conforming mesh. Ω^- and Ω^+ are sub-domains of Ω on both sides of the interface such that $\Omega^- \cup \Gamma \cup \Omega^+ = \Omega$ and $\Omega^- \cap \Omega^+ = \emptyset$. When the level set is used to represent implicitly a structure embedded within the level set support, Ω^- is the structure and Γ is the boundary. In this paper, the terms *internal* and *external* regions respectively correspond to Ω^- and Ω^+.

2.2. Usage of level set within a finite element simulation

In the context of this work, when the level set is used within a finite element simulation, Ω is considered to be a 2D triangular first-order mesh. Decomposition to simplicial elements is performed for other element types. Level set values are computed at every node belonging to Ω. For nodes which are very close to the interface (e.g., $| \Phi(x) | < 0.01\, e$, where e is the element edge size), the level set value is arbitrarily set to 0 to avoid narrow sub-elements for integration.

In the finite element context, for the imposition of Neumann boundary conditions on the interface, an *Interface-mesh* is reconstructed from the level set. The interface, defined by the iso-0 level set, i.e. a curve on which $\Phi(x) = 0$, is obtained by evaluating a level set field with help of the finite element shape functions. In case of first-order shape functions, the intersection points on edges are determined with a linear interpolation between the level set values calculated at the end-points of each edge.

In order to create bijective relations between mesh entities (vertex, edge and face) of the level set support and intersection points, a tagging system is introduced for linking entities. As illustrated on Figure 1, an interface-tag TAG-I is defined, so that any side of a relation can be retrieved by knowing the other side and the tag. Connecting these intersection points with line segments results in the construction of the *Interface-mesh* which is a poly-line in 2D. As illustrated in the Figure 1, each of these poly-line segments is also tagged to the corresponding element.

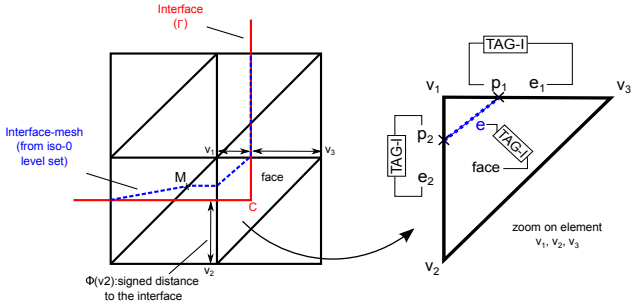


Figure 1: Classical level set approach: creation of the *Interface-mesh* and associated tagging system on the element containing the corner C

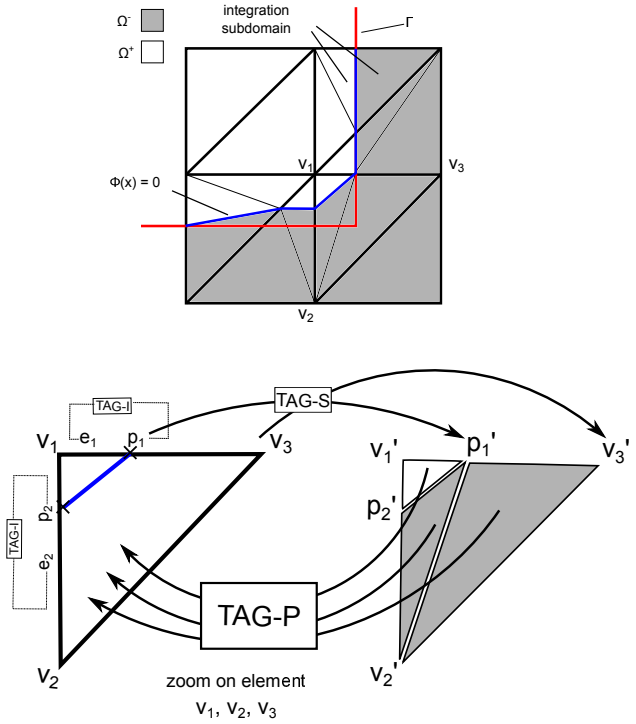


Figure 2: Classical level set approach: creation of the associated *Submesh* on the element containing the corner

In addition to the *Interface-mesh*, a *Submesh* associated to each element cut by the *Interface-mesh* is constructed for the purpose of integration over the element. Typically, different integration rules can be used on both sides of the interface (i.e. materials interface) or even no integration at all on Ω^+ in case of an implicitly defined volume. It is important to notice that the use of *Submesh* is limited to integration and visualization purposes, without introduction of additional degrees of freedom to the finite element problem.

Creation of the *Submesh* is shown on Figure 2. Elements crossed by the iso-0 level set are subdivided into sub-elements (triangles) whose edges are conforming to the iso-0 level set. Similar Gaussian quadrature rule is used on each sub-element as for the uncut elements.

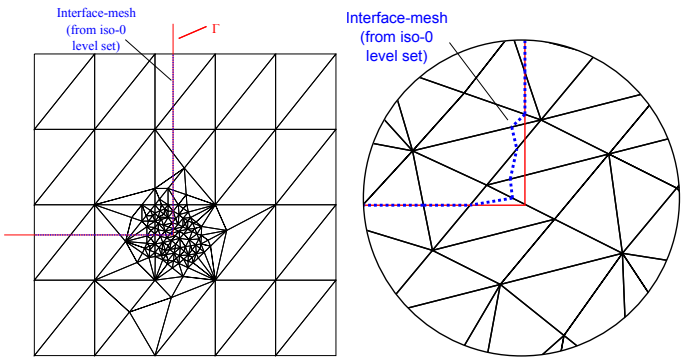
The subdivision algorithm is the following. Intersection points which are detected during construction of the *Interface-mesh* are retrieved from support entities using the interface-tag (TAG-I) and then subdivision is performed for each element so that each sub-element is located on either side of the *Interface-mesh*. As shown in Figure 2, intersection points and the vertices are duplicated and related by using the submesh-tag (TAG-S). Sub-elements are constructed from the duplicated vertices. The position (internal or external) of the sub-elements is determined by discussing the sign of the level set values on the vertices of the parent element. Alternatively Moumnassi *et al.* (18) detect the position of sub-elements using sign of level set at their centroids. All sub-elements are linked to their parent elements using partitioning-tag (TAG-P).

2.3. On the capture of corners with classical level sets

The classical level set is shown to be a versatile, robust and efficient technique for a wide class of problems. However, the level set description as presented above fails at representing precisely geometries which contain corners or small details compared to the level set support characteristic length. As illustrated in the Figure 1, the classical level set tends to erase or smooth out the corner C when using linear interpolation inside an element.

The effect of this drawback is not only limited to the elements containing corners. As shown in Figure 1, the signed distance of vertex v_1 to the interface (red line) is the distance to the vertical leg of the interface (i.e. at the intersection with the edge $v_1 - v_3$), but the same value is also used for computing the intersection point M on the other edge associated to v_1. Therefore, the intersection point M obtained from the interpolation along this edge is badly located.

In order to increase the geometrical accuracy, mesh refinement in the vicinity of elements containing corners might be used (see Figure 3(a)). However, as depicted in Figure 3(b), even if the interface representation tends to be more accurate while refining mesh around the corners, it still fails in the sense that the geometric feature associated to the corners is lost and cannot be used, for instance, for the imposition of boundary conditions.



(a) Mesh refinement around corner (b) Zoom around mis-captured iso-0 level set

Figure 3: Mesh refinement around mis-captured corner

Another alternative is to use the multiple level sets approach as proposed by Moumnassi *et al.* (18) which guarantees the accuracy of implicitly represented object around corners with minimal dependency on level sets supports. This method is an adaptation of the parametric description of interface combined with implicit representation so that parametric definition of each interface segment is converted into an implicit form which is defined on the background mesh. By using this technique, the corner depicted in Figure 3 is correctly captured using two level sets, each one representing one segment of Γ, without the need of mesh refinement. However with this technique multiple level set values are required in vicinity of sharp features which can result in an increase of computational cost in terms of memory. The issue with that technique is that the level sets are global; It means that the values of the level sets are known every where leading to a memory footprint which is roughly proportional to the number of sharp features and memory footprint of one global level set. Of course, we could reduce the memory footprint by reducing the support of each level set to its narrow band. In what follows, we have the procedure that achieves the same complexity automatically. There is, however, another drawback in the use of multiple level sets for representing concave corners beside convex ones. This flaw appears when applying Boolean operations between level sets.

3. Interface representation with level set plus

This section introduces the concept of level set plus and the associated additional information to the classical level set for correct capture of sharp features. This is done in two steps : (i) corners detection from the interface Γ and (ii) classification of these corners. Accordingly, modification on the *Interface-mesh* and the *Submesh* become necessary.

3.1. The level set plus method

The level set plus is a semi-implicit technique which combines the classical level set approach with the explicit extraction of the geometrical corners. Contrary to the methods mentioned in section 2.3, no mesh refinement is needed and a single level set is used to represent the interface accurately.

The algorithm proposed for the level set plus technique is presented in Figure 4. The light blocks represent the essential steps which are needed for capturing the interface with the classical level set approach as described in section 2.1 while the dark blocks are the additional steps of the level set plus algorithm to correctly capture corners. Each additional block is described hereafter.

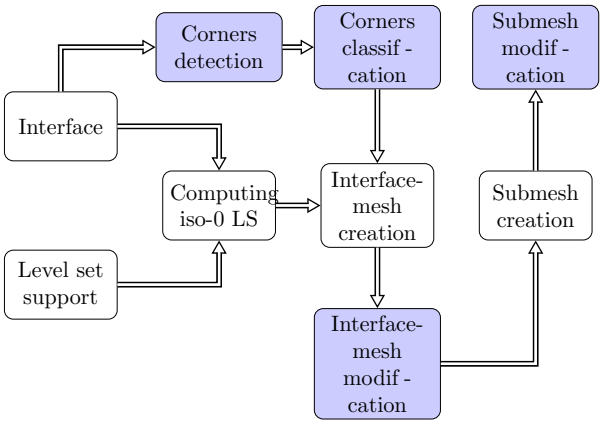


Figure 4: Algorithm for representation of the interface; light blocks: essential steps needed by the classical level set; dark blocks: additional operations required by the level set plus technique

3.1.1. Corners detection

In order to represent sharp features, the first step is to identify corner points from the interface. These corners can be obtained from angular filtering of interface geometry or mesh, or, if available, directly from geometrical information contained within the mesh. These techniques can keep the topological features that are geometrically smooth such as corners located between almost collinear edges. However, detecting corners based on the internal angle between edges gives an opportunity to set a tolerance based on the desired accuracy. Corners which are detected from the interface are stored in a list of corners which is added to the classical level set data structure.

3.1.2. Corners classification

Considering the different configurations of the corner placement with respect to the face in which it lies, the next step is to classify these corners. Corners classification is all about adding necessary data to each corner so that the suitable algorithm is selected during upcoming steps in the *Interface-mesh* and *Submesh* modifications. Ten different configurations are identified and illustrated in Figure 5.

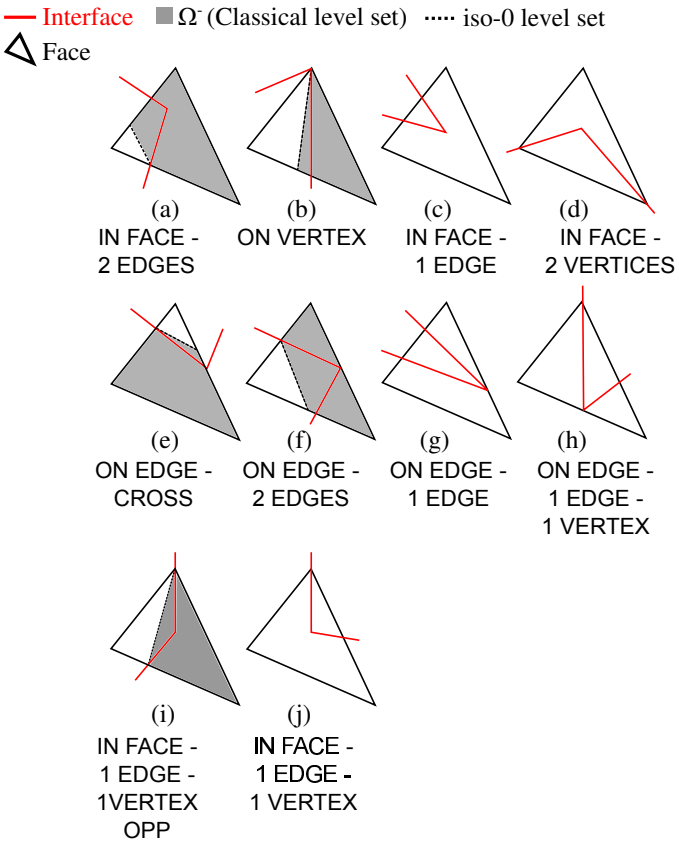


Figure 5: Corners classification

Beside the configurations above, other configurations do exist as depicted in Figure 6. However, as some of the interface edges are much smaller than characteristic element size, those configurations will not be handled by the present algorithm but smoothed by the classical level set approach. This has the advantage of not representing features which are too small with respect to the mesh size.

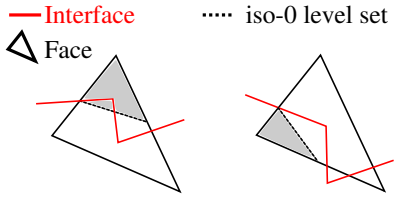


Figure 6: Two additional corner configurations which are not considered within corners classification

Based on the proposed classification, three main attributes are considered for each detected corner plus a fourth one for a limited number of cases only:

- c_{xy} stores the coordinates of the corner.
- c_{elm} is the face of the level set support in which the corner is located. As loop over all faces is time-consuming, algorithm performance is improved by detecting the closest vertex to the corner (with an implementation of the

Nearest-Neighbor algorithm (20)), and then check all faces that are connected to that particular vertex (with the Crossing Number method (21)). The latter algorithm counts the number of times a ray starting from the corner C crosses the face boundary edges. Since the result of the algorithm may be incorrect if the corner lies very close to the face boundary (due to the rounding error), following process applies before checking corner inclusion within the face.

If a corner lies on a vertex (case (b) on Figure 5), c_{elm} attribute is one of the faces (chosen arbitrarily) which is connected to the vertex and is crossed by the interface (a face which has vertices with opposite level set signs).

If a corner lies on an edge (cases (e), (f), (g) and (h) on Figure 5), c_{elm} is assigned as follows. If the interface crosses the edge, case (e), one of the two faces connected to the edge is chosen arbitrarily. For cases (f), (g) and (h), c_{elm} attribute needs more investigation and will be assigned while processing c_{type} (see Equations 2f - 2h).

- c_{edge} holds the edge on which the corner is located (cases (e), (f), (g) and (h) in Figure 5). In other cases c_{edge} remains empty.
- c_{type} indicates the corner classification according to Figure 5. Classification is based on the sign of the level set evaluated at vertices of the element c_{elm}. V is the set of all vertices v belonging to c_{elm} and V_e is the set of vertices belonging to c_{edge}. Classification is done by applying following assortment:

$$
\begin{aligned}
&c_{type} = \text{IN FACE - 2 EDGES} \Leftrightarrow \\
&\quad c_{edge} = \emptyset \;\&\; \forall v \in V : \Phi(v) \neq 0 \;\& \\
&\quad \exists v_1, v_2 \in V : \Phi(v_1)\Phi(v_2) < 0 \qquad (2a) \\
&c_{type} = \text{ON VERTEX} \Leftrightarrow \\
&\quad \exists v_1 \in V : \Phi(v_1) = 0 \;\& \\
&\quad \exists v_2, v_3 \in V : \Phi(v_2)\Phi(v_3) < 0 \qquad (2b) \\
&c_{type} = \text{IN FACE - 1 EDGE} \Leftrightarrow \\
&\quad c_{elm} \neq \emptyset \;\&\; c_{edge} = \emptyset \;\& \\
&\quad \forall v_1, v_2 \in V : \Phi(v_1)\Phi(v_2) > 0 \qquad (2c) \\
&c_{type} = \text{IN FACE - 2 VERTICES} \Leftrightarrow \\
&\quad c_{elm} \neq \emptyset \;\&\; \exists v \in V : \Phi(v) \neq 0 \;\& \\
&\quad \forall v_1, v_2 \in V : \Phi(v_1)\Phi(v_2) = 0 \qquad (2d) \\
&c_{type} = \text{ON EDGE - CROSS} \Leftrightarrow \\
&\quad c_{edge} \neq \emptyset \;\&\; \forall v \in V : \Phi(v) \neq 0 \;\& \\
&\quad \exists v_1, v_2 \in V_e : \Phi(v_1)\Phi(v_2) < 0 \qquad (2e) \\
&c_{type} = \text{ON EDGE - 2 EDGES} \Leftrightarrow \\
&\quad c_{edge} \neq \emptyset \;\&\; \forall v \in V : \Phi(v) \neq 0 \;\& \\
&\quad \forall v_1, v_2 \in V_e : \Phi(v_1)\Phi(v_2) > 0 \;\& \\
&\quad \exists v_1, v_2 \in V : \Phi(v_1)\Phi(v_2) < 0 \qquad (2f) \\
&c_{type} = \text{ON EDGE - 1 EDGE} \Leftrightarrow \\
&\quad c_{edge} \neq \emptyset \;\&\; \forall v \in V : \Phi(v) \neq 0 \;\& \\
&\quad \forall v_1, v_2 \in V_e : \Phi(v_1)\Phi(v_2) > 0 \;\& \\
&\quad \forall v_1, v_2 \in V : \Phi(v_1)\Phi(v_2) > 0 \qquad (2g) \\
&c_{type} = \text{ON EDGE - 1 EDGE - 1 VERTEX} \Leftrightarrow \\
&\quad c_{edge} \neq \emptyset \;\&\; \exists v \in V : \Phi(v) = 0 \;\& \\
&\quad \forall v_1, v_2 \in V_e : \Phi(v_1)\Phi(v_2) > 0 \qquad (2h) \\
&c_{type} = \text{IN FACE - 1 EDGE - 1 VERTEXOPP} \Leftrightarrow \\
&\quad c_{elm} \neq 0 \;\&\; c_{edge} = 0 \;\&\; \exists v_1 \in V : \Phi(v_1) = 0 \;\& \\
&\quad \exists v_2, v_3 \in V : \Phi(v_2)\Phi(v_3) < 0 \qquad (2i) \\
&c_{type} = \text{IN FACE - 1 EDGE - 1 VERTEX} \Leftrightarrow \\
&\quad c_{elm} \neq 0 \;\&\; c_{edge} = 0 \;\&\; \exists v_1 \in V : \Phi(v_1) = 0 \;\& \\
&\quad \exists v_2, v_3 \in V : \Phi(v_2)\Phi(v_3) > 0 \qquad (2j)
\end{aligned}
$$

As shown in Figure 7, there are cases such that even if the corner lies outside the level set support, it still has an effect on the iso-0 level set. These cases will be handled in the *Interface-mesh* and *Submesh* modifications, as described hereafter.

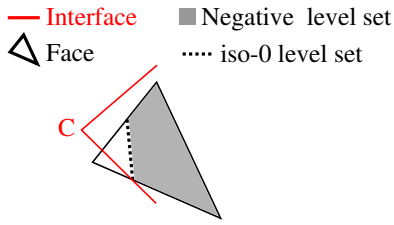


Figure 7: Effect of the corner outside level set support on an element within level set support

Correct classification and assignment of the attributes are crucial for the subsequent operations.

3.2. Usage of level set plus within a finite element simulation

As presented in the section 2.2, *Interface-mesh* and *Submesh* are needed for, respectively, the imposition of a Neumann boundary conditions and integration. As illustrated on Figure 4, modifications of the algorithms are required according to the corner classification.

3.2.1. Interface-mesh modification

As illustrated on Figure 4, modification of the *Interface-mesh* is based on the one obtained from the iso-0 of the classical level set. *Interface-mesh* is adapted in elements containing corners as well as in neighboring elements. Knowing all corners stored in the list of corners, the elements containing corners can be retrieved by checking the c_{elm} attribute.

The first modification consists in correcting the intersection of the *Interface-mesh* with the edges of all these elements containing a corner (e.g. in Figure 1, the edge connecting vertices v_1 and v_2). As shown in Figure 8(b), a wrongly calculated intersection point, p_2, moves to p_2' on Figure 8(c) and no additional tag is required for this modification. Correct position of p_2 is

obtained by intersecting the fictitious line connecting the embedded corner to the previous/next corner inside list of corners and the edge on which p_2 is located. Modifications are then considered within adjacent elements crossed by the interface until an element in which no modification is required or already containing a corner is reached.

The second modification consists in modifying the *Interface-mesh* by splitting the edge inside the face c_{elm} into two segments and directly inserting corner C. This operation is needed only for corners classification of types $(a-f)$ and (i). All types of corners are related to their corresponding face c_{elm} using the corner-interface tag (TAG-IC in Figure 8).

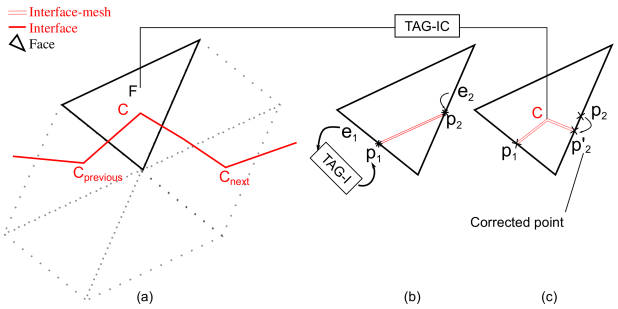
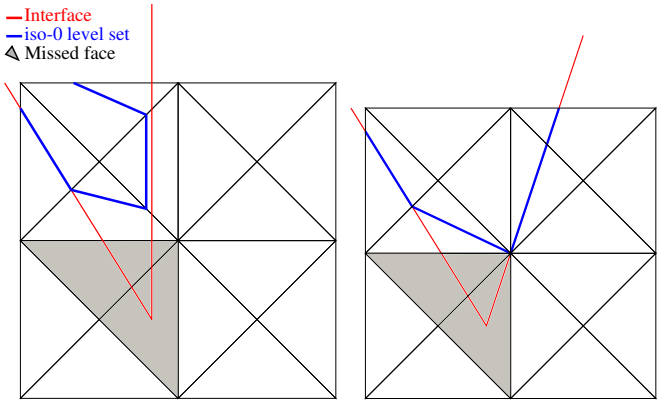


Figure 8: (a) Tagging corner and intersection points during creation/modification of *Interface-mesh* in face F containing a corner C, (b) classical level set, (c) level set plus

It is clear from Figure 5 that with the classical level set approach, no interface is created for the corner classifications (c, d, g, h, j). In this section, only cases (c) and (j) are discussed as cases (g) and (h) are considered as extensions. For case (d), the general treatment described above applies.

As illustrated on Figure 9 in configuration (c), all vertices have either positive or negative level set values which results in pertaining the element to the external or internal region respectively. In the configuration (j) only one of the vertices with zero level set value appears on the the interface. However this contribution is only related to the vertex itself. For these two particular cases, the *Interface-mesh* is modified not only by inserting the corner point, but also additional edges. The effect of mis-capturing the iso-0 level set (blue line in Figure 9) for these cases can be more severe than for a general case, as no contribution to the *Interface-mesh* is captured at all within the gray area of Figure 9.

Once again, the additional edges which are tagged to the corresponding gray elements can be used later in the *Submesh* modification. This tag for level set plus is different from the one used with classical level set as it is not detectable during creation of the *Submesh* and hence, it has no negative effect on creation of the *Submesh* by classical level set method.



(a) Case (c) (b) Case (j)

Figure 9: iso-0 level set (blue line) representing the interface (red line) using classical level set approach for cases (c) and (j)

The difference between cases (c) and (j) arises during the extension of the *Interface-mesh* from the element containing the corner point to the neighboring ones. For the case (j), since each leg of the corner crosses different neighboring elements, the correction of the *Interface-mesh* is achieved separately for each of them, similarly to the general case. However, for the configuration (c), since both legs of the corner cross only one edge and enter to the same neighbor element, constructing/modifying *Interface-mesh* is simultaneous for both legs.

3.2.2. Submesh modification

This section describes how the *Submesh* is modified to take corners into account. The modification is applied to the elements containing corners and, whenever needed, to their neighbors as well. Depending on the corner classification c_{type}, different solutions are proposed for the *Submesh* modification. The more general approach is described first and then some particular treatments for different corners classifications are examined.

The algorithm loops over all the corners. For a given corner and its c_{elm} attribute, the first step consists in deleting the associated sub-elements computed with the classical level set by breaking the partition-tag (TAG-P, Figure 2).

Figure 10 illustrates the algorithm for a face c_{elm} containing a corner C. As depicted in Figure 10(b), the corner C is retrieved by the corner-interface tag (TAG-IC). Similarly to the classical technique of adding a vertex to the *Submesh*, corner C is duplicated in the *Submesh* using the corner-submesh tag (TAG-SC).

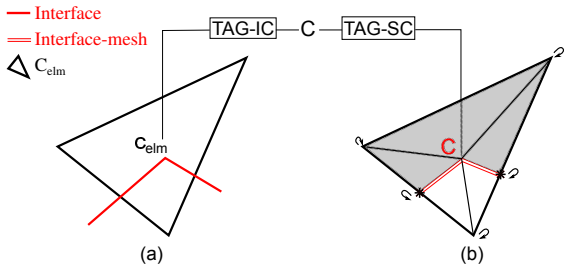


Figure 10: Tagging corner during Submesh modification : (a) *Interface-mesh* and (b) *Submesh*

Sub-elements associated to elements containing a corner according to their classification are illustrated in Figure 11. The approach based on the corner classification ensures that no degenerated triangle is produced. Vertices duplication from the *Interface-mesh* is sufficient for the creation of the *Submesh* face entities inside an element. In the last step of the *Submesh* modification, all generated sub-elements are tagged to the internal or external region of the level set support similarly to the classical level set approach.

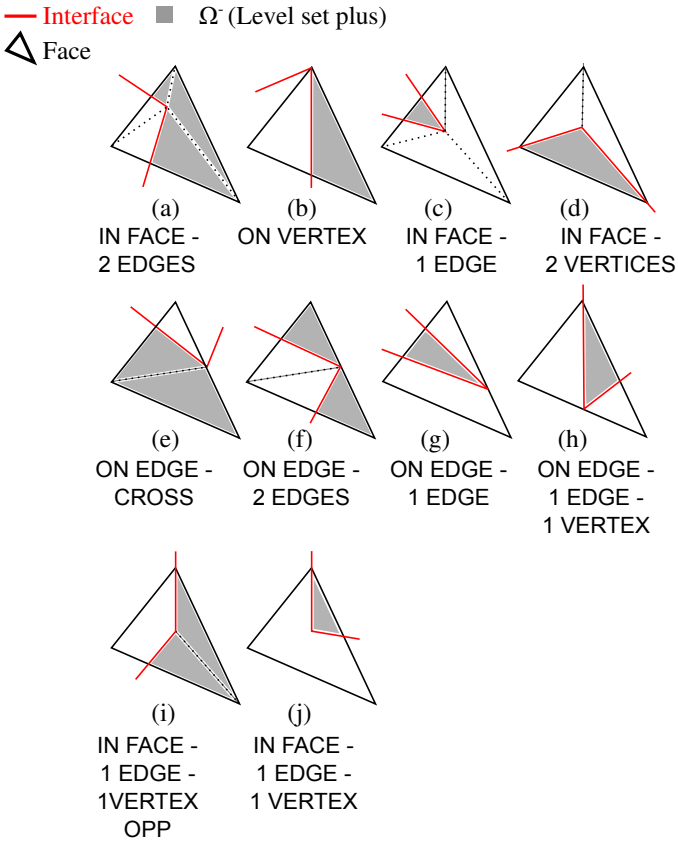
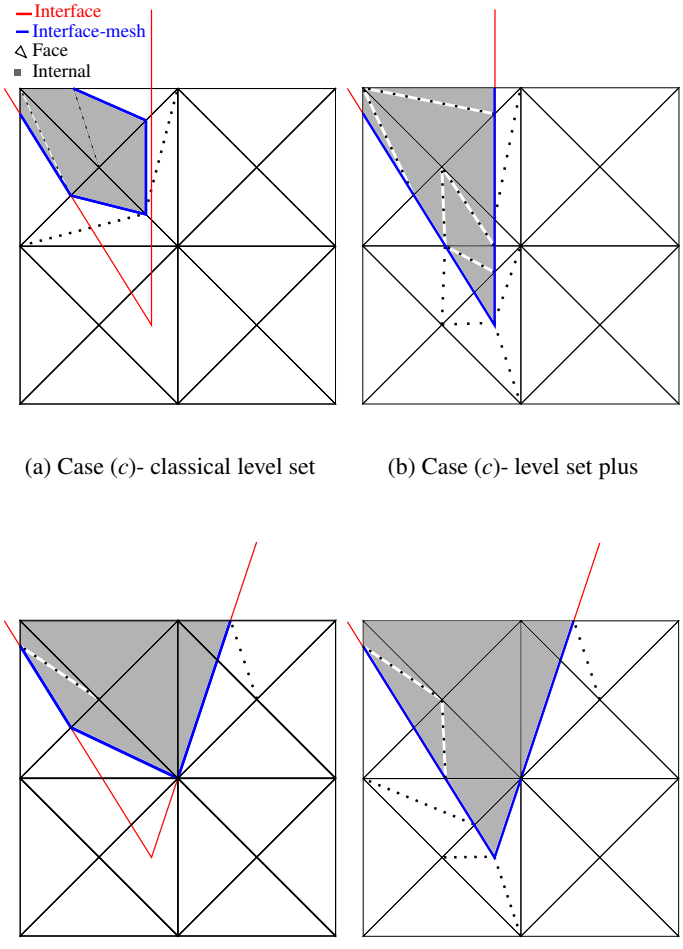


Figure 11: *Submesh* modification according to corner classification

Corner classification (*e*) is exempted of the *Submesh* modification as this corner already lies on an edge, only position of the intersection on the edge needs to be modified. No additional interface-corner tag is defined for this particular case.

As already mentioned in section 3.2.1, for corner classifications (c, g, h, j), the effect of the corner is not limited to the element containing the corner but also to neighboring elements (see Figure 12). Since additional edges during modification of the *Interface-mesh* are tagged to the elements using corner-interface tag TAG-IC, they cannot be recalled during the creation of *Submesh*. Hence, additional *Submesh* corrections are needed for these types. Figure 12 shows subdivision of the elements around corner type (*c*) and (*j*) using the classical level set approach and the level set plus technique.



(a) Case (*c*)- classical level set

(b) Case (*c*)- level set plus

(c) Case (*j*)- classical level set

(d) Case (*j*)- level set plus

Figure 12: Creation of *Submesh* for cases (*c*) and (*j*) by classical level set and level set plus

4. Combining level sets

One of the advantages of the implicit representation of interfaces with level sets is the simplicity of performing Boolean operations. Boolean operators prove to be very useful in different fields such as topography simulations (22), machining processes (23) and simulations involving multi-materials parts (24).

4.1. Boolean operations for classical level set

By representing interfaces implicitly with level sets, Boolean operations on the interfaces can be described as operations on

the corresponding level set functions (25). Considering the interfaces Γ_a and Γ_b represented by the level sets functions Φ_a and Φ_b, Boolean operations can be expressed as:

$$Union : \Gamma_c = \Gamma_a \bigcup \Gamma_b \Longleftrightarrow \Phi_c = min(\Phi_a, \Phi_b) \quad (3)$$

$$Intersection : \Gamma_c = \Gamma_a \bigcap \Gamma_b \Longleftrightarrow \Phi_c = max(\Phi_a, \Phi_b) \quad (4)$$

$$Complement : \Gamma_c = \Omega \backslash \Gamma_a \Longleftrightarrow \Phi_c = -\Phi_a \quad (5)$$

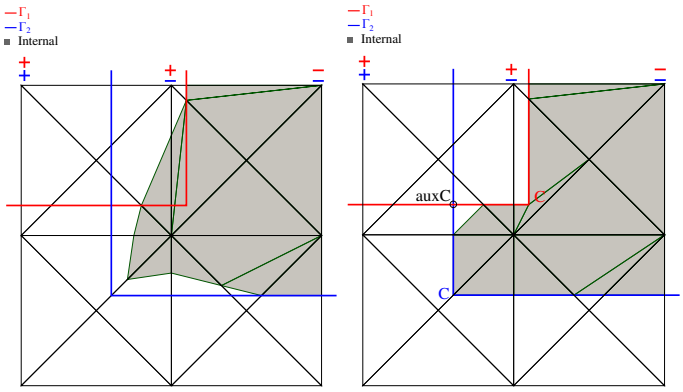
$$Difference : \Gamma_c = \Gamma_a \backslash \Gamma_b \Longleftrightarrow \Phi_c = max(\Phi_a, -\Phi_b) \quad (6)$$

Based on the Equations 3-6, in the classical level set approach, Boolean operations are performed directly on the level sets values.

4.2. Boolean operations for level set plus

Although the classical level set approach allows treatment of merging interfaces as mentioned above, not only the corners of the interfaces are missed but also those generated during intersection of interfaces, as illustrated at the intersection of blue and red lines on Figure 13(a).

The overlapping area of internal regions of two level set functions can be obtained by applying the intersection operator. As illustrated on Figure 13(b), it can be perceived that applying a Boolean operator only on level sets values of the level sets plus is not enough to correctly capture the interface. Although explicit corners on each level set are detected and added to the *interface-mesh*, auxiliary corners (denoted *auxC* in Figure 13(b)) may arise during Boolean operations and are stored in an ordered merged list of corners.



(a) Classical level set

(b) Level set plus - basic Boolean operation only

Figure 13: Intersection operation - Gray area in the remaining part

Auxiliary corners also need to be classified similarly as it was done for a single level set plus (see section 3.1.2). Possible configurations for intersecting two interfaces according to their position within the face are illustrated on Figure 14.

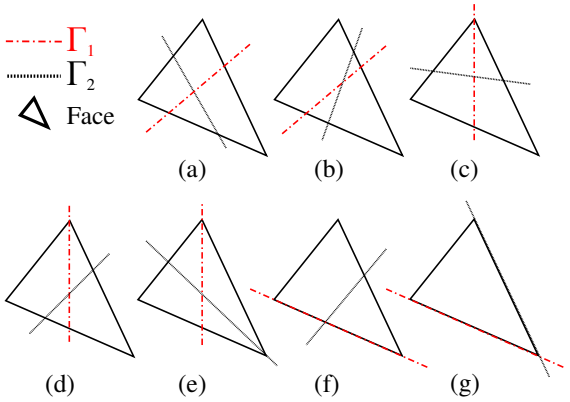


Figure 14: Possible configurations of intersecting two interfaces

Above configurations are discussed as follows:

- case (a): Two interfaces are crossing all edges and no vertex.
- case (b): Two interfaces are crossing two edges.
- case (c): Two interfaces are crossing all edges and one vertex
- case (d): Two interfaces are crossing two edges and one vertex
- case (e): Two interfaces are crossing two edges and two vertices
- case (f): One interface crosses two edges and the other lies on one of the crossed edge.
- case (g): Two interfaces lie on two edges.

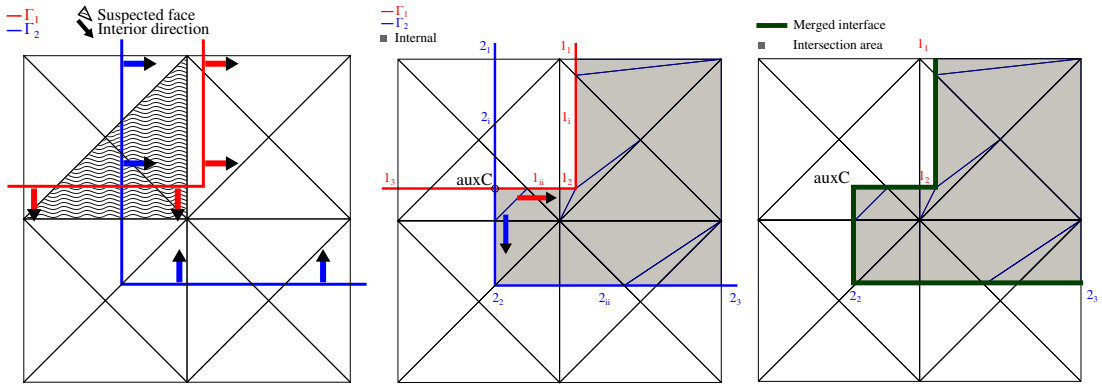
Among these configurations, cases (a, b, c, d, e), the most common cases, are taken into account in the context of this work. The conditions in Algorithm 1 are chosen so that the element is detected as a suspected face regardless of the internal or external regions of interfaces.

The procedure of updating the corners list during application of the Boolean operation is done in two steps. First, one detects suspected faces in which there are potential auxiliary corners. The second step adds auxiliary corners and removes ineffective corners associated to each level set plus.

Detection of suspected faces is presented in Algorithm 1 and illustrated on Figure 15(a).

Since there is a possibility of having two interfaces with no intersection but sign conversions, the sign conversions positive to negative *P2N* and negative to positive *N2P* are separated, in Algorithm 1, in order to avoid detection of faces between two interfaces as suspected faces.

The second step consists in updating the list of merged corners associated to the level set plus by inserting auxiliary corners and removing ineffective ones. This step contains several stages which are presented in Algorithm 2 and illustrated in Figure 15(b). Forasmuch as the correction of intersections inside an element is done by intersecting the edge and the line



(a) Detection of suspected face (Alg.1)

(b) Update list of corner (Alg.2)

(c) Final merged interface (Alg.2) and intersection area

Figure 15: Intersection operation

Algorithm 1: Detect suspected faces for cases $(a, b, c, d$ and $e)$

Input: LS_1^+, LS_2^+
Output: Suspected faces (F)
foreach *face* f *in* Ω **do**
-Compute Nr. of vertices with positive sign for both LS_1^+ and LS_2^+ $(P2P)$;
-Compute Nr. of vertices with negative sign for both LS_1^+ and LS_2^+ $(N2N)$;
-Compute Nr. of vertices with sign conversion from negative for LS_1^+ to positive for LS_2^+ $(N2P)$;
-Compute Nr. of vertices with sign conversion from positive for LS_1^+ to negative for LS_2^+ $(P2N)$;
-Compute Nr. of vertices with sign conversion from/to zero for either LS_1^+ or LS_2^+ $(0PN)$;
if

- $N2N + P2P = 3$ *with* $(N2N\&P2P > 0) \Rightarrow$ *(case a & b) or*
- $P2N + N2P = 3$ *with* $(P2N\&N2P > 0) \Rightarrow$ *(case a & b) or*
- $N2N + P2P + (P2N$ *or* $N2P) = 3$ *with* $(N2N\&P2P\&(P2N + N2P) > 0) \Rightarrow$ *(case a & b) or*
- $P2N + N2P + (P2P$ *or* $N2N) = 3$ *with* $(P2N\&N2P\&(P2P + N2N) > 0) \Rightarrow$ *(case a & b) or*
- $(P2P$ *or* $N2N) + (P2N$ *or* $N2P) + 0PN = 3$ *with* $(P2P$ *or* $N2N)\&(P2N$ *or* $N2P)\&0PN) > 0 \Rightarrow$ *(case c) or*
- $P2P + N2N + 0PN = 3$ *with* $(P2P\&N2N\&0PN > 0) \Rightarrow$ *(case d) or*
- $P2N + N2P + 0PN = 3$ *with* $(P2N\&N2P\&0PN > 0) \Rightarrow$ *(case d) or*
- $0PN = 2 \Rightarrow$ *(case e)*

then
Insert f in F
end
end
return F

Algorithm 2: Update list of corners

Input: LS_1^+ ans LS_2^+ lists of corners
Output: Updated merged list of corner
-Create map between LS_1^+ and LS_2^+ intersection points and interfaces line-segments numbers (CornerMap);
-Split LS_1^+ and LS_2^+ interfaces into line-segments (Fig. 15(b)): $[n_i, n_{ii}, ...]$ where n is the interface number;
foreach *suspected face* **do**
  Find the first segment of each interface that intersects with the suspected face;
  if *line-segments intersect* $(2_i, 1_{ii})$ **then**
    Insert the intersection point (auxiliary corner, auxC) into the CornerMap;
  end
end
foreach *intersection point (key of CornerMap)* **do**
  Find correct directions for LS_1^+ and LS_2^+ based on the level set values of the corresponding element (Fig. 15(b));
end
-Insert auxiliary corner into line-segments;
-Order each line-segment (e.g. multiple intersections on a same line-segment);
-Convert back line-segments into a list (Fig. 15(b)): $\begin{cases} for\ LS_1^+ : [1_1, 1_2, auxC, 1_3] \\ for\ LS_2^+ : [2_1, auxC, 2_2, 2_3] \end{cases}$;
-Classify corners inside lists of corners;
-Update merged list of corners based on the above lists and the directions of the auxiliary corner. The merged list of corners: $[1_1, 1_2, auxC, 2_2, 2_3]$ (Figure 15(c)) ;

connecting two corners (see section 3.2.1), finding the correct placement inside merged list of corners for auxiliary corners is necessary.

For the case where an auxiliary corner is located in an element already containing an usual corner, a smoothed solution is formed by taking into account only the usual corner, in accordance with the non handled corner classification case (see Figure 6).

Considering the possibility of forming discontinuous interface as a result of the Boolean operation as shown on Figure 16, the merged list of corners stores corners in a matrix having columns with different lengths. Each column of the matrix represents one merged interface.

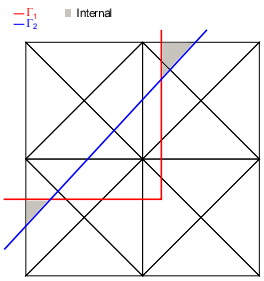
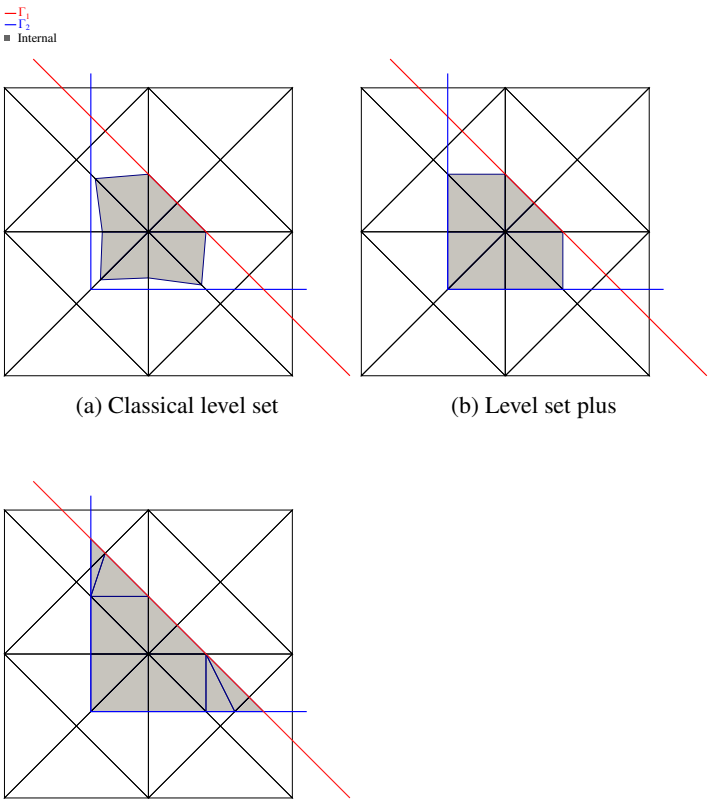


Figure 16: Forming discontinuous interface

There is also a possibility of forming a closed merged interface as a result of the Boolean operation. A new attribute is added to each merged interface representing the mode of the interface (open, closed). This attribute also helps to avoid failure in finding the correct position of the interface intersections based on the next/previous corner point. Figure 17 depicts the closed attribute for different methods of capturing initial interfaces.



(a) Classical level set (b) Level set plus

(c) level set plus with auxiliary corners

Figure 17: Closed merged interface mode

When considering several Boolean operations, two strate-

gies can be adopted. For the first one the result of each operation is preserved, while for the second strategy the outcome of an operation overwrites the result of the previous one so that a single interface representation is maintained.

With the first strategy, the level set corresponding to a combination of $(n-1)$ Boolean operations can be expressed as

$$LS_n^{+*} = ((LS_1^+, LS_2^+), ..., LS_n^+) \tag{7}$$

where $(.,.)$ represents one Boolean operation.

With the second strategy, the level set after operation $(n-1)$ reads

$$LS_n^{+*} = (LS_{n-1}^{+\ *}, LS_n^+)\,. \tag{8}$$

The second strategy can generally be solved by an iterative algorithm which is running in linear time and requires constant storage. The first strategy, in contrast, may require exponential time and storage. Therefore, the second strategy is often more efficient than the first one. Moreover, for most of the practical examples, keeping the history of the Boolean operations is not needed. The second strategy is adopted for the numerical examples in section 5.

4.3. Corners storage

Storing all the data related to the corners is not necessary for a case involving a single smooth interface. However, all data are stored inside the corner data structure itself, this information is updated after each Boolean operation. For example, the c_{type} attribute of a corner may vary after each operation. Therefore, it is necessary to create links connecting corners and modified intersections (i.e. point p_2' on Figure 8) to the corresponding entities. These links are preserved and tracked during each update of the level set. Adding this feature in accordance with the Boolean operation, it is certain that the interface is correctly captured for all steps with no need of extra recalculations of modified intersections. Storing corners is sufficient as long as the level set support is unchanged.

5. Numerical validations

The proposed approach has been validated on various numerical examples. The strategy is first applied in the context of a pure geometric validation. The robustness of the level set plus technique is then studied when a large number of Boolean operations is involved. Finite element simulation of an L-shaped geometry is discussed for the purpose of error analysis. Finally, a comprehensive finite element simulation is performed by taking advantage of both the geometrical accuracy and Boolean operation capability of the proposed method.

5.1. Geometrical validation - Compass

To clarify the accuracy of the level plus method, it is useful to compare it on a rather complex geometry. A compass with 8 branches and 16 corners is illustrated in Figure 18. The support of the level set is a square domain of side $L = 6.5$ meshed with 338 structured triangles of size 0.5.

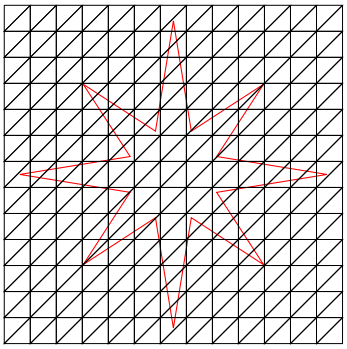
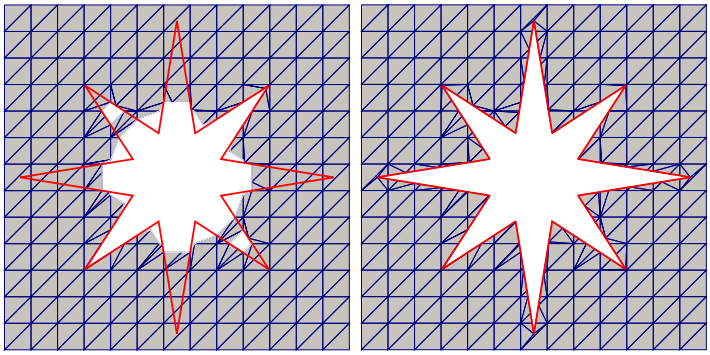
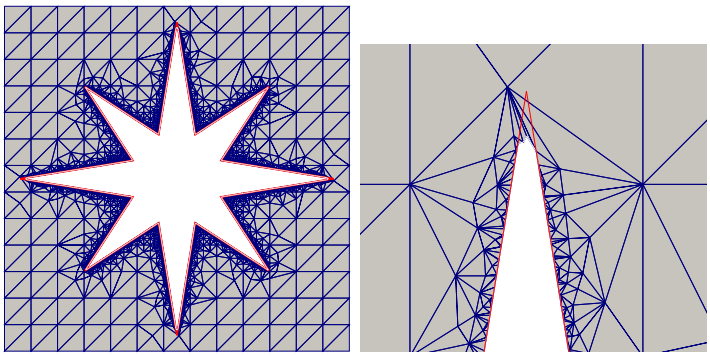


Figure 18: A 16 corners compass

On Figure 19(a), the implicit representation of the compass is obtained with the classical level set technique. The implicit representation is very bad. In addition disconnected domains or voids are observed. This may lead to a convergence problem in the context of finite element simulations. Exact representation of the compass using level set plus technique is illustrated in Figure 19(b). As expected, the semi-implicit representation is conforming with the initial geometry. As shown in Figure 19(c), if the level set support is refined in the vicinity of the iso-0 level set, a better representation of the interface is captured. However, as discussed in section 2.3, using mesh refinement technique in the region of the iso-0 level set, Figure 19(d), cannot ensure an exact representation of the interface.



(a) Classical level set (b) Level set plus



(c) Classical level set with mesh refinement (d) Classical level set with mesh refinement - Zoom around one corner

Figure 19: Implicit representation of the compass with different methods

Figure 20 illustrates the relative geometrical error obtained with the classical level set when the refinement increases while it is zero for the level set plus technique independently of the mesh size provided that all corners are classified as one the types mentioned in section 3.1.2. The relative geometrical error measures the area which is captured by classical level set with respect to the exact area.

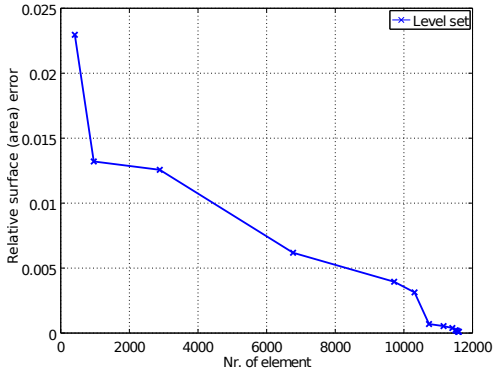
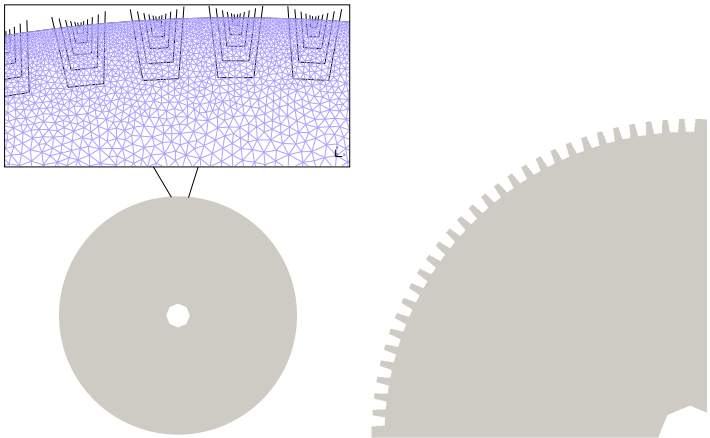


Figure 20: Relative surface (area) error obtained by classical level set as function of number of elements in the support

5.2. High number of Boolean operations - Gear

The performance of the level set plus algorithm is tested on an example containing a high number of Boolean operation steps (719 differences obtained by Equation 6). Cutting steps of a 120 tooth gear, each tooth with 6-stages cutting process, is illustrated in Figure 21. As shown in Figure 21(a), level set support is non-uniform with high refinement in the vicinity of the disk circumference in order to avoid cutting same element by multiple interfaces. Although interior layers of interfaces have no effect on the final geometry of the gear (each of them lies inside internal region of the next step), this example insists on showing robustness and performance of the level set plus technique. Figures 21(b, c) show the sequence of the Boolean operations in different stages. According to the algorithm 2 presented in section 4.3, the list of corners is updated at each Boolean operation in order keep it as minimal as possible. The evolution of the corners list size is reported in Figure 22 and compared to the total number of corners. This strategy clearly highlights the advantage of filtering the corner list. The difference between the two line charts indicates the number of corners which are filtered and increases over the whole cutting process.



(a) Initial configuration and different cutting steps

(b) Quarter cut, Level set plus

(c) Final cut, Level set plus

Figure 21: 120 tooth gear with 720 cutting process

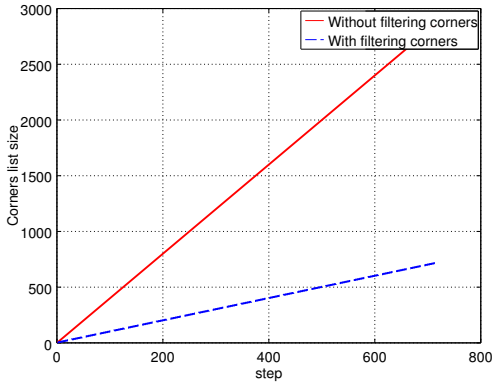


Figure 22: Corners list evolution during cutting progress with/without filtering corners

5.3. Error estimation - L-shape domain under mode I load

The L-shape domain is a classical simulation problem for which an analytical solution is available in linear elasticity. It was firstly formulated by M. Williams (26) and later by B. Szabo and I. Babuška (27). The dimensions and prescribed boundary conditions for this problem are illustrated in Figure 23.

The black dashed lines of the boundary are loaded with traction computed from the analytical solution given in Equation 9

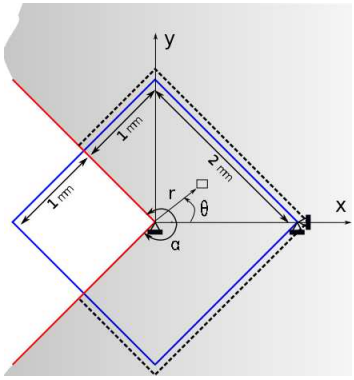


Figure 23: L-shape model dimensions and boundary conditions

while the remaining boundaries are stress free.

$$\boldsymbol{\sigma}(r,\theta) = \underbrace{K_I \lambda_I r^{\lambda_I - 1} \boldsymbol{\Phi}_{\mathbf{I}}(\lambda_I, \theta)}_{pure \quad Mode \quad I} + \underbrace{K_{II} \lambda_{II} r^{\lambda_{II} - 1} \boldsymbol{\Phi}_{\mathbf{II}}(\lambda_{II}, \theta)}_{pure \quad Mode \quad II} \tag{9}$$

where r and θ are radial distance and angular position. This example only studies L-shape model under pure mode I load. K_I represents the generalized stress intensity factor which linearly determines the intensity of the stress field in the vicinity of sharp corner. λ_I is an eigenvalue which determines the magnitude of the singularity and is only related to the corner angle alpha and can be computed as the smallest positive root of following equation:

$$sin(\lambda_I \alpha) + \lambda_I sin(\alpha) = 0, \tag{10}$$

$$\alpha = 3\pi/2 \;\Rightarrow \lambda = 0.544484$$

Tensor Φ_I is a set of trigonometric functions which is given by Equation 11:

$$\boldsymbol{\Phi}_{\mathbf{I}}(\lambda_I, \theta) = \begin{cases} (2 - Q(\lambda_I + 1))cos(\lambda_I - 1)\theta - (\lambda_I - 1)cos(\lambda_I - 3)\theta \\ (2 + Q(\lambda_I + 1))cos(\lambda_I - 1)\theta + (\lambda_I - 1)cos(\lambda_I - 3)\theta \\ Q(\lambda_I + 1)sin(\lambda_I - 1)\theta + (\lambda_I - 1)sin(\lambda_I - 3)\theta \end{cases} \tag{11}$$

where $Q = \frac{cos\left((\lambda_I - 1)\frac{\alpha}{2}\right)}{cos\left((\lambda_I + 1)\frac{\alpha}{2}\right)}$ is a constant for given angle α. The analysis is carried out assuming plane strain conditions. The material parameters are Young's modulus $E = 1000\,MPa$, and Poisson's ratio $\nu = 0.3$. The interested reader can find more details in (28). Using Equation 9, the relative error in energy norm $\overline{E}_{en}$, computed from Equation 12, for classical level set and level set plus is shown in Figure 24.

$$\overline{E}_{en} = \frac{\left(\frac{1}{2}\int(\sigma - \sigma^{exact}) : (\varepsilon - \varepsilon^{exact}) d\Omega\right)^{1/2}}{\left(\frac{1}{2}\int \sigma^{exact} : \varepsilon^{exact} d\Omega\right)^{1/2}} \tag{12}$$

It shows that level set plus always gives lower numerical error compared to classical level set. This guarantees the quality of the implementation of both *Interface-mesh* and *Submesh*. Oscillations come from mesh quality in vicinity of the corner. As a reference, conforming finite element simulation is performed. The difference between level set plus and FEM results on Figure 24 is due to the difference on numerical integration. Integration scheme is briefly discussed in the Appendix A.

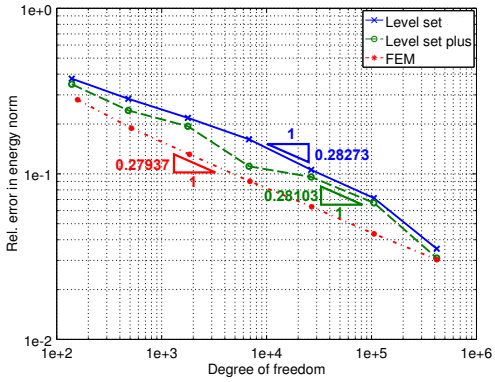


Figure 24: Relative error in energy norm

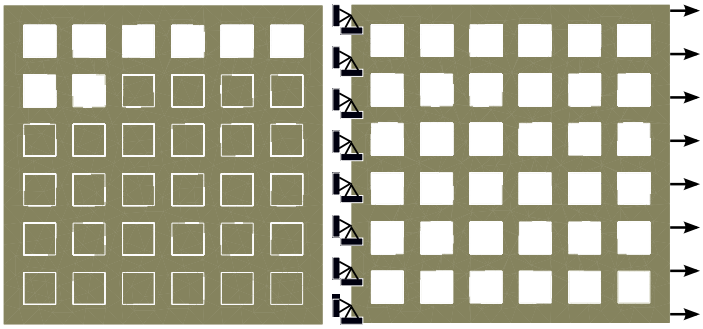
According to the B. Szabo *et al.* (29), asymptotic rate of convergence of L-shape model for h-refinement equals to:

$$\beta = \frac{1}{2} min(p, \lambda) = 0.272242 \tag{13}$$

where p is the polynomial degree.

5.4. Netted plate

This problem involves a square plate with a side of $2\,m$, featuring 36 squares cutout with a side of $0.1\,m$ each, that is subjected to pure tension. The square holes are sequentially extracted by use of level set plus technique as illustrated in Figure 25(a,b).



(a) 8^{th} cut, Level set plus

(b) Imposed boundary condition to the plate

Figure 25: Netted plate cut and boundary conditions

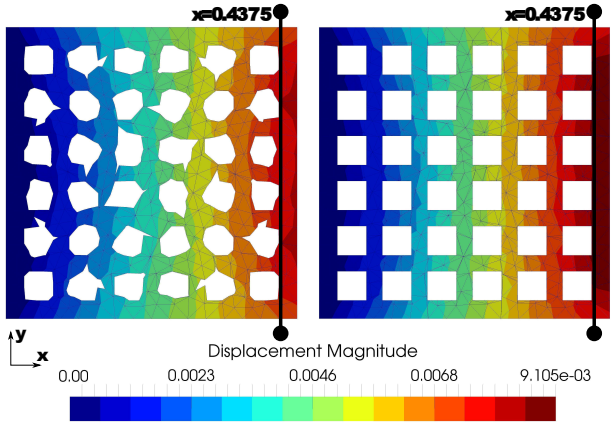


Figure 26: Magnitude of displacement field for level set (left) and level set plus (right)

The plate is clamped along the left edge and has a uniform load of 1000 MPa imposed on its right side (see Figure 25(b)). Plane strain conditions and $E = 200\,GPa, \nu = 0.3$ (under linear elasticity condition) are assumed for this example. The number of degrees of freedom for this example is $2.8K$. Distribution of the magnitude of the displacement field is illustrated for both level set and level set plus techniques on Figure 26. As for the example of section 5.1, classical level set results in poor representation of the geometry. Since there is no exact solution available for this problem, a reference finite element solution on a fine conforming mesh with about $500K$ degrees of freedom is considered. Relative displacement in x direction with respect to the reference solution is plotted over line $x = 0.4376$ as illustrated on Figure 27. Non-zero relative displacement for the *conformed* case is due to the dependency of the result accuracy on the mesh size. The difference between *conformed* result and level set plus is due to the adopted integration scheme (see Appendix A). Displacement fluctuates much more when corners are not accurately represented. As depicted in Figure 27, level set plus technique provides better result than classical level set.

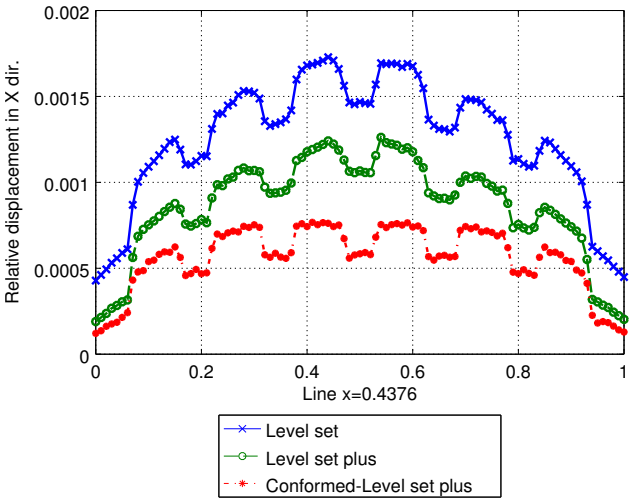


Figure 27: Relative displacement in x direction plotted over line $x = 0.4376$

6. Conclusions

The level set plus technique presented in this paper is able to represent semi-implicitly sharp features of a boundary. In addition to the implicit representation of the classical level set technique, topological information from the boundary is considered. In the context of a finite element simulation, the goal is to update *Interface-mesh* and *Submesh* in order to accurately impose boundary conditions and perform the numerical integration. This process is done in several steps. First of all, sharp features must be detected from the boundary. Those are then classified according to their position within the face of the level set support to which they belong. Thanks to this classification and additional information associated to the corners as the face or edge on which they lie, *Interface-mesh* and *Submesh* obtained with the classical level set technique are modified to be coherent with the initial boundary containing sharp features. All adopted algorithms are presented in details for the different configuration cases.

Combining level set plus with Boolean operations is another challenge considered in this work . Not only existing sharp features of interfaces must be considered, but also those appearing during the operations. Also, final corner list must be sorted and filtered to eliminate potential useless corners. All this is done without any modification or mesh refinement of the level set support.

All these developments are illustrated and validated on several examples such as compass shape (geometrical validation), gear (robustness in Boolean operations), L-shape model (convergence study), and Netted plate (general finite element simulation). For the different meshes, the error related to the level set plus always lies in between the one obtained with the classical level set (which contains geometrical errors as well) and the one from a conforming finite element simulation.

A first extension of this work is to generalize the algorithms (detection and classification of sharp features) for 3D geometries. In this 2D case, we were able to explicitly generate all different configurations by hand. For the 3D case, the number of configurations is much higher. Therefore, we plan to switch the classification part of the algorithm so that these explicit enumeration is not needed anymore. It is currently under development.

Another research topic is regarding implementation aspects, especially for three dimensional specimens. Parallelization of the proposed algorithms should be considered to be in line with what is done for classical implicit representation.

Finally, in addition to capturing sharp features, implicit representation of curvatures ((14; 13)) should be coupled with this approach.

Appendix A. Numerical integration

Numerical integration has to be treated by care on the elements cut by iso-0 level set. Therefore, these element split into integration cells (*Submesh*) as discussed in section 2.2. Then, regular integration rule is performed on *Submesh*. Figure A.1 shows distribution of stress differences between level set plus

and a conforming finite element simulation that is based exactly on the same *Submesh* that is used only for integration purposes. The difference here is that the sub-elements are used as the support for building the interpolation. Significant difference is observed on elements in vicinity of the corner. These differences arise due to different degrees of freedom. In the level set plus approach the connected elements to the red nodes in Figure A.1 have partial contribution during numerical integration. Of course, introducing new degrees of freedom to the level set plus approach will provide same solution.

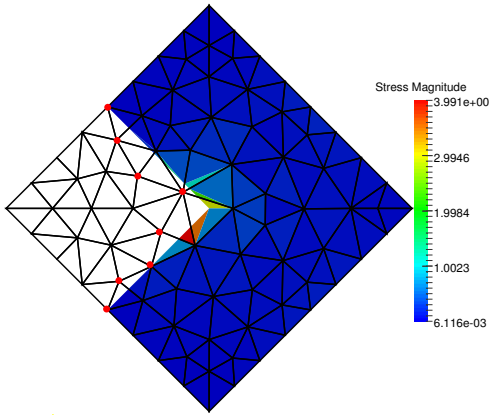


Figure A.1: Distribution of stress difference between level set plus and conforming finite element simulation

Acknowledgment

The authors highly appreciate the financial support provided within Framework Programme 7 Initial Training Network Funding under Grant No. 289361 "Integrating Numerical Simulation and Geometric Design Technology (INSIST)". The authors also thank Y. Jin, Cenaero research engineer, for assistance in the context of error analysis, and L. Arboui, Cenaero research engineer, for comments that greatly improved the numerical validations.

References

[1] S. Osher, J. A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations, Journal of computational physics 79 (1) (1988) 12–49.

[2] J. A. Sethian, Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science, Vol. 3, Cambridge university press, 1999.

[3] S. Osher, R. Fedkiw, Level set methods and dynamic implicit surfaces, Vol. 153, Springer Science & Business Media, 2003.

[4] S. Chen, B. Merriman, S. Osher, P. Smereka, A simple level set method for solving stefan problems, Journal of Computational Physics 135 (1) (1997) 8–29.

[5] S. Osher, R. P. Fedkiw, Level set methods: an overview and some recent results, Journal of Computational physics 169 (2) (2001) 463–502.

[6] M. Duflot, A study of the representation of cracks with level sets, International Journal for Numerical Methods in Engineering 70 (11) (2007) 1261–1302.

[7] L. A. Vese, T. F. Chan, A multiphase level set framework for image segmentation using the mumford and shah model, International journal of computer vision 50 (3) (2002) 271–293.

[8] M. Sussman, E. Fatemi, P. Smereka, S. Osher, An improved level set method for incompressible two-phase flows, Computers & Fluids 27 (5) (1998) 663–680.

[9] E. Olsson, G. Kreiss, A conservative level set method for two phase flow, Journal of computational physics 210 (1) (2005) 225–246.

[10] J. A. Sethian, A. Wiegmann, Structural boundary design via level set and immersed interface methods, Journal of computational physics 163 (2) (2000) 489–528.

[11] M. Y. Wang, X. Wang, D. Guo, A level set method for structural topology optimization, Computer methods in applied mechanics and engineering 192 (1) (2003) 227–246.

[12] G. Allaire, F. Jouve, A.-M. Toader, Structural optimization using sensitivity analysis and a level-set method, Journal of computational physics 194 (1) (2004) 363–393.

[13] K. W. Cheng, T.-P. Fries, Higher-order xfem for curved strong and weak discontinuities, International Journal for Numerical Methods in Engineering 82 (5) (2010) 564–590.

[14] T.-P. Fries, S. Omerović, Higher-order accurate integration of implicit geometries, International Journal for Numerical Methods in Engineering.

[15] M. Kästner, S. Müller, J. Goldmann, C. Spieler, J. Brummund, V. Ulbricht, Higher-order extended fem for weak discontinuities–level set representation, quadrature and application to magneto-mechanical problems, International Journal for Numerical Methods in Engineering 93 (13) (2013) 1403–1424.

[16] G. Legrain, N. Chevaugeon, K. Dréau, High order x-fem and levelsets for complex microstructures: uncoupling geometry and approximation, Computer Methods in Applied Mechanics and Engineering 241 (2012) 172–189.

[17] G. Legrain, C. Geuzaine, J.-F. Remacle, N. Moës, P. Cresta, J. Gaudin, Numerical simulation of cad thin structures using the extended finite element method and level sets, Finite Elements in Analysis and Design 77 (2013) 40–58.

[18] M. Moumnassi, S. Belouettar, É. Béchet, S. P. Bordas, D. Quoirin, M. Potier-Ferry, Finite element analysis on implicitly defined domains: An accurate representation based on arbitrary parametric surfaces, Computer Methods in Applied Mechanics and Engineering 200 (5) (2011) 774–796.

[19] A. Tran, J. Yvonnet, Q.-C. He, C. Toulemonde, J. Sanahuja, A multiple level set approach to prevent numerical artefacts in complex microstructures with nearby inclusions within xfem, International Journal for Numerical Methods in Engineering 85 (11) (2011) 1436–1459.

[20] H. Brönnimann, G. Melquiond, S. Pion, The design of the boost interval arithmetic library, Theoretical Computer Science 351 (1) (2006) 111–118.

[21] W. R. Franklin, Pnpoly-point inclusion in polygon test, Web site: http://www. ecse. rpi. edu/Homepages/wrf/Research/Short_Notes/pnpoly. html.

[22] O. Ertl, S. Selberherr, A fast level set framework for large three-dimensional topography simulations, Computer Physics Communications 180 (8) (2009) 1242–1250.

[23] O. Pierard, J. Barboza, M. Duflot, L. D'Alvise, A. Perez-Duarte, Distortions prediction during multi-pass machining simulations by using the level-set method, International journal of material forming 1 (1) (2008) 563–565.

[24] J. Liu, K. Duke, Y. Ma, Computer-aided design–computer-aided engineering associative feature-based heterogeneous object modeling, Advances in Mechanical Engineering 7 (12) (2015) 1687814015619767.

[25] K. Museth, D. E. Breen, R. T. Whitaker, A. H. Barr, Level set surface editing operators, in: ACM Transactions on Graphics (TOG), Vol. 21, ACM, 2002, pp. 330–338.

[26] M. Williams, Stress singularities resulting from various boundary conditions, Journal of Applied Mechanics 19 (4) (1952) 526–528.

[27] B. A. Szabo, I. Babuška, Finite element analysis, John Wiley & Sons, 1991.

[28] O. A. González-Estrada, S. Natarajan, J. J. Ródenas, H. Nguyen-Xuan, S. P. Bordas, Efficient recovery-based error estimation for the smoothed finite element method for smooth and singular linear elasticity, Computational Mechanics 52 (1) (2013) 37–52.

[29] B. Szabó, I. Babuška, Introduction to Finite Element Analysis: Formulation, Verification and Validation, Vol. 35, John Wiley & Sons, 2011.