



**HAL**  
open science

## On use of the thick level set method in 3D quasi-static crack simulation of quasi-brittle material

Alexis Salzman, Nicolas Chevaugeon, Nicolas Moes

► **To cite this version:**

Alexis Salzman, Nicolas Chevaugeon, Nicolas Moes. On use of the thick level set method in 3D quasi-static crack simulation of quasi-brittle material. *International Journal of Fracture*, 2016, 202 (1), pp.21 - 49. 10.1007/s10704-016-0132-8 . hal-01692311

**HAL Id: hal-01692311**

**<https://hal.science/hal-01692311>**

Submitted on 25 Jan 2018

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# On use of the thick level set method in 3D quasi-static crack simulation of quasi-brittle material

Alexis Salzman · Nicolas Moës · Nicolas Chevaugeon

**Abstract** This work demonstrates the 3D capability of the thick level set (TLS) method, first introduced by Moës et al. (Int J Numer Methods Eng 86:358–380, 2011. doi:10.1002/nme.3069) and later in Stolz and Moës (Int J Fract 174(1):49–60, 2012. doi:10.1007/s10704-012-9693-3). The thick level set approach is a non-local damage method embedding fracture mechanics discontinuity. Enhanced numerical implementation for elastic quasi-brittle materials in 2D under quasi-static loading conditions was presented in Bernard et al. (Comput Methods Appl Mech Eng 233–236:11–27, 2012. doi:10.1016/j.cma.2012.02.020). The present work focuses on using this enhanced numerical implementation in a 3D context. This work adds a new way to construct the crack faces by use of a “double cut algorithm”. The regularization computation, part of the non-local feature of the model, is also reviewed to improve its accuracy. As 3D models are computationally intensive, CPU aspects are discussed. Five test cases are presented. The first one illustrates the capability of the method to deal with crack coalescence, which is quite unique for this kind of simulation. Three other cases point out a comparison with literature examples (numerical and experimental) and good agreement is observed. One is a more complex example, which deals with an engineering oriented application. This work confirms good performance of

the thick level set method in 3D context. The use of the new “double cut” algorithm is giving well discretized crack path and allows for discontinuous displacement.

**Keywords** TLS · Signed vector distance function · Double cut algorithm · Crack coalescence · Crack initiation · 3D · Parallel · Damage mechanics · X-FEM

## 1 Introduction

The thick level set (TLS) method was first introduced by Moës et al. (2011) and further studied in Stolz and Moës (2012). The work of Bernard et al. (2012) is the basis for the implementation used here. More recently Moës et al. (2014) have generalized the TLS method to mix local and non-local damage procedure in the same framework. The present work focuses on 3D simulation usage.

The ingredients in the TLS method regarding the material are twofold. The first is given by the local damage constitutive law and the second by the damage profile imposed in the localization zone. In this sense, there is not, per se, a fracture criteria in the model. This information results in the TLS of a combination between the local material strength and the necessity for localization to spread over some length with a given profile.

Compared to pure fracture mechanic propagation methods such as X-FEM or remeshing techniques, the TLS method offers the ability to create crack via dam-

---

A. Salzman (✉) · N. Moës · N. Chevaugeon  
UMR CNRS 6183, Ecole centrale de Nantes, GeM  
Institute, 1 rue de la Noë, 44321 Nantes, France  
e-mail: alexis.salzman@ec-nantes.fr

age initiation without ad hoc crack insertion. Crack coalescence and branching are also automatically taken into account by the TLS. This is not easy or even possible with pure fracture mechanic propagation technique. This opens simulation to a broader range of problems.

Now other non-local approaches exist to model quasi-brittle failure and avoid spurious localization. In this paper, the goal is not to discuss in detail all these methods. We, however, point out how the TLS relate or not to existing techniques. The TLS shares similarities with the so-called non-local integral approach of (Pijaudier-Cabot and Bažant 1987; Bažant and Jirasek 2002) in which weighted averages are performed over segments (1D), disks (2D), and spheres (3D) of fixed sizes. In the TLS, however, weighted averages are always performed on segments whatever the dimension of the body and over a length that is not fixed in time, but evolves from zero to a maximum length  $l_c$ . Also, the segment orientation is always aligned with the damage gradient. The TLS is a discontinuous approach and by no means turns the crack into a diffuse zone as in the phase-field approach (Karma et al. 2001; Miehe et al. 2010; Spatschek et al. 2011) or the variational approach to fracture (Francfort and Marigo 1998; Bourdin et al. 2008). Finally, readers interested by a comparison between phase-field and TLS may look at Cazes and Moës (2015).

Key TLS features (level set, averaging damage along segments, automatic introduction of crack discontinuity, ...) naturally extend to 3D. The only effort imposed by problem dimension is related to discretization, in particular for the crack and the computation cost. The latter is mainly treated by parallel computational technique. The former is treated with a new method called the “double cut algorithm,” which provides adapted finer 3D discretization for the TLS framework.

Five 3D examples are presented in this work to illustrate the TLS framework versatility (in particular, coalescence and initiation) and are compared with other numerical methods or tests.

In this paper, we consider a local stress-strain curve with abrupt softening for most cases. This is not a limitation of the TLS, and for quantitative comparison example we choose, like in Parrilla Gómez et al. (2015), to consider smoother softening regime.

In Sect. 2, the TLS method main theoretical and algorithmic aspects are presented. Then in Sect. 3, the proposed “double cut algorithm” is explained and its introduction in the TLS framework is discussed. In

Sect. 4, we will describe some improvement to the averaging techniques used to compute the energy release rate. Simulation time consumption aspects are presented in 5. Numerical tests are then exposed in Sect. 6. Finally, Sect. 7 concludes and gives some perspectives.

## 2 TLS method

### 2.1 Review

In the TLS framework, the domain  $\Omega$  of interest is decomposed into three zones (possibly empty): a zone  $\Omega_-$  in which damage  $d$  evolves in a local manner, a localizing zone  $\Omega_+$  in which the evolution is non-local, and, finally, a zone  $\Omega_c$ , in which the material is completely damaged ( $d = 1$ ). This is detailed in (Moës et al. 2014). In this paper, as in Bernard et al. (2012), we consider damage constitutive models without hardening, leading to damage localization as soon as damage starts. Thus, damage is always zero in  $\Omega_-$ .

The interface between  $\Omega_+$  and  $\Omega_-$  is denoted  $\Gamma_0$ , and it is located by the iso-zero level set of a signed distance function  $\phi$  (counted negative in  $\Omega_-$ ). The “distance function” characteristic is achieved by:

$$\|\nabla\phi(x)\| = 1 \quad (1)$$

Damage is then expressed in terms of  $\phi$  by a scalar function.

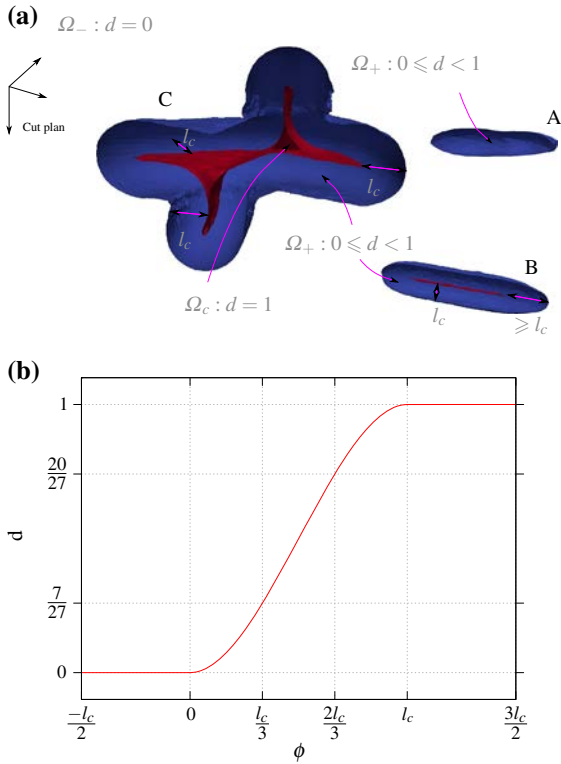
$$d = d(\phi(x)) \quad (2)$$

The function  $d(\phi)$  is material data that must satisfy a set of requirements, including that damage reaches 1 only beyond a distance  $l_c$ . The interface between  $\Omega_+$  and  $\Omega_c$  is thus the iso- $l_c$ <sup>1</sup> and is denoted  $\Gamma_c$ .

The use of a level set is a powerful aspect of the method, offering support for complicated  $\Gamma_0$ , with potential branching and coalescence, as will be seen in Sect. 6.1. Another powerful aspect is the introduction in the model of this characteristic length  $l_c$ , which achieves two goals:

- it introduces in an automatic manner a crack, considering iso- $l_c$  values of the level set, as the crack front. In this work a more precise discretization of  $\Gamma_c$  is proposed in Sect. 3 and its introduction in the TLS framework discussed in Sect. 3.5.

<sup>1</sup> The iso- $l_c$  is determined by a  $-l_c$  offset of  $\phi$  iso-zero.



**Fig. 1** Model parameters. **a** Domain and interface definitions: in blue  $\Gamma_0$ , in red  $\Gamma_c$ , **b** damage function  $d(\phi)$  used in four cases of this work:  $\left(\frac{\phi}{l_c}\right)^2 \left(3 - 2\frac{\phi}{l_c}\right)$  for  $0 \leq \phi \leq l_c$

- it introduces a maximum length over which an averaging of damage is performed.

Figure 1a shows the above defined ingredients. It represents different situations, using a planar cross section to see inside  $\Gamma_c$ , where:

- A is a damage initializing zone where all points are at a distance less than  $l_c$  from  $\Gamma_0$  (in blue), and for which  $d < 1$ .
- B is a simple crack represented by  $\Gamma_c$  (in red). In this case one crack tip is situated at a distance greater or equal to  $l_c$  from  $\Gamma_0$  tip. This illustrates a complex damage growth pattern in front of a crack tip.
- C illustrates a branching situation where the crack first develop “horizontally” and then “vertically” after a change in loading (in figure perspective).

The damage shape function  $d()$  must be

- increasing
- continuous
- bounded to  $[0, 1]$

- null outside the damage zone in  $\Omega_-$  (when  $\phi(x) \leq 0$ )
- equal to 1 in the fully damaged zone  $\Omega_c$  (when  $\phi(x) \geq l_c$ )

Specific choices of  $d()$  have permitted comparison with the phase-field method in Cazes and Moës (2015) or with the cohesive zone model in Parrilla Gómez et al. 2015. The chosen  $d()$  function taken for four cases in this paper is given in Fig. 1b.

## 2.2 Governing equations

As in previous work, we use an asymmetric constitutive law. This asymmetric behavior is mandatory to avoid damage growing in the compression direction. In this paper, however, we do not deal with contact on crack faces.

In  $\Omega$ , equilibrium and kinematics equations are:

$$\begin{cases} \nabla \cdot \sigma = 0 \text{ on } \Omega \\ \sigma \cdot \mathbf{n} = \mathbf{f} \text{ on } \partial\Omega^N \\ \varepsilon = \frac{1}{2} (\nabla \mathbf{u} + (\nabla \mathbf{u})^t) \text{ on } \Omega \\ \mathbf{u} = \bar{\mathbf{u}} \text{ on } \partial\Omega^D \end{cases} \quad (3)$$

where  $\sigma$  is the Cauchy stress tensor,  $\varepsilon$  the strain tensor,  $\mathbf{f}$  the external loading on part  $\partial\Omega^N$  of  $\Omega$  boundary,  $\mathbf{n}$  the outgoing normal vector of the domain,  $\bar{\mathbf{u}}$  the prescribed displacements on part  $\partial\Omega^D$  of  $\Omega$  boundary and  $\mathbf{u}$  the displacement field. Here small strains and displacements are assumed.

Stress and energy release rate  $Y$  are derived from the free energy as follows:

$$\begin{cases} \sigma = \frac{\partial \varphi}{\partial \varepsilon} \\ Y = -\frac{\partial \varphi}{\partial d} \end{cases} \quad (4)$$

with

$$\varphi(\varepsilon, d) = \frac{\lambda}{2} (1 - \alpha d) \text{tr}(\varepsilon)^2 + \mu \sum_{i=1}^3 (1 - \alpha_i d) \Lambda_i^2 \quad (5)$$

where:

- $\lambda$  and  $\mu$  are the Lamé elasticity coefficients
- $\Lambda_i$  are the eigenvalues of the strain tensor
- $\alpha_i$  and  $\alpha$  are coefficients introduced to take into account an asymmetric behavior in traction/compression:

- $\alpha_i = \beta$  if  $\Lambda_i < 0$
- $\alpha_i = 1$  if  $\Lambda_i \geq 0$
- $\alpha = \beta$  if  $tr(\varepsilon) < 0$
- $\alpha = 1$  if  $tr(\varepsilon) \geq 0$

-  $\beta$  is a parameter, bounded to  $[0, 1]$ , to drive the asymmetry.

When  $\beta$  is equal to 1,  $\alpha_i$  and  $\alpha$  terms are always equal to 1 and the free energy is then a linear elastic damage potential that can be rewritten using Hooke's tensor  $E$  as follows:

$$\varphi(\varepsilon, d) = \frac{1}{2}(1-d)\varepsilon : E : \varepsilon \quad (6)$$

When  $\beta < 1$  the potential becomes non-quadratic. Damage participation to free energy becomes negligible in compression when  $\beta$  tends to 0. When  $\beta$  is null, no damage growth is obtained in compression because in this case  $Y = 0$ . Intermediate  $\beta$  values are allowed to fit to material properties where irreversible degradation modify slightly the stiffness in compression.

For the damage evolution law, detailed in Bernard et al. (2012), the local damage growth model:

$$Y \leq Y_c, \dot{d} \geq 0, (Y - Y_c)\dot{d} = 0 \quad (7)$$

is regularized into its non-local counterpart:

$$\bar{Y} \leq \bar{Y}_c \quad (8a)$$

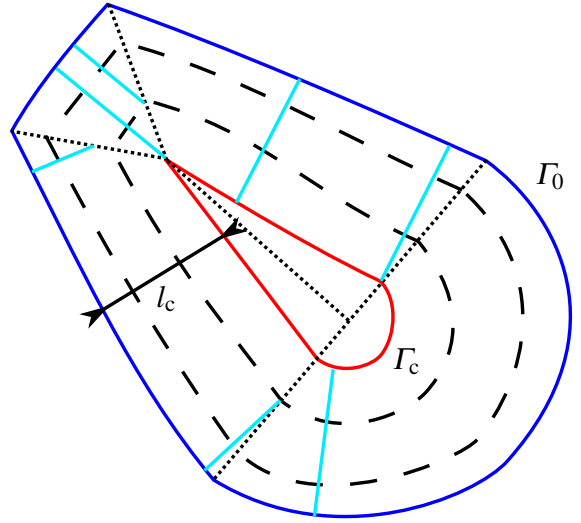
$$\dot{d} \geq 0 \quad (8b)$$

$$(\bar{Y} - \bar{Y}_c)\dot{d} = 0 \quad (8c)$$

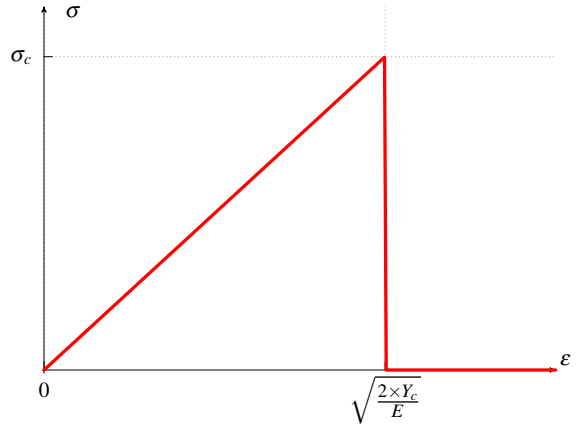
where  $\bar{Y}$  and  $\bar{Y}_c$  are the regularized energy release rate and the regularized critical energy release rate respectively.

An averaging operator is introduced to fulfill non-local estimation of the energy (critical energy) release rate on the damage front. By construction, (1) with (2) impose damage along  $\phi$  gradient segment in  $\Omega_+$  (see Fig. 2). Thus, any point of  $\Omega_+$  may not have their damage value modified without changing damage values of all gradient segment points it belongs to. This introduces a natural averaging process along all points related by condition (1).  $\bar{Y}$  ( $\bar{Y}_c$ ) regularizes  $Y$  ( $Y_c$ ) by using  $\nabla\phi$  to drive operation and a weighting function to smooth the computed values (more details are given in Sect. 4).

The quantity  $Y_c$  may be a constant (as in Fig. 3) or may depend on damage, in which case it is expressed as  $Y_c^0 h(d)$  where  $Y_c^0$  is a constant and  $h(d)$  a softening function.



**Fig. 2**  $\bar{Y}$  graphical representation (in 2D for visualization, but the same principle holds in 3D): In *small dashed line*  $\phi$  skeleton. In *large dashed line* some intermediate iso-value of  $\phi$ . In *blue*  $\Gamma_0$ . In *red*  $\Gamma_c$ . *Cyan* segments are aligned with the gradient of  $\phi$ . Over these segments,  $\bar{Y}$  is uniform and is the average of the underlying  $Y$  field



**Fig. 3** Abrupt stress/strain local response considered in test cases 6.1, 6.2, 6.3, and 6.4

### 2.3 Staggered algorithm

The TLS solver used for this work is given by Algorithm 1 where  $\mu^i$  is the load factor corresponding to load step  $i$ .

The  $g$  operator is related to the damage growth model (8). The  $K$  and  $F$  operators are related to the structural equilibrium. Finally,  $f_k$  operators are related to the damage criteria (8a).

---

**Algorithm 1** Schematic staggered algorithm scheme.

---

Starting from  $\mathbf{u}^i$ ,  $d^i$ , and  $\mu^i$

**repeat**

Find  $d^{i+1}$  such that  $d^{i+1} = g(\mu^i, \mathbf{u}^i)$  (I)

Find  $(\mathbf{u}^{i+1}, \mu^{i+1})$  such that

$$\begin{cases} K(d^{i+1}, \mathbf{u}^{i+1}) \mathbf{u}^{i+1} = F(\mu^{i+1}) & (II) \\ \max_k (f_k(\mu^{i+1}, \mathbf{u}^{i+1})) = 0 & (III) \end{cases}$$

**until** Complete failure or user given load level

---

In Algorithm 1, (I) (corresponding to (29) in Bernard et al. 2012) uses an explicit resolution. In (II), the  $\mathbf{u}^{i+1}$  solution is obtained with a Newton–Raphson algorithm due to the non-linear nature of  $K$ . Condition (III) is solved with an explicit resolution. Solving (III) is eased by the fact that  $\mathbf{u}^{i+1}$  has a linear relationship with respect to  $\mu^{i+1}$  (as long as it does not change sign).

The displacement field  $\mathbf{u}^i$  is discretized in space using a classical finite element approximation plus a ramped Heaviside enrichment (Bernard et al. 2012) to represent the existing cracks. The determination of  $\Gamma_c$  is greatly improved compared to Bernard et al. (2012) by the use of the so-called double cut algorithm detailed in Sect. 3.

The solution  $(\mathbf{u}^i, d^i, \mu^i)$  at any load step is thus in equilibrium (in the finite element sense) and satisfies fully (8a) and (8b). Equation (8c) is, however, slightly violated (see Bernard et al. 2012 for discussion).

The load step is not given by the user. It is part of the solution process. What is provided is the maximum damage (or  $\phi$  to be more precise) increase load step. It enters in the definition of  $g$ . The  $g$  operator is needed both to grow the existing damage zone and initiate new ones.

Note that in Algorithm 1, we have made clear an explicit dependence of  $K$  on  $\mathbf{u}^{i+1}$ . Indeed, the asymmetric damage model, introduced in (5), yields a non-linear elastic problem. This dependence of  $K$  on  $\mathbf{u}^{i+1}$  is in fact very limited in space since it affects only  $\Omega_+$ . Thus, the tangent matrix used to solve structural equilibrium is implemented to spare CPU time as a fixed part, assembled once per load step and a varying part, reassembled at each Newton–Raphson iteration. The same strategy is used also for the Newton–Raphson residual vector where the linear part of the internal forces is simply obtained by multiplication of the fixed tangent matrix part by the  $\mathbf{u}$  associated part.

Regarding the initialization of the staggered scheme, we find first the elastic loading for which damage

occurs. This gives  $(\mathbf{u}^0, d^0 = 0, \mu^0)$ . If needed, a non-zero initial damage field may be prescribed.

Regarding the finite element mesh, it can either be given at the start (and fine enough in zones that will be affected by damage, where element size should be at most a quarter of  $l_c$ ) or evolving during the simulation. In particular, octree adaptive grids are quite efficient. All numerical examples dealt with in this paper used a constant grid during the simulation.

To be precise, the definition of  $g$  follows the paper of Bernard et al. (2012) except for one improvement, the computation of  $\bar{Y}$ . This change is detailed in Sect. 4.

### 3 Crack lips definition improvement

#### 3.1 Crack representation

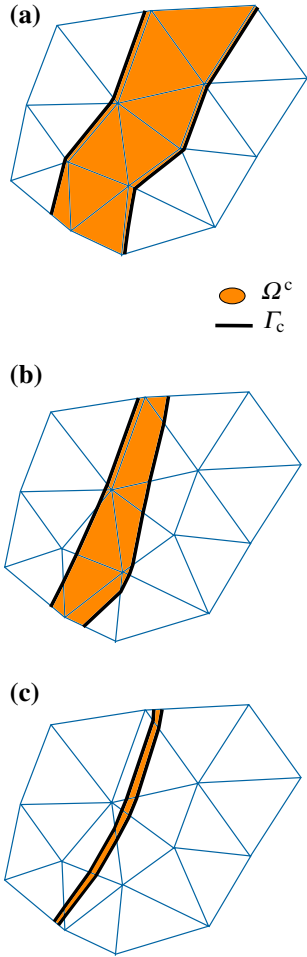
Usual treatment of level set in the context of X-FEM starts by determining the iso-zero location. This definition is then used to generate integration cells (called sub-elements hereafter) embedded in elements crossed by iso-zero. Depending on problem type, some or all sub-elements are used to integrate the problem using approximation of elements crossed by iso-zero. Some extra enrichment aspects may also be added and depend on this initial treatment. In the TLS framework this scheme is followed for both  $\Gamma_0$  and  $\Gamma_c$ . In this work we focus on iso- $l_c$  improvement, which represents the crack discontinuity.

From a numerical point of view, if no enrichment is introduced around iso- $l_c$ , crack lips will move apart until an element is fully in the damage zone (see Fig. 4a). This permits obtaining the expected discontinuous displacement by having a fully damaged element in between the lips of the crack. But depending on the element size, this may dissipate more energy than wanted, as the fully damaged zone depends then on mesh definition.

To avoid this, ramped Heaviside enrichment, explained in Bernard et al. (2012) and Sect. 3.5.1, offers a numerical mechanism to introduce displacement discontinuity by adding extra degrees of freedom. These are added along the iso- $l_c$  on entities having their support divided into at least two parts by the fully damaged zone. This condition is directly related to the definition of  $\Gamma_c$ , which depends on the level set representation.

Since the level set is defined by algebraic values at mesh nodes with linear interpolation between them, an





**Fig. 4** Location of  $\Omega_c$  (orange) depending on the technique used to construct  $\Gamma_c$  and enrichment scheme. **a** Single cut without discontinuous enrichment, **b** single cut with discontinuous enrichment: crack lip can only appear if there is at least one node between them, **c** double cut with discontinuous enrichment

edge may only be cut once by the  $\text{iso-}l_c$ . This restriction implies that elements may not be cut into two parts by the fully damaged zone. Consequently, enrichment will only occur if at least two elements are cut by the fully damaged zone (one element per crack lip). This imposes that the crack zone grows (or shifts) enough to have discontinuous displacement. This may again dissipate more energy than in reality (or may lead to a wrong crack path) if the mesh size is too big (see Fig. 4b).

A new approach permits passing over the restriction imposed by the level set. First, a versatile tool, the vector distance function, from the work of Gomes and Faugeras (2003), is reused and presented in Sect. 3.2. In

this work a signed form of this tool is proposed, which leads to the signed vector distance function concept. It introduces extra information compared to the classical level set tool. Secondly, adapted to signed vector distance function concept, a new algorithm permits cutting an edge twice (the “double cut” algorithm), which allows a fully damaged zone to cut a single element (see Fig. 4c). This double cut algorithm first cuts all edges and then splits all elements. This is explained in Sects. 3.3 and 3.4.

### 3.2 Signed vector distance function

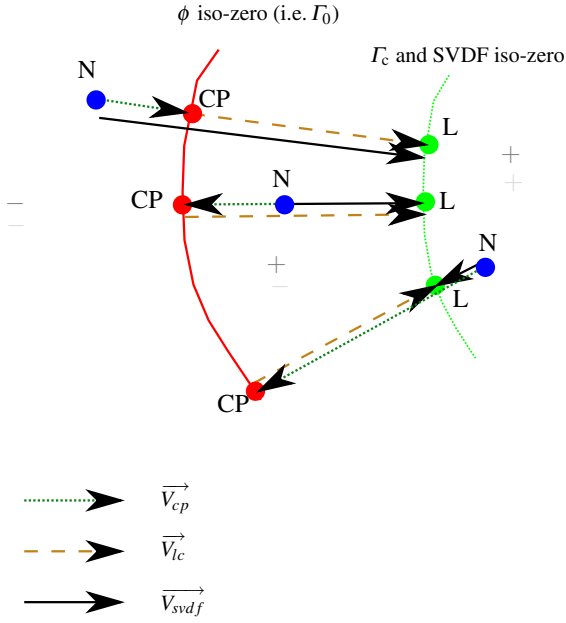
Given a manifold, a vector distance function (VDF) gives at each surrounding point a vector pointing to the closest point on the manifold (Gomes and Faugeras 2003). The manifold is also denoted as the VDF iso-zero in what follows. The manifold of interest in this section is  $\Gamma_c$ .

To keep domain partitioning available, a sign has to be associated with vectors (so we use the concept of signed vector distance function (SVDF) and not simply VDF). This means a node is in the negative or positive domain delimited by SVDF iso-zero.

In the TLS framework, the SVDF replaces the offset level set for the definition of the  $\text{iso-}l_c$ . Indeed, offset level set technology is not able to find multiple cuts within the same element, as in Fig. 4c.

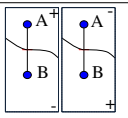
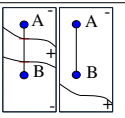
In this context no intrinsic evolution of the SVDF is taken into account as it could be (see Gomes and Faugeras 2003). It is only used as an enhanced tool to represent the  $\Gamma_c$  manifold, reset at every load step by using a new  $\phi$  definition. It follows the same sign convention as the level set it replaces (i.e. positive for the fully damaged zone and negative for all other parts of the domain). It is constructed for each mesh node, from the level set defining the front, by the following steps illustrated in Fig. 5:

- Find the closest point (CP) on  $\Gamma_0$  from node (N).
- Compute the vector  $\vec{V}_{cp}$  from N to CP.
- From this vector, construct the vector  $\vec{V}_{lc} = \frac{\text{sign } l_c \vec{V}_{cp}}{\|\vec{V}_{cp}\|}$  starting from CP where *sign* is 1 if level set sign for  $N$  is “−” and −1 if level set sign for  $N$  is “+”.
- Signed vector distance function vector  $\vec{V}_{svdf}$  for  $N$  is then  $\vec{V}_{cp} + \vec{V}_{lc}$  starting from  $N$  ending in point call L.



**Fig. 5** Signed vector distance function construction from iso-zero front. *Vector* are in fact all superposed, but to distinguish them on this figure they are drawn shifted if needed. Here, representation is in a plane, but it has to be understood in 3D space

**Table 1** Possible cut cases for an edge AB

Position of A and B			
Number of cuts	0	1	0 or 2
Illustration of figure 6	6h	6b,6c,6d	6e,6f,6g

- Signed vector distance function sign for  $N$  is “+” if level set sign for  $N$  is “+” and  $\|\vec{V}_{cp}\| > l_c$ . Otherwise, it is “-”.

### 3.3 Double cut algorithm first step: cutting edges

From SVDF information, one must first compute  $\Gamma_c$  cutting point for all mesh edges. Conditions to have an edge AB cut, given in Table 1, are from logical consideration of the placement of edge relatively to domain parts.

The algorithm uses then the following general geometrical predicate:

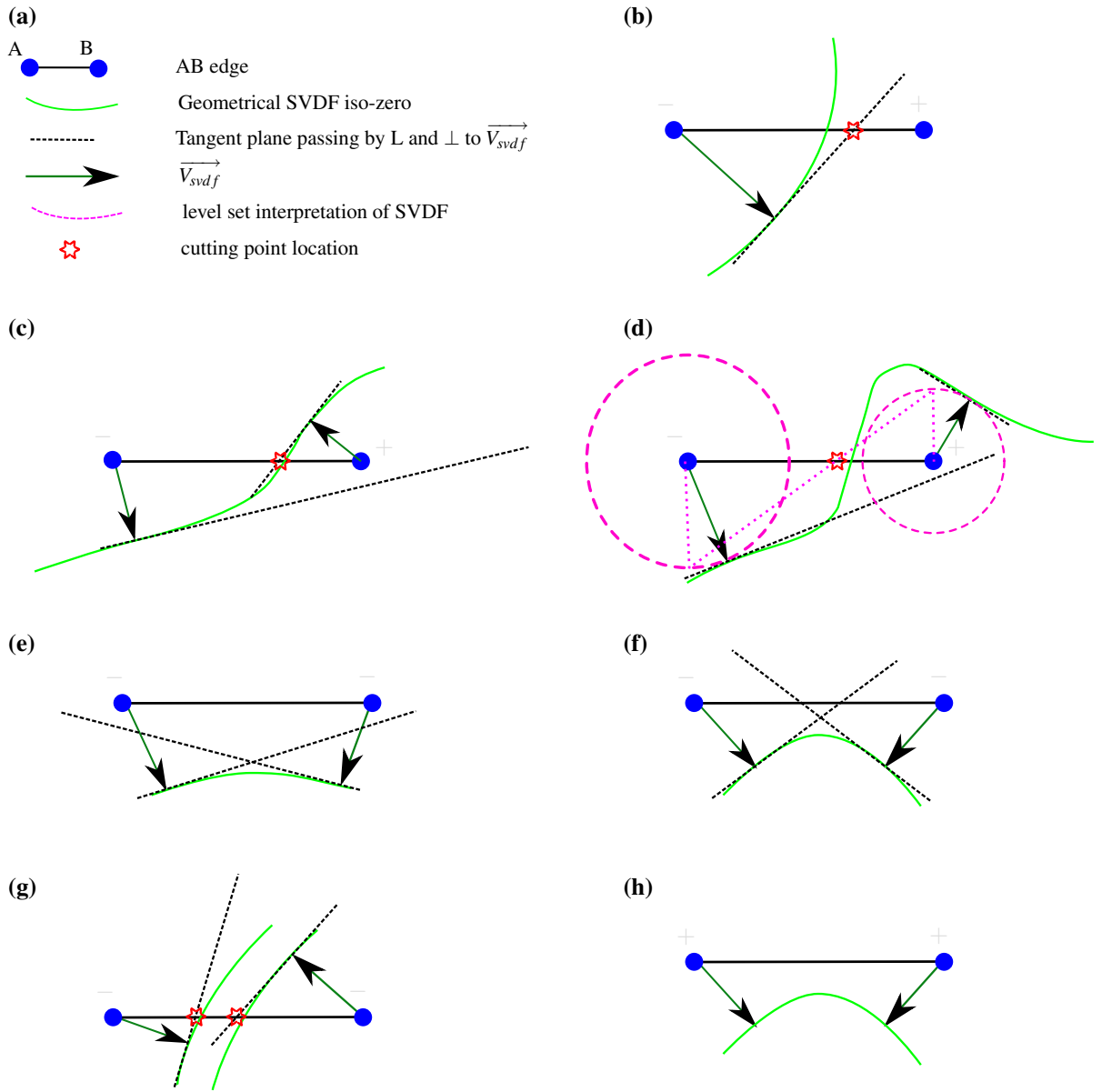
- Geometrical SVDF iso-zero passes through ending point L.
- Geometrical SVDF iso-zero tangent plane at L point is orthogonal to  $\vec{V}_{svdf}$ .
- Geometrical SVDF iso-zero tangent plane on L points corresponding to each node A and B of an edge may cut it.

The algorithm to obtain discrete SVDF iso-zero points on edges (i.e.  $\Gamma_c$  points) is based on these geometrical predicates. Tangential plane intersection(s) is(are) considered under the following conditions illustrated in Fig. 6 (and logical consideration of Table 1) as SVDF iso-zero point(s):

- one cut case (+-/-++)
- Cut is searched first with negative SVDF information (Fig. 6b). This follows the assertion that negative SVDF information is potentially of better quality than positive SVDF information, which is closer to hard tracked skeleton location.
- If the above failed, cut is searched with positive SVDF information (Fig. 6c).
- If the above failed, a linear interpolation of magnitude of SVDF vector taken as level set values is done (Fig. 6d).
- zero or two cut case (- -)
  - If tangent planes do not cut the segment, then there is no cut (Fig. 6e)
  - If tangent planes cut the segment twice, but the order of cut point is not “A, cut from point A, cut from point B, B,” then there is no cut (Fig. 6f). This condition avoids topological inconsistency (unwanted intersection) when following SVDF iso-zero.
  - If tangent planes cut the segment twice in correct order (see above) there are two SVDF iso-zero points (Fig. 6g).

An important aspect to consider in this process is what should be done when an identified cut point lies close to A or B. First, a definition of “close” must be given. A cut point is close to an edge node when its





**Fig. 6** Edge cut construction illustrating cut point determination. Representation is planar, but it generalizes to 3D. **a** Legend, **b** preferred one cut solution, **c** one cut first alternative, **d** one cut

using level set value as last resort, **e** no cut by construction, **f** no cut by wrong order of cutting points, **g** two cuts, **h** no cut by Table 1 rule

abscissa<sup>2</sup>  $s$  in the edge coordinate system is such that  $s \in [0, \varepsilon_{metric}[$  or  $s \in ]1 - \varepsilon_{metric}, 1]$  where  $\varepsilon_{metric}$  is an arbitrary small number (we choose  $10^{-5}$  in this work).

<sup>2</sup>  $s$  is dimensionless and vary from 0 (edge node origin location) to 1 (other edge node location).

With single cut algorithm, usually, a simple treatment that modifies the level set is enough so that its iso-zero surface passes through the node. But it is no longer possible to do the same with double cut algorithm as there is no way to identify crack lips if we merge information on nodes. Thus, the cut node is placed on an

edge node if close to it, but it still belongs to the edge from a topological point of view.

Regarding double cut position on an edge, a similar proximity consideration must be taken into account. Are two cutting nodes on an edge close enough to be considered coincident? Again, the same  $\varepsilon_{\text{metric}}$  is used to compare abscissa (in the edge coordinate system)  $s_1$  and  $s_2$ : if  $|s_1 - s_2| \leq \varepsilon_{\text{metric}}$  then both cut points are considered metrically at the same location ( $\frac{s_1+s_2}{2}$ ) and topologically distinct.

Another case is also considered when looking for coincident node treatment. Independently of what the above plane cut algorithm might give, if the SVDF magnitude at a node divided by the length of the emanating edge is small (less than  $\varepsilon_{\text{metric}}$ ) then the cut is considered to exist and is located at the node. To avoid dissimilar treatment around a node due to edge length fluctuation, we first compute the mean length of edges connected to a node. This gives a consistent metric reference across edges to evaluate the magnitude of a SVDF.

All these overlapping position identifications are then used when constructing sub-elements for integration and identifying enrichment (see “Element cutting” section of “Appendix 1” and Sect. 3.5.1 for further details).

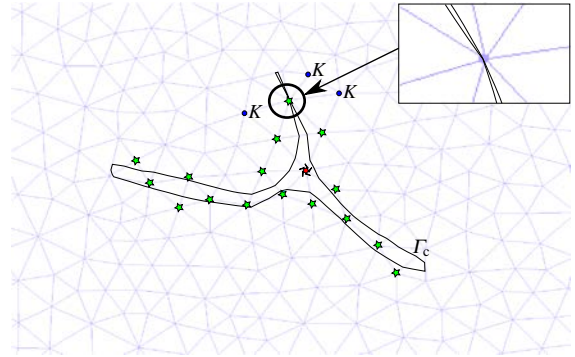
### 3.4 Double cut algorithm second step: cutting elements

Having edge cut positions and topologies, one is then able to construct the geometric domain using the following assertions (also available in 2D):

- The convex hull generated by the set of cut points and positive nodes corresponds to  $+$  domain ( $\Omega_c$  in TLS framework). It is a polytope<sup>3</sup> (polyhedron or polygon, depending on the case).
- The subtractions (geometric operation sense) of this  $+$  domain to the original element give the  $-$  domain polytope ( $\Omega_+$  in TLS framework).
- The common boundary of the  $+$  and  $-$  domains gives the SVDF iso-zero ( $\Gamma_c$  in the TLS framework).

By using elementary topological rotations and symmetries, element cut pattern cases may be reduced to a small set of unique cases. Depending on the way one

<sup>3</sup> See Coxeter (1973, p. 118) for a general definition.



**Fig. 7** Selection of nodes to be enriched by the ramped Heaviside. *Star* nodes are enriched once and the “sun” node is enriched twice. Note that *K* nodes are not enriched because both cracks lips run on the boundary of their support

chooses to implement the cutting algorithm, generation of null volume parts (from cut point merged with other nodes) may be eliminated from integration or not. This leads to an increased number of element cut pattern cases or not. All these cases are depicted as 3D in “Appendix 1”. You will find also in this Appendix a pure geometric illustration of the performance of the double cut algorithm with SVDF compared to the single level set cutting algorithm.

### 3.5 Discussion on new $\Gamma_c$ definition

The impact of using the “double cut” algorithm for discretization of  $\Gamma_c$  in the TLS framework is twofold:

1. the way ramped Heaviside enrichment is computed. This will be discussed in Sect. 3.5.1
2. the way  $\phi$  values are evaluated in  $\Gamma_c$  vicinity. Section 3.5.2 briefly comments on this aspect.

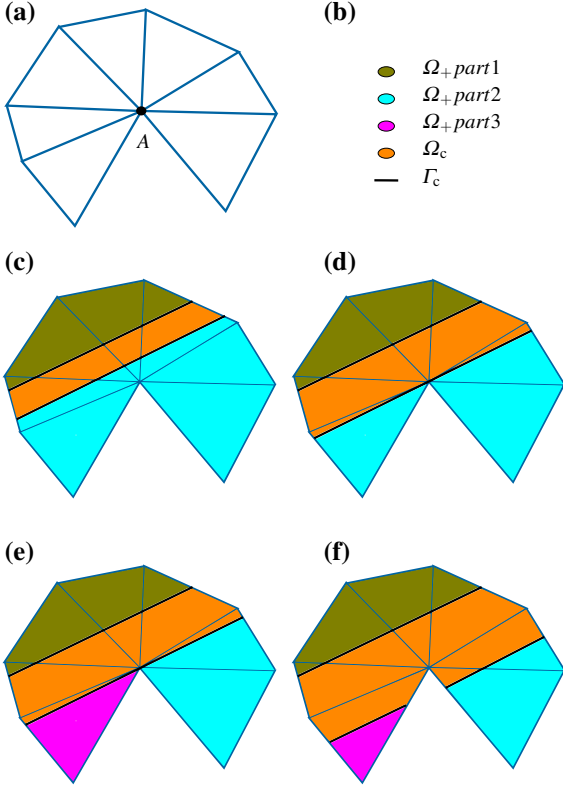
#### 3.5.1 Enrichment impact

The fact that an element may be cut twice by the same iso- $l_c$  has little influence on the process of finding dof<sup>4</sup> to enrich describe in Bernard et al. (2012). The general guidance is still to consider for a given dof its support<sup>5</sup> and count how many unconnected parts from  $\Omega_+$  are created by splitting it by  $\Omega_c$  (see Fig. 7).

As soon as there is more than one  $\Omega_+$  part, enrichment must be used to allow crack opening.

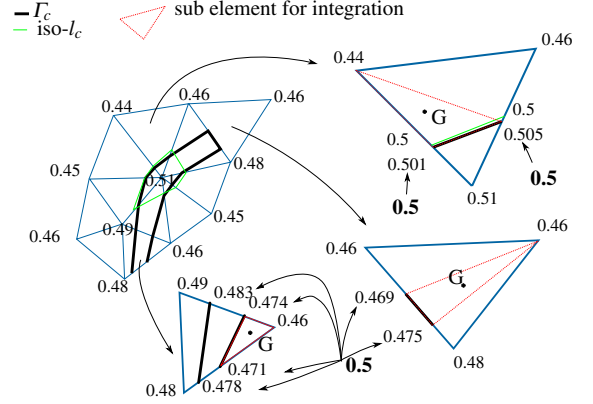
<sup>4</sup> Degrees of freedom.

<sup>5</sup> The support of a dof is the set of elements over which the approximation function associated with the dof is non-zero.



**Fig. 8** Decomposition of the support of a node  $A$  (a) into sound  $\Omega_+$  parts. Two sound  $\Omega_+$  parts appear in (c) and (d), whereas three sound  $\Omega_+$  parts appear in (e) and (f). In (d),  $\Gamma_c$  runs through node  $A$  while cutting edges above. In (e),  $\Gamma_c$  runs through node  $A$  while cutting edges below. **a** Mesh, **b** legend, **c**  $A \in \Omega_+$ , **d**  $A \in \Omega_+$  and  $A$  close  $\Gamma_c$ , **e**  $A \in \Omega_c$  and  $A$  close  $\Gamma_c$ , **f**  $A \in \Omega_c$

The “double cut algorithm” presented in this work carefully handles cases where the crack runs close to nodes (see Sect. 3.3 and Appendix 1). This has an impact on the way  $\Omega_+$  parts are identified and counted. In Fig. 7 the zoom shows a basic example of  $\Gamma_c$  passing on a mesh node. In this case a topological  $\Omega_+$  part exists, but is of zero measure. Those parts do not count for enrichment decision. Subtle impact on  $\Omega_+$  parts identification is illustrated in Fig. 8. Consider the mesh given in Fig. 8a around node  $A$ . In Fig. 8c, f, where  $\Gamma_c$  does not run through node  $A$ , we have either two sound (undamaged) parts (Fig. 8c) or three sound parts (Fig. 8f), depending on whether node  $A$  lies outside  $\Omega_c$  or not. In Fig. 8d,  $\Gamma_c$  runs through node  $A$  (from the above), there are two sound parts (as in Fig. 8c). In Fig. 8e,  $\Gamma_c$  runs through node  $A$  (from below) and there are three sound parts (as in Fig. 8f). Recall (see Sect. 3.3) that even though  $\Gamma_c$  runs through a node, we



**Fig. 9** Reshaping  $\phi(x)$  around  $\Gamma_c$ . In **bold**, new values used to compute  $\phi(x)$  on Gauss points  $G$ .  $l_c = 0.5$ , 2D

keep the information of which edges are cut around the node by  $\Gamma_c$  (enabling tracking of  $\Omega_+$  part definition).

### 3.5.2 $\Gamma_c$ new definition: impact on $\phi(x)$

The “double cut algorithm” uses a signed vector distance function instead of an offset level set for  $\Gamma_c$  construction. As exposed in Sect. 3, “Appendix 1” and in the following test cases, this gives good results for  $\Gamma_c$  discretization.

But it introduces some differences in the vicinity of  $\Gamma_c$ , between the  $\phi$  level set value and the crack front location. Figure 9 shows the issue on a 2D example (with  $l_c = 0.5$ ). In this example a little mesh (upper right) supports a level set (values attached to nodes). Using standard linear interpolation, an offset  $iso-l_c$  curve is plotted in green. Using the new double cut algorithm, the  $\Gamma_c$  curve is in bold black. From this situation three elements are zoomed to provide insight on  $\phi(x_G)$  computation. Here,  $G$  is the Gauss point of the integration sub-element depicted by  $\Gamma_c$  cutting those elements. Now there are two ways to interpolate  $\phi(x)$  on  $G$ :

- geometric linear interpolation of the element level set values
- geometric linear interpolation of the sub-element level set values

Clearly, in either case if we use level set values we won’t have a value in accordance with the presence of  $\Gamma_c$ . For example, the crack tip element (upper right element of the mesh) has following level set values: 0.46, 0.46, 0.48.  $\phi(x)$  on  $\Gamma_c$  edge node location is 0.469 and

0.475 from geometric linear interpolation of element level set values (bottom right zoom).

To be always consistent with SVDF construction, the idea is to consider that level set values on  $\Gamma_c$  nodes are set to a value of  $l_c$ . Then, computation of  $\phi(x_G)$  using geometric linear interpolation of sub-element modified level set values will be in accordance with the presence of  $\Gamma_c$ .

Doing so,  $\phi(x)$  is no longer a distance function in the vicinity of  $\Gamma_c$ . And somehow the response of function  $d()$  is no longer what we want in this region (It is locally another  $d()$  function), but damage is now 1 on  $\Gamma_c$ .

#### 4 New averaging computation

This paper modifies the computation of  $\bar{Y}$  (illustrated in Fig. 2), the regularized damage energy release rate. The following variational formulation is used to compute  $\bar{Y}$ :

Find  $Y \in \mathcal{Y}$ ,  $\lambda_1 \in \mathcal{Y}$ ,  $\lambda_2 \in \mathcal{Z}$  such that

$$\left\{ \begin{array}{l} \int_{\Omega_+} \bar{Y} d'(\phi) \hat{Y} + \frac{h^2}{l_c} \nabla \bar{Y} \cdot \nabla \hat{Y} d\Omega \\ \quad + \int_{\Omega_+} l_c \nabla \lambda_1 \cdot \nabla \phi \nabla \hat{Y} \cdot \nabla \phi d\Omega \\ = \int_{\Omega_+} Y d'(\phi) \hat{Y} d\Omega \quad \forall \hat{Y} \in \mathcal{Y} \end{array} \right. \quad (9a)$$

$$\left\{ \begin{array}{l} \int_{\Omega_+} l_c \nabla \hat{\lambda}_1 \cdot \nabla \phi \nabla \bar{Y} \cdot \nabla \phi d\Omega \\ \quad + \int_{\Gamma_0} l_c \lambda_2 \cdot \hat{\lambda}_1 d\Gamma = 0 \quad \forall \hat{\lambda}_1 \in \mathcal{Y} \end{array} \right. \quad (9b)$$

$$\left\{ \begin{array}{l} \int_{\Gamma_0} l_c \hat{\lambda}_2 \lambda_1 d\Gamma = 0 \quad \forall \hat{\lambda}_2 \in \mathcal{Z} \end{array} \right. \quad (9c)$$

where:

–  $\mathcal{Y}$  and  $\mathcal{Z}$  represent the following:

$$\mathcal{Y} = \{y \text{ "regular" on } \Omega_+\}$$

$$\mathcal{Z} = \{z \text{ "regular" on } \Gamma_0\}$$

–  $d(\phi)$  is defined by Eq. (2).

–  $d'(\phi) = \frac{\partial d(\phi)}{\partial \phi}$

–  $h$  is the mean size of element in  $\Omega_+$ .

Equation (9a) is dedicated to averaging process. The right hand side term corresponds to  $Y$  weighting summation on  $\Omega_+$  where  $d'(\phi)$  is the weighting function. The term  $\frac{h^2}{l_c} \nabla \bar{Y} \cdot \nabla \hat{Y}$  is diffusion added to the smooth averaging process.

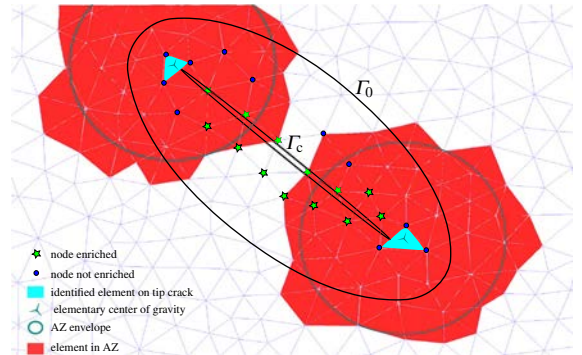
Equation (9b) imposes  $\bar{Y}$  to have its gradient  $\nabla \bar{Y}$  orthogonal to  $\nabla \phi$ . This imposes an averaging process along “flow” lines of the level set (see Fig. 2). In fact, (1) implies that  $\phi$  “flow” lines are segments connecting  $\Gamma_c$  to  $\Gamma_0$  (independently of the problem dimension). Equation (9c) imposes that  $\lambda_1$  is zero on  $\Gamma_0$ . If this condition was not imposed, the remaining Eqs. (9a) and (9b) would form a singular system ( $\lambda_1$  would be defined up to a constant on each segment aligned with a gradient of  $\phi$ ). Equation (9c) is the main difference with the system given in Bernard et al. (2012) where condition (9c) was a priori imposed in space as a Dirichlet condition.

#### 5 CPU time reduction

In damage mechanics, in the wake of the crack tip, no evolution of damage is expected. Unless boundary conditions impose drastic modifications or another crack joins from elsewhere, the damage and crack lips definition won’t change. From this observation, one can try to isolate those crack tips to focus computation effort on these zones.

These regions, called active zones, or AZ hereafter, are illustrated in Fig. 10 with an artificial TLS on a 2D coarse mesh. In this figure two crack tips have been identified using enrichment information, and a generic envelope (circles) is constructed (see “Appendix 2” for the envelope creation algorithm). Using this envelope, elements are separated in two groups:

- the AZ group (colored elements in Fig. 10, which are cut or inside the AZ envelope)
- the fixed group (white elements in Fig. 10)



**Fig. 10** Tracking crack tip by use of enrichment information: order 1 example in 2D

The idea is to construct an AZ group for a certain amount of load steps so that the TLS front may progress into it. As soon as the moving part of  $\Gamma_0$  reaches the AZ boundary, a new active zone has to be considered. But during this progression, information about the fixed zone ( $\phi, d, \dots$ ) is considered constant, and the computational cost may be optimized.

The bigger the AZ group, the more load steps that benefit from optimization of the fixed group (but the smaller this group is, the bigger is the computation for the AZ).

In this work the matrix associated with fixed group elements is condensed on the boundary of the AZ. During an AZ's lifetime, the fixed group is viewed only by its condensed stiffness on the frontier. The non-linear  $K$  operator of Algorithm 1 related to a fixed group is linearized at construction of the AZ. The Schur complement resulting from this condensation needs parallel distribution technique to handle memory consumption related to this dense matrix. Use of direct parallel sparse solver MUMPS<sup>6</sup> offers a good solution both to reduce overall linear solving time consumption and to reduce Schur complement creation time by doing incomplete block factorization. Following a domain decomposition approach, the AZ group itself is also condensed on its frontier with the fixed group. Then at each Newtown-Raphson iteration (Algorithm 1, (II)) a dense problem representing global system is created (by adding the fixed group Shur complement to the current AZ condensed problem) and solved. In this work it is solved with a parallel dense solver Scalapack<sup>7</sup> (but an iterative solver may also be considered).

One extra feature tested in this work is the use of mixed shape function approximation order. By setting a high order in the AZ and an order 1 outside, one may gain computational time. For that, the size of the common frontier between the AZ group and the fixed group has to be kept on the order of 1 (i.e. order transition must be done in the AZ boundary element) to get the least dense problem size possible.

Use of parallelism induces load balancing strategy. In this work a rather specific partitioning evoked in "Appendix 4" has been tested to stick to the AZ concept. AZ performance in the context of the following numerical example is discussed in "Appendix 5".

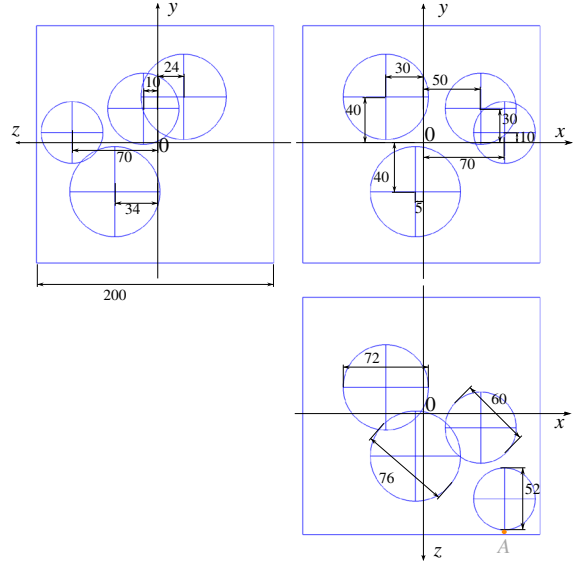


Fig. 11 Spherical holes test case: dimensions in mm

## 6 Numerical example

All material characteristics and simulation settings are given in "Appendix 6".

### 6.1 A cube with spherical holes

This test case illustrates crack merging and "search for damage initiation" capability of the TLS framework. A cube with four embedded spherical holes (see Fig. 11) is under tension with a uniform loading condition on two faces (perpendicular to  $y$  axes).

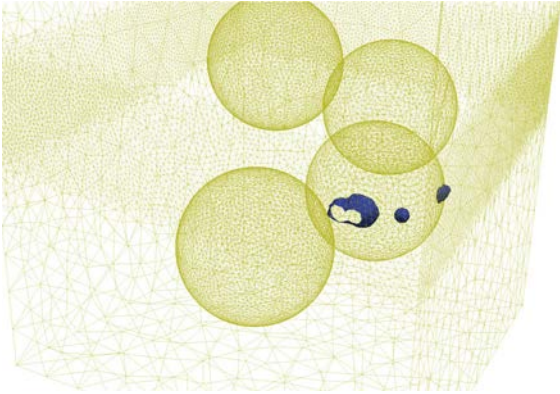
Simulation starts with an undamaged model. The initial step searches for the maximum damage criterion location. A spherical iso-zero level set is placed at this location introducing a small damaged zone. It is added around point A (see Fig. 11). Then, at each step, a search is made for extra locations violating locally the damage criterion ( $Y \leq Y_c$ ) outside the damaged zone that has already started. As the load factor is still slightly increasing during this part of the simulation, extra local zones are discovered almost at every load step (see Fig. 12 at step 4) while propagation enlarges the initial damaged zone.

The material section between the spherical hole and the cube face is so thin that its local degradation does not affect the overall stiffness too much. After five steps, the load starts decreasing, and at step 10 a crack appears

<sup>6</sup> 4.10 version.

<sup>7</sup> 1.8 version.





**Fig. 12** Spherical holes test case: damage initiation locations at step 4

(i.e.  $\Gamma_c$ ). From this point almost no extra locations are found during the simulation that violate the damage criterion.

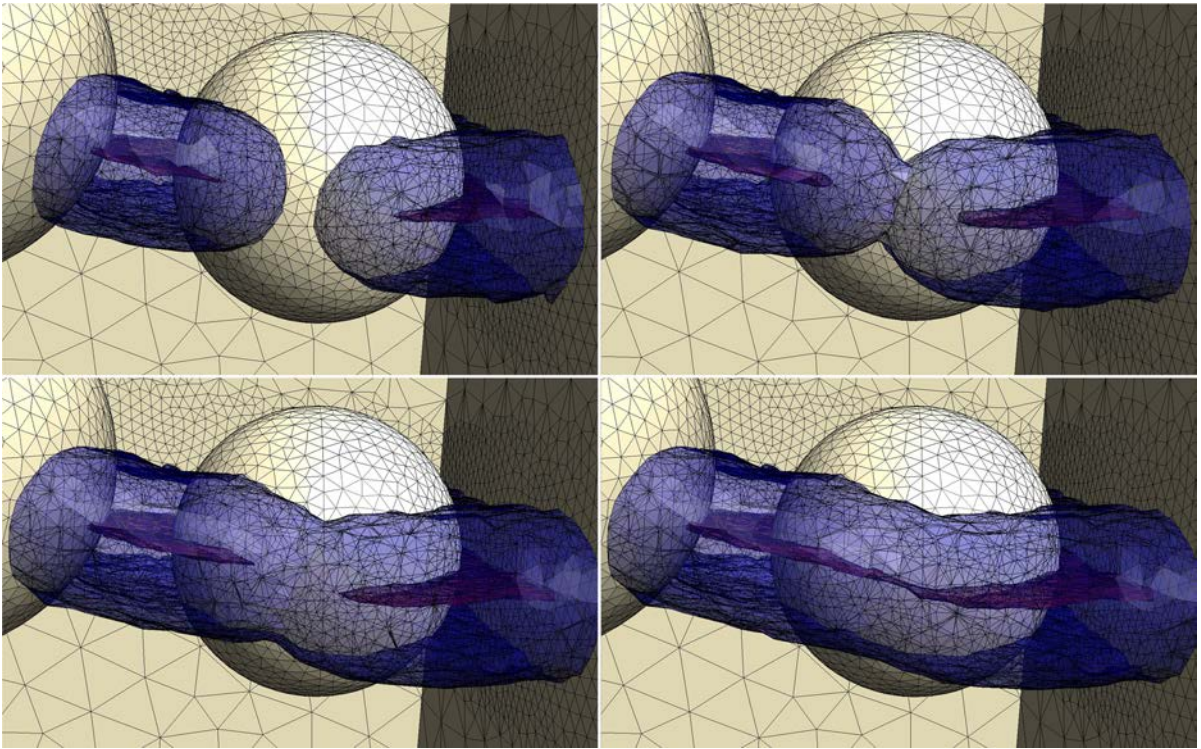
As given in Appendix 6, we consider for the cube example a damage law without hardening (corresponding to Fig. 3). It is clear that hardening would yield first a diffuse damage in the cube before localization. Hardening was considered in the TLS framework by Van Der Meer and Sluys (2015) and Moës et al. (2014).

During propagation, one observes merging capabilities of TLS. Figure 13 shows the behavior of the  $\Gamma_0$  and  $\Gamma_c$  when, after turning around the first spherical hole, fronts join together. First, the iso-zero of both sides of the front merge (upper right view). As distance from hole is less than  $l_c$ , propagation continues with rather steady  $\Gamma_c$  location (lower left view). Propagation continues and the front is sufficiently far from the hole (more than  $l_c$ ), which generates a quick coalescence of  $\Gamma_c$  front (lower right view). Here, fronts collide in a rather simple manner as both sides are more or less at the same  $y$  position (vertical of picture). Note that the merging of  $\Gamma_0$  fronts does not imply automatic merging of  $\Gamma_c$  fronts. Figure 14 illustrates this scenario.

Final crack location is given in Fig. 15 after 386 load steps. The cube is split into two components.

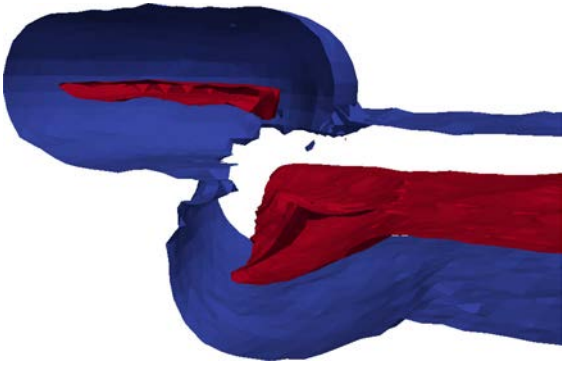
## 6.2 Chalk under torsion

This test case was studied in Bordas et al. (2008) with a mesh-free method. A cylindrical chalk bar is twisted with two opposite torques at its extremities. Geometry and loading are presented in Fig. 16.

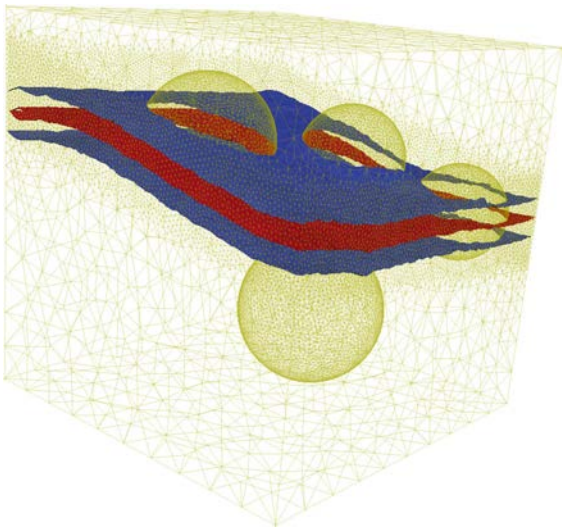


**Fig. 13** Spherical holes test case: merging (inside view with only skin mesh)

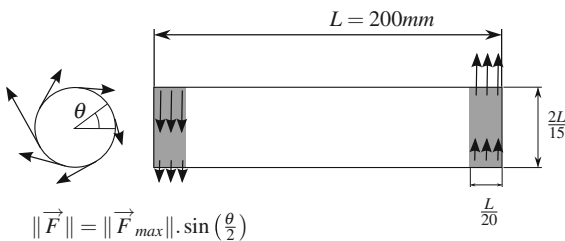




**Fig. 14** Undocumented test case (one crack diving under the other) presenting correct partial merging of  $\Gamma_0$  fronts, but not  $\Gamma_c$  fronts



**Fig. 15** Spherical holes test case: iso-surface after 336 load steps,  $\Gamma_0$  in blue,  $\Gamma_c$  in red



**Fig. 16** Chalk test case

Loading is applied in a tangential direction to the chalk surface. To stick to Bordas et al. (2008), loading magnitude depends on the angle  $\theta$  defined in Fig. 16.



**Fig. 17** Chalk comparison of the final state between simulation and experiment presented in Bordas et al. (2008) (Reprinted from Bordas et al. (2008) Copyright (2007) with permission from Elsevier)

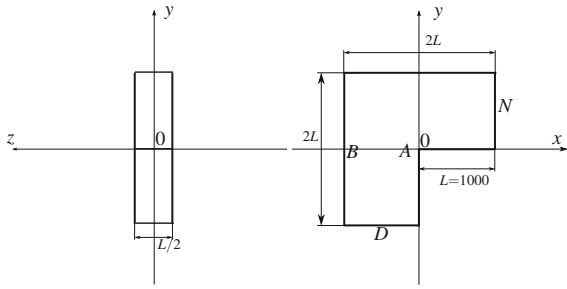
As the object is axisymmetric, a defect must be introduced to start at a defined location. In this work a simple initial damage is set by a  $\Gamma_0$  small sphere with a center at  $L/2$  along cylindrical axes on the chalk surface. This gives the opportunity to reduce time consumption by using a mesh with a refined slice where a crack will start and is expected to develop. Figure 17 presents the result of the simulation after 103 load steps, when the chalk is completely separated into two parts.

One observes a good agreement of crack shape between the experiment and the simulation: the development of a helicoid (bottom left and middle view) is followed by a blunt finish (bottom right view) where the crack front shape is more straight.

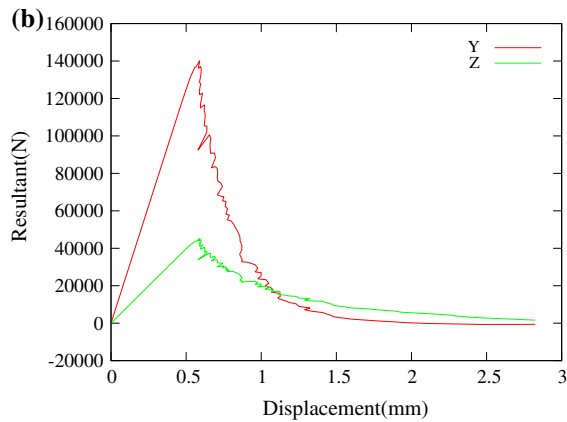
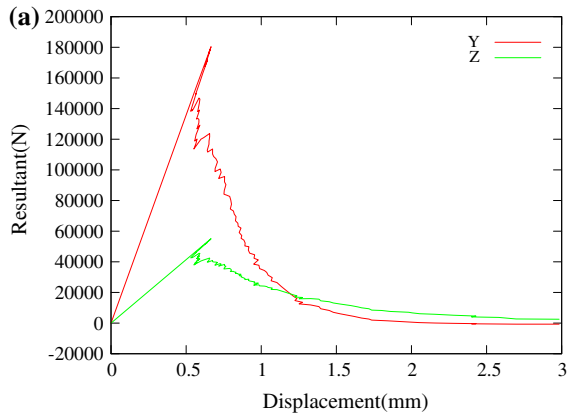
### 6.3 L shape: mode I + III

Lorentz and Godard (2011) modified the standard L Shape–mode I test case by introducing a supplementary lateral imposed displacement. This way mode III is activated and the crack path is slightly modified. Geometry and loading are presented in Fig. 18. Mesh is refined around corner  $A$  up to extremity  $B$ .

Two types of computations have been conducted. One with automatic damage initiation and the other with forced damage initiation (along the corner by setting an initial cylindrical  $\phi$  of radius  $1.05 \times l_c$ ).

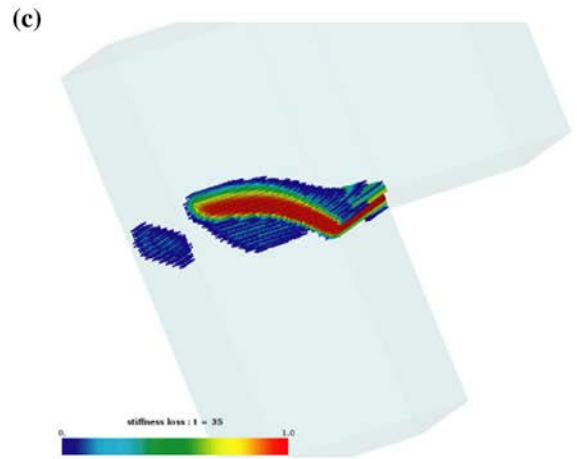
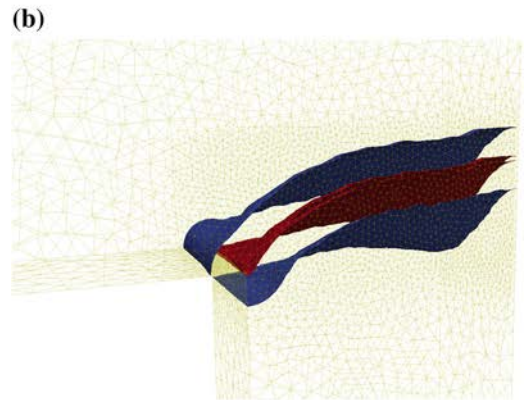
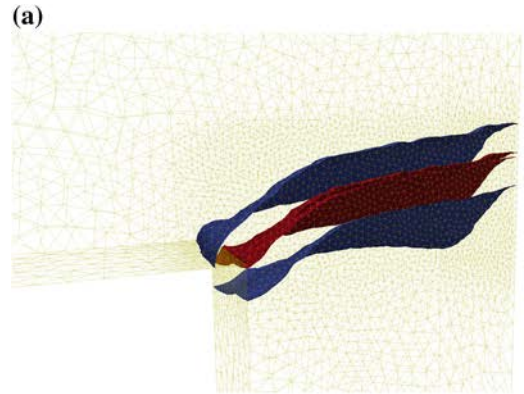


**Fig. 18** L shape: geometry and boundary conditions, dimensions in mm, D face with displacements fixed to 0 in all directions, N face with displacements imposed along y and z directions



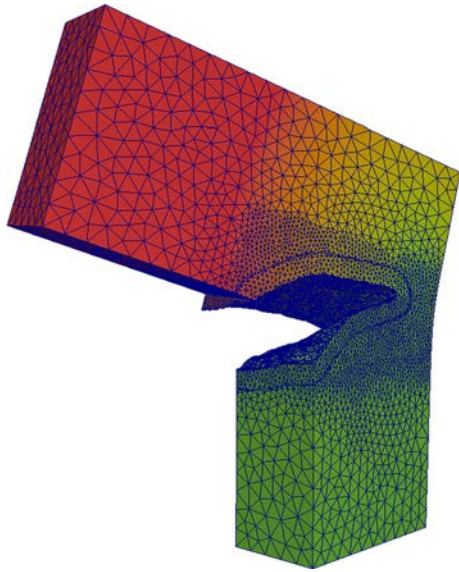
**Fig. 19** L shape: force–displacement curves (along Y an Z) with automatic (a) or forced (b) damage initiation

This second case is to compare better with results of Lorentz and Godard (2011), who obtained in the corner a rather straight damage distribution. Force–displacement curves are given in Fig. 19, and final situations are presented in Fig. 20.



**Fig. 20** L shape: TLS results (a, b): iso-surface at the end of computation,  $F_0$  in blue,  $F_c$  in red. Gradient method results (c). a) TLS with automatic damage initiation, b) TLS with forced damage initiation, c) results reprinted from Lorentz and Godard (2011) Copyright (2010) with permission from Elsevier. Damage distribution with lateral z effort inverted compared to this paper

Looking at Fig. 19b, one can see that forced damage initiation removes the initial steep snap back. Load–displacement curves are comparable in shape to those



**Fig. 21** L shape: displacement field (scaled) at load step 120 (with automatic damage initiation)

found in Lorentz and Godard (2011). A less brutal softening mechanism like in Parrilla Gómez et al. (2015) where  $Y_c$  is a function of  $d$  (or the inclusion of a hardening part), will most likely remove the steep initial snap back even with automatic damage initiation.

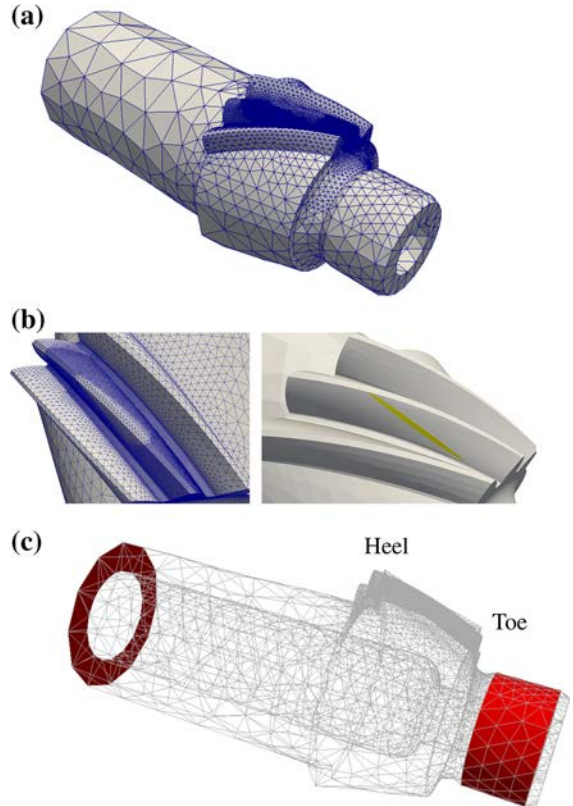
Comparison in Fig. 20 shows that there is little difference between final crack shapes with respect to damage initiation strategy except in the vicinity of corners. Compared to Lorentz and Godard (2011), the crack twists the same way, but simulation was conducted further in this work as compression is taken into account in the free energy (5) in  $\Omega_+$ . This test illustrates interest in automatic damage initiation, which permits obtaining a more complex crack path in vicinity of a corner.

Displacement field at load step 120 in Fig. 21 illustrates correct enrichment and representation of  $\Gamma_c$ . The opening of the crack faces is very clear.

Finally, this simulation shows that the TLS model gives equivalent results compared to another numerical tool which uses a damage gradient method.

#### 6.4 A spiral bevel gear

This test case mimics an industrial study on a spiral bevel pinion gear of a helicopter transmission system first appearing in a report by NASA Spievak et al. (2000) and published in Spievak et al. (2001). Later



**Fig. 22** Spiral bevel pinion gear: mesh, boundary condition and loading. **a** Mesh general view, **b** central tooth mesh view, **c** applied loading zone (yellow), **d** boundary condition (red)

Ural et al. (2005) looked again at this problem with a new computational technique. Those studies were conducted to help pinion gear designers by giving them a crack path with use of a simulation. The purpose of this test case is to show that the TLS method, with complex geometry, gives good crack shape information without specific mesh refinement and without any initial crack placement.

From Ural et al. (2005), Spievak et al. (2001) an approximate geometry has been rebuilt from scratch with only three teeth. In this work no fatigue study with varying complex loading is done as testing implementation only deals with quasi-static loading. The load location corresponds to the highest point of single tooth contact (HPSTC see Spievak et al. 2000). An ellipse  $E$  on an arbitrary CAD plane  $P^8$  is projected onto the tooth to follow Hertz contact shape. In Fig. 22c

<sup>8</sup> Not shown here, but roughly located in front of studied tooth, locally parallel to its face.

it appears as a yellow zone where mesh on the surface follows its boundary. Loading magnitude  $F$  in this zone is computed with the following expression:

$$F = H_c \cdot F_0$$

where:

- $F_0$  is the maximum load in this zone
- $H_c$  coefficient is

$$H_c = \sqrt{1 - \left(\frac{x}{a}\right)^2 - \left(\frac{y}{b}\right)^2}$$

where  $x, y$  are the projected coordinates of a point in this zone on plane P, with Cartesian coordinates corresponding to center and axes of ellipse E.  $a$  and  $b$  correspond, respectively, to major and minor radius of ellipse E.

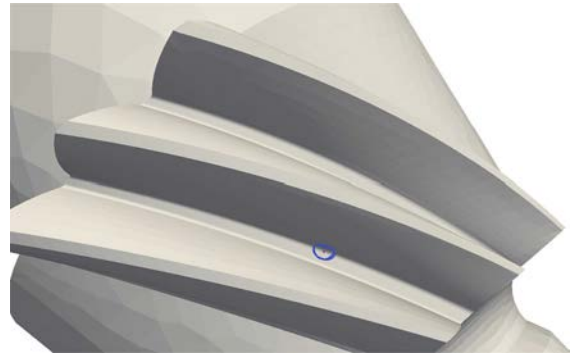
Figure 22d shows, in red, the clamped face (ring at the end of the long shaft) and the connection sliding pivot (cylinder on the surface of the smaller shaft). Figure 22a, b shows a general and focused view of the mesh. As seen, the mesh is overall much finer on the central tooth. Element size in the refined zone is at least  $\frac{l_c}{4}$ . The mesh contains 453,824 nodes.

This simulation produces the following results:

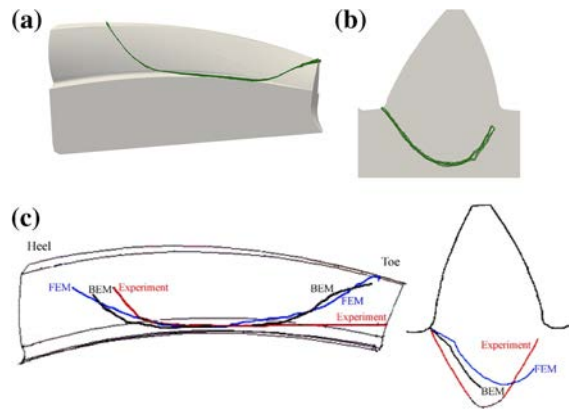
- Damage initiation (automatic) is found in the fillet concave part at almost mid-distance (8% shift) from toe and heel sides (see Fig. 22d for sides location). This is where Ural et al. (2005) places a crack. Only a small damaged zone is placed at this location, which does not make any assumption on future crack path.
- The first loading step extends this zone, and at the 9th load step a crack ( $\Gamma_c$ ) is automatically placed. It emanates from the damage initiation location (see Fig. 23).
- Crack shape is consistent with experimental results given in Ural et al. (2005) and reported in Fig. 24.

The general situation after 246 and 380 load steps is depicted with iso-surfaces in Fig. 25 and with a displacement field in Fig. 26.

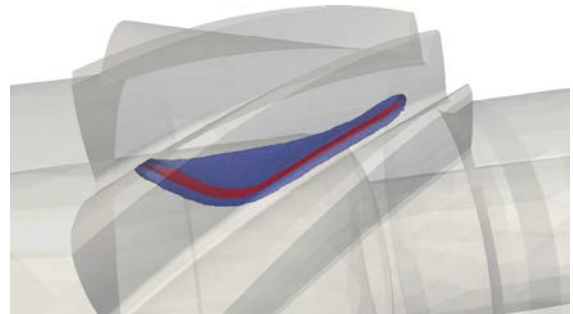
Note that in the simulation performed, the chosen  $l_c$  length is rather large with respect to actual material process zone size. However, the chosen material strength and length  $l_c$  do combine ( $Y_c \times l_c$ ) to produce



**Fig. 23** Spiral bevel pinion gear at load step 9: in blue  $\Gamma_0$  and the little red spot is the initial crack location

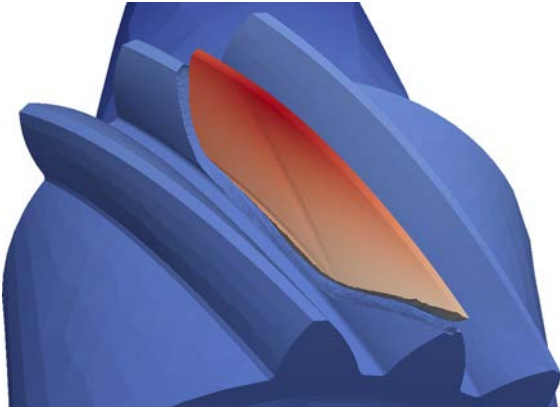


**Fig. 24** Spiral bevel pinion gear at load step 385: comparison with Ural et al. (2005). **a** TLS: lateral view (toe on right), **b** TLS: cut view passing at initiation location, **c** results reprinted from Ural et al. (2005) Copyright (2004) with permission from Elsevier



**Fig. 25** Spiral bevel pinion gear at load step 246: Iso-surfaces  $\Gamma_0$  (blue) and  $\Gamma_c$  (red)





**Fig. 26** Spiral bevel pinion gear at load step 380: displacement field

the proper order of magnitude for the toughness of the material.

### 6.5 A quantitative comparison: three-point bending test of a notched beam

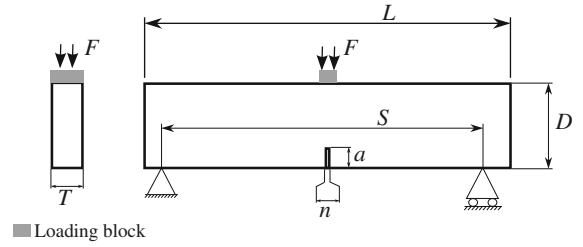
In this section, we try to provide some insight of the TLS method quantitative quality by doing a comparison with a test. We chose the work by (Hoover et al. 2013), which provides data for a size effect study on a concrete beam under three-point bending conditions. General beam geometry and loading are depicted in Fig. 27. Two (out of 18) size combinations are chosen from Hoover et al. (2013): the Bc and Cb cases (see dimensions given by Table 2).

Force displacement curves given in Fig. 28 presents the experimental response. CMOD (crack mouth opening displacement) is measured at the bottom of the beam between two points located symmetrically with respect to the notch (and separated by 137 mm and 59 mm in test cases Bc and Cb, respectively).

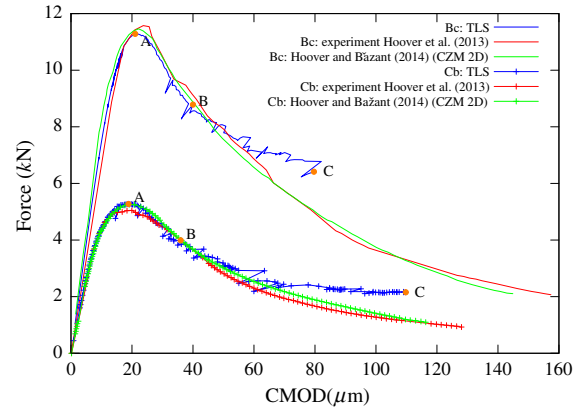
Cohesive zone model (CZM) simulations corresponding to the test case are given in Hoover and Bažant (2014) and are reproduced in Fig. 28.

**Table 2** Three-point bending test of a notched beam: dimensions (in mm)

Label	L	D	a	S	n	T
Bc	516	215	16.125	467.84	1.5	40
Cb	223.2	93	13.95	202.37	1.5	40



**Fig. 27** Three-point bending test of a notched beam: generic geometry and boundary conditions



**Fig. 28** Load displacement curves for notched beam cases Bc and Cb

The idea is to compare TLS simulations with this set of results (experiment and CZM). To model properly behavior of concrete, the brutal softening regime of Fig. 3 may not be used. To adopt a different softening regime, we consider the work of Parrilla Gómez et al. (2015) where  $Y_c$  now depends on  $d$ :

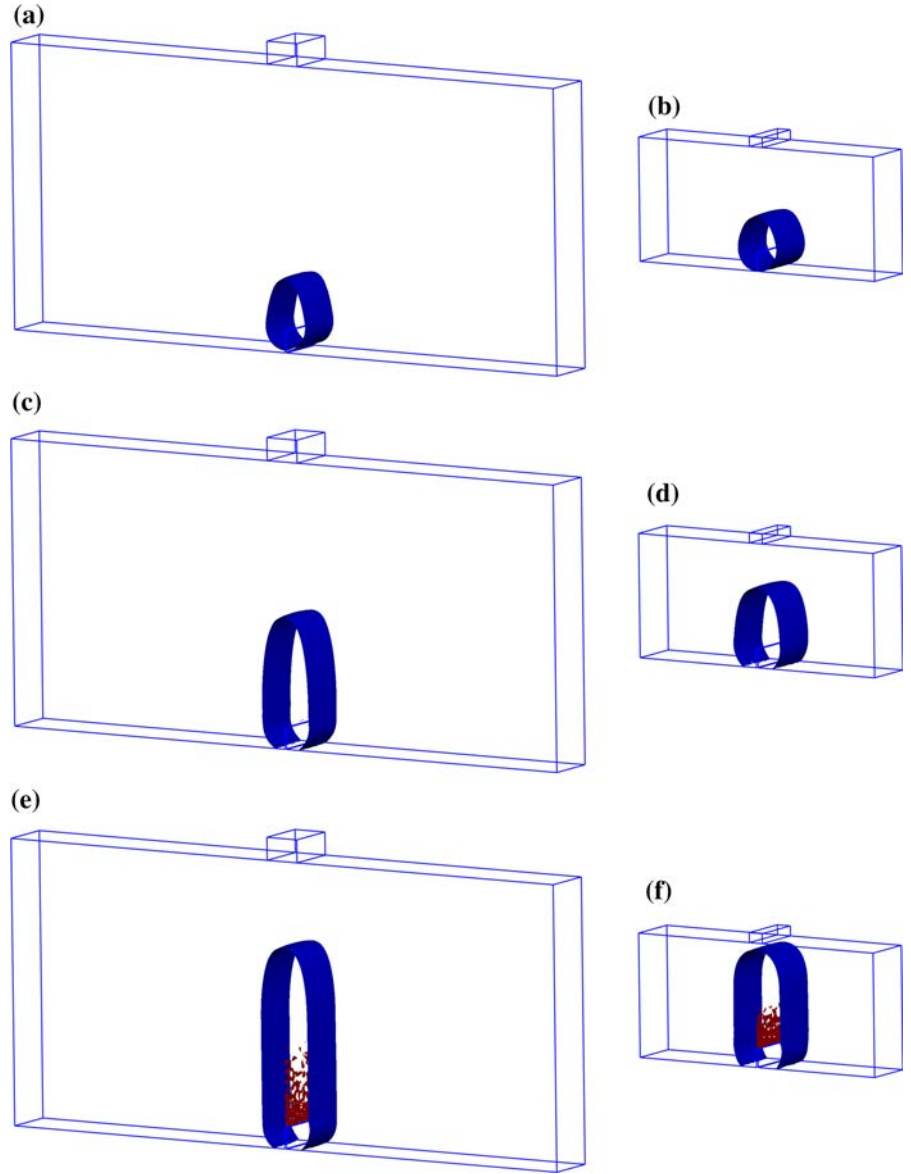
$$Y_c(d) = Y_c^0 h(d) \quad (10)$$

where  $Y_c^0$  is a constant.

Using Parrilla Gómez et al. (2015), the CZM (Hoover and Bažant 2014) parameter are transformed into TLS parameters ( $h(d)$  and  $Y_c^0$ ).

Results given in Fig. 28 show that TLS simulations are rather close to the test and CZM simulations. The test peak load is obtained with an error of 2.5 and 4.5 %, for the Bc (top) case and Cb (bottom) case, respectively (Note that the CZM simulations do give errors of 1 % (Bc) and 4.1 % (Cb) with respect to the test peak load). There is a discrepancy between the last part of the post-peak and the test. In Parrilla Gómez et al. (2015) load-

**Fig. 29** Notched beam case Bc (a, c, e) and Cb (b, d, f): iso surface at point A (a,b), B (c,d), and C (e,f) of Fig. 28,  $\Gamma_0$  in blue,  $\Gamma_c$  in red



CMOD curves on another test case show more accurately this post-peak region. In ongoing work, Parrilla Gomez et al. do a full comparison with Hoover et al. (2013) in 2D and will investigate this aspect.

In Fig. 29,  $\Gamma_0$  and  $\Gamma_c$  are presented for points A, B, and C. Those points, shown in Fig. 28, correspond to curve peak (A), first appearance of  $\Gamma_c$  (B), and simulation end (C). One observes that point B for both cases arrives quite late in the process. Damage develops first in the very long process zone. In Fig. 29e, we see that  $\Gamma_c$  appears in a non-uniform manner. This is related

to the long process zone where, on the skeleton of  $\phi$ , damage is almost 1. Transition to damage equal to 1 with automatic introduction of  $\Gamma_c$  discontinuity is then hard to achieve softly as the long part of the skeleton may vary abruptly. In those simulations we chose to slow down the damage front advance (which gives this non-uniform intermediate state), but one other possible solution is to smooth more the transition to  $\Gamma_c$  creation by using a specific intermediate enrichment. This might also have some effect in the last part of post-peak curves.



## 7 Conclusions and perspectives

The numerical examples presented did show the capability of the TLS to model 3D quasi-static failure of quasi-brittle material. Although based on a damage model, the approach exhibits displacement discontinuities and crack growth. Initiation and crack coalescence are taken into account.

The extra effort required by non-locality is restricted to a narrow zone close to cracks ( $\Omega_+$ ). This is in contrast with other non-local approaches requiring computations over the whole domain. Also, the damage update is explicit and does not require an iterative scheme.

On the implementation side, a couple of new ideas have been proposed. Crack extraction is based on a double cut algorithm using a signed vector distance function. The width of the fully damaged zone is now quite insensitive to the mesh size. The active zone concept allows drastically reducing the computational time by focusing only on zones where damage growth is active.

We now discuss perspectives.

Certainly, the type of damage model considered in this paper has some limitations. The dissymmetric behavior in tension and compression is well modeled, but damage is only given by a scalar quantity. Damage anisotropy effect would need a better description of damage in the local model. Also, test cases were chosen so that the generated cracks do not suffer contact. The introduction of contact and more complex damage representation at the local level are currently being investigated.

As shown in this paper, the only matrix that needs to be built and solved for damage evolution is in the evaluation of the average energy release rate. This matrix assembly and solution may in fact be replaced by an averaging technique along modes built on the damage front and extended into the localization zones with a fast marching technique (in Moreau et al. 2015).

Techniques using fast marching instead of the signed vector distance function to extract the crack are also interesting alternatives. Two-dimensional experiments (in Chevaugeon et al. 2014) indicate that this allows for a good precision of the crack tip location, thus  $\phi$  reshaping (Sect. 3.5.2) will no longer be needed.

The capability of the TLS to exhibit cracks and their extraction with the double cut algorithm also allows considering important unrefinements in the wake of the crack and the use of classical X-FEM strategy in these zones. Current investigations indicate that this

could reduce dramatically the number of degrees of freedom.

**Acknowledgments** The authors are grateful for the research support of the European Research Council through ERC starting Grants ERC-XTLS N291102. The authors also thank Gilles Markmann regarding his help to provide CAD of the spiral bevel pinion gear from scratch, and Felipe Bordeu regarding his help with some paraview treatments for the double cut algorithm.

## Appendix 1: Double cut algorithm details

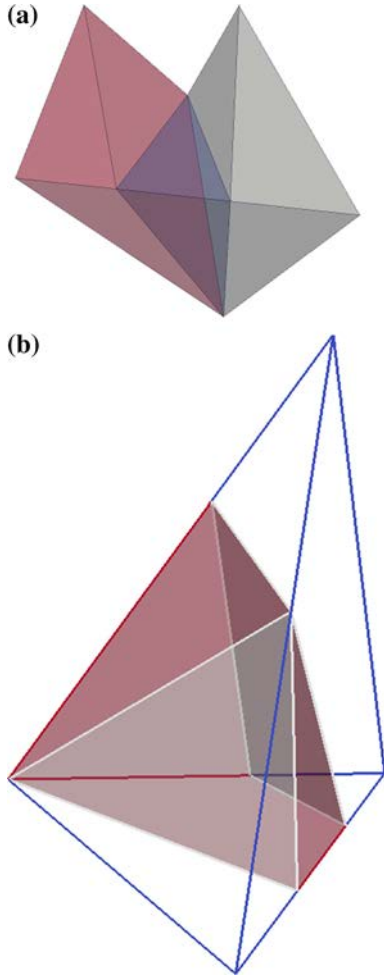
### Element cutting

Following the scheme given in Sect. 3.4, in 3D using Hert and Schirra (2013) for convex hull computation and Hachenberger and Kettner (2013) for subtractions, one obtains 17 reference element cut patterns for a tetrahedron element as shown in Fig. 31. A cut pattern corresponds to a set of edge cut points and tetrahedron node signs that lead to a unique topological element cut. The 17 patterns do not take into account the specific metric (single cuts are placed at the edge middle and double cuts at one third and two thirds of the edge).

We may group some of the 17 patterns into four different categories:

- Patterns 1 to 3 correspond to the simple cut that one can obtain with a classical scalar level set.
- Patterns 3, 5, 6, and 9 have a potentially  $\Gamma_c$  warped surface (four points surface) where the choice of diagonal is arbitrary.
- Patterns 11 to 14 give non-convex negative domain polytope ( $\Omega_+$ ), which have to be subdivided into convex polytopes to correctly generate sub-element tetrahedrons. This is what appears as extra blue edges on some element faces. It describes the sub-cut used to split polytopes into convex ones. See Fig. 30a for illustration of pattern 13 where a negative domain polytope was split into three convex ones.
- Patterns 15 to 17 are termination patterns.  $\Gamma_c$  is only present on tetrahedron boundary and no positive domain is present. For example pattern 17 may terminate positive zone of pattern 6 (bottom face). Those choices are arbitrary.

In Table 3, the number of possible permutations from each pattern is given. Without close node treatment, a total of 111 permutations have to be handled. With close



**Fig. 30** Convexity and close node treatment illustrations. **a** Tetrahedron negative domain polytope splitting: out of the initial domain, three convex polytopes are created for pattern 13 of Fig. 31, **b** pattern 5 of Fig. 31 with two cut nodes from the same edge collapsing, and one cut node for two edges collapsing on one node of the tetrahedron

node treatment, some polytopes change and no longer give null sub-elements for integration. See Fig. 30b for an illustration where one of the negative domain parts reduces to a single tetrahedron producing one instead of three tetrahedrons for integration. This is interesting when some specific extra computation is made with those sub-elements (for example, some fluid flow in the crack) and null volume is an issue. For this work we could have used those null sub-elements. But, in creating a tool to handle a double cut algorithm, we chose to be a bit more general from the start, and eliminated null volume sub-elements. This leads to close

**Table 3** Number of possible topological permutations obtained by using elemental topological rotation and symmetry

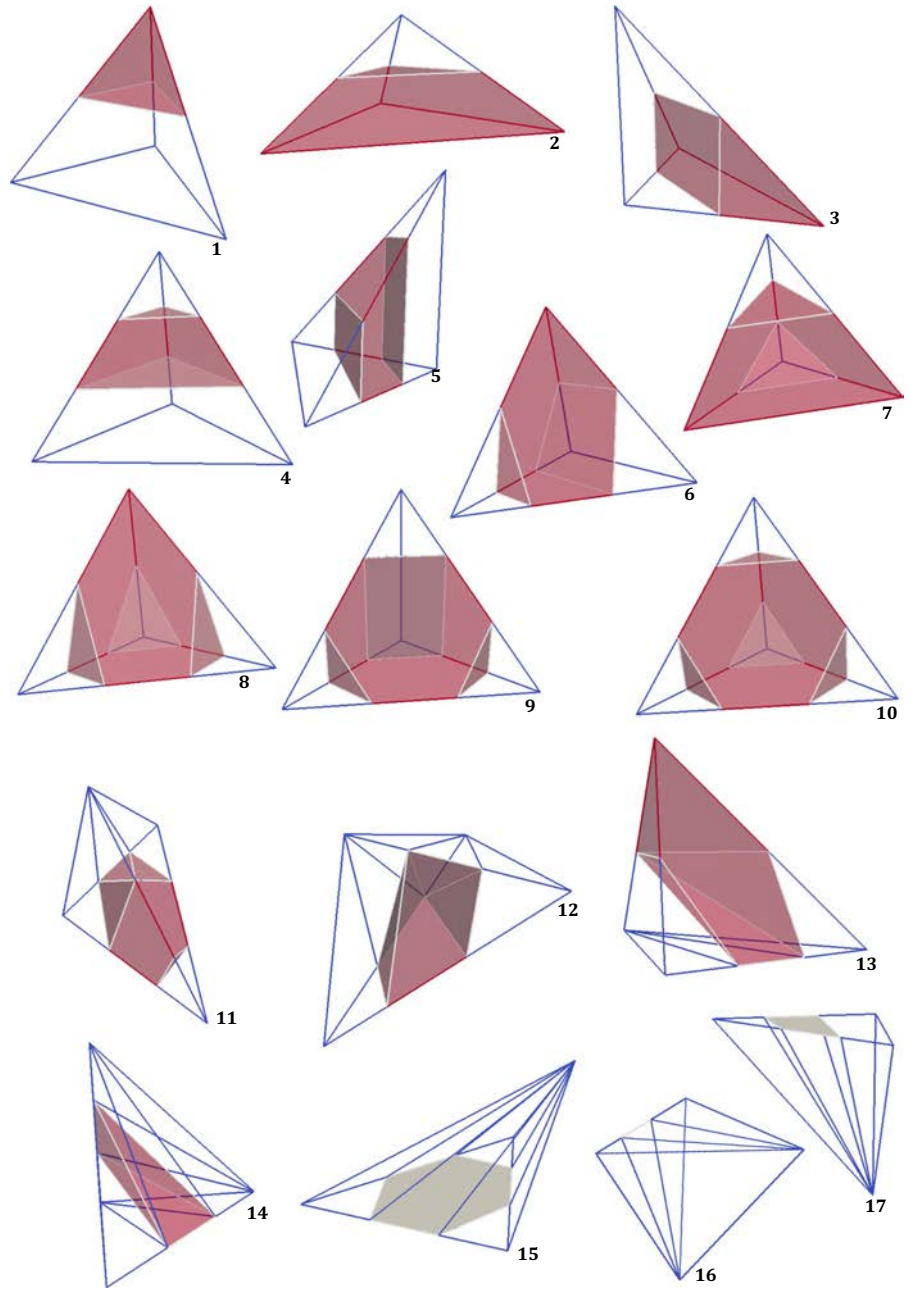
Patterns	Number of possible permutations with or without close node treatment	
	Without	With
1	4	108
2	4	108
3	6	486
4	4	208
5	3	336
6	12	1056
7	6	294
8	4	740
9	6	1488
10	1	545
11	12	1368
12	12	624
13	12	504
14	3	72
15	4	180
16	6	42
17	12	240
Total	111	8399

node treatment handling 8399 permutations. From an implementation point of view, a natural choice is to use a database to hold all those patterns. The cutting procedure reduces then to cut edges and query the database with edge cut pattern as a search key. The database gives then integration cells and topological relations for enrichment identification (number of independent support parts).

#### Comparison between single and double cut algorithms

To illustrate how single and double cut algorithms perform, we consider the surface of a hammerhead shark (from Lutz Kettner's home page at the Max Planck Institute: <https://people.mpi-inf.mpg.de/~kettner/proj/obj3d/> discretized with triangles) plunged into unstructured meshes of an increasing number of elements (see Fig. 32). For each mesh a level set and a signed vector distance function are computed from the shark's surface. Associated cutting algorithms are then used. Fins appear sooner with the double cut algorithm, as

**Fig. 31** Tetrahedron topological cut patterns: *red zone* corresponds to positive domain. On each edge, the *red segments* are in the positive domain, the *blue segments* are in the negative domain, and the *grey segments* are in the iso- $l_c$

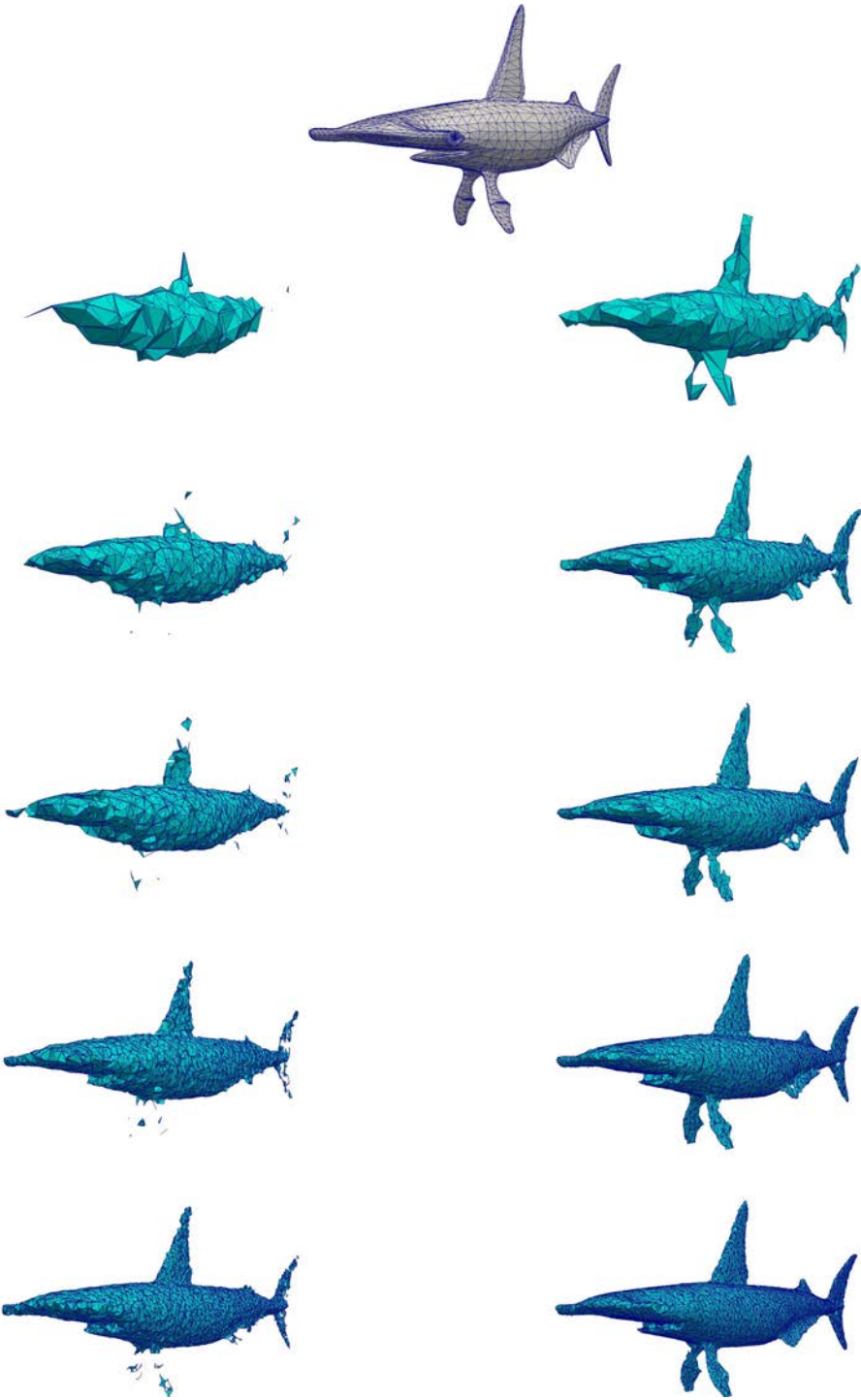


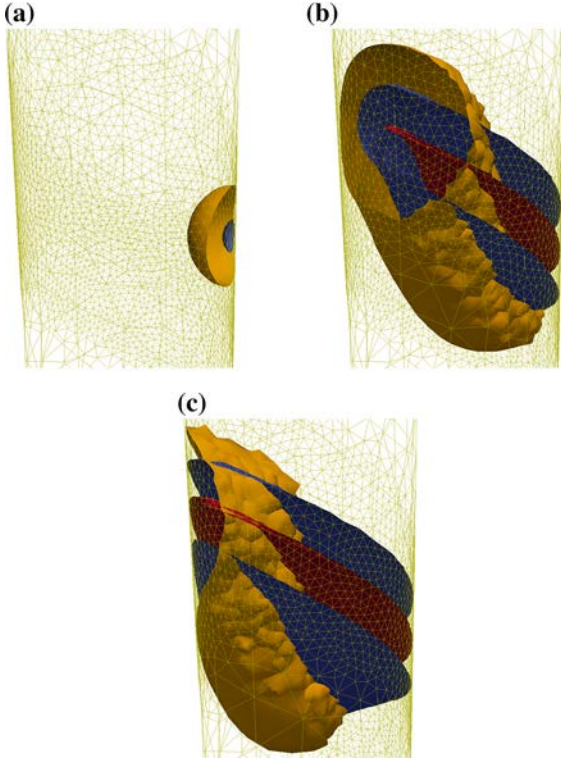
expected since this algorithm detects layers even if they are embedded inside an element. Less expected is that even in rather simple zones such as the shark's body the use of a signed vector distance function gives better accuracy. Two reasons may be pointed out, more accurate cut locations with the SVDF and the possibility of a warped iso-contour inside elements (patterns 3, 5, 6, and 9).

## Appendix 2: Active zone envelope creation

In this work the AZ was constructed by using enrichment information around  $\Gamma_c$ . In Fig. 10 all nodes of elements topologically cut by  $\Gamma_c$  have been analyzed. Blue circle nodes correspond to non-enriched nodes and green stars to enriched ones. The idea is to consider that at crack tips, there always exists at least

**Fig. 32** Iso-contours obtained with a single (*left*) or double (*right*) cut algorithm for a hammerhead shark reference shape (*top*). Meshes used are more and more refined from *top* to *bottom*





**Fig. 33** Chalk test case: in orange the AZ envelope embedding crack tip, in blue  $\Gamma_0$  and in red  $\Gamma_c$ . **a** Simulation start, **b** middle of the simulation, **c** end of the simulation

one element made only of non-enriched nodes. This is not possible in the wake of the crack as  $\Gamma_c$  is always splitting into two (or more) parts, support of nodes. In Fig. 10 identified elements (in cyan) are both used to compute (elementary center of gravity) centers of circle (sphere in 3D) used to localize AZ. The union of those circles is what we call the AZ envelope. Following the above explanation, a proposed method is given in Algorithm 2. It uses  $R_{AZ}$ , a chosen radius, to construct a sphere corresponding to the AZ envelope.

Figure 33 illustrates crack tip tracking by AZ envelope (in orange) at some stages in simulation of the chalk test case.

When searching for damage initiation outside the damaged zone and a small damaged sphere is added, a new AZ must be created to encapsulate this extra zone.

Algorithm 2 provides a good to moderate AZ envelope. It is in default when the  $R_{AZ}$  is too small and  $\Gamma_0$  moves more rapidly than  $\Gamma_c$ . Another problem, observed in tests, is the presence of a zone of crack where lips are not fully detached or a crack region

---

### Algorithm 2 Algorithm used to create AZ envelop.

---

```

for a given  $R_{AZ}$  radius
for each element  $e$  topologically cut by  $\Gamma_c$  do
   $a = 0$ 
  for each node  $n$  of  $e$  do
    compute the number  $k$  of distinct parts of  $n$  support
    if  $k = 1$  then
       $a = a + 1$ 
    end if
  end for
  if  $a$  equals the number of nodes of  $e$  then
    store  $e$  gravity center point in table  $T$ 
  end if
end for
if extra spheres are added by damage initiation search then
  store or delete their center points in table  $T$  if sphere is
  isolated or absorbed by a front
end if
if  $T \neq \emptyset$  then
  create AZ envelope by taking union of spheres with points
  of  $T$  for center and  $R_{AZ}$  for radius
else
  use a shifted iso-zero location for AZ envelope
end if

```

---

including a full element, which induces the presence of points in  $T$  for unnecessary reasons.

### Appendix 3: Modification in stagger algorithm

Use of AZ modifies stagger Algorithm 1 and give new Algorithm 3. The  $\tilde{\cdot}$  represent condensed operator.  $G$  is the expansion operator from condensed to full domain.

---

### Algorithm 3 Modified staggered algorithm scheme.

---

```

Starting from  $\mathbf{u}^i$ ,  $d^i$ , and  $\mu^i$ 
repeat
  Find  $d^{i+1}$  such that  $d^{i+1} = g(\mu^i, \mathbf{u}^i)$ 
  Eliminate fixed zone from problem for any new AZ
  Find  $(\mathbf{u}^{i+1}, \mu^{i+1})$  such that
  
$$\begin{cases} \tilde{K}(d^{i+1}, \tilde{\mathbf{u}}^{i+1}) \tilde{\mathbf{u}}^{i+1} = \tilde{F}(\mu^{i+1}) \\ \mathbf{u}^{i+1} = G\tilde{\mathbf{u}}^{i+1} \\ \max_k (f_k(\mu^{i+1}, \mathbf{u}^{i+1})) = 0 \end{cases}$$

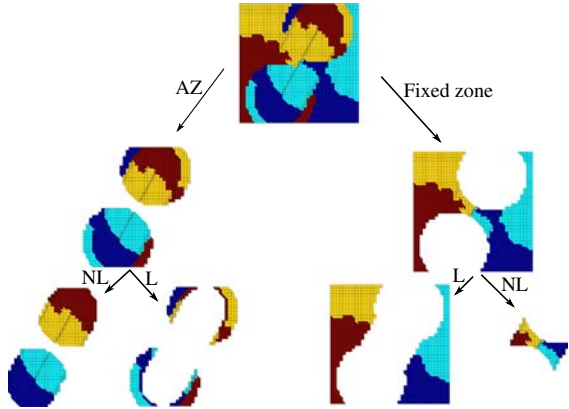
until Complete failure or user given load level

```

---

In Algorithm 3 application of the  $g$  operator must be analyzed to check that  $\Gamma_0$  evolution is only observed in the AZ envelope. Otherwise, a new AZ must be constructed. This is because the nonlinear material law of damaged elements in the fixed zone is assumed to be





**Fig. 34** Partitioning: a four-process example in 2D, each *color* represents a set of elements treated by a process

linearized at the creation of the AZ, and the associated condensed tangent matrix is kept constant during all load steps until the next AZ creation.

#### Appendix 4: Load balancing

With Algorithm 3 four assemblies appear. AZ groups are separated into linear (in  $\Omega_-$ ) and nonlinear (in  $\Omega_+$ ) element groups. Fixed groups are also separated into linear (in  $\Omega_-$ ) and nonlinear (in  $\Omega_+$ ) element groups. Those four groups of elements are then partitioned to obtain a good load balancing for assembly computation.

Partitioning zones are treated with ParaMetis<sup>9</sup> in this work. Figure 34 depicts in a 2D example those four partitioning zones. On the right, the fixed zone is created only when a new AZ is computed and linear and nonlinear assemblies are created independently. On the left, the AZ, here made of two circles around the tips of an oblique crack, is also split for linear and nonlinear independent assembly. During load steps those two partitioning sets are kept unchanged. A mix of them is used, depending on  $\Gamma_0$  location. This is expected to offer a good load balancing during all load steps until the next AZ creation.

A more ad hoc partitioning would be to start from a partitioned  $\Gamma_0$  and by some coloring algorithm expand this initial partitioning to AZ and fixed zone. This would give a priori a more equilibrated partitioning with less frontiers. This would open up the possible

<sup>9</sup> 3.1.1 version.

**Table 4** Test elapsed time in hours (h) or days (d) with four cluster nodes made of four AMD Opteron 6328 at 3.192 GHz (eight cores) and InfiniBand network 4× DDR

Test case	Elapse time	Number of processes
L shape <sup>(6.3)</sup>	6.17 h	16
Chalk <sup>(6.2)</sup>	5.43 h	24
Spherical Holes <sup>(6.1)</sup>	9.25 h	24
Spiral bevel gear <sup>(6.4)</sup>	1.6 d (up to step 246)	48

strategy of splitting the AZ (which grow in 3D) to do a kind of oriented domain decomposition.

#### Appendix 5: Computation time performance

In terms of computation time performance, state-of-the-art elapsed time for all test cases is given in Table 4 except for Sect. 6.5 test case.<sup>10</sup> Elapsed time is a rather blunt measure as it takes into account network bottleneck, un-optimized parts, and operating system loading. But it gives a general idea of time consumption on a modest cluster with the in-house code implemented in C++. Note that the older spherical holes test case was not re-computed with the AZ. The elapsed time given in the table corresponds to a pre-AZ version without condensation.

The AZ was very helpful for the spiral bevel pinion gear test case where the model is large (453,824 nodes) from the start. The gain is especially high for the first load step, thanks to a small AZ envelope. As mentioned in “Appendix 2”, AZ determination is in default in some cases, as in this simulation. Instead of a decreasing envelope size at the end of the computation as the damage front reaches part of the boundary (as in Fig. 33, for example), the AZ increases due to spurious detection. Times given in Table 4 correspond to computations up to load step 246 (see Fig. 25) where the AZ envelope is almost what we expect. To reach load step 385, an extra 4.4 days were used. Again, here the AZ is not ideal during those last load steps, and future work will hopefully reduce the time consumption.

<sup>10</sup> For these notched beam tests, element size ( $l_c/20$ ) is too small for current in-house code implementation and performance is irrelevant (parts not treated by the method exposed in this paper are too expensive).



**Table 5** Spiral bevel pinion gear computation times in hours (the 246 first load steps)

Item	Elapsed time	% of total
Total	37.93	100
Mechanical problem resolution <small>Algorithm I II</small>	23.68	62.42
TLS update <small>Algorithm I I+III</small>	13.12	34.59
AZ creation	0.32	0.84
Other	0.82	2.16

**Table 6** Spiral bevel pinion gear detailed computation times for mechanical problem resolution in hours (the 246 first load steps)

Item	Elapsed time	% of total	% of task
Mechanical problem resolution <small>Algorithm I II</small>	23.68	62.42	100
System creation (Integration + assemble)	0.63	1.66	2.65
Algebraic system resolution	18.68	49.25	78.91
Dof update and results output	2.21	5.82	9.33
Matrix structure creation, parallel task, nonlinear resolution, ...	2.16	5.69	9.11

**Table 7** Spiral bevel pinion gear detailed computation times for TLS update in hours (the 246 first load steps)

Item	Elapsed time	% of total	% of task
TLS update <small>Algorithm I I+III</small>	13.12	34.59	100
$\bar{Y}$ computation <small>Sect. 3.5</small>	5.24	13.8	39.9
Update domain ( $\Gamma_0$ and $\Gamma_C$ creation)	2.72	7.17	20.74
Update $\phi$ and other	5.16	13.61	39.35

For insight into which part of the computation consumes CPU time in a rather intensive simulation, the spiral bevel pinion gear test case was profiled. Results for the first 246 load steps are given in Tables 5, 6, and 7. The first table presents global dispatching of computation time.

First, regarding the TLS update task detailed in Table 7, it participates for 34.59% of the total simulation elapsed time, which is not negligible. The main consumers are  $\bar{Y}$  and  $\phi$  update computation. Neither task is well parallelized. As mentioned in the conclusion, fast marching techniques will replace those com-

**Table 8** Computation times in hours for the L shape test case with different order strategies and 16 processes

Order	Elapse time
1	1.75
2	21.8
Mixed 1/2	6.17

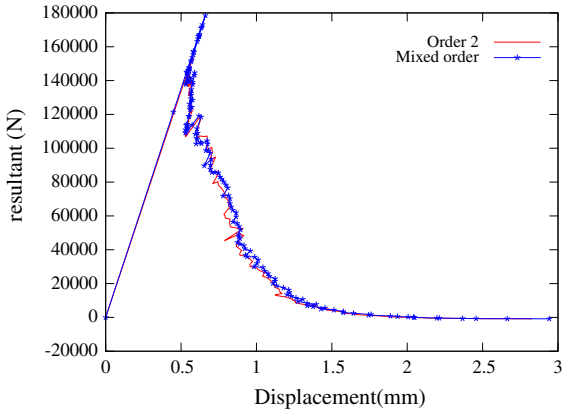
putations and, in particular, the algebraic system resolution attached to them.

Note that the new double cut algorithm proposed in this paper is not a high computation time consumer. It is embedded in the update domain item, which represents only 7.17% of the total simulation elapsed time without being parallelized.

Secondly, regarding global dispatching, effort put into reducing mechanical problem resolution with the use of parallelism, AZ techniques, and mixed order are well justified as this task remains the most important consumer. Detailed in Table 6 it is mostly (78.91%), penalized by algebraic resolution (factorization, condensation, solving, ...). Matrix creation (elementary creation and assembly), despite use of sub-elements for integration (X-FEM technique), are not greedy in terms of time consumption. Parallelism and good load balancing (see ‘‘Appendix 4’’) keeps consumption low. Remaining sequential computation like dof updates will have to be treated in parallel in the future.

Mixed order extra features have been tested on the L shape test case. With the same mesh, 16 processes in all cases, elapsed time given in Table 8 shows big gain (divided by 3.5) of mixed order compared to full order 2, mainly by having a smaller dense problem to create and solve. From a numerical point of view, results are the same as shown by the force-displacement curve presented in Fig. 35. This mixed order strategy will make more sense when used with coarser mesh.

Finally, AZ performances are illustrated again with the spiral bevel pinion gear test case intensive simulation. In Table 9 elapsed times after 80 load steps are given in different configurations. Upper left results correspond to sequential computations with no AZ or condensation. It is more or less the implementation of Bernard et al. (2012) in terms of mechanical problem resolution. It is somehow the starting point of this work. On one hand, introduction of AZ and condensation (upper right) reduces time consumption by 6.35.



**Fig. 35** Mixed order comparison: force displacement curves,  $Y$  component, L shape test case (abscissa is the imposed displacement, ordinate is the resultant force on clamped face)

This by itself validates interest in this technique. On the other hand, applying parallel computation on hot points (lower left) reduces time consumption by 8.56. Then, applying parallel computation on AZ technique (lower right) decreases time consumption by 3.25. If CPU times for mechanical problem computation is extracted from the global time, it is a ratio of 8.08, which is in fact encountered. This illustrates the good choice of parallel technique with condensed resolution. We can add that memory consumption becomes an issue with condensation if no parallel distribution of the dense system

**Table 9** Elapsed time computations for spiral bevel pinion gears (Sect. 6.4) in hours up to the 80th load step. Comparison between sequential and parallel versions with or without AZ. Ratio corresponds to comparison per column, line, and diagonal

	Without AZ	With AZ	ratio
1 process	113.18	17.82	6.35
48 processes	13.22	5.48	2.41
Ratio	8.56	3.25	20.64

is done. In parallel, use of AZ technique reduces time consumption by 2.41, which is less than in sequential, but still interesting. Compared to the non-AZ sequential version, adding both strategies, as proposed in this work, reduces time consumption by 20.64. This last result somehow relativizes the remark above about non-corrected AZ in the last load step of this simulation.

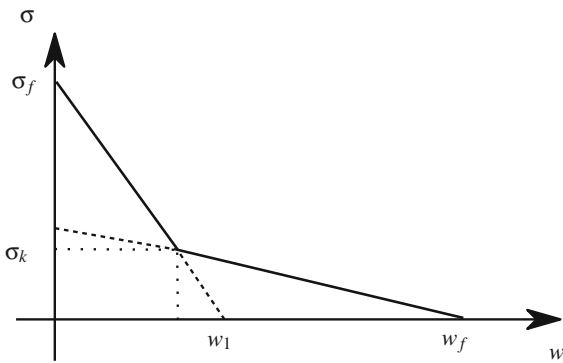
## Appendix 6: Material characteristics and simulation parameters

Material characteristics and parameters of the model used in this article are in Table 10.

For the test case in Sect. 6.5, the critical energy release rate is a function of the damage. This function is constructed by use of an equivalence with a bilinear cohesive zone model has stated in Parrilla Gómez et al.

**Table 10** Simulation parameter set

Test case material	L shape <sup>(6.3)</sup> PMMA	Chalk <sup>(6.2)</sup> Chalk	Spherical Holes <sup>(6.1)</sup> PMMA	Spiral bevel gear <sup>(6.4)</sup> Steel AISI9310	Tree point bending <sup>(6.5)</sup> Concrete
Elastic Parameter:					
Young Modulus ( $N/mm^2$ )	25850	2000	2800	206844	41240
Poison Ration	0.18	0.18	0.38	0.3	0.174
Numerical approximation:					
Order	1	mixed 1/2	1	1	1
Number of nodes	20 809	12 934	92 746	453 824	452 598 (Bc) 171 240(Cb)
Local damage model Parameter:					
$Y_c$ ( $N/mm^2$ )	0.00095	0.0125	0.06	54.83	$Y_c^0 \cdot h(d)$
Traction/compression parameter $\beta$	0	0	1	0	0
TLS Parameter:					
$l_c$ (mm)	100	4	10	0.7	20
$d(\phi)$	$\begin{cases} 0 & \text{for } \phi < 0 \\ \left(\frac{\phi}{l_c}\right)^2 \left(3 - 2\frac{\phi}{l_c}\right) & \text{for } 0 \leq \phi \leq l_c \\ 1 & \text{for } l_c < \phi \end{cases}$			$\begin{cases} 0 & \text{for } \phi < 0 \\ \frac{\phi}{l_c} \left(2 - \frac{\phi}{l_c}\right) & \text{for } 0 \leq \phi \leq l_c \\ 1 & \text{for } l_c < \phi \end{cases}$	



**Fig. 36** Bilinear cohesive zone softening law in terms of stress versus opening displacement

**Table 11** Bilinear cohesive zone softening law coefficients from Hoover and Bažant (2014)

	Values
$\sigma_f$	3.92 MPa
$\sigma_k$	0.588 MPa
$w_1$	25.3 $\mu\text{m}$
$w_f$	94.8 $\mu\text{m}$

(2015). The four extra details mandatory to describe  $Y_c^0 h(d)$  are from a description of the bilinear softening law given in terms of stress versus opening displacement (see Fig. 36). Table 11 gives value fitted by Hoover and Bažant (2014) and used in this paper.

## References

Bažant ZP, Jirasek M (2002) Nonlocal integral formulations of plasticity and damage: survey of progress. *J Eng Mech* 128:1119–1149

Bernard P-E, Moës N, Chevaugeon N (2012) Damage growth modeling using the thick level set (tls) approach: Efficient discretization for quasi-static loadings. *Comput Methods Appl Mech Eng* 233–236:11–27. doi:10.1016/j.cma.2012.02.020 (ISSN 0045-7825)

Bordas S, Rabczuk T, Zi G (2008) Three-dimensional crack initiation, propagation, branching and junction in non-linear materials by an extended meshfree method without asymptotic enrichment. *Eng Fract Mech* 75(5):943–960. doi:10.1016/j.engfracmech.2007.05.010 (ISSN 00137944)

Bourdin B, Francfort GA, Marigo J-J (2008) The variational approach to fracture, vol 91. doi:10.1007/s10659-007-9107-3

Cazes F, Moës N (2015) Comparison of a phase-field model and of a thick level set model for brittle and quasi-brittle fracture. *Int J Numer Methods Eng*. doi:10.1002/nme.4886

Chevaugeon N, Salzman A, Moës N (2014) Vector level set contouring algorithm and associated enrichment functions for the extended finite element method. In: 11th World congress on computational mechanics, Barcelona

Coxeter HSM (1973) Regular polytopes, 3rd edn. Dover Publications, New York

Francfort GA, Marigo J-J (1998) Revisiting brittle fracture as an energy minimization problem. *J Mech Phys Solids* 46:1319–1412

Gomes J, Faugeras O (2003) The vector distance functions. *Int J Comput Vis* 52:161–187

Hachenberger P, Kettner L (2013) 3D Boolean operations on Nef polyhedra. In: CGAL user and reference manual. CGAL Editorial Board, 4.3 edn. <http://doc.cgal.org/4.3/Manual/packages.html#PkgNef3Summary>

Hert S, Schirra S (2013) 3D convex hulls. In: CGAL user and reference manual. CGAL Editorial Board, 4.3 edn. <http://doc.cgal.org/4.3/Manual/packages.html#PkgConvexHull3Summary>

Hoover CG, Bažant ZP (2014) Cohesive crack, size effect, crack band and work-of-fracture models compared to comprehensive concrete fracture tests. *Int J Fract* 187(1):133–143. doi:10.1007/s10704-013-9926-0

Hoover CG, Bažant ZP, Vorel J, Wendner R, Hubler MH (2013) Comprehensive concrete fracture tests: description and results. *Eng Fract Mech* 114:92–103. doi:10.1016/j.engfracmech.2013.08.007

Karma A, Kessler D, Levine H (2001) Phase-field model of mode III dynamic fracture. *Phys Rev Lett* 87(4):045501. doi:10.1103/PhysRevLett.87.045501

Lorentz E, Godard V (2011) Gradient damage models: toward full-scale computations. *Comput Methods Appl Mech Eng* 200(21–22):1927–1944. doi:10.1016/j.cma.2010.06.025

Miehe C, Welschinger F, Hofacker M (2010) Thermodynamically consistent phase-field models of fracture: variational principles and multi-field FE implementations. *Int J Numer Methods Eng*. doi:10.1002/nm.2861

Moës N, Stolz C, Bernard P, Chevaugeon N (2011) A level set based model for damage growth: the Thick level set approach. *Int J Numer Methods Eng* 86:358–380. doi:10.1002/nme.3069

Moës N, Stolz C, Chevaugeon N (2014) Coupling local and non-local damage evolutions with the thick level set model. *Adv Model Simul Eng Sci* 1(1):16. doi:10.1186/s40323-014-0016-2

Moreau K, Moës N, Cazes F (2015) The thick level set approach, towards simulations coupling local and non-local evolutions. In: CFRAC, fourth international conference on computational modeling of fracture and failure of materials and structures, Ecole Normale Supérieure de Cachan (Paris), France

Parrilla Gómez A, Moës N, Stolz C (2015) Comparison between thick level set (TLS) and cohesive zone models. *Adv Model Simul Eng Sci*. doi:10.1186/s40323-015-0041-9

Pijaudier-Cabot G, Bažant ZP (1987) Nonlocal damage theory. *J Eng Mech ASCE* 113:1512–1533

Spatschek R, Brener E, Karma A (2011) Phase field modeling of crack propagation. *Philos Mag* 91(1):75–95. doi:10.1080/14786431003773015 (ISSN 1478-6435)

- Spievak LE, Wawrzynek PA, Ingrassia AR (2000) Simulating fatigue crack growth in spiral bevel gears. Technical Report May 2000, NASA/CR, Glenn Research Center
- Spievak LE, Wawrzynek PA, Ingrassia AR, Lewicki DG (2001) Simulating fatigue crack growth in spiral bevel gears. *Eng Fract Mech* 68(1):53–76. doi:10.1016/S0013-7944(00)00089-8 (ISSN 00137944)
- Stolz C, Moës N (2012) A new model of damage: a moving thick layer approach. *Int J Fract* 174(1):49–60. doi:10.1007/s10704-012-9693-3 (ISSN 0376-9429)
- Ural A, Heber G, Wawrzynek PA, Ingrassia AR, Lewicki DG, Neto JBC (2005) Three-dimensional, parallel, finite element simulation of fatigue crack growth in a spiral bevel pinion gear. *Eng Fract Mech* 72:1148–1170. doi:10.1016/j.engfracmech.2004.08.004
- Van Der Meer FP, Sluys LJ (2015) The Thick Level Set method: Sliding deformations and damage initiation. *Comput Methods Appl Mech Eng* 285(285):64–82. doi:10.1016/j.cma.2014.10.020. www.elsevier.com/locate/cma