Thick Level Set model implementation in 3D parallel context

Alexis Salzman

THESE DE DOCTORAT DE

L'ÉCOLE CENTRALE DE NANTES
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 602
Sciences pour l'Ingénieur
Spécialité : Mécanique des Solides, des Matériaux, des structures et des surfaces

Par
Alexis SALZMAN

Développement de l'approche « Thick Level Set » dans un cadre 3D parallèle

Thèse présentée et soutenue à Nantes le 25/10/2019
Unité de recherche : Institut de Recherche en Génie Civil et Mécanique - UMR CNRS 6183

Abstract

A complex 3D fracture simulation of quasi-brittle material in quasi-static is still hard to tackle nowadays. Many methods and models propose partial solutions to this problem. The Thick Level Set (TLS) model, which uses an approach mixing damage mechanics and explicit crack representation, provides an easy fracture initiation, a complex crack growing capability (coalescing or branching) and an accurate tortuous fracture path. In this thesis we will demonstrate that the implementation of this model in a parallel 3D context provides an accurate and versatile tool that potentially scales.
Regarding the accuracy, a novel tool called the "double cut algorithm" has enhanced the existing TLS implementation by letting pass a straight fully damaged zone in a mesh element without conditions on its size. This tool also brings a way to optimize the discretization by coarsening the mesh in a crack front wake. This adaptation reduces the size of the discrete mechanical problem and therefore the effort for the linear algebra resolution.
As far as the scaling is concerned, the bottleneck is the linear algebra resolution time and its associated memory consumption. The parallel solving strategy developed in this thesis to tackle this problem starts first with a basic approach. Then by switching to a method close to domain decomposition and later to a two-scale method, it permits increasing scalability. The other TLS tasks are also partially parallelized. The principal concern is to obtain a tool that either runs faster or can treat a more significant problem if we provide much more computational units. Finally, some test cases illustrate the obtained results with a parallel 3D implementation.

Acknowledgments

The author is grateful for the research support of the European Research Council through ERC starting Grants ERC-XTLS N°291102. "Ecole Centrale de Nantes" school through the GeM laboratory is also acknowledged for its financial help.
The author also wants to warmly thanks Nicolas Moës who has allowed this work to exist, to come to fruition and to succeed. The whole GeM MS team is also to be associated with this work for their constant help, advice and support.

"We can only see a short distance ahead, but we can see plenty there that needs to be done."

Alan Turing

Contents

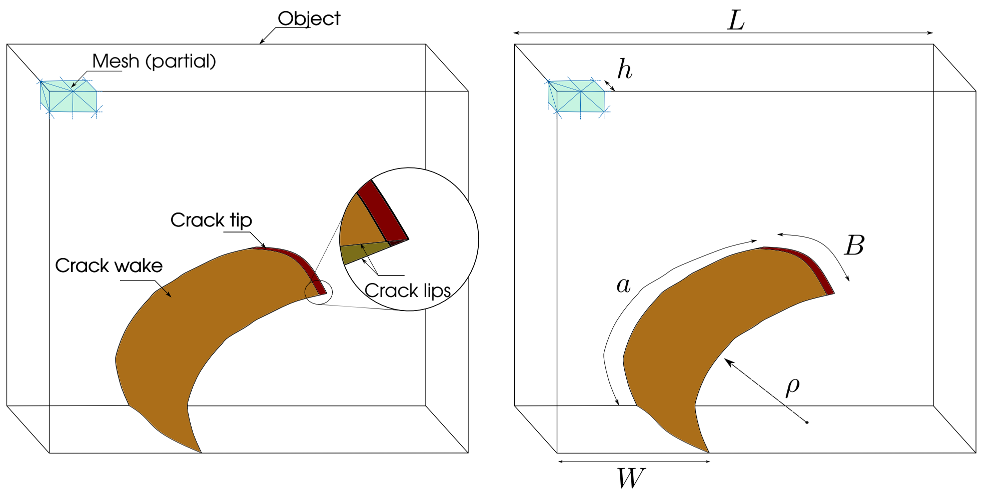
Chapter I
Context and objectives

This thesis was written at "Ecole Centrale de Nantes" at the Gem Laboratory and is related to the complex 3D fracture simulation of quasi-brittle material. The term complex implies here that the simulations can initiate one or more cracks which develop by describing a set of tortuous paths which can coalesce or branch out

Three categories composed of strong, weak or discrete discontinuity approaches cover such kind of simulations where the cracks opening must be represented somehow. The first one, associated with the fracture mechanics is able to reproduce the crack discontinuous displacement behavior by using an explicit discretization of the crack. It, therefore, gives a correct crack path. However, it needs a priori crack initiation and coalescing or branching are hard to achieve. The second, associated with the damage mechanics, gives an easy crack initiation, branching or coalescing. Nevertheless, it needs a regularization and does not provide an explicit crack path (and the associated discontinuous displacement). The third one, related to the discrete element method, uses beam or trust failure to represent the crack which leads to some mesh dependency.

The recent Thick Level Set model used in this thesis sets a bridge between these first two discontinuity approaches by mixing a damage model with an explicit crack representation. Used in the 2D simulation it already offers a valuable tool in terms of results quality and computational consumption. In this thesis the objective is to maintain such quality for a 3D simulation and, in particular, to get a reasonable simulation computation time.

This chapter develops the fracture simulation context and details the meaning of what "maintains such quality" and "reasonable simulation computation time" mean.



(a) Vocabulary (b) Characteristic length

Figure I.1: General fracture problem

I.1 General context

In many situations the robustness of the objects or constructions that one manufactures is a security problem. Sometimes, a crack initiation in a structure is not affordable. Satellite, for example, is not expected during the launching phase to be deteriorated as it cannot be repaired. Sometimes having a way to predict a crack creation and its development in a structure helps to design it. For example, to evaluate the security risk after a blast in an Atomic power plant, one would try to evaluate the leaks of the contaminated fluid that may pass through cracks and then define the structure of the wall in accordance. In other cases, the crack path knowledge may help to set the appropriate reinforcements to avoid these cracks. On the contrary, in some situations one wants to crack objects. For example, in hydraulic cracking, people are eager to optimize water injection to obtain the best-fractured ground. In any case, the cracks and their propagation are phenomena that one wishes to control.

To control such phenomena, one needs to reproduce the cracks to be able to act on them. A perfect tool to achieve this goal is numerical simulation. It reproduces the complex behavior of objects in a low-cost and accurate way. More precisely we will focus on the quasi-static failure simulation of quasi-brittle material in 3D structures.

The general vocabulary and the characteristic lengths usually associated with a fracture problem are presented in figure I.1. The characteristic lengths, given in this figure, help compare the problem dimension together. Characteristic lengths do not represent the exact dimension but only its order of magnitude. L is associated to object sizes. In this object, a crack is described by a and B, respectively associated with its length and its tip width. This crack may develop at a distance associated with W, from the edge of the object. It may open (i.e., its lips may separate from each other) under some loading conditions. It may grow, with crack tips advancing in the material (a and/or B may increase). ρ is associated with the crack path curvature. Finally, h is associated with the size of the spacial problem discretization (illustrated here with a finite element mesh). In this document h will specifically be named h_t, h_w or h_o for respectively mesh size in the crack tip vicinity, mesh size in the crack wake region or mesh size outside both regions.

With these conventions, the following chapter introduces the numerical model retained in this work.

I.2 Fracture and damage mechanics

Fracture modeling implies considering the discontinuity in the displacement field, which is introduced by the cracks (lips separate from each other). The numerical methods to simulate such problems can be classified into three approaches: strong, weak and discrete discontinuity. In the strong discontinuity approach, the crack introduces a discontinuous displacement in the solution field explicitly. In the weak one, the displacements stay continuous but the material strength drops in a region where the crack is expected to be located. It provides a crack opening behavior without a clear identification of lips. In the discrete approach, inherited from the Discrete Element method devised by Cundall and Strack (1979), the failure of the discrete elements, such as lattices of trusses and beams with Voronoi tessellation (Bolander and Saito (1998)), is interpreted as a discontinuous phenomenon (a crack). This last approach is not studied here. However, the two others are the cement for the model used in this work.

The strong discontinuity approach, based on Griffith (1921) and Irwin (1957) fracture mechanics theory uses the finite element method with an explicit cracks discretization. In other words, it means that both crack lips are meshed independently; they may have their own life because of the allowed discontinuous displacement across the crack. A fine discretization to capture singularity is also mandatory near the crack tip. It is possible then to use the assumption of fracture mechanics to compute the crack growth. This assumption is based on the stress intensity factor computed at the crack tip. The two problems related to this approach are:

- The size of the crack compared to the dimension of the object studied is so small ($a \ll L$ and $b \ll L$) that the mesh size needs to be adapted around it. When the crack curvature (ρ) is small, the path also needs a fine discretization. It introduces a considerable meshing effort (with robustness issues) and gives a rather large system to solve.
- When a crack grows, the mesh must be almost completely recreated.

Many strategies were used to minimize those issues. Among them there is the crack box. It corresponds to a strategy where a region (a box) is assumed to encapsulate the crack during all the simulation. This zone is finely discretized to capture the crack behavior and is connected somehow to a coarser mesh. It is supposed to reduce the global problem size and to ease re-meshing by limiting it to the box region. Samcef (from Siemens (2019)) formerly uses "glue" to connect the box to the coarser mesh. Franc3D (from Fracture Analysis Consultants (2019)) creates a box region by extracting a portion of a 3D finite element model. A re-meshing of this portion with mesh boundary constraints is made so that the connection to the model remains identical.

Alternatively, the boundary element method (used in the first Franc3D version) proposes a more straightforward discretization approach. Only the crack and the studied object boundary are meshed. The 3D crack progression is simply associated with a 2D meshing tool. However, the system matrix is dense with this method. It leads to unacceptable linear algebra computational effort.

Still chasing for a solution without meshing issues, researchers propose meshfree methods. In those techniques (Belytschko et al. (1995), Bordas et al. (2008)), the use of element-free Galerkin methods gives among other things, an easy way to propagate a crack by setting moving particles to follow the crack path.

Nevertheless, the real jump in performance was introduced by the use of the eXtended Finite Element Method (X-FEM) in Moës et al. (1999)). The definition of the crack no longer relies on a fine mesh discretization. The mesh is now of a uniform fixed size, and the level set tools (J.A. Sethian (1999)) are used to represent crack and place it where it has to be in the object. Enrichment technique is then used to describe the discontinuity (crack opening) and the particular strain field at the crack tip. The crack growth, still based on fracture mechanics, is achieved only by moving the level set with the mesh remaining fixed. The main drawback is that the crack initiation needs to be done manually (there must be an existing crack somewhere to start the computation). Moreover, the crack coalescence or branching is hard or even impossible to achieve.

The weak discontinuity approach associates, on its side, irreversible microscopic phenomena such as micro-cracks or micro-voids to the material rigidity loss. It is done through the use of an internal variable, which represents a micro-default density. In its scalar form[1], called further d, it ranges from 0 (sound material) to 1 (completely damaged material without any rigidity). For instance the free energy density of linear elastic damage law can be written using Hooke's law tensor $\mathbf{C}$, strain tensor ϵ and d as follows:

$$\varphi(\epsilon, d) = \frac{1}{2}(1-d)\epsilon : \mathbf{C} : \epsilon \tag{I.1}$$

The damage energy release rate Y associated with this potential ($Y = -\frac{\partial\varphi}{\partial d}$) is then used to settle the evolution laws that are used to make damage evolving in the studied part. Compared to the strong discontinuity approach, the damage mechanics provide an easy way to initiate a crack from a sound situation. It also gives access to a complex path with coalescence or branching. However, this local treatment of the damage induces some spurious localizations with a pathological mesh dependency. To overcome this problem, different models

[1]only this scalar form will be considered in this work

were proposed to regularize the damage evolution. They share the introduction of a characteristic length that tunes the averaging process used for regularization.

Integral-type or gradient-type model families are commonly proposed in the scientific literature to smooth the quantity Y. The former, the integral-type family, proposes a spatial weighted averaging process at the studied material point. It is based on the integration of some weighted quantity, which is performed over a domain of fixed sizes. The possible smoothed quantities can be directly Y or the associated strain as proposed in Pijaudier-Cabot and Bažant (1987), Bažant and Jirasek (2002). The latter, the gradient-type family, devised by Peerlings et al. (1998), is based on the use of an averaged Y implicit gradient formulation. This regularized Y is obtained from a solving of a partial differential equation where Y is the source term and where arbitrary null boundary condition is imposed. In terms of the computational cost, the former adds an extra non-local averaging task for each integration at Gauss point and the latter adds a resolution of an auxiliary Laplacian problem.

Other approaches add to the free energy, a term based on gradient of damage (Frémond and Nedjar (1996), Lorentz and Godard (2011)) that penalizes the sudden variations of damage.

To conclude this non-exhaustive review, a more recent model assimilates d to a crack phase-field such as in Karma et al. (2001), Miehe et al. (2010). The regularization of d is transformed into a minimization problem of a diffusive crack topology.

The TLS model has taken the best of both discontinuity approach. Like in damage mechanics, it uses a damage variable d but it considers its treatment in a configurational mechanics manner. A surface (front) separates sound from damaged material. The damage evolution is then linked to the front movement. It leads to compute Y on this front. Furthermore, to again avoid spurious localization on the front, a regularization such as provided in the integral-type model, performs a Y weighted averages on segments[2] originated from front. The length of the segments is not fixed during time and can evolve from zero to a maximum length l_c. The orientation of these segments is al-

[2]whatever the dimension of the body

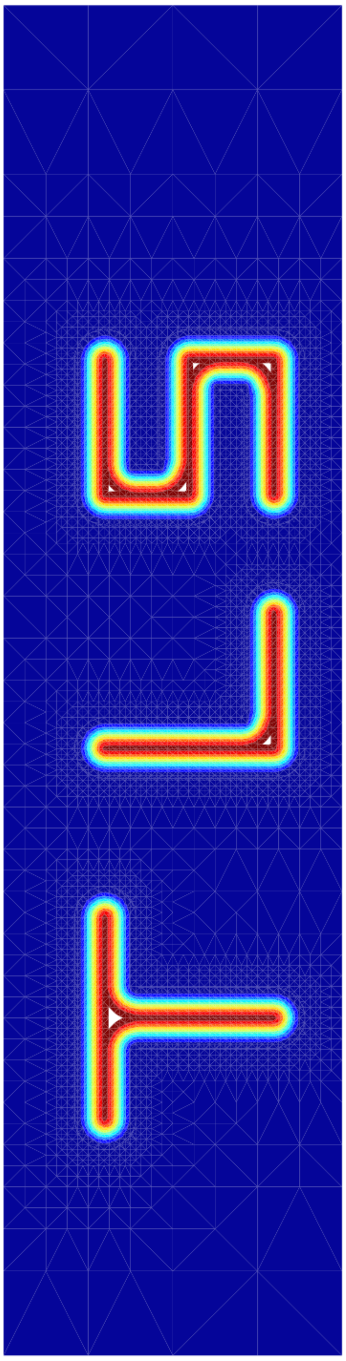
ways aligned with the damage gradient. This weak discontinuity treatment provides to the TLS model, its capacity to initiate a damaged zone from a sound situation. The damage progression, linked to the front progression, is made very versatile (tortuous path, merging, forking, ...). Moreover, the model also adds automatically crack lips in the damaged zone where $d = 1$. Using the enrichment techniques from the strong discontinuity approach, those lips become independent. It adds to the model a precise discontinuous displacement location that lacks in damage mechanics. This model is detailed in the dedicated chapter II.

I.3 Thesis objectives and outlines

The global goal of this thesis is to illustrate the capabilities of the TLS model in the quasi-static 3D simulation context of quasi-fragile materials. From an implementation point of view, switching from 2D to 3D has not been a significant effort.The TLS library created during the work of Bernard et al. (2012) and incorporated in eXlibris later on, was from scratch designed to deal with 2D or 3D problems. However, due to 3D discretization, computation consumption becomes an issue. Behind this computation consumption, the causes need to be distinguished from the consequences. The causes are the discretization choices induced by the model, which can be optimized (which mesh size, where to adapt, which polynomial order to choose for interpolation and where to apply it, ...). The consequences are the resolution method choice to handle very large system of equations. In this work, both are addressed (causes in chapters III and IV, consequences in chapter V).

In both cases, the parallel paradigm needs to be considered. Parallelism is nowadays provided by almost all computational hardware. From a laptop to an enormous cluster computer, it offers a way to reduce simulation elapsed time and to overpass memory consumption issues. This thesis is focused on the scalability of the TLS problem resolution. The strong scaling will indicate if significant problems are achievable in a reasonable time with the parallelism. And the weak scaling will indicate if more significant problems can be solved using the parallelism.

The outlines of the thesis are the following:
Chapter II starts by describing the TLS model used in the context of this thesis. **Chapter III** introduces an improvement of the description of this model based on the "double cut" algorithm. This tool meets the thesis goal by opening the spatial discretization, treated in **chapter IV**, to coarser meshes. The main TLS resolution aspects are treated in **chapter V**. Then a set of test cases in **chapter VI** provides an insight on results that can be obtained in a quasi-static quasi-fragile 3D simulation with TLS model using some of the numerical improvements proposed in this thesis. Finally, **chapter VII** is the conclusion of what has been demonstrated. It also describes as a prospect, what remains to be done to value this work.



Chapter II
The TLS model a bridge between fracture and damage

The Thick Level Set model sets a bridge between fracture and damage mechanics by mixing a damage model with an explicit crack representation. The former offers complete freedom on how crack may initiate, where it progresses or forks and how it coalesces. The latter, using enrichment techniques, adds a strong discontinuous behavior for the crack opening .
This chapter presents the TLS ingredients:

- ϕ the signed distance function used, with the help of level set tools, to split the domain in sound, damaged or fully damaged region,
- $d(\phi)$ the function provided by the user to associate d damage variable to ϕ,
- l_c the characteristic length that is used, such as in other damage models, to regularize the damage energy release rate but here, only along the ϕ gradient segment. It is also used to automatically place the crack lips when they do exist, giving support for the enrichment and the discontinuous behavior.

In this chapter, the governing equations are also provided, showing that d is used in a rather conventional way, with the asymmetric constitutive law from Mazars (1986). Moreover, as mentioned above, the damage evolution law is regularized to avoid some spurious damage localization. In the end, the resolution scheme is summarized in the presented staggered algorithm.

This chapter slightly modified in this thesis have been published in Salzman et al. (2016)

II.1 The Thick Level Set model components

In the TLS framework, as detailed in Moës et al. (2014), the domain Ω of interest is decomposed into three zones (possibly empty): a zone Ω_- in which damage, d, evolves in a local manner, a localizing zone Ω_+ in which the evolution is non-local and, finally, a zone Ω_c in which the material is completely damaged $(d = 1)$. Nevertheless, in this thesis, we consider constitutive damage models without hardening that leads to a damage localization as soon as the damage starts. So, the damage is always zero in Ω_-. This choice is done in Bernard et al. (2012) and has been used in most of our TLS model implementation.

The interface between Ω_+ and Ω_- is denoted Γ_0 and it is located by the iso-zero level set of a signed distance function ϕ (counted negative in Ω_-). The "distance function" characteristic is achieved by :

$$\|\nabla\phi(x)\| = 1 \tag{II.1}$$

The damage is then expressed in terms of ϕ by a scalar function.

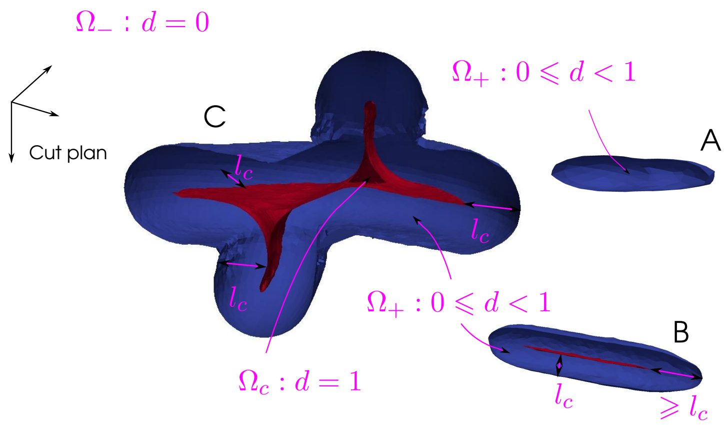
$$d = d(\phi(x)) \tag{II.2}$$

The function $d(\phi)$ is a material data that must satisfy a set of requirements, among which the fact that the damage reaches one, only beyond a distance l_c. The interface between Ω_+ and Ω_c is thus the iso-l_c[1] and is denoted Γ_c.

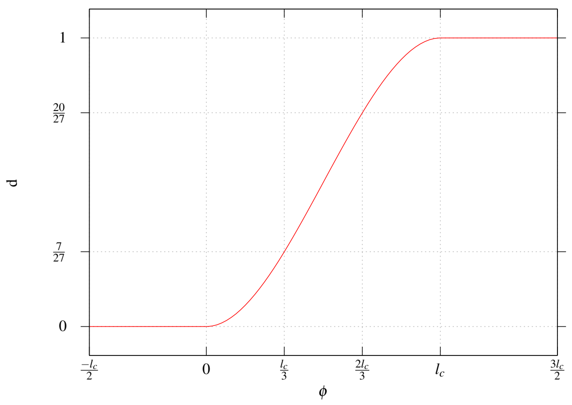
The use of a level set is a powerful aspect of the method which offers support for complicated Γ_0, with potential branching and coalescence, as it will be seen in chapter VI.1. Another powerful aspect is the introduction in the model of this characteristic length l_c which achieves two goals:

- The automated introduction of a crack, considering the iso-l_c values of the level set, as the crack lips. In this thesis a more accurate discretization of Γ_c is proposed in chapter III.
- The introduction of a maximum length over which an averaging of damage is performed.

[1] The iso-l_c is determined by a $-l_c$ offset of ϕ iso-zero.



(a) Domain and interface definitions: in blue Γ_0 , in red Γ_c



(b) Damage function $d(\phi)$ used in four cases of chapter VI: $\left(\frac{\phi}{l_c}\right)^2\left(3-2\frac{\phi}{l_c}\right)$ for $0 \leqslant \phi \leqslant l_c$

Figure II.1: Model parameters.

Figure II.1a shows the above-defined ingredients. It represents different situations, using a planar cross-section to see inside Ω_+, where:

- A is a damage initializing zone where all points are at a distance less than l_c from Γ_0 (in blue) and for which $d < 1$.
- B is a simple crack represented by Γ_c (in red). In this case, one crack tip is located at a distance greater or equal than l_c from Γ_0 tip. It illustrates a complex damage growth pattern in front of the crack tip.
- C is illustrating a branching situation where the crack first develops "horizontally"[2] and then moves "vertically"[2] after a change of the loading conditions.

[2]in figure perspective

The damage shape function $d(\phi)$ must be

- increasing,
- continuous,
- bounded in $[0, 1]$,
- null outside the damage zone, in Ω_- (when $\phi(x) \leqslant 0$),
- equal to 1 in the fully damaged zone Ω_c (when $\phi(x) \geqslant l_c$)

Specific choices of $d(\phi)$ have permitted some comparisons with the phase-field method in Cazes and Moës (2015) or with the cohesive zone model in Parrilla Gómez et al., 2015. The chosen $d(\phi)$ function of four cases in chapter VI is depicted in figure II.1b.

II.2 Governing equations

As in Bernard et al. (2012) work the asymmetric constitutive law from Mazars (1986) is used here. This asymmetric behavior is mandatory to avoid damage growth in the compression direction. In this thesis, however, we do not deal with contact on the crack faces.

In Ω, the equilibrium and kinematics equations are:

$$\begin{cases} \nabla.\sigma = 0 \; on \; \Omega \\ \sigma.\vec{n} = \vec{f} \; on \; \partial\Omega^N \\ \epsilon = \frac{1}{2}\left(\nabla\vec{u} + (\nabla\vec{u})^t\right) \; on \; \Omega \\ \vec{u} = \vec{\bar{u}} \; on \; \partial\Omega^D \end{cases} \tag{II.3}$$

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

The stress and the damage energy release rate Y are derived from the free energy density as follows:

$$\begin{cases} \sigma = \frac{\partial\varphi}{\partial\epsilon} \\ Y = -\frac{\partial\varphi}{\partial d} \end{cases} \tag{II.4}$$

with:

$$\varphi(\epsilon, d) = \frac{\lambda}{2}(1 - \alpha d)\, tr(\epsilon)^2 + \mu\sum_{i=1}^{3}(1 - \alpha_i d){\Lambda_i}^2 \tag{II.5}$$

where:

- λ and μ are the Lamé elasticity coefficients
- Λ_i are the eigenvalues of the strain tensor
- α_i and α are coefficients introduced to take into account an asymmetric behavior in traction/compression:
  - $\alpha_i = \beta \; if \; \Lambda_i < 0$
  - $\alpha_i = 1 \; if \; \Lambda_i \geqslant 0$
  - $\alpha = \beta \; if \; tr(\epsilon) < 0$
  - $\alpha = 1 \; if \; tr(\epsilon) \geqslant 0$
- β is a parameter, bounded in $[0, 1]$, to drive the asymmetry.

When β is equal to 1, α_i and α terms are always equal to 1 and the free energy density is then a linear elastic damage potential (see (I.1))

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

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

$$Y \leqslant Y_c, \; \dot{d} \geqslant 0, \; (Y - Y_c)\, \dot{d} = 0 \tag{II.6}$$

is regularized into its non-local counterpart:

$$\bar{Y} \leqslant \bar{Y_c} \tag{II.7a}$$
$$\dot{d} \geqslant 0 \tag{II.7b}$$
$$\left(\bar{Y} - \bar{Y_c}\right) \dot{d} = 0 \tag{II.7c}$$

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

An averaging operator is introduced to fulfill a non-local estimation of the energy (critical energy) release rate on the damage front. By construction, (II.1) with (II.2) impose damage along the ϕ gradient segment in Ω_+ (figure II.2). Thus, any point of Ω_+ may not have their damage value modified without changing the damage values of all gradient segment points belonging to it. It

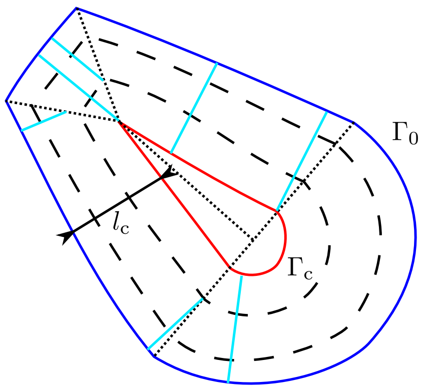


Figure II.2: graphical representation of $\bar{Y}$ (in 2D for better understanding but all principles also hold in 3D) : In the small dashed line, ϕ skeleton. In the large dashed line, some intermediate iso-values of ϕ. In blue Γ_0. In red Γ_c. Cyan segments are aligned with the gradient of ϕ. Over these segments, $\bar{Y}$ is uniform and is the average of the underlying Y field.

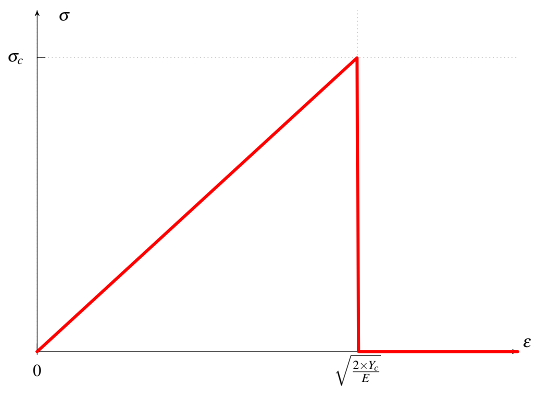


Figure II.3: Abrupt stress/strain local response considered in test cases VI.1, VI.2, VI.4 and VI.5.

introduces a natural averaging process for all points related by the condition (II.1). $\bar{Y}$ ($\bar{Y_c}$) regularizes Y (Y_c) by using $\nabla\phi$ to drive operation as well as a weighting function to smooth the computed value.

The quantity Y_c can be a constant (as in figure II.3) or can depend on damage; in this last case, it is expressed as $Y_c^0\ h(d)$ where Y_c^0 is a constant and $h(d)$ a softening function.

II.3 Staggered algorithm

The TLS solver used for this work is given by algorithm 1 where μ^i is the load factor corresponding to load step i.

Algorithm 1 Schematic staggered algorithm scheme.

```
Starting from $\vec{u}^i$, $d^i$, and $\mu^i$
repeat
    Find $d^{i+1}$ such that $d^{i+1} = g\left(\mu^i, \vec{u}^i\right)$    (I)
    Find $\left(\vec{u}^{i+1}, \mu^{i+1}\right)$ such that
    $\begin{cases} K\left(d^{i+1}, \vec{u}^{i+1}\right)\vec{u}^{i+1} = F\left(\mu^{i+1}\right) & (II) \\ \max\limits_k \left(f_k\left(\mu^{i+1}, \vec{u}^{i+1}\right)\right) = 0 & (III) \end{cases}$
until Complete failure or user given load level
```

The g operator is related to the damage growth model (II.7). The K and F operators are related to the structural equilibrium. Finally, the f_k operators are related to the damage criteria (II.7a).

In algorithm 1, (I) (corresponding to (29) in Bernard et al. (2012)) uses an explicit resolution (as it is based on regularized quantities, the averaging operator resolution[3] has to be added to this step). In (II) the solution $\vec{u}^{i+1}$ is obtained through a Newton-Raphson algorithm, because of the non-linear nature of K. The condition (III) is solved with an explicit resolution. Solving (III) is eased by the fact that $\vec{u}^{i+1}$ linearly depends on μ^{i+1} (as long as it does not change sign).

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

The solution $(\vec{u}^i, d^i, \mu^i)$ at any load step is thus in equilibrium (in the finite element sense) and fully satisfy (II.7a) and (II.7b) even if (II.7c) is slightly violated (see Bernard et al. (2012) for discussion).

The user does not give the load step. It is part of the solution process. Only the maximum damage (or ϕ to be more precise) increase over each load step is provided. This increase falls within the definition of g. The g operator is needed both to grow the existing damage zone and to initiate the

[3]Averaging operator computation was not examined in this thesis despite that the slightly modified version of Bernard et al. (2012), used in chapter VI simulations, adds two linear system to solve. In Moreau et al. (2015) the introduction of an explicit modal space replace one linear resolution by a projection. And in Moreau et al. (2017), the use of Fast Marching techniques remove the other linear resolution. So in this work, even if this averaging operator resolution does have a computational cost, the use of those last techniques will hopefully alleviate it.

new ones[4]. Regarding the initialization of the staggered scheme, we find the elastic loading for which the damage occurs first. It gives $(\vec{u}^0,\, d^0 = 0,\, \mu^0)$. If needed, a non-zero initial damage field may be prescribed.

In algorithm 1, we have revealed an explicit dependence of K on $\vec{u}^{i+1}$. Indeed, the asymmetric damage model, introduced in (II.5), yields a non-linear elastic problem. This dependence of K with respect to $\vec{u}^{i+1}$ is, in fact, very limited in space since it affects only Ω_+. Thus, the tangent matrix used to solve structural equilibrium is implemented, to spare CPU time, as two parts. A fixed part, which is assembled once per load step. And a varying part, which is reassembled at each Newton-Raphson iteration. The same strategy is also used for the Newton-Raphson residual vector, where the linear part of the internal forces is simply obtained by multiplying the fixed tangent matrix part by $\vec{u}$. This early optimization does not change, as presented in chapter V.1, the fact that the non-linear problem resolution is the major computational bottleneck in 3D. The chapter V will address this problem.

Finally, the finite element mesh requirement for this algorithm in 3D is discussed in chapter IV.

[4]In the current implementation, the initiation corresponds to the introduction of a new damaged location. It is made by merging ϕ with a level set which is describing a little sphere encapsulating the new default.

Chapter III
The double cut algorithm or the art of describing a complex crack pattern

On the one hand, the level set method allows representing a curve (2D) or a surface (3D) using an iso-value of a function. Generally this function, which is discretized as a field on the studied mesh, is used to identify a possible iso-value position over all edges. Those points that are also called edge cut gives, the iso-curve or iso-surface mesh according to an element pattern meshing strategy. On the other hand the Thick Level Set model uses such iso-curve/surface (iso-l_c level set obtained from signed distance function ϕ) to represents the crack lips. Those lips can be very close compared to the element size and can cut twice an edge. In this case standard level set technology is faulty since it provides only one cut per edge.
The new approach presented in this chapter allows cutting an edge twice. Firstly, a versatile tool, the vector distance function, that comes from the work of Gomes and Faugeras (2003) is reused. In this work a signed form of this tool is proposed and leads to the signed vector distance function concept (abbreviated to SVDF). It introduces extra information compared to the standard level set tool. Secondly, adapted to SVDF, a new algorithm, the "double cut" algorithm or DCA, gives a way to cut an edge twice. Associated with a database of element cut pattern (which is introduced for computation performance), this new algorithm allows passing a fully damaged zone in a mesh element without any condition on its size.
It leads to a crack lips representation quite insensitive to the mesh size, allowing one to discretize over bigger elements.

This chapter slightly modified in this thesis have been published in Salzman et al. (2016)

III.1 Double vs single cut algorithm

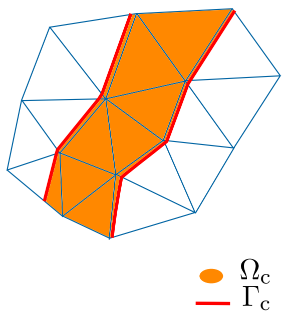
Usual treatment of the level set in the context of XFEM starts with the determination of the iso-zero location. This definition is then used to generate the integration cells (called sub-element hereafter) embedded in the elements crossed by the iso-zero. Depending on the type of problem, some or all those sub-elements are used to integrate the problem using the approximation of elements crossed by iso-zero. Some additional enrichment aspects may also be added and depends on this initial treatment. In the TLS framework this scheme is followed for both Γ_0 and Γ_c. In this chapter we are going to focus on the iso-l_c improvement which represents the crack discontinuity.

From a numerical point of view, if no enrichment is introduced around the iso-l_c, the crack lips will move apart until an element is entirely in the damaged zone (figure III.1a). It allows obtaining the expected discontinuous displacement by having a fully damaged element inside the lips of the crack. However, depending on the size of the elements, more energy can be dissipated than wanted since the fully damaged zone depends on the mesh definition.

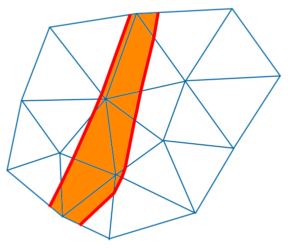
To avoid this issue, the ramped Heaviside enrichment described in Bernard et al. (2012), offers a numerical mechanism to introduce the displacement discontinuity by adding extra degrees of freedom. These latter are added along the iso-l_c of entities, having their support divided into two parts at least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

Since the level set is defined by algebraic values at the mesh nodes, all what can be done with them on an edge is their linear interpolation to find cuts. It implies that edges can only be cut once by the iso-l_c. This restriction implies that the elements can not be cut in two parts by the fully damaged zone. In consequence, the enrichment will only occur if at least two elements are cut by the fully damaged zone (one element per crack lip). It imposes that the crack zone grows (or shifts) enough to start having a discontinuous displacement. It can again dissipate more energy than in reality (or can lead to a wrong crack path) if the mesh size is too big (figure III.1b).

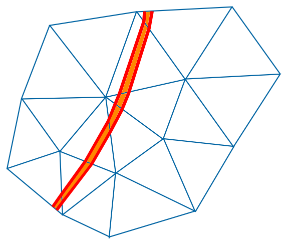
The new approach allows passing over the restriction imposed by the standard level set method. The vector distance function, coming from the work of Gomes and Faugeras (2003), is reused and presented in a signed form, the signed vector distance function, in chapter III.2. It introduces at mesh nodes, a vector vision instead of a scalar vision of the mathematical function used



(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

Figure III.1: Location of Ω_c (orange) depending on the technique used to construct Γ_c and enrichment scheme.

to define the level set. It allows with the new algorithm called the "double cut" algorithm (DCA hereafter) to cut an edge twice. This allows a fully damaged zone to cut a single element (figure III.1c). This double cut algorithm first cuts all edges and then splits all elements. It is explained in chapter III.3.1 and III.3.2.

III.2 The Signed Vector Distance Function concept

Given a manifold, a vector distance function (VDF) gives at each surrounding point a vector pointing out 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 chapter is Γ_c.

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

In the TLS framework, the SVDF replaces the offset level set for the definition of the iso-l_c. Indeed, the offset level set technology can not find multiple cuts within an element as depicted in figure III.1c.

In this context no intrinsic evolution of the SVDF is considered as it is the case in Gomes and Faugeras (2003). It is only used as an enhanced tool to represent the Γ_c manifold, reset at every load step using a new ϕ definition. It follows the same sign convention as the level set it replaces (i.e. positive for fully damaged zone and negative for all other parts of the domain). The SVDF can be constructed for each mesh node, from the level set that defines the front, by the following steps, illustrated in Figure III.2:

- Find the closest point (CP) on Γ_0 from the node (N).
- Compute the vector $\overrightarrow{V_{cp}}$ from N to CP.
- From this vector, construct the vector $\overrightarrow{V_{lc}} = \frac{sign\ l_c\ \overrightarrow{V_{cp}}}{\left\|\overrightarrow{V_{cp}}\right\|}$ 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 $\overrightarrow{V_{svdf}}$ for N is then $\overrightarrow{V_{cp}}+\overrightarrow{V_{lc}}$ starting from N and ending at the point called L.
- Signed vector distance function sign for N is "+" if level set sign for N is "+" and $\left\|\overrightarrow{V_{cp}}\right\| > lc$. Otherwise, it is "-".

This procedure is the one used in this thesis. However, we can notice that a fast marching tool, introduced in eXlibris recently by Nicolas Chevaugeon,

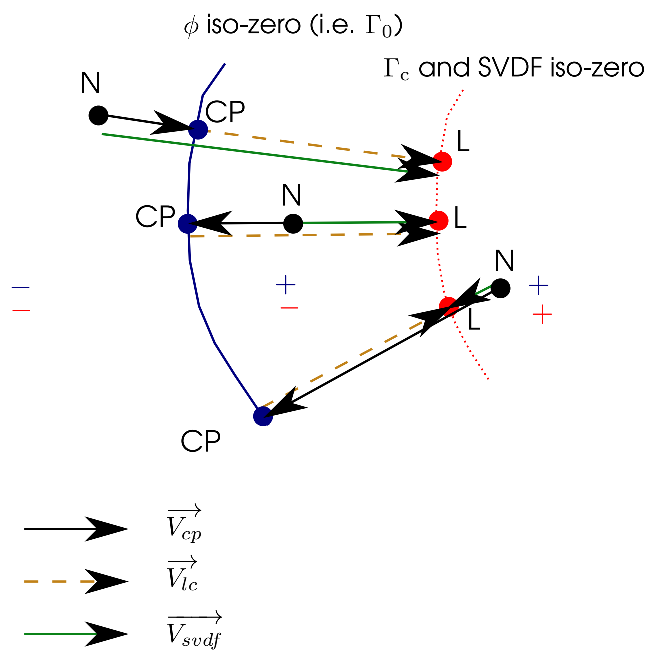

ϕ iso-zero (i.e. Γ_0)
Γ_c and SVDF iso-zero
N
CP
L
CP
N
L
N
L
CP
$-$
$-$
$+$
$-$
$+$
$+$
$\overrightarrow{V_{cp}}$
$\overrightarrow{V_{lc}}$
$\overrightarrow{V_{svdf}}$

Figure III.2: Signed vector distance function construction from the iso-zero front. Vectors are superposed but to distinguish them in this figure, they are drawn shifted if needed. It is a flat representation but this construction must be understood as carried out in 3D space.

gives a way to obtain the SVDF more accurately using ϕ and its gradients computed during the marching.

III.3 The Double Cut Algorithm (DCA)

III.3.1 First step of DCA: cut mesh edges

From the SVDF information, one first has to compute the Γ_c cutting points for all mesh edges. The conditions for having an edge AB cut, come from a logical consideration of the position of the edge with respect to the parts of the domain (table III.1).

The algorithm uses then the following general geometrical predicates:

- Geometrical SVDF iso-zero passes through the ending point L.
- Geometrical SVDF iso-zero tangent plane at point L is orthogonal to $\overrightarrow{V_{svdf}}$.
- Geometrical SVDF iso-zero tangent plane at point L corresponding either to node A or B of an edge may cut it.

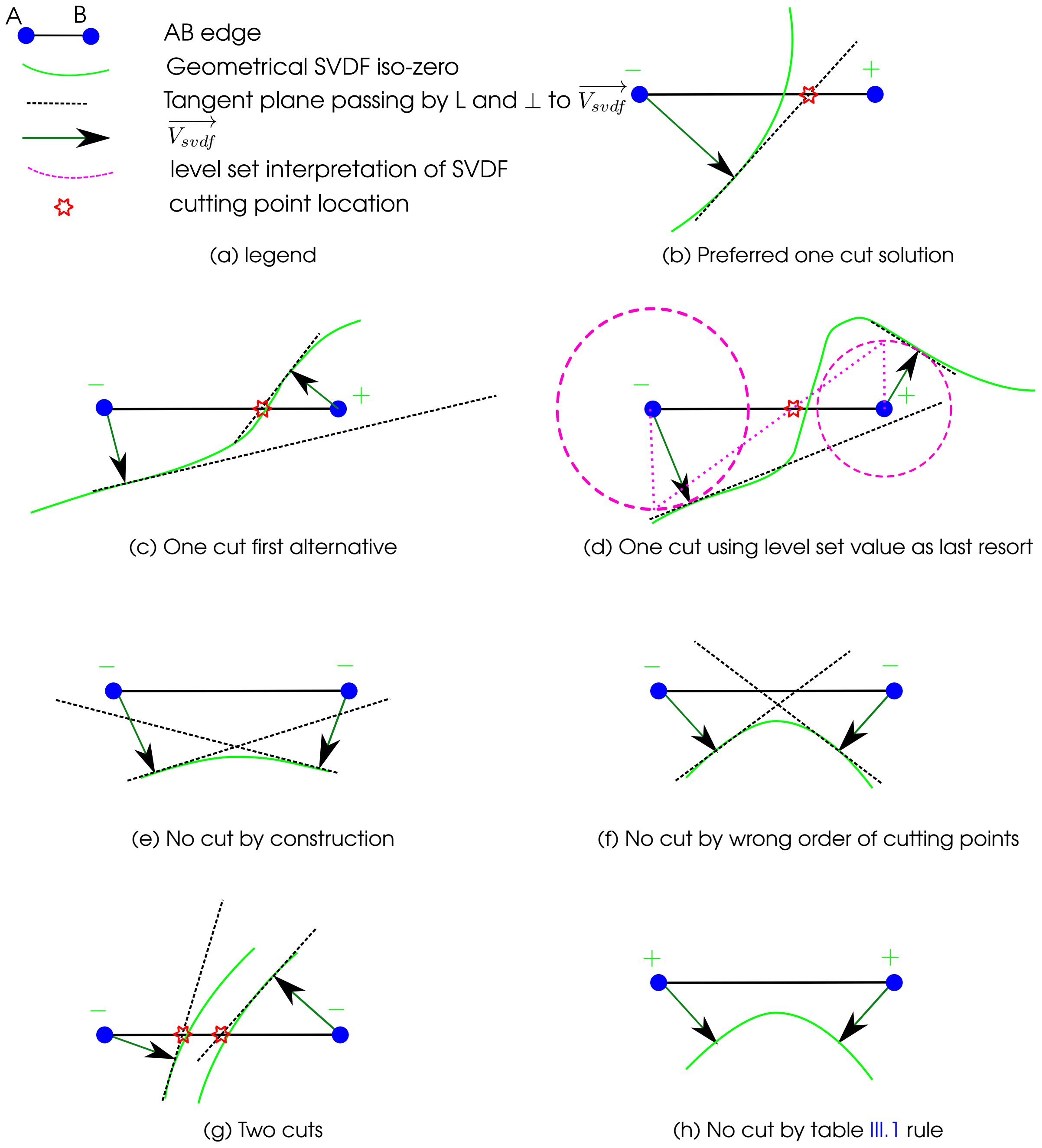

A
B
AB edge
Geometrical SVDF iso-zero
Tangent plane passing by L and $\perp$ to $\overrightarrow{V_{svdf}}$
$\overrightarrow{V_{svdf}}$
level set interpretation of SVDF
cutting point location
(a) legend
$-$
$+$
(b) Preferred one cut solution
(c) One cut first alternative
(d) One cut using level set value 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 III.1 rule

Figure III.3: Edge cut construction illustrating the cut point determination. Representation is 2D but it can be extended to 3D.

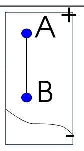
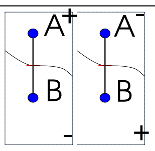
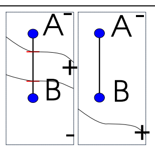
Positions of A and B			
	A and B are in the positive domain (++)	A and B are in two separate domain (+-/-+)	A and B are in the negative domain (- -)
Number of cuts	0	1	0 or 2
Illustration of figure III.3	III.3h	III.3b, III.3c, III.3d	III.3e, III.3f, III.3g

Table III.1: Possible cut cases for an edge AB.

The algorithm to obtain the discrete SVDF iso-zero points on edges (i.e. Γ_c points) is based on these geometrical predicates. Tangent plane intersection(s) is(are) considered as SVDF iso-zero point(s) under the following conditions illustrated in Figure III.3 (taking also into account the logical consideration of table III.1) :

- one cut cases (+-/-+)
  - The cut is first identified with the negative SVDF information (III.3b). It agrees with the assertion that the quality of the negative SVDF information is potentially better than the one of the positive SVDF information (that are closer to skeleton location which is hard to track).
  - If the above fails, the cut is based on the positive SVDF information (III.3c).
  - If the above fails once again, a linear interpolation of the magnitude of the SVDF vector taken as level set values is done(III.3d).
- zero or two cut cases (- -)
  - If tangent planes do not cut the segment then there is no cut (III.3e)
  - 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 (III.3f). This condition avoids some topological inconsistency (unwanted intersections) when following the SVDF iso-zero.
  - If tangent planes cut the segment twice in the right order (see above) there are 2 SVDF iso-zero points(III.3g).

An important aspect to consider in this process is what should be done when the 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 abscissa[1] s in the edge coordinate system is such that $s \in [0, \varepsilon_{metric}[$ or $s \in]1 - \varepsilon_{metric}, 1]$ where ε_{metric} is an arbitrary small number (we choose 10^{-5} in this work).

With a single cut algorithm, a simple treatment that modifies the level set is usually enough so that its iso-zero surface passes through the node. Nevertheless, it is no longer possible with the double cut algorithm because there is no way to identify the crack lips if we merge information at nodes. Thus, the cut node is placed on an edge node if it is close to it, but it still belongs to the edge from a topological point of view.

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

Another case is also taken into account during the coincident node treatment. Independently of what might give the above plan cut algorithm, if the SVDF magnitude at a node divided by the length of the emanating edge is small (less then ε_{metric}) then a cut is considered to exist and is located at the node. To avoid different treatment around a node due to edge length fluctuation, we first compute the mean length of edges connected to a node. It gives a consistent metric reference across edges to evaluate the magnitude of the SVDF.

All these overlapping position identification are then used during the construction of the sub-elements for the integration and during the identification of the enrichment (see annex A and chapter III.4.1 for further details).

III.3.2 second step of DCA: cut mesh elements

Once the edge cut positions and topologies are determined, the construction of the geometric domain using the following assertions (also available in 2D) can be achieved:

- The convex hull generated by the cut points and positive node-set corresponds to a + domain (Ω_c in the TLS framework). It is a polytope[2] (polyhedron or polygon depending on the case).

[1] s is dimensionless and varies from 0 (edge node origin location) to 1 (other edge node location).

[2] see (Coxeter, 1973, p. 118) for a general definition

- The subtraction (geometric operation sense) of this + domain to the original element gives the - domain polytope (Ω_+ in the TLS framework).
- The common boundary of the + and - domains gives the SVDF iso-zero (Γ_c in the TLS framework).

The element cut pattern cases can be reduced to a small set of unique cases using the elementary topological rotations and symmetries. Depending on the way one chooses to implement the cutting algorithm, generation of the null volume parts (coming from the cut point merged with other nodes) can be eliminated from the integration or not. It leads to a more significant number of element cut pattern cases or not.

All those cases are depicted for the 3D case in annex A. This annex also presents the cutting database introduced to accelerate the DCA computations. Moreover, it presents a pure geometric illustration of the performance of the double cut algorithm with SVDF compared to the level set single cut algorithm. Note that the library that provides DCA has also been implemented in a distributed version, not presented here, but used in chapter V.4.

III.4 Discussion about the new Γ_c representation

The use of the "double cut" algorithm for the discretization of Γ_c in the TLS framework has two impacts:

1. On the way ramped Heaviside enrichment is computed. It will be discussed in chapter III.4.1
2. On the way ϕ values are evaluated in the Γ_c vicinity. Chapter III.4.2 briefly comments on this aspect.

III.4.1 Enrichment impact

The fact that an element can be cut twice by the same iso-l_c is of small influence on the process that finds the dof to be enriched in an element as described in Bernard et al. (2012). The general guidance is still to consider for a given dof its support and to count the unconnected parts from Ω_+ created while splitting it by Ω_c (figure III.4).

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

The "double cut algorithm" presented in this work handles with care the cases where the crack runs close to the nodes (chapter III.3.1 and annex A). It has an impact on the way Ω_+ parts are identified and counted. In figure III.4, the zoom shows a basic example of Γ_c passing on a mesh node. In this

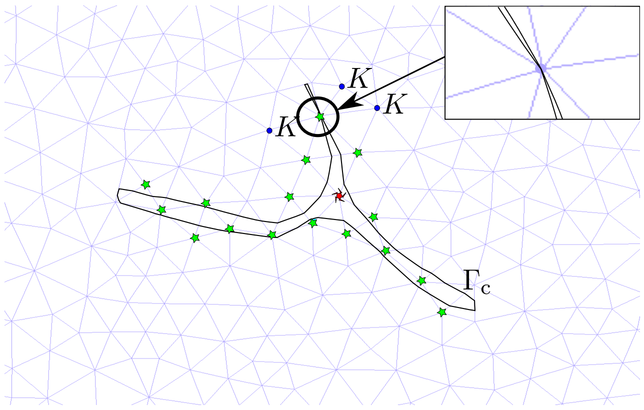


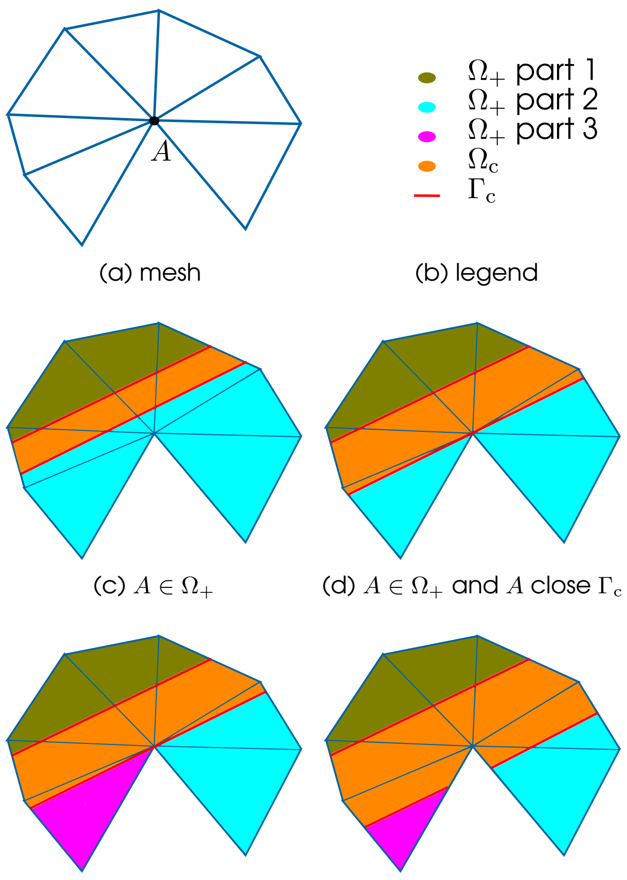
Figure III.4: Selection of nodes to be enriched by the ramped Heaviside. Star nodes are enriched once and the 'sun' node is enriched twice. Note that K nodes are not enriched because both cracks lips run on the boundary of their support.

case, a topological Ω_+ part exists but it is of zero measure. Those parts do not count for the enrichment decision. Subtle impact on Ω_+ parts identification is illustrated in figure III.5. Let's consider the mesh given in figure III.5a and A one of its node . In figures III.5c and III.5f, where Γ_c does not run through node A, either we have 2 sane parts (III.5c) or 3 sane parts (III.5f) depending on whether the node A lies outside Ω_c or not. In figure III.5d, Γ_c runs through node A (from the above), there are two sane parts (such as in figure III.5c). In figure III.5e, Γ_c runs through node A (from below) and there are 3 sane parts (like in figure III.5f). We remind that even though Γ_c runs through a node, the information about which edges are cut around the node by Γ_c (which enables the tracking of Ω_+ part definition) is kept.

III.4.2 ϕ in Γ_c neighborhood

The "double cut algorithm" uses a signed vector distance function instead of an offset level set for Γ_c construction. As exposed in the previous chapter and annex A, it gives a suitable Γ_c discretization. Nevertheless, it introduces some differences in Γ_c vicinity, between the ϕ value and the crack front location.

Figure III.6 shows the issues with an element extracted from a mesh in a 2D context (with $l_c = 0.5$). Figure III.6a depicts the following. The vertex element ϕ values are : 0.48, 0.497, 0.51. The SVDF construction of triangle upper vertex (based on figure III.2 steps) has discovered the closest point CP on the Γ_0 discretization at a distance of 0.499. This value is in contradiction with the 0.51 ϕ value. It is possible because Γ_0 is a discretization of the ϕ iso-zero that can induce a difference with the true iso-zero (in dot). The consequence in this case is that the SVDF sign at this vertex is negative! Consequently, the DCA uses the '- -' rule of table III.1 and there are only cuts at the bottom edge.



(a) mesh (b) legend

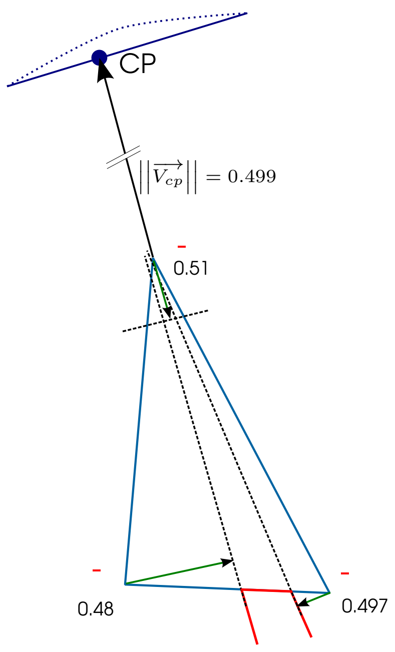
(c) $A \in \Omega_+$ (d) $A \in \Omega_+$ and A close Γ_c

(e) $A \in \Omega_c$ and A close Γ_c (f) $A \in \Omega_c$

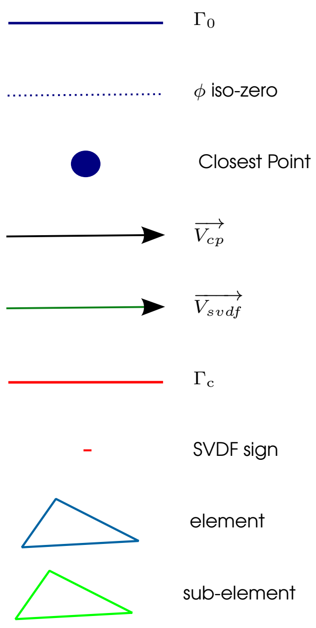
Figure III.5: Decomposition of the support of a node A (a) into sane Ω_+ parts. Two sane Ω_+ parts appear in (c) and (d) whereas three sane Ω_+ parts appear in (e) and (f). In (d), Γ_c runs through node A while cutting edges above. In (e), Γ_c runs through node A while cutting edges below.

The Γ_c appears then on it. Now, if considering ϕ value instead of $\left|\left|\overrightarrow{V_{cp}}\right|\right|$ the upper vertex would have been in Ω_c. This problem is treated modifying the ϕ value on this vertex (for example, by using neighborhood values) so that it is not greater than l_c as shown in figure III.6d. Note again that the SVDF fast marching alternative raised in chapter III.2 would suppress such difference (and would give a better vector orientation).

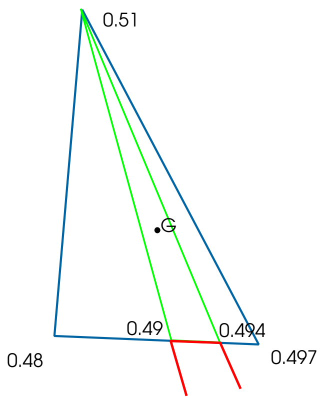
The other issue (not linked to the SVDF construction) is related to integration. Figure III.6c presents in green a sub-element introduced to take into account the Γ_c presence. As the integration is made at the Gauss point G for this sub-element, one must know the ϕ value at this point. There are two ways to interpolate ϕ on G :



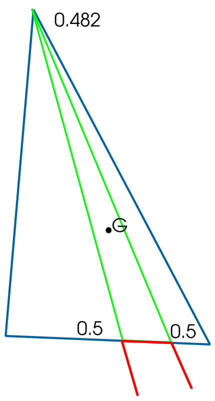
(a) Construction



(b) Legend



(c) Error



(d) Correction

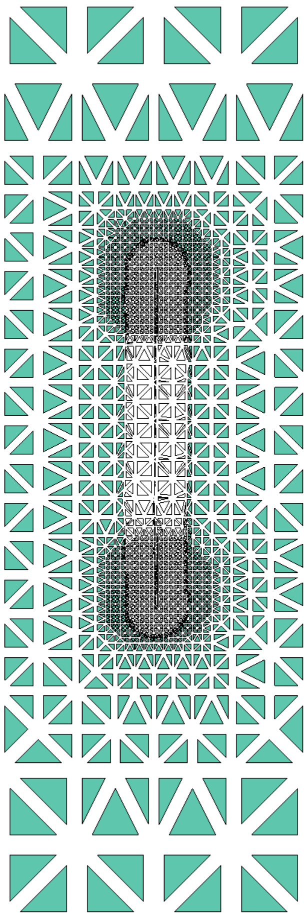
Figure III.6: Reshaping ϕ around Γ_c. III.6a SVDF construction and edges cut processing, III.6c error on Gauss points G computation corresponding to original ϕ, III.6d corrections applied to ϕ . lc = 0.5, 2D

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

Clearly in both cases, using the ϕ level set will not give a computed value corresponding to the presence of Γ_c. For example in figure III.6c, at the Γ_c edge nodes locations, ϕ should be 0.49 and 0.494 from geometric linear interpolation of element level set values.

The idea, to be always consistent with DCA construction, is to consider that the level set values on Γ_c nodes are imposed to a l_c value. Then, the computation of ϕ on G, using the geometric linear interpolation of the sub-element modified level set values, will be in accordance with the presence of Γ_c as shown in figure III.6d.

These two modifications lead to the fact that ϕ is no longer a distance function in the vicinity of Γ_c. It implies that somehow $d()$ function response is not anymore what we expect in this region (It is locally another $d()$ function). But the damage is now 1 on Γ_c and no spurious fully damaged location does appear anymore.



Chapter IV
Mesh adaptation strategy

Let us recall h_t, h_w and h_o definitions as respectively the mesh size in the crack tip vicinity, the mesh size in the crack wake region and the mesh size outside damage band. With the Thick Level Set model, the discretization requirements are of two kinds: the accurate capture of the mechanical behavior at the crack tips (that imposes h_t to be roughly equal to $\frac{lc}{5}$ for a right approximation of the strain field) and the right discretization of the crack path (that imposes h_w to be between h_t and h_o depending on the crack wake curvature for a correct integration computation). h_o must only fulfill the general mechanical constraint over the domain (such as boundary condition, fillet, hole , ...).
In a 3D simulation, the lazy option that imposes h_t everywhere or in a rather large defined region, leads to a huge system of equations that will be hard or impossible to solve on a computer (because of lack of memory or irrelevant computation time). Some specific computational strategies can help as exposed in chapter V. But adapting the mesh size is almost essential to reduce the effort of the linear algebra computation.
A first easy adaptation consists of tracking the damaged band and imposes within it, h_t (with $h_w = h_t$) and h_o elsewhere. It has been studied in this chapter with an octree, unstructured and two-scale embedded mesh. This last mesh is related to the two-scale strategy exposed in chapter V where adaptation is made at a fine-scale level.
An alternate adaptation strategy that sticks to the TLS requirement given above, consists of using h_w such that $h_t < h_w \leq h_o$. In particular, the "double cut" (chapter III) algorithm capability will be advantageously used to fulfill this new requirement in the octree mesh context.

In the TLS model, $\bar{Y}$ computed from the strain field, needs to be of excellent quality to provides the right crack progression. As the strain field is singular in the crack tip vicinity, a suitable discretization in this region is mandatory. That is the reason why h_t must be roughly equal to $\frac{l_c}{5}$ to capture the strain singularity accurately (Bernard et al. (2012)).

In the crack wake, only a correct material integration and a proper enrichment do mater. Both are related to ρ and crack complexity. If the crack curvature is small, the discretization must be rather precise to capture lips geometry correctly: h_w is close or equal to h_t. If ρ is large then the crack is almost straight and h_w can be equal to h_o. The same holds for branching or merging scenario where the lips are described with sharp angles.

In the region outside the damaged band, h_o needs only to fulfill constraint imposed by the mechanical problem: local Neumann boundary condition, holes in the part, fillet ... In this work, h_o is used with a constant order of magnitude. However, in reality, the mesh size is adapted around the holes, the fillet, ... This adaptation will be considered as a fixed input in this chapter.

The TLS mesh size requirement can then be summarized as $h_t \approx \frac{l_c}{5} \leqslant h_w \leqslant h_o$ with h_w being a parameter to adapt.

On the one hand, one may implement the TLS model without dynamic mesh size adaptation. In this case, he will be forced to impose h_t everywhere or in a rather large defined region. Nevertheless, no re-meshing adaptation nor ϕ transport has to be done. In 3D, it will lead to a system size difficult or impossible to solve on a computer (lack of memory or irrelevant computation time). However, all simulations of chapter VI have followed this principle. It has required a specific solving strategy, called active zone (see V.3), to partially bypass memory issues and time consumption.

On the other hand, if the mesh adaptation tools are available, a good choice of h_o and h_w can reduce the size of the system. The next chapter, proposes a mesh adaptation strategy that tunes h_w taking h_o (fixed adaptation) has an optimal input.

IV.1 Adapting mesh in damaged band

It corresponds to the simplest but not the most efficient strategy. h_w is arbitrarily taken equal to h_t. It is simple because it doesn't require algorithmic efforts to separate the crack tip vicinity from the crack wake. Only damaged band localization is requested and ϕ gives an easy way to do so.

IV.1.1 Octree context

It is the TLS historical adaptation strategy . It uses eXlibris Octree library that provides an effortless coarsening/refinement of cells in a cube or arrangement of cubes. The octree implementation uses the Linear octree concept (Gargantini (1982)). It stores the cell status in a linear array instead of a tree data structure. The array size is fixed by the maximum refinement level imposed by the user. At level 0 only one cell is defined: the original hexahedron (or set of hexahedron). At Level 1, 8 cells are corresponding to the first cube split in 8 equal hexahedron. At Level 2 there are 8^2 cells, and so on.

The library adaptation functionality takes as input a set of quadruplet: a level set, a level to reach in the octree, an offset from iso-zero of the level set and the level set side to use for refinement. The octree cells at the appropriate level are activated following those parameters with an imposed 2:1 constraint: a cell at a level $level_i$ can not have a neighboring cell with a level higher than $level_{i+1}$ or lower than $level_{i-1}$. It ensures a priori smooth mesh size transition and only one hanging node on edges or faces (annex C).

The TLS implementation uses this library via an octree/unstructured mesh conversion tool since all cutting and enrichment features are acting only on tetrahedrons. The unstructured mesh here, as inherited from an octree mesh, is mostly a regular mesh turned into simplex. The conversion tool transports back and forth the level set function from an unstructured mesh to/from an octree mesh. When passing from an octree to an unstructured mesh, the hanging nodes are treated either with supplementary linear kinematic equations or additional connecting elements (annex C). Of course, the level set function used for the octree adaptation is ϕ. In $\Omega_+ \bigcup \Omega_c$, we impose h_t and enlarge the refined zone using an l_c percentage offset.

This offset is essential in this strategy since the adaptation does cost (mesh creation, octree definition, level set function transfer, ...). By having this margin around Γ_0, the front can progress during several load steps without the need to reshape the unstructured mesh. The only constraint is that Γ_0 remains in the refined area. As soon as this condition is not fulfilled anymore, a new adaptation is required. In the future, the offset has to be tuned. For now, it is taken all around Γ_0 which is unnecessary most of the time. An "Only where Γ_0 will progress" criteria can tune the offset appropriately and can reduce the problem size slightly. However, this "Only where Γ_0 will progress" criteria is complex to tackle because it implies to know where the crack will progress before computing its progression.

Many simulations, not presented here, did successfully use this adaptation but the octree strategy is only available for parallelepiped shaped objects. Mentioned by members of the team, to bypass this limitation, a level set function can be used to define objects embedded in a cube. However, it only gives a discrete definition of its boundary (the geometric representation depends on the maximum octree refined level). It also requires a complex implementation to deal with an element cut by several level sets. Alternatively, using a mapping from/to a cube to/from a specific shape can work for a simple object. It has to be tested. In the end, coping with this shape limitation is investigated in the next chapter, transposing the octree refinement on an unstructured mesh.

IV.1.2 Unstructured context

The same principle as the octree refinement with a 2:1 constraint (see previous chapter) is used in this unstructured distributed mesh adaptation tool. Annex D provides the full parallel algorithm 5 and the details of the used refining procedure. This tool takes as input a mesh and a criterion indicating whether an element should be refined or not. In the context of damaged band mesh adaptation, this criterion is a function that asks to refine the element if it is in or partially in $\Omega_+ \bigcup \Omega_c$ and if its size is greater than h_t. As in the octree context, adding a smart offset in this criterion would be a good mechanism to reduce the number of adaptation. However, it as not been investigated yet.

In terms of scalability load balancing is a crucial point (as illustrated in annex D with a dedicated test). If the initial mesh is always distributed so that each process gets elements to split, the load may be balanced and the scalability will be present. Besides, in this rather crude tool, the splitting history is not kept. It limits the operations only to the refining from the original mesh: neither successive adaptation nor coarsening are possible. A future development can provide them and will also potentially ameliorate the scalability. However, for now this implementation provides an experimental adaptation tool to quickly test the assumptions formulated in this work using the appropriate mesh. Furthermore, the next chapter shows, in particular, how this unstructured adaptation tool has been used in the two-scale context.

IV.1.3 Two-scale context

The two-scale method is presented in chapter V.4 but it is interesting to recall, in this part, its discretization mechanism. At this point, all we need to consider for the two-scale method is that it needs to have a fine mesh embedded into a coarse mesh. A fine mesh element must know in which coarse mesh element it is embedded. Reciprocally, a coarse mesh element must know all its embedded fine mesh elements. This two-scale method applied to the TLS can use the adaptation criterion suggested in this chapter: $h_w = h_t$. In fact, the unstructured adaptation tool of chapter IV.1.2 may be used with a coarse element set to create all the attendees described above. The procedure may be summarized as follows:

1. Selection of the coarse elements that cover the damaged band. It will be, hereafter, named the $\mathcal{C}$ element set. An SVDF description of ϕ on coarse mesh is needed for that. Eventually paired with DCA, it will allow determining if the damaged band crosses a coarse element or not. Note that a small isolated damaged band can be untraceable by the DCA. It has to be investigated in the future.
2. Distribution of $\mathcal{C}$ using ParMetis weighted partitioning (an element that covers a more significant part of the damaged band gets higher weight). It is mandatory for a proper load balancing of the adaptation(as mentioned in annex D.3) as well as of the two-scale resolution (chapter V.4.3)
3. Use of $\mathcal{C}$ to create a new mesh, the SuperPatch, that will correspond to the fine scale. A connection between the $\mathcal{C}$ elements and their equivalent in SuperPatch is also created.
4. Adaption of the SuperPatch using the tool and criteria of chapter IV.1.2. A transport function has to be added to maintain a connection between the fine mesh elements newly created and their ancestors in $\mathcal{C}$.
5. Creation of the reverse connection from $\mathcal{C}$ to SuperPatch using the connection between the SuperPatch elements and their ancestor in $\mathcal{C}$.

Compare to the adaptation in chapter IV.1.2, the main difference, in this case, is the presence of two data structures to store the initial and final (refined) meshes instead of only one transformed mesh . It has two consequences. Firstly, it allows launching a new adaptation from a coarse level easily: only the fine data structure has to be cleaned. Secondly, it provides different results since the 2:1 constraint does not propagate in the same way for the two approaches. In this two-scale context, the SuperPatch is limited to the $\mathcal{C}$ set. Hence, the neighboring elements of $\mathcal{C}$ are not impacted by 2:1 constrain as they are in the adaptation of chapter IV.1.2.

IV.2 Coarsening mesh in the crack wake

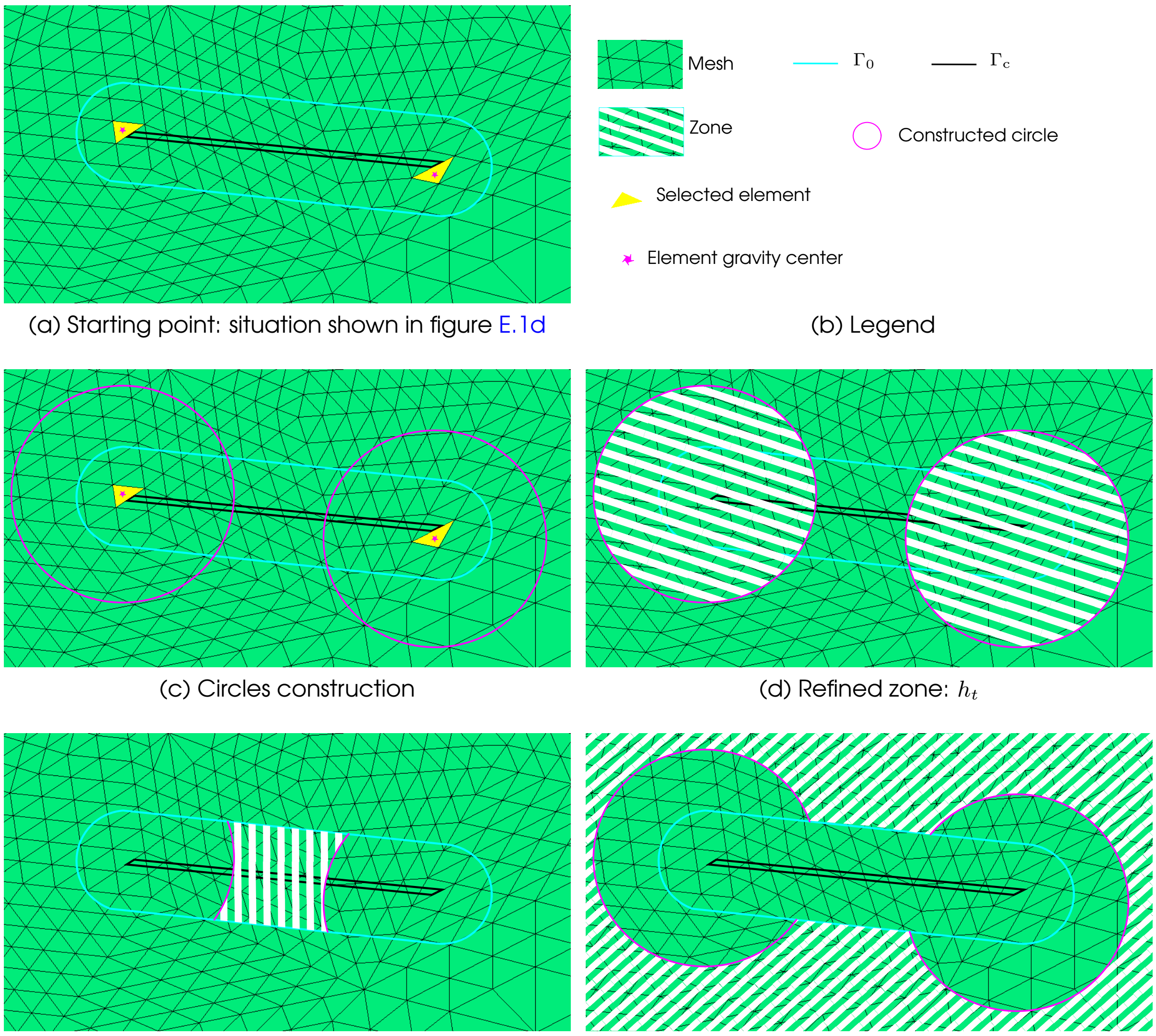
This strategy proposes h_t in the crack tip vicinity, h_o outside the damaged band and h_w in the crack wake with: $h_t \leqslant h_w \leqslant h_o$. Three questions have to be associated with this new approach. The first one, fairly obvious, is what value should be specified for h_w and should it be uniform in the crack wake? This question is not addressed in this work. Nevertheless, considering $h_w \approx max\left(h_t, min\left(h_o, \frac{\rho}{L}\right)\right)$ should be a good starting point. The second one is related to the crack tip vicinity definition itself. Where should vicinity stop and where does wake start? This question is treated in chapter IV.2.1. The third one is what tools are mandatory to represent the TLS framework correctly in the coarsened crack wake? This last question is related to the Γ_0 and Γ_c description. If we want to keep both of them in the coarsened crack wake, a standard level set tool is not appropriate. The DCA (chapter III) will let Ω_c pass inside an element. It lets h_w grows up to a limit where an edge is cut more than twice (2 times by Γ_c and one or two times by Γ_0). So with current tools, we must have $h_w < l_c$ without further treatment.

IV.2.1 Identifying the crack tip vicinity

A priori, the crack tip vicinity can only be described if the tip is located. The Γ_c discretization provides the tip and lips. However, where is the tip on this curve (in 2D) or surface (in 3D)? For now, no topological information is given by the tool that creates Γ_c. Investigation, not presented here, based on graph manipulation and zoning, gives a bad approximate of the tip localization on Γ_c. Besides, the computational effort is important.

Alternatively, since only the crack tip vicinity does matter, why trying to isolate the tip accurately? Knowing some points, close to the crack tip, can be enough. In this work, the enrichment information around Γ_c are used to find those points as presented in annex E. A quick analysis based on the current mesh and the TLS framework location, gives the elements which are close (at least in contact) to the crack tip. The points sought are then only the centers of gravity of these selected elements.

Figures IV.1a and IV.2a show, in a 2D context, two situations where point locations are given by enrichment information around Γ_c (see annex E for the steps that select those points). It is then possible to define a tip neighborhood by: constructing spheres (3D) or circles (2D) using those points as centers (figures IV.1c and IV.2c); taking their convex hull (see the areas with the white stripes in figures IV.1d and IV.2d). The sphere (or circle) radius is intended to be large enough so that the damaged band front is covered. It imposes a radius at least greater than l_c. But reducing the adaptation number, as in chapter IV.1, is possible by taking a larger radius value. It allows staying with the same mesh for several times steps as long as Γ_0 remains in the refined zone. Once having those refined regions, it is not difficult to remove them from the damaged band. Figures IV.1e and IV.2e show the resulting regions where a coarsening will be done with a h_w element size target. The adaptation tool will ensure that 2:1 constraint is valid. Furthermore, the DCA will permit the construction of Γ_c in those mesh size transitions. Finally, figures IV.1f and IV.2f show the mesh that will not be adapted (excepted marginally to ensure the 2:1 constraint). Note that this zoning is better than the offset strategy



(a) Starting point: situation shown in figure E.1d

(b) Legend

(c) Circles construction

(d) Refined zone: h_t

(e) coarsening region: h_w

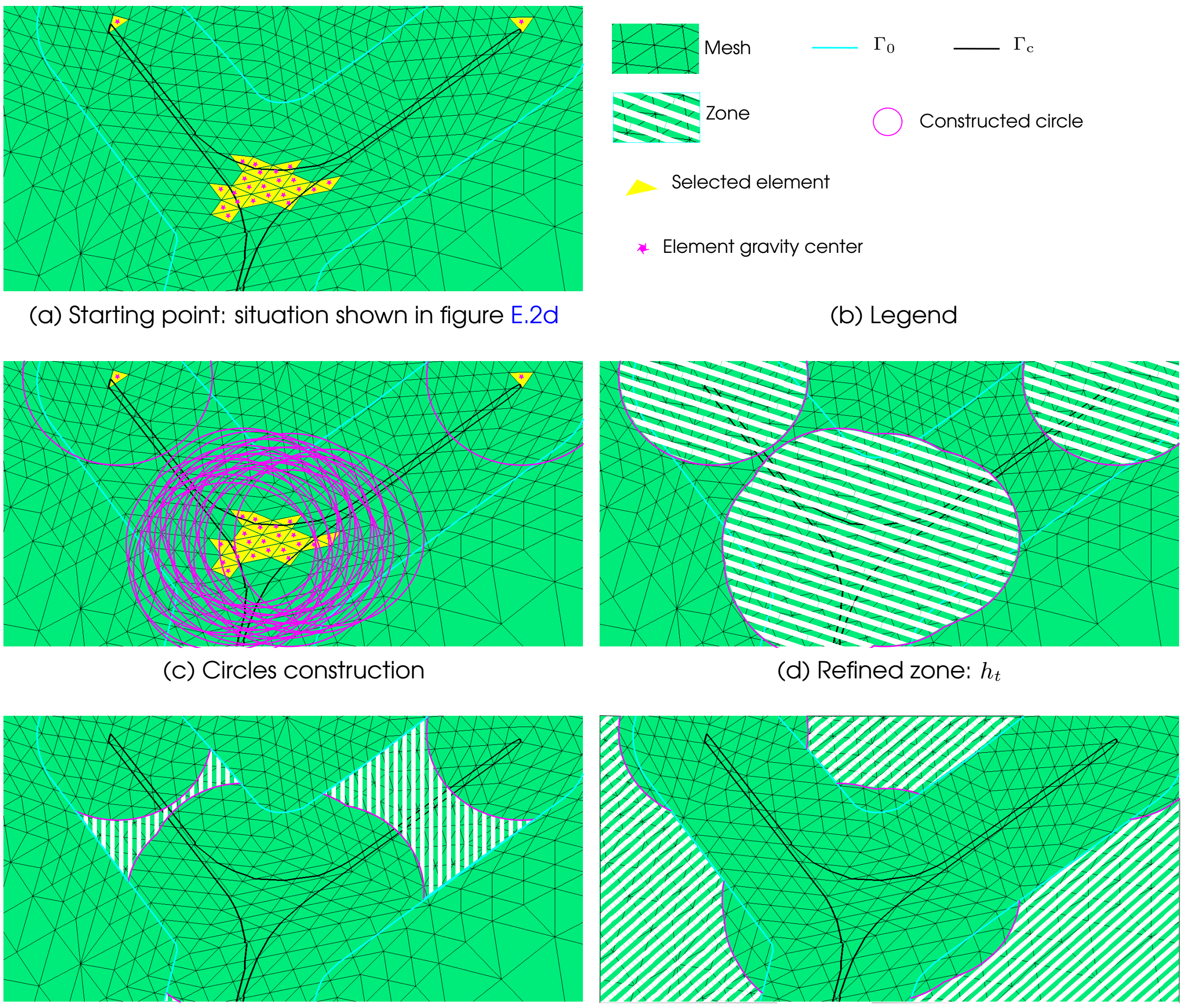
(f) Outside band and refinement:h_0

Figure IV.1: Simple zoning example in 2D

presented in chapter IV.1, even without coarsening. To limit the phases of adaptation, the additional refined elements are applied only in the vicinity of the crack tip.

Note that some care is required in the absence of cracks (during the initiation phase).In this case, no point close to the tip can be located because there is no tip. The workaround solution is to switch to the damaged band adaptation presented in chapter IV.1.

Finally, as discussed in annex E, the forking case presented in figure IV.2 shows that the proposed algorithm also identifies a region where the crack lips are not close to each other, as a tip vicinity. In this specific example, it is



(a) Starting point: situation shown in figure E.2d

(b) Legend

(c) Circles construction

(d) Refined zone: h_t

(e) Coarsening region: h_w

(f) Outside band and refinement:h_0

Figure IV.2: Complex zoning example in 2D

a benefit. It would have certainly been rather complex to coarsen the mesh in such a region. However, other circumstances, like a small region where lips are still connected (figure E.3), show the limitations of the proposed identification of the crack tip. The use of a more sophistical crack tip localization may in the future, be more accurate and efficient (see for example Coupez (2011)).

IV.2.2 Experimentation in the octree context

This strategy has only been tested in the octree context. The adaptation tools presented in chapter IV.1.1 are used with an extra level set. It corresponds to the convex hull introduced in IV.2.1 and is created simply by taking circle or

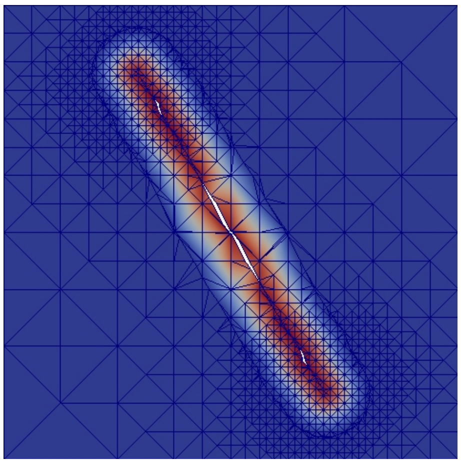


Figure IV.3: TLS simulation in 2D quad tree context with a crack wake coarsening (damaged field)

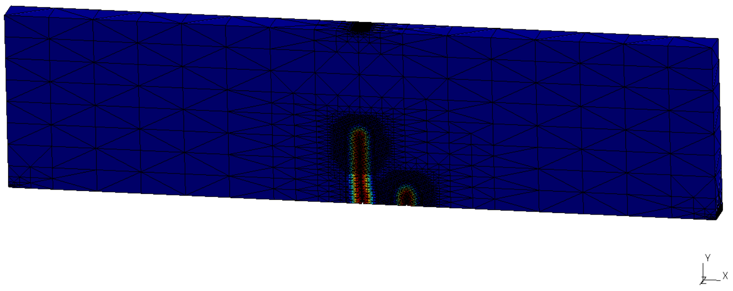


Figure IV.4: TLS simulation in 3D octree context with crack wake coarsening (damaged field)

sphere level set union. Two simulation results are presented in figure IV.3 and IV.4.

The first one corresponds to a plate (2D) under a shear loading. The boundary conditions correspond to tearing the plate in a specific direction (30 degrees from the vertical figure axis). It leads to a crack path not aligned with any edges of the mesh. The simulation is started with a seed (little damaged circle) at the plate center since the problem is initially symmetric. Then, the damaged band grows without an explicit crack. The octree adaptation imposes h_t in the band. When the damage front grows enough, Γ_c is introduced and the crack tip identification starts. On the first adaptation following this event, the tips at both ends are too close and the refined zone covers all the band. Then, later on in the simulation, both tips are far enough so that a coarsening region appears as depicted in figure IV.3.

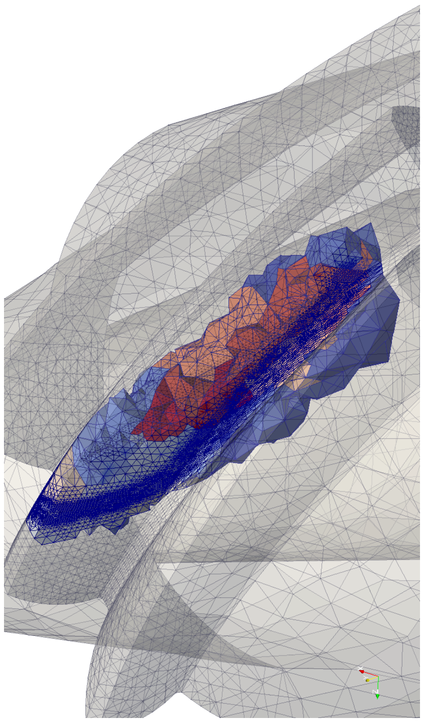
The second simulation is a three-points bending test with two initial notches at the bottom of the beam. The load is applied at the top center of the beam. One notch is centered (just below loading) and the other is shifted from the center. In the beginning,both notches, represented by the TLS framework, are of the same length. The first adaptation covers both. The two notches start growing and the shifted one stops obviously while the centered one continues toward the top of the beam. In figure IV.4, the tip of the centered crack is far enough from the bottom of the beam so that coarsening occurs.

During those two simulations, slightly modified Γ_0 and Γ_c positions have been observed in the coarsened regions. It is related to level set transport during the adaptation and this aspect must be investigated in the future.

The work presented in this chapter is more a proof of concept than a clean quantitative evaluation. This one has been postponed to focus on the a priori more profitable solutions studied in chapter V.

IV.2.3 Other experiments

Note that an anisotropic mesh adaptation (Coupez (2011)) has been tested with the TLS model by other team members. The idea was to replace the isotropic coarsening in the crack wake by an anisotropic meshing. It was done with the ICI-lib from ICI-Tech suites (ICI laboratory at ECN). The control of ρ and more precisely the curvature of Γ_0, give a parameter to adapt the mesh in all regions. At the tip, Γ_0 has a high curvature and the elements are kept isotropic. In the crack wake, Γ_0 is more straight and the elements can be extended in the crack path direction reducing their numbers in this direction. The tests were encouraging but some level set transport from one library to the other added numerical issue. It would make sense to investigate this path again.



Chapter V
Parallel and multi-scale solving strategy: distributed TLS approach

We show, in this chapter that in the implementation of the Thick Level Set model, the computational throughput is mainly linked to the performance of linear system resolution, as in many finite element applications. The system creation and the damage front advance are, respectively, second and third CPU time consumer.
Two general approaches are commonly used to solve a linear system: The "frontal" strategy that takes the problem as it is and uses direct or iterative solver to obtain a solution. The second approach corresponds to the "indirect". In this case, this strategy tries upstream manipulations to reduce the problem size before using a direct or iterative solver. In this chapter, both strategies have been investigated and parallelism has been chosen to chase High-Performance-Computing with a high scalability objective.
Regarding the "indirect" strategy, among all available possible solutions, two numerical methods have been studied: the domain decomposition method with a specific application called the "active zone" and the two-scale or GFEM^{gl} computational framework. The first one reduces the system size using a linear algebra condensation technique. The second one keeps low the system sizes by working on a global discretization (relatively small) which is enriched by some local fine-scale information. Both have been chosen for their potential scalability.
Even though every thing is not fully stated with solving strategies proposed in this chapter, positive preliminary results comfort the numerical and implementation choices.

	Load step	
Id	2	84
Elapsed time in s	832,4	7304,05
Nb of Newton Raphson iteration	3	12
Task	Elapsed time in % of total	
System resolution	97,4%	55,8%
System creation	2,1%	43,6%
Damage update	0,5%	0,6%

Table V.1: Chalk (see VI.2) simulation with a sequential profiling on titan (see J.2)

V.1 Initial profiling and strategies for improvement

Sequential TLS profiling shows that the greatest computational effort is related to the nonlinear resolution ((II) in algorithm 1). This iterative resolution (Newton-Raphson) can be split into two main tasks: the system of equation creation (definition of sparse matrix structure, integration and assembly of tangential rigidity matrix and right-end side residual vector) and the system of equation resolution (direct solver factorization and solving at each Newton-Raphson iteration). The computational performances of these tasks and of the damage updates (solving of equation (II.7) and ϕ updating) are presented in table V.1. They correspond to a chalk test case simulation (chapter VI.2) in a sequential computation context. It is representative of 3D elapsed time consumption. This table gives results for load steps 2 and 84. In both cases, the damage front advance computation is negligible. At load step 2, the simulation begins and a small damaged zone is only initiated at the chalk edge. The Newton-Raphson resolution is done in 2 iterations and the linear algebra resolution is proportionally the most important consumer. The assembly does not cost much because the non-linear part (damaged band) is small and is only computed twice. At load step 84, the crack has reached the chalk center and the Newton-Raphson resolution is done in 12 iterations. Consequently, the assembly task is proportionally more important than in step 2. There are more elements to assemble and this task is done more time. However, the linear algebra resolution remains the hot spot. Note that this last assertion was made on the basis of the resolution of linear systems with a direct solver.

To decrease the time consumption, one first solution consists in reducing the number of Newton-Raphson iterations by "simplifying" non-linearity. This point, not investigated here, is related to the asymmetric non-linear potential(II.5). Smoothing this law at tension/compression switch may ease the non-linear resolution (team proposal).

Now keeping the same non-linearity,the Newton-Raphson time consumption can be reduced by modifying the update frequency of the tangent matrix. With the modified Newton-Raphson or other variants, the tangent matrix factorization is done only a few times (depending on the modification applied to the original Newton-Raphson algorithm). Then, between these factorizations, only cheap forward and backward substitutions are used to solve the linear systems. It has been tested in a three-points bending simulation with an update of the tangent matrix every arbitrary N iterations. It is not present here but the results were not positive. The nonlinear resolution takes more iteration to converge decreasing the benefits given by reducing the number of factorizations. Nevertheless, it might be interesting to continue digging this track.

If the nonlinear resolution is not modified, it remains to decrease the computational time of the linear systems resolution. Two modus operandi have to be considered for that. The former is call "frontal" since it takes the linear system as it is and tries to find the fastest algorithm to solve it. The latter is called "indirect" because it first tries to reduce the problem size and then uses an adapted solver to solve the reduced system. The "frontal" approach is described in this chapter as the frontal strategy presented in chapter V.2.

For "indirect" approach, many strategies are available. The domain decomposition methods are good candidates. As stated in Saad (2003), these techniques "revolve around the principle of divide and conquer". Stepping in the resolution process very early, they divide the initial problem in a set of smaller ones. Those smaller problems, called domains, contribute to the global resolution by providing computed information individually. This domain treatment independency gives excellent scalability properties to those methods. In this thesis, non-overlapping domains are considered (eliminating Schwarz alternating method which seems complex with enrichment) and a element-base partitioning is chosen (the vertex or edge-base partitioning are also eliminated). This last choice is strongly related to the eXlibris tools capabilities in a distributed context. A first basic direct Schur complement resolution with two domains is proposed for the TLS model in Chapter V.3. Then in chapter V.5, an iterative Schur complement approach is benchmarked (however not with the TLS model) against the other strategies proposed in this chapter.

Alternatively, an "indirect" approach can reduce the problem size by the use of multi-scaling strategies. The most basic form has been investigated in previous chapter IV where the mesh adaptation copes with the different problem scale phenomena. Besides, mentioned in the introductory chapter I.2, a crack box can be seen as a specific scale where the discontinuous crack behavior is well captured. The rest of the part, embedding the box, corresponds to another scale only able to reproduce a overall mechanical behavior. Another technique, the multi-model and multi-scale Arlequin framework (Ben Dhia (1998)), introduces a generic way to superimpose a fine-scale model to a large-scale model combined at the energy level by coupling operator (often Lagrange Multiplier). The first attempt with this technique proposes to follow the crack evolution with a fine mesh covering it. Then, using XFEM, in Ben Dhia and Jamond (2010), only the crack tips vicinity is tracked by a fine-scale problem. At the global level, the enrichment with a Heaviside function ensures the discontinuous behavior in the crack wake[1]. A similar approach can be applied to the TLS. Despite the complexity linked to the Arlequin/TLS mixture, it is not clear whether the above proposals would have had the right scalability property. By nature, the Arlequin system structure is well suited for the domain decomposition-like solvers. But that implies that the crack tip vicinity (in 3D it's a pretty big problem in itself) has to be treated efficiently (in many domains in parallel) which is not obvious from bibliographic study. Around the same period Pereira et al. (2011) propose, for the same kind of simulation, a multi-scaling with the use of enrichment to "connect" scales. The associated two-scale (TS) or GFEM^{gl} computational framework (first introduced in Duarte and Kim (2008)) links a global scale to a set of overlapping local (or fine) scale problems (called patches). This last point offers a priori good scalability properties (Kim et al. (2011)) despite suffering from redundancy. This justifies studying how the TLS model can be introduced in this two-scale framework. Chapter V.4 presents this work.

V.2 Frontal strategy

This strategy may be summed up as choosing the fastest appropriate linear solver for the TLS non-linear resolution. To make the appropriate choice, the linear system matrix nature must be analyzed. The tangent matrix is a sparse symmetric positive definite matrix. This is interesting because it involves the use of a simpler and faster resolution algorithm compared to an indefinite matrix. However, the matrix conditioning is not very good due to the ramp Heaviside enrichment. In this thesis, this last point has contributed to eliminate the iterative solvers based on Krylov subspace from the possible solutions. Those types of sparse solvers, although they provide good scalability properties, are very sensitive to the matrix condition number. A vast research field is dedicated to preconditioning techniques that improve the solved matrix properties and then the iterative solver performance. A specific work on that aspect would be necessary to find the best preconditioner for the TLS model. Once done, other investigation about Newton-Krylov methods would be fruitful to really gain performance at the non-linear resolution level. Note that the multi-grid method may also play a role in this kind of solver. Finally, mixing the direct

[1]Note that in weak discontinuity approach community, the same performance concerns are also addressed. Moreover, the same mixing approach is proposed in Giovanardi et al. (2017) with XFEM in the wake and phase-field patch at the crack tip.

and iterative solvers in Newton-Raphson resolution might give some results easily (the direct solver computes, at some non-linear iterations, the costly tangent matrix factorization (eventually to be fast with low precision : MUMPS BLR) and this factorization is used as a pres-conditioner for all linear resolution done with iterative solver. It was successfully tested a few years ago at EDF R&D by Olivier Boiteau (no citation here sorry, we just talked together)).

After eliminating the iterative solver, the direct solver, less sensitive to poor conditioning is then the key solution. Many libraries are providing good algorithm for such sparse symmetric defined matrix: SuperLu (https://portal.nersc.gov/project/sparse/superlu/), Taucs (https://www.tau.ac.il/~stoledo/taucs/), Pardiso (https://www.pardiso-project.org/), PaStiX (https://gitlab.inria.fr/solverstack/pastix), MUMPS (http://mumps-solver.org/), ... The results of table V.1 show that a sequential approach is not adapted to the kind of problem addressed in this work. So parallel versions of those codes are a must. However, different parallel paradigms are proposed. For example, Pardiso proposes a multi-threaded solution that implies a computer with lots of unified memory connected to many cores (even though Pardiso developers switch now to hybrid direct/iterative distributed resolution). The message passing protocol with or without multi-threading is, on his side, more hardware flexible (it can be used indifferently on a set of personal computers networked together or a cluster interconnected with high bandwidth network or ...). For example our target platforms (annex J.2) are more adapted (especially Liger) to the message passing protocol. In this regard, MUMPS, PaStiX or SuperLU are better candidates. They all propose a version using MPI with multi-threaded or GPUs hybrid computation. Mumps has been retained for its well-known performance, its large set of features and its maintainers' group size (guarantee of longevity).

Now, how is it possible to introduce the MPI parallel solver into the TLS framework sequential implementation? A first try was to isolate the sequential code in process 0 and span, with a master/slave paradigm, solver steps in all processes. It gives quickly a way to scale the linear resolution but it has a limited action: the system creation becomes the bottleneck. The MUMPS solver offers a distributed input capability. The next move was therefore to fully distribute the TLS resolution using an element-based partitioning. It allows the system creation tasks parallelization which well scales (quite ideally if load is well-balanced). This move has not been achieved yet. Because many algorithms used in the TLS framework, take time to be adapted to MPI communication (such as the double cut algorithm of chapter III, for example). Nevertheless, this goal is slowly getting closer.

This strategy though, for the non-linear resolution, is only relying on the MUMPS scalability to perform. A recent version of this solver proposes multi-threading capabilities as well as block low rank (BLR) approximation (Amestoy et al. (2015)): they push both the performance at a high level. A point in figure H.8c gives insight of those promising feature, with the uniform benchmark test (with 64 MPI process, elapsed time for MUMPS resolution has been divided by 3.5 through the multi-threading). However, for sure, BLR must be studied at some point. Chasing access to higher performance, the condensation techniques to reduce the problem size have been tested in the next chapter V.3 with the active zone strategy.

V.3 Active zone strategy

V.3.1 General concept

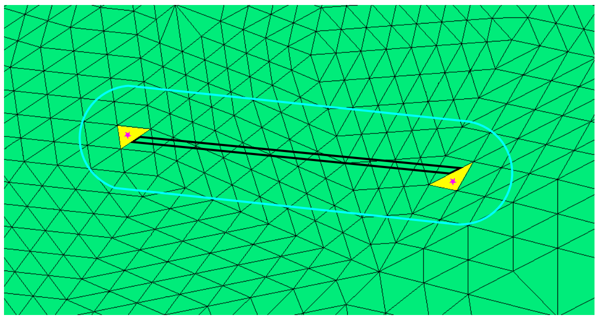
In the crack wake, the damage evolution is not expected unless the boundary condition does impose drastic modifications or unless another crack joins from elsewhere. If the damage does not generally change in those regions, it means that the mechanical properties are locally constant (the lips are at the same location and d is then unchanged). This said, one can try to isolate the crack tips to focus the computation effort on these variable zones (called Active Zones or AZ hereafter). It meets the concern of chapter IV.2: finding the crack tip vicinity. So naturally, the same enrichment base algorithm 6 (annex E) is used to locate the points in the crack tip vicinity. The simple example of an artificial TLS framework applied to a 2D mesh is presented in figure V.1. As the adaptation, a generic envelope is constructed base on a circle (2D) or a sphere (3D) (figure V.1c) centered on those points (figure V.1a). In figure V.2, this envelop is presented in a 3D simulation context. Using this envelope, the elements are separated into two groups :

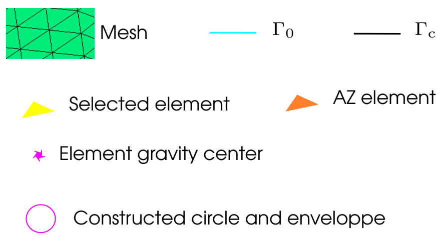
- the AZ group (orange elements in figure V.1d which are cut or inside the AZ envelope)
- the fixed group (green elements in figure V.1d)

The idea is to construct an AZ group for a certain amount of load steps so that the damage front may progress into it. As soon as the Γ_0 moving part reaches the AZ boundary, a new active zone has to be considered. However, during all this progression, information of the fixed zone (ϕ,d,...) are considered constant and their computational cost can be optimized.

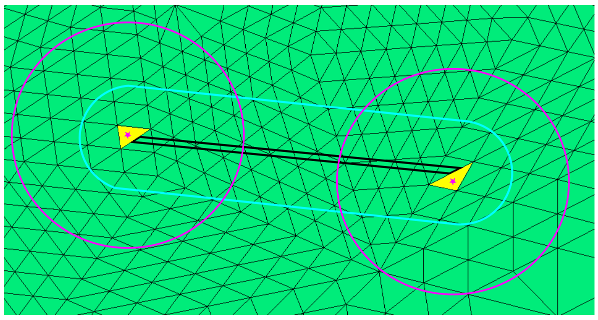
The larger is the AZ group, the higher is the number of load steps that benefit from the optimization of the fixed group (but the smaller is this group and the greater is the computation for the AZ).

In this work, the fixed and AZ groups are considered as two domains where decomposition techniques are applied. The Schur complement contribution and the condensed right-hand sides for both domains are computed. Then,


(a) Starting point: situation shown in figure E.1d



(b) Legend


(c) Circles construction

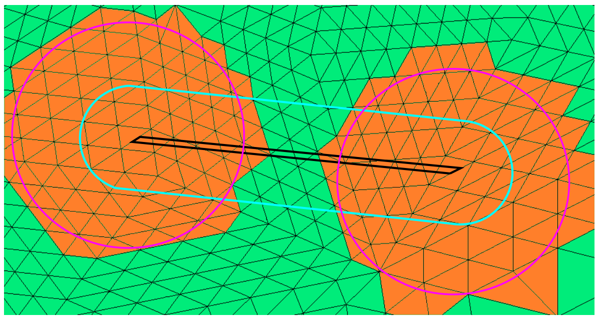
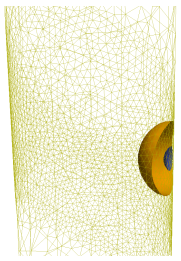
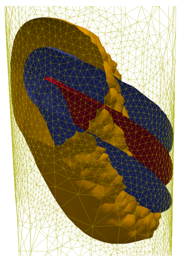

(d) AZ group

Figure V.1: Simple AZ envelope example in 2D


(a) Begin of the simulation


(b) middle of the simulation

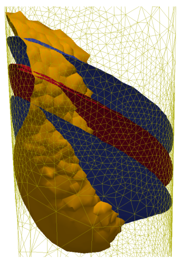

(c) end of the simulation

Figure V.2: AZ envelop examples with chalk test simulation of chapter VI.2: in orange, the AZ envelop embedding crack tip, in blue, Γ_0 and in red, Γ_c.

Schur complement reduced linear system is solved. It represents a part of algorithm 1 resolution II. However, during all the AZ lifetime, the fixed group is considered constant. So this domain can be condensed only once at AZ construction. Then fixed domain linearized form, the Schur complement contribution, is simply added to reduced problem at resolution. In this way, the factorization effort for the fixed group is amortized over many load steps and non-linear iterations. The AZ domain itself remains non-linear and is subject to the ϕ variation. Hence, its condensation is systematic for each Newton-Raphson iteration during its lifetime. It gives the new stagger algorithm 2 where the $\tilde{}$ corresponds to the condensed space and G is the expansion operator (from the condensed to the full domain).

Algorithm 2 Modified staggered algorithm scheme with AZ.

```
Starting from $\vec{u}^i$, $d^i$, and $\mu^i$
repeat
    Find $d^{i+1}$ such that $d^{i+1} = g\left(\mu^i, \vec{u}^i\right)$
    Eliminate fixed group from problem for any new AZ
    Find $(\vec{u}^{i+1}, \mu^{i+1})$ such that
      $\begin{cases} \tilde{K}\left(d^{i+1}, \vec{\tilde{u}}^{i+1}\right)\vec{\tilde{u}}^{i+1} = \tilde{F}\left(\mu^{i+1}\right) \\ \vec{u}^{i+1} = G\vec{\tilde{u}}^{i+1} \\ \max\limits_k \left(f_k\left(\mu^{i+1}, \vec{u}^{i+1}\right)\right) = 0 \end{cases}$
until Complete failure or user given load level
```

In algorithm 2,it is in the application of the g operator that the Γ_0 evolution is checked against the AZ boundary. Any progression, close to this boundary, launches a new AZ construction. Note also that when a damage initiation outside the damaged band occurs a small damaged sphere is added to ϕ and a new AZ must be created to encapsulate this extra zone.

Like in frontal strategy (chapter V.2) message-passing paradigm is chosen to gain computational performance with parallelism. In the current AZ strategy, only two domains are used. So parallelism will not be, like in many domain decomposition methods, related to domains parallel treatment. Instead, the parallelism will be employed to reduce the algebraic resolution cost of both domains and the reduced system. MUMPS is again chosen since it provides excellent performance for Schur complement creation by simply adapting its parallel multi-frontal factorization to an incomplete block factorization. It also nicely gives Schur complement dense matrix in a distributed cyclic block format compatible with the Scalapack library. It simplifies the manipulation of reduced problem and drives to use the Scalapack distributed dense solver to resolve it. The load balancing induced by parallelism is treated in this thesis, with a specific partitioning that tries to stick as much as possible to the AZ concept. It is depicted in annex F.1.

Note that an extra feature not presented in this thesis has been tested. It corresponds to the use of a mixed shape function approximation order. One may want to use a higher-order to solve the problem. With the concept of focusing the computational effort on crack tip, it is rather natural to restrict this higher-order to the AZ group. An order 1 for the fixed group is enough as any way it is only used in a linearized way. Compared to a full higher-order problem, gains in computational time are expected. Though this requires that the size of the common frontier between the AZ and the fixed groups is kept at order 1 (i.e. order transition must be done in the AZ boundary element) to get the smallest possible size of dense problem.

V.3.2 Feed back

This strategy has been used in many test cases of chapter VI. Note that none of these simulations uses a mesh adaptation strategy from chapter IV: a fixed region using h_t mesh size covers a priori the crack path. This choice was guided by the wish to eliminate any mesh adaptation interference in 3D TLS model validation (independent damage initiation, no level set transport interference during the adaptation, ...). The AZ strategy itself has also justified it. The condensation and the mesh adaptation strategy are similar: they concentrate the work load in the crack tip vicinity and amortize it over few load steps. Hence, all optimizations that the mesh adaptation can provide are only used in the fixed group. The cost of this domain condensation being amortized over many load steps/Newton-Rapshon iterations, the mesh adaptation optimization is less worthwhile.

Not using the mesh adaptation also simplifies the implementation. This one, like the frontal strategy in chapter V.2, starts from the TLS framework sequential implementation. It must be parallelized to integrate the AZ strategy. A full distributed implementation would be the best. However, for the same reasons seen in the frontal approach, a mid-term solution has been adopted. It loads the full mesh in each process. Then, the distributed algorithm is only introduced for key tasks used in the AZ strategy. All other steps remain sequential (but done on each process).

A full description of obtained performance on these test cases is given in annex I. What has to be retained is first that the enrichment base algorithm 6 (as said in annex E.2) provides excellent to moderate locations for the AZ envelope creation. The algorithm is faulted when a not fully detached crack lips zone exists or when the crack region includes a full element. In both cases, the AZ group becomes larger than expected and the computation slows down. The crack tip vicinity localization is to be improved. Nevertheless, when the AZ envelop is well identified, this strategy gives, as shown in table I.5, a good time consumption reduction. However, the scalability performances are hard to investigate with this mixed sequential/parallel implementation. What can be said is that, like the frontal strategy, the scaling of nonlinear resolution depends mostly on MUMPS parallel performance (Scalapack also comes into play). And if a 3D crack tip becomes very long, the associated AZ can have a size that is no more appropriate to a direct solver treatment of the incomplete block factorization. Extending domain decomposition approach, with the AZ divided into many domains, can allow regaining scalability. However, more domains imply a larger Schur complement (even larger if the fixed domain is also split into many domains). The reduced matrix direct resolution with Scalapack can then become an issue. Switching to an iterative solver for reduced system resolution would then be a natural solution. Nevertheless, as evoked in V.2 (even if Schur complement does improve the conditioning), the matrix conditioning will impose to find an ad hoc TLS model preconditioner. All this work has been postponed to try a completely different approach. It is based on a multi-scaling strategy which should give better scalability. This new strategy is presented in the next chapter V.4. However, the idea of this AZ enhanced version (with many domains) as been evaluated indirectly in the benchmark of chapter V.5.

V.4 Two-scale strategy

Algorithm 3 Scale algorithm scheme: smx maximum number of scale iterations, ε arbitrary numerical convergence criteria, $\vec{U}_s$ solution at global level and scale iteration s, $\vec{u}_s^{k\in I_e^g}$ set of fine-scale problems solution at iteration s, I_e^g set of enriched global mesh entities identification number.

```
Starting from the global problem solution $\vec{U}_{s=0}$
**repeat**
    $s \leftarrow s+1$
    **for** $k \in I_e^g$ **do**
        Impose fine-scale problem $k$ boundary conditions with $\vec{U}_{s-1}$
        Solve fine-scale problems $k$ to obtain $\vec{u}_s^k$ solution
    **end for**
    Compute global enriched problem with $\vec{u}_s^{k\in I_e^g}$ as enrichment functions to give $\vec{U}_s$ solution.
**until** $s \geq smx$ or $\frac{\|\vec{U}_s-\vec{U}_{s-1}\|}{\|\vec{U}_{s-1}\|} < \varepsilon$
```

V.4.1 General concept

The two-scale (TS) or GFEM^{gl} computational framework (first introduced in Duarte and Kim (2008)) uses interdependent mechanical problems for the simulation. The global level problem treats the entire domain with a finite element discretization which is only able to capture the global phenomena. The related fine[2] level problems capture localized specific behaviors with a fine discretization. A fine level problem definition corresponds to the region covered by a coarse entity support that includes the localized specific behaviors to capture. Consequently, fine level problems overlap.

The local behavior can be of different natures such as welds spot in Li and Duarte (2018), crack in Pereira et al. (2011), thermo-structural effects in Plews and Duarte (2015), ... In all cases, the same approach is used for the interaction between the global problem and the set of fine problems:

1. The global level solution is used to impose the boundary conditions of the fine level problems.
2. The fine level solutions are used directly or not, as enrichment functions to enrich the global space using the partition of the unity framework.

The enrichment function construction from the fine level solution is explained more in detail in chapter V.4.4.

Different algorithms are conceivable using those two interactions. For simple fixed phenomena, a single loop between the scales provides the global solution as shown in algorithm 3. For evolving behavior such as crack growth, a first approach would be to use the algorithm 3 to solve the mechanical problems at each dissipating load step that makes the crack advances.

But with evolutionary behavior problems, the scale loop can also be mixed with a general solving process. This is what has been done in Pereira et al. (2011) where the authors consider that the growth of the cracks is sufficiently small so that the global solution evolves smoothly. It leads to the general algorithm 4 where the TS enrichment functions are updated using the global solution of the previous load step. Intermixing the load steps and the scale loops is the point that can make the difference compared to other resolution strategies. Because, in this case, a part of the resolution process cost is amortized between load steps.

V.4.2 Integrating the TLS model into the two-scale method

Following the general concept presented in chapter V.4.1, the discontinuous problems represented by the TLS model will be treated at the fine level in the two-scale framework. Here, one possible approach would be to focus on the crack tips only for the TLS computation at the fine scale. Moreover, in the crack wake, a Heaviside enrichment function at the global level would

[2]In this thesis most of the time "fine" is used in place of "local" original GFEM^{gl} designation

Algorithm 4 Scale algorithm integrated into a general growth loop: tmx maximum number of dissipation load step, CS_t crack status at load step t , $\vec{U}_t$ solution at global level and load step t, $\vec{u}_t^{k \in I_e^{g^t}}$ set of fine-scale problems solution at load step t, $I_e^{g^t}$ set of enriched global mesh entities identification number at load step t. To draw a parallel with TLS , (†) correspond to step (I) and (‡) more or less to step (II) and (III) of algorithm 1, and CS_t is related to ϕ level set

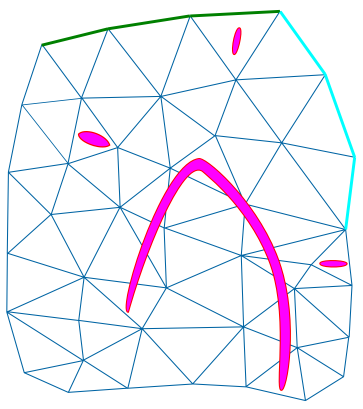
```
Starting with a crack status $CS_{t=0}$ compute global enriched problem solution $\vec{U}_{t=0}$ with algorithm 3
repeat
    $t \leftarrow t+1$
    (†) Compute new crack shape $CS_t$ from $\vec{U}_{t-1}$ and $CS_{t-1}$
    for $k \in I_e^{g^t}$ do
        Set or reset fine-scale problem $k$ with $CS_t$
        Impose fine-scale problem $k$ boundary condition with $\vec{U}_{t-1}$
        Solve fine-scale problems $k$ to obtain $\vec{u}_t^k$
    end for
    (‡) Compute global enriched problem with $\vec{u}_t^{k \in I_e^{g^t}}$ to give $\vec{U}_t$ solution.
until $t \geq tmx$ or crack stop growing
```

do the job. This is what has been proposed in Pereira et al. (2011) with the GFEM^{gl} method and in Ben Dhia and Jamond (2010) and Giovanardi et al. (2017) with other methods. However, to maintain the TLS model versatility (late fork in the crack wake, merging, ...) and to respect the crack path during a simulation (small ρ), the whole TLS framework has to be discretized. It does not imply that in the crack wake there is a need to pass by the TS enrichment function to obtain a discontinuous displacement at the global level. Using at the global level, a Ramp Heaviside enrichment function, built on top of the fine-scale discretization, is cheaper.

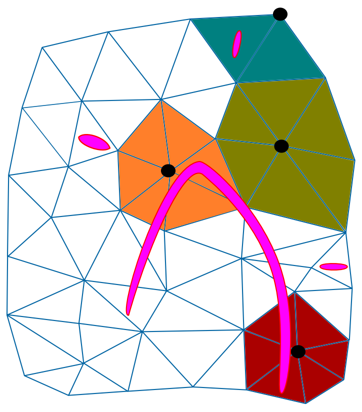
The alternative is then to maintain at the fine level an appropriate discretization for the TLS model in the whole damaged band. And at the global level, one must use enrichment with the TS function only[3] if the patch may not be treated by a Ramp Heaviside function. Note also that the weak discontinuity of the TLS model (damaged material) is taken into account at the global level: macro elements are integrated using the fine-scale elements so that Ω_-, Ω_+ and Ω_c are distinguished in these global element stiffnesses.

This solution is illustrated in figure V.3, in a fictitious 2D simulation context. Figure V.3a presents the global-scale mesh with a damaged band envelop

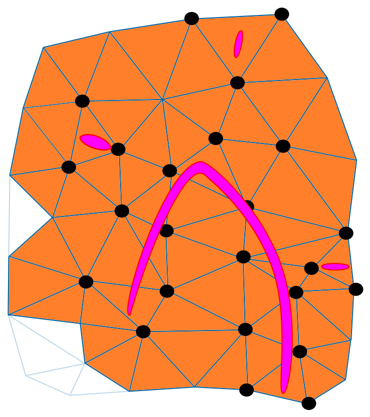
[3]Future work may prove that some enrichment overlapping can be mandatory to get better results. In this case some nodes can be enriched with both TS and ramp Heaviside function.



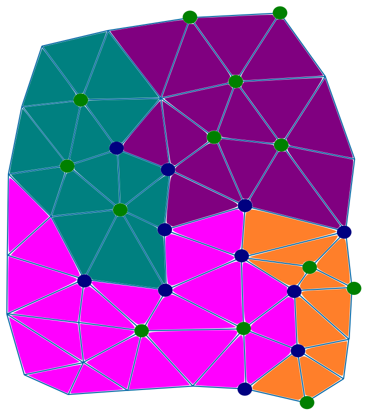
(a) Global-scale problem



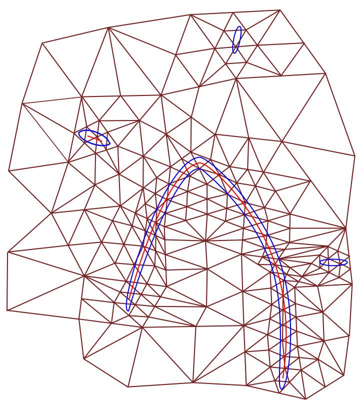
(b) 4 colored supports out of 24



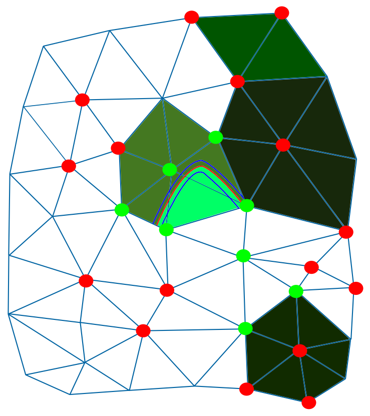
(c) Support union



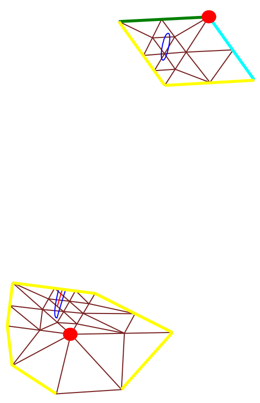
(d) Global element dispatching on 4 processes



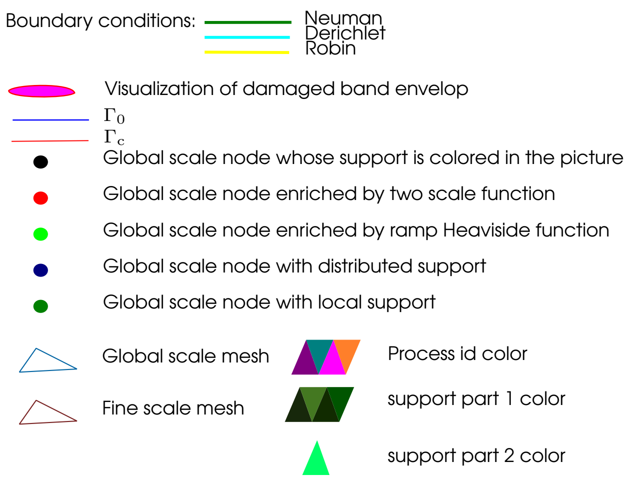
(e) SuperPatch: refined mesh



(f) Enriched node (with support parts)



(g) Local scale problems (2 out of 16)


Boundary conditions:
Neuman
Derichlet
Robin
Visualization of damaged band envelop
Γ_0
Γ_c
Global scale node whose support is colored in the picture
Global scale node enriched by two scale function
Global scale node enriched by ramp Heaviside function
Global scale node with distributed support
Global scale node with local support
Global scale mesh
Fine scale mesh
Process id color
support part 1 color
support part 2 color

(h) Legend

Figure V.3: TLS framework imbrication with the TS method presented in a fictitious 2D problem context

known somehow (it could be a mathematical definition at the beginning of the simulation , or a level set defined in another mesh obtained from a previous simulation step, ...). It is represented here just for visualization purposes since it is not defined at this scale.

A first task is to identify the nodes[4] that do have their support covering the damaged band (it is illustrated in figure V.3b with 4 supports of different color). All nodes and their support covering the damaged band are visible in figure V.3c. Then, as mentioned above, the fine-scale problems have to be created for each of them. By recalling that in the TLS model, ϕ is discretized on nodes of the mesh, this field should logically be defined at the fine scale within the TS approach (to provide a correct model description). Now does the ϕ definition have to be replicated on each patch? And, does each patch must have its own fine mesh discretization? It looks more handy to have the level set defined on a unique mesh representing the fine scale. This unique mesh called the "SuperPatch" (similar to the master-local domain in Li and Duarte (2018)) would correspond to the union of fine-scale elements of all patches if they were all discretized in the same way. This last point (uniform discretization) provided by the use of a unique fine-scale mesh is essential for the integration: a fine-scale element is unique for its covering supports . Hence, at its Gauss points, all associated enrichment functions are defined (it would not be the case with a different discretization per patch) when integrating the matrix at the global level. Using a SuperPatch also simplifies (I) and (III) of the algorithm 1. Moreover, it allows running only once the cutting algorithm (in particular, the DCA presented in chapter III). The SuperPatch construction, by itself, follows the unstructured adaptation strategy of chapter IV.1.3. It is illustrated in figure V.3e for the simple 2D example where $\mathcal{C}$ corresponds to the colored elements of figure V.3c (i.e. the support union).

Having the SuperPatch, the level set ϕ is transported on this mesh and Γ_0 and Γ_c are computed (shown on the same figure V.3e even if they are drawn far too precisely compared to the non-realistic refined mesh that we see). Once done, one can analyze the node supports of global mesh, to count, using the fine-scale definition of Ω_c, the number of parts of these supports divided by the crack. If there is only one part, the TS enrichment is used (red node in figure V.3f). Otherwise, a ramp Heaviside enrichment function is chosen (green node in figure V.3f). Figure V.3f visualizes these parts for 3 supports. For the ramp Heaviside, no extra computation is required. However, for the TS enrichment the enriched function has to be computed for each patch.

The patches are created by extracting the SuperPatch elements related

[4]Note that in this thesis, the TLS model integration into the TS method has been mainly evaluated in the context of first-order approximation for fields. This leads to talk only of enriched nodes.

to their support as illustrated in figure V.3g. Regarding the patch boundary conditions, they come from the global level displacement and strength fields solutions, which are imposed as Robin boundary conditions[5]. Here, some preliminary tests lead to the sole use of loosely imposed displacement by a penalization. Other global boundary conditions are inherited. The TLS framework is naturally present in the patches since the elements are the SuperPatch one. In particular, the damage and the ramp Heaviside enrichment (at fine level, not to be confused with global level enrichment) are already created in the SuperPatch. The resulting patch displacement solution is then used as a TS enrichment function at the global level as detailed in chapter V.4.4 and scale loop can be used.

The scale loop integration with the TLS non-linear resolution (II of algorithm 1) is still under analysis. In a very preliminary work (Salzman et al. (2017)), the solution of embedding Newton-Raphson loop into a scale loop has been tested with a sequential TLS/TS framework version. Fine-scale problems were interpreted as linear problems where the strains used in (II.5) that give traction/compression states are taken from the global scale. When entering the scale loop, the previous load step solution is used to compute fine-scale problems (boundary conditions and strains). Then, the problem at the global scale is solved (Newton-Raphson iterations) with a fixed TS enrichment function. At the convergence, the fine-scale problems are again computed with the new global solution. And the non-linear resolution is launched again. The scale loop is stopped when the fine-scale updates do not change the non-linear solution. The test shows that the convergence is slow with this scheme. It could be improved by solving at fine-scale the non-linearity. Alternatively, using a completely different approach, the scale loop can be reduced to only update the TS enrichment functions from the previous load step. It corresponds to disregard the error introduced by not looping again after the Newton-Raphson resolution in the current load step (as proposed in algorithm 4). This is left as a prospect.

Finally, as ϕ progress, two adaptation criteria should be taken into account to update the SuperPatch: the conventional element size criterion (chapter IV.1.2) and the fact that Γ_0 passes in a support not activated yet. In this thesis, the SuperPatch adaptation has not been tested.

V.4.3 Two-scale method in a distributed context

In the TLS approach, the crack tip in 3D may generate a significant amount of patches that do overlap. This redundancy may be masked only if the patches represent tiny problems or if their treatment is done in parallel. In Kim et al. (2011), the authors use a multi-threaded parallelism to compute each

[5]note that no boundary condition is imposed in Ω_c

patch. However, from the TLS simulations done with an AZ strategy (chapter V.3), millions of dofs can easily be encountered in the case of long crack tips. Using only multi-threaded in this context is expected to be limited by memory issues. So, as in the frontal approach (chapter V.2), the idea is to distribute the global mesh on processes relying on MPI, to exchange useful information amongst themselves. In this strategy, an element is only present in one process. This choice impacts the TS method treatment:

As fine-scale problems are concerned, how do we treat a patch that has its global elements in more than one process [6]?

One possible solution is to consider that this kind of patches are duplicate somehow on each process. Then, each process computes the same patch but this is too costly. An intermediate solution is to consider that the patch is dispatched as its macro element on several processes. However, only one process holds the full definition of this patch and compute it. The obtained solution must then be scattered on the other processes related to this patch. It is complicated, because part of the mesh (or the contribution to the matrix system from this mesh part) must be duplicated on the process performing the resolution of the patch. It does not respect the rule "one element is held only by one process" and implies many developments in eXlibris. It also introduces load balance asymmetries by leaving only one process of the patch to do most of the work.

The other solution is to dispatch the patch as their macro elements and to assume that the patch problem itself is treated in a distributed manner. It introduces some complexity in the patch treatment as explained below, but it offers a double-level of parallelisms. It can be useful in the TLS context. With the mesh adaptation from chapter IV.1.3, the patches are not of the same size: only the damaged band is locally refined so a patch entirely inside the band will be bigger (more dofs) than the one which is partially in the band. As soon as there are differences, one can arrange the coarse mesh partitioning (dispatching of coarse elements on all available processes) so that largest patches are treated in parallel and the smallest one in sequential. The load balancing is then expected to be good. A further argumentation about this double-level of parallelism, related to the size jump between scales, is given in the benchmark test analysis (of chapter V.5). However, the greatest complexity related to this double-level parallelism comes from the fact that the patch treatment can no longer be done in random order. Two patches defined on the process 1 and 2 for the first one and 1 and 3 for the second one, may not be computed at the same time on process 1 (even though it is

[6] Illustrated in Figure V.3d with the global mesh of the 2D simple fictitious example distributed over 4 processes. In this figure, green and blue nodes correspond to patches respectively entirely embedded in one process or split across two or more processes

possible on process 2 and 3). The patch treatment has to be ordered across all processes. This sequencing algorithm, quickly presented here, retake the order proposed in Kim et al. (2011) ("bigger" patch treated first). Besides that, it imposes sequences where the possible distributed patches are chosen so that the greatest number of them are computed in the same sequence. If in a process, no distributed patch can be computed in the current sequence, a local patch is taken. And if no local patch is available, nothing is done (which is the worst case).

When all distributed patches are treated, the remaining local patches are computed independently, following the local ordering rule "biggest firstly" (processes are not synchronized anymore). For that, in one process, the natural sequential computation can be used but a multi-threaded computation can also be a solution. This last choice would turn this application into a hybrid MPI/multi-threaded simulation code. For now, many eXlibris components (assembly, sparse solver) are not thread-safe. That is the reason why a hybrid TS was not implemented but this is not impossible from a conceptual point of view. In terms of efficiency, it has to be compared with MPI: does the usage of many threads with few MPI processes (and fewer distributed patches) better/same/worse than the usage of only many MPI processes? In any case, it would give great flexibility to the user. This question is unresolved and left as a future work prospect.

Apart from the patch processing, one must not forget that the assembly must also be well distributed. As integrating global elements leads in some cases to review the associated fine-scale elements, the assembly cost is not the same across the global-scale elements. Hence, to sum up, the load balancing and the associated global mesh partitioning is mainly driven by :

- Number of fine-scale elements per global element, used for the integration.
- Number of enriched vertices per global element.
- Cost of the patch resolution (as said above the "biggest" patches should themselves be distributed) which, in general is related to its number of fine-scale elements.
- Use of the TS or the ramp Heaviside enrichment at the global level.

In this work, the load balancing has not been investigated in detail and this question is left as a perspective. A simple crude partitioning is used in the current testing implementation (based on the number of patches associated with a global element)

The TS method implementation in the distributed context proposed in this chapter has been carefully evaluated in chapter V.5 (but in a non TLS context). It has also been used successfully (giving correct results in parallel) in various tests not presented here.

V.4.4 Enrichment strategy

Let us consider the standard first order finite element shape functions $N_i(\vec{x})$, $i \in I^g = \{1, 2, ..., N\}$, associated with a global node $\vec{x_i}$, with a support given by the union of all finite elements sharing the node $\vec{x_i}$, in a global mesh discretization with N nodes. In this chapter, only the TS enrichment function is retained at the global level. This function that can be in scalar or vector form is considered here in its vector form to simplify the presentation. Let us call it $\vec{F}_k(\vec{x})$, $k \in I_e^g = \{1, 2, ..., T\}$. It is associated with the global node $\vec{x_k}$, it has the same support as $N_k(\vec{x})$ and I_e^g represents the T global nodes enriched by the TS method. Kinematic equation used to describe the global displacement field is then:

$$\vec{U}(\vec{x}) = \sum_{i \in I^g} N_i(\vec{x})\, \vec{U}_i + \sum_{k \in I_e^g} A_k N_k(\vec{x})\, \vec{F}_k(\vec{x}) \tag{V.1}$$

At the fine-scale level for a given patch k, we consider the classical first-order finite element shape functions $n_j(\vec{x})$, $j \in I^{l^k} = \{1, 2, ..., n^k\}$, associated with the fine-scale node $\vec{x_j}$, with a support given by the union of all finite elements sharing the node $\vec{x_j}$, in the fine mesh discretization with n^k nodes. The TLS scalar ramp Heaviside function associated withe the k patch is called RH_r^k,$r \in I_e^{l^k} = \{1, 2, ..., t^k\}$. It is associated with the global node $\vec{x_r}$ and has the same support as $n_r(\vec{x})$. $I_e^{l^k}$ represents the t^k fine-scale nodes enriched by the TLS model. Kinematic equation used to describe the fine-scale displacement field is then:

$$\vec{u}^k(\vec{x}) = \sum_{j \in I^{l^k}} n_j(\vec{x})\, \vec{u}_j^k + \sum_{r \in I_e^{l^k}} n_r(\vec{x}) RH_r^k(\vec{x})\, \vec{a}_r^k \tag{V.2}$$

In scientific publications, many solutions are proposed to create $\vec{F}_k(\vec{x})$ with the displacement $\vec{u}^k(\vec{x})$, solution of the fine-scale problem k. The simplest solution is to use the fine-scale displacement without further manipulations:

$$\vec{F}_k(\vec{x}) = \vec{u}^k(\vec{x}), k \in I_e^g \tag{V.3}$$

It is the solution that has been used in chapter V.5 (where all nodes are enriched). However, when the blending elements[7] are present, a more accurate solution is possible. Proposed in Gupta et al. (2015) for 3D fracture

[7]elements that have a mix of enriched and non-enriched vertices

simulation with GFEM (similar to XFEM), the Stable GFEM (or SGFEM) method removes from the enrichment function its projection on linear standard finite element space. In the TS context, it corresponds to a fine-scale displacement field projection over the global-scale space removal. The enrichment function becomes:

$$\vec{F}_k(\vec{x}) = \vec{u}^k(\vec{x}) - \sum_{p \in I^g_{sk}} N_p(\vec{x})\vec{u}^k(\vec{x_p}), k \in I^g_e \tag{V.4}$$

where I^g_{sk} is the set of the global nodes used to discretize the global node k support. A test, not presented here, has been done with the problem described in annex H. Similarly, as in annex D.3, the plate has been refined in its center to have some blending elements. It confirms that results with a non-TLS framework are improved with (V.4). Nevertheless, in this test as well as in the TS literature, $\vec{u}^k(\vec{x_p})$ always exists at $\vec{x_p}$ nodes. With the TLS model, it is not anymore the case. The node $\vec{x_p}$ may be in Ω_c and as such eliminated from the fine-scale displacement field. This issue, that can be addressed by taking $\vec{u}^k_s(\vec{x_p}) = \vec{U}_{s-1}(\vec{x_p})$ (with s being the currently considered scale loop and $s-1$ the previous one, see algorithm 3), remains an open question.

Otherwise, note that $\vec{F}_k(\vec{x})$ as it uses $\vec{u}^k(\vec{x})$ is not anymore a dimensionless quantity such as the Heaviside function values. Depending on the chosen unities, the tangent matrix can have some terms with very high or low values compared to other standard non-enriched dofs. The conditioning is then impacted. A scaling of $\vec{F}_k(\vec{x})$ can be a solution. For (V.4), the scaled enriched function becomes:

$$for\ k \in I^g_e \begin{cases} \vec{G}_k(\vec{x}) = \vec{u}^k(\vec{x}) - \sum_{p \in I^g_{sk}} N_p(\vec{x})\vec{u}^k(\vec{x_p}) \\ \vec{F}_k(\vec{x}) = \dfrac{\vec{G}_k(\vec{x})}{\max\limits_{j \in I^{lk}} \left(\max\limits_{i \in \{1,2,3\}} \left(\vec{G}_k(\vec{x_j}).\vec{e_i} \right) \right)} \end{cases} \tag{V.5}$$

with $(\vec{e_1}, \vec{e_2}, \vec{e_3})$ an orthonormal basis of 3D Cartesian space.

Using the same test that check (V.4), the matrix conditioning has been improved with this scaling. Discovered later in the literature, a scaling SGFEM has already been proposed in Sillem et al. (2015) where the authors call it the scaled SGFEM or sSGFEM. An illustration of this enrichment function is given in annex G. Note that this scaling factor is a good indicator concerning the enrichment pertinence. If it is close to zero it means that $\vec{u}^k(\vec{x})$ can already be represented in the global-scale space. Moreover, in this situation, the global node k must not be enriched to avoid a singular problem (independently of the fact that dividing by zero for scaling is not possible). It has allowed switching off enrichment automatically, when using (V.5).

Introducing the TLS model in the TS method is still a work in progress at time of writing this thesis. The interaction of the ramp Heaviside enrichment at a coarse level (not parallelized yet) with the TS enrichment is still subject to questions. Those two enrichments overlap. It can even be chosen to superimpose them (by forcing the full overlapping). The idea would then be to remove, not the projection (such as in SGFEM), but a piece-wise linear projection, to $\vec{u}^k(\vec{x})$ so that the contribution of the global-scale ramp Heaviside enrichment is removed from $\vec{F}_k(\vec{x})$. This idea has been proposed with a different approach (not a piece-wise linear projection but a double Heaviside function application) in Sanchez-Rivadeneira and Duarte (2019). Note that such a method can alleviate the issue pointed above with the problematic global nodes in Ω_c.

V.4.5 Feed back

The first thing to mention is that finding a reference to compare TS enrichment functions and test ramp Heaviside/TS overlapping is not easy. Using the first-order frontal approach as a reference is difficult: first, since the SuperPatch is built on a subset of global elements, the refined mesh can differ in the frontal and the TS problem; second, with the SGFEM enrichment a second-order approximation is introduced by this function that first-order reference will not be able to represent. Moreover, no simple mathematical reference has been found yet. Nevertheless, many tests (one of them is used as an illustration in annex G) have been conducted using a symmetric version of II.5 (i.e. $\beta = 1.$) and using a fixed ϕ (no evolution). These tests permit to check:

- That the TLS ingredients are well introduced in the TS framework using the SuperPatch.
- That the scale loop does converge to a similar solution to the one of an equivalent problem treated by a frontal strategy.
- That the choice of the TS enrichment functions does influence this convergence. Finding the right function is an essential point to be able to progress on this subject.
- That the parallel features proposed in chapter V.4.3 does work.

This last point is illustrated in the next chapter.

V.5 Uniform test case benchmark

This test is an artificial benchmark dedicated to comparing the strategies performance of chapter V. All detailed information for this test case are given in annex H. It corresponds to a simple plate under a known complex loading that gives a way to compare scalability of:

- The frontal strategy (chapter V.2) called "frontal" in this benchmark.
- Some form of the active zone strategy (in fact it corresponds to the enhanced version with many domains proposed in chapter V.3.2) called "domain" in this benchmark.
- The TS strategy (chapter V.4) called "two-scale" in this benchmark.

This plate, more or less refined, has to be seen as a 3D crack tip with a far simpler strain field without discontinuous displacements. The simulation is intended to evaluate strategies performances when the crack tips become spatially very wide. The overall part and the crack wake discretization are de-facto neglected in this study. Their real impact on the performance depends on the strategy. Let us hereinbelow use the term "reality" to refer to a complete TLS simulation.

With the TS strategy, neglecting the part and the crack wake is rather true because the global level is in charge of them. And normally it is expected to remain a fast step even if the crack wake global-scale integration is impacted by the fine-scale discretization. So, in this benchmark, the "two-scale" results are rather close to the reality.

Regarding the frontal strategy, adding overall part and refined/coarsened crack wake discretization implies a larger system size to be solved. So, in this benchmark, the "frontal" results are optimistic compared to the reality.

For the active zone strategy, what has been tested is the decomposition of the active zone itself. Each process holds a domain which is condensed. The resulting distributed condensed dense system is solved with a home-made parallel Conjugate Gradient iterative solver, with a Block Jacobi preconditioner. Note that using such iterative resolution is possible in this benchmark due to the right problem conditioning. In reality, this conditioning aspect, as already mentioned, can be an issue for an iterative resolution. Apart from this point, the Schur complement size (i.e. condensed problem size) should be increased because of the interface between the AZ group and the fixed group (corresponding precisely to what has been neglected in this benchmark). Besides that, the time for the fixed domain condensation should be added. So, in this benchmark, the "domain" results are optimistic compared to the reality.

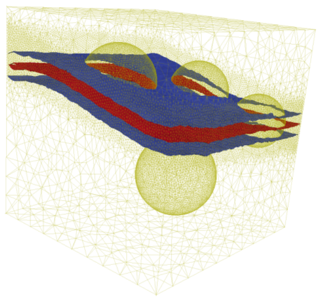
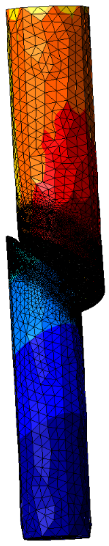
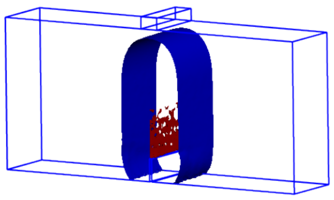
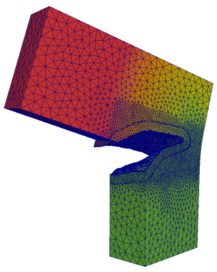
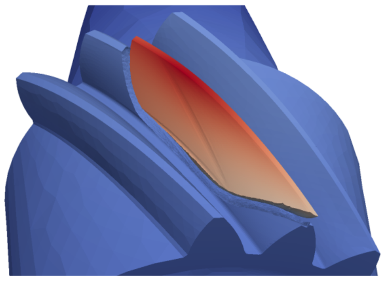
Note that for the "domain" approach, fixing only one domain per process looks rather unfair: reducing domain sizes is only done by increasing the number of processes, which can slow down the iterative resolution at some point. Maybe, a more efficient implementation would have used an independent number of domains and processes (i.e a process may handle one or more domains). It would have permitted, for example, to have many domains treated in sequential, and obtain an efficiency for a constant domain number. Due to the complexity of the implementation as well as lack of time, only the one domain per process solution has been tested. It already gives some fruitful information.

What we can retain from results given in annex H.3 is the following:

- The TS strategy both for strong scaling and weak scaling has better efficiency than the two other strategies.
- Modifying the active zone, using many domains looks promising because of an excellent strong scaling efficiency compared to the original strategy that can be assimilated to the "frontal" results in this benchmark (i.e. MUMPS is used in parallel to factorize matrices, in both cases). A multi-domains per process strategy may change the observed results positively.
- The TS strategy even with small problem, still has a good strong scaling capability.
- The most critical case shows that the TS strategy has a good strong scaling efficiency up to the point where:
  - The parallel global-scale resolution slows down the computation. The resolution scalability corresponds to the performance of the direct sparse solver. Due to the global level moderate problem size, using the same distribution for both scales is then not efficient. Gathering a global level problem matrix on a smaller set of processes can provide a way to use more efficiently the direct solver with hopefully a small computational effort spent in communicating.
  - All the patches are themselves distributed which looks like playing a role int the performance.
- As expected, the sequential TS strategy is slower than other strategies even if we can notice that when the fine-scale discretization grows, this assertion becomes less valid. Up to the point where lack of memory becomes an issue. And in this case only the TS strategy can obtain a sequential resolution. The same ascertainment may be done with a moderate parallelism and small problems: with few processes, a frontal strategy defeats, in absolute terms, the TS strategy.
- For a given strategy, the energy error which is computed in this benchmark is independent of the number of processes used. This evidence only proves that there is no bug in the implementation.

- As expected, the size jump between scales in the TS strategy drives the computational performance. If the jump is high then the patch problem size will be large and can reach the computational effort of a global frontal approach. That is the reason why in Kim et al. (2011), the authors use a multi-scaled strategy so that the size jump between levels remains small: A global level has "small" patches, themselves considered as the global level of new TS strategies, themselves having "small" patch ... In this thesis, we only use one scale level because:
  - The distributed context with a distributed patches resolution is expected to be equivalent to the use of many scale levels if the biggest patches (i.e. those with a large problem size) are distributed over more than one process. This equivalence need still to be proved. However, we can consider that this proof is almost already given through this benchmark. Using extra scales or using distributed patches resolution is similar, respectively to the use of the TS strategy or the use of the frontal strategy. Regarding small to moderate size problems with few processes it turns in favor of a frontal solution.
  - It simplifies this first testing implementation in a distributed context. Maintaining local and distributed patch at all scales adds a lot of extra work to afford the right load balance.
- The amortization of the TS strategy computation over several loading steps has been studied in this benchmark by recovering the time elapsed by iteration of the scale loop. This quantity is not counting the factorization step of fine-scale problems resolution that can be the same between non-linear iterations (chapter V.4.2) or load steps (ϕ constant in the patch). Considering this measure reveals the potential of the TS strategy compared to other strategies even for small size problems.

In conclusion, this benchmark has comforted the idea that using a TS strategy appears as a valuable solution for the TLS 3D simulations targeted by this thesis.

Chapter VI
Simulation gallery: confrontation to complex 3D problems

This chapter illustrates the capabilities of the Thick Level Set model in 3D. However, a full simulation campaign (with convergence rate analysis, l_c testing, ...), already well studied in 2D (Bernard et al. (2012),...), was not carried here in 3D. Only a few qualitative and quantitative performance of this method compared to some experiments and other numerical models are presented.
A first simulation corresponding to a cube with spherical holes, under traction, starts presenting the TLS model features, such as the automatic damage initiation or the merging capability. Then, a twisting chalk simulation gives a first clue to the correct quality of this method when compared qualitatively to the experiment. After, a more quantitative comparison to experimentations is given by a set of notched beam simulations. With a different setting of the TLS parameter, those beams are loaded under three points bending conditions. Next, an L-shape simulation with modes I+III loading gives a way to compare with another numerical method qualitatively. Finally, a spiral bevel pinion gear of a helicopter transmission system, is studied under a critical loading. It allows some good comparison with tests and simulations carried out with a more complicated loading.
Note that, most of those simulations have been used to illustrate the computational performances of the AZ strategy exposed in chapter V.3.

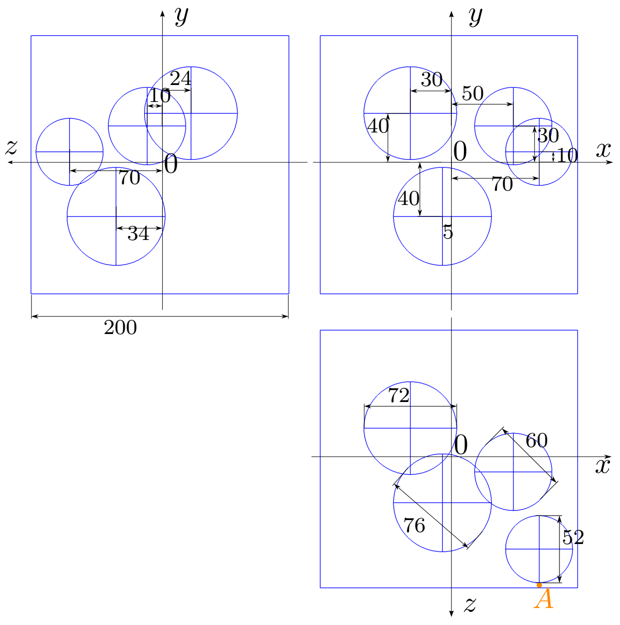


Figure VI.1: Spherical holes test case (dimensions in mm).

This chapter slightly modified in this thesis have been published in Salzman et al. (2016)

All material characteristics and simulation settings are given in annex J. Annex I gives the computational cost of those tests. Note that most of those tests use the AZ strategy (chapter V.3) to solve the problem. Moreover, none of those simulations use the mesh adaptation strategy of chapter IV as explain in section V.3.2. The DCA (chapter III) is used in all simulations to obtain Γ_c.

VI.1 A cube with spherical holes

This test case shows the capabilities of the TLS framework in terms of initiation and merging of crack. A cube with 4 embedded spherical holes (Figure VI.1) is in tension with a uniform loading condition applied on 2 faces (perpendicular to y-axis).

The simulation starts with an undamaged model. In the first initial step, a search is performed to find the location of the maximum damage criterion. A spherical iso-zero level set is put at this location introducing a small damaged zone. It is added around the point A (figure VI.1). Then at each step, a search for an additional location that locally violates the damage criterion ($Y \leqslant Y_c$), is done outside the already initialized damaged zone. As the load factor is still slightly increasing during this part of the simulation, extra local zones

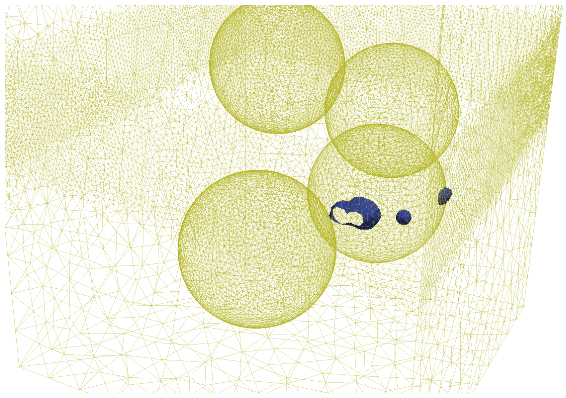


Figure VI.2: Spherical holes test case: initiation locations at step 4.

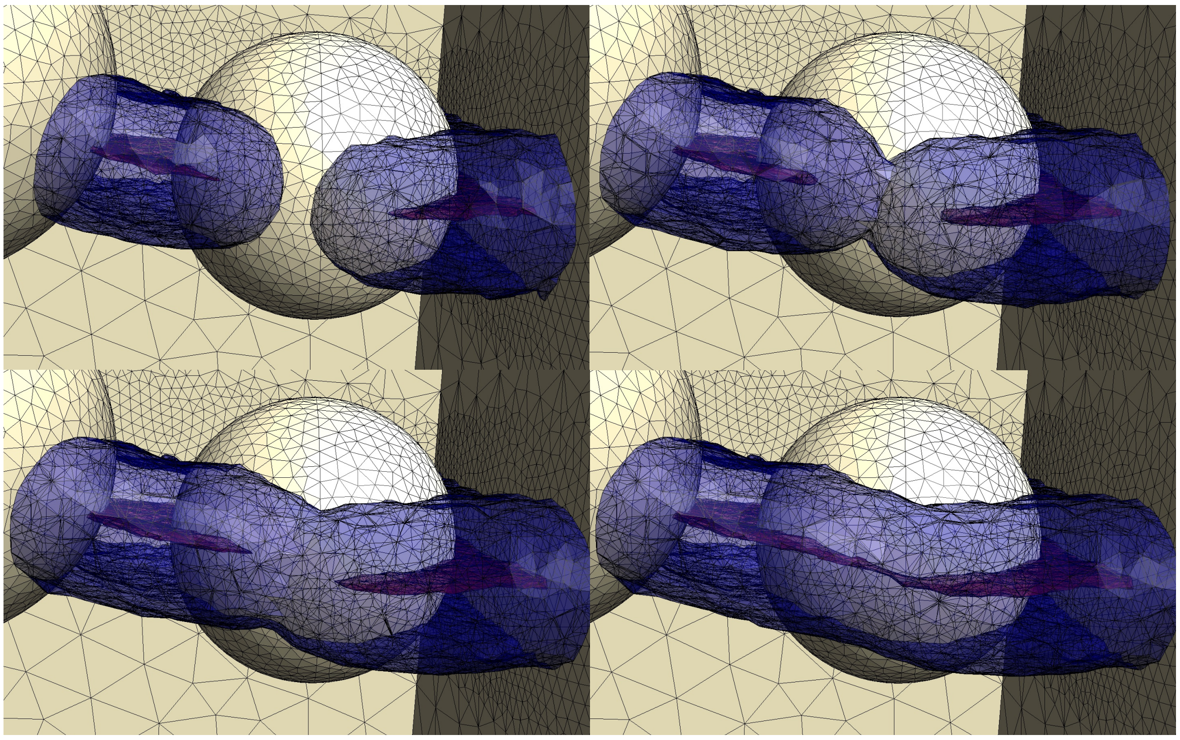


Figure VI.3: Spherical holes test case: merging (inside view with only skin mesh).

are discovered at almost every load step (see Figure VI.2 at step 4) while propagation is enlarging 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 5 steps, the load starts decreasing and at step 10, a crack appears (i.e. Γ_c appears). From this point, there is almost no more additional location violating the damage criterion found during the simulation.

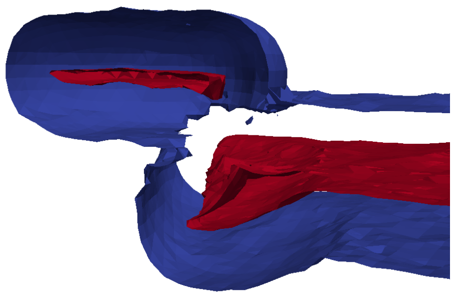


Figure VI.4: Undocumented test case (one crack diving under the other): presenting of the correct partial merging of Γ_0 fronts but not Γ_c fronts.

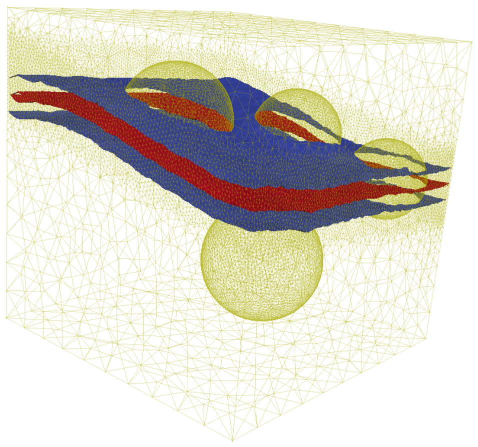


Figure VI.5: Spherical Holes test case: iso surface after 336 load steps, Γ_0 in blue, Γ_c in red.

As given in table J.1, we are considering, for the cube example, a damage law without hardening (corresponding to figure II.3). It is clear that the hardening would first yield to a diffuse damage in the cube before any localization. Hardening was already considered in the TLS framework by Van Der Meer and Sluys (2015) and Moës et al. (2014).

During the propagation, one can observe the merging capabilities offered by the TLS. Figure VI.3 exhibits the behavior when, after turning around the first spherical hole, the fronts are joining together. First iso-zeros of both sides of the front merge (top-right view). As the distance from the hole is less than l_c, the propagation continues with a rather steady location of Γ_c (bottom-left view). The propagation continues and the front gets far enough from the hole (more than l_c) which generates a quick coalescence of the Γ_c front (bottom-right view). Here, the fronts were colliding in a rather straight-

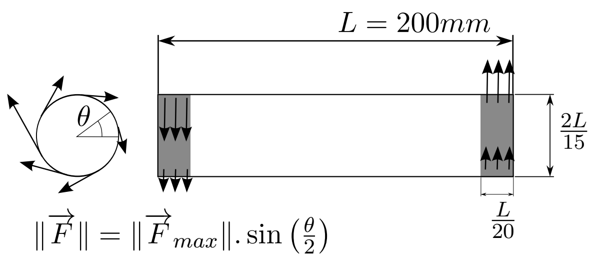

$L = 200mm$
θ
$\frac{2L}{15}$
$\frac{L}{20}$
$\|\overrightarrow{F}\| = \|\overrightarrow{F}_{max}\|.\sin\left(\frac{\theta}{2}\right)$

Figure VI.6: Chalk test case.

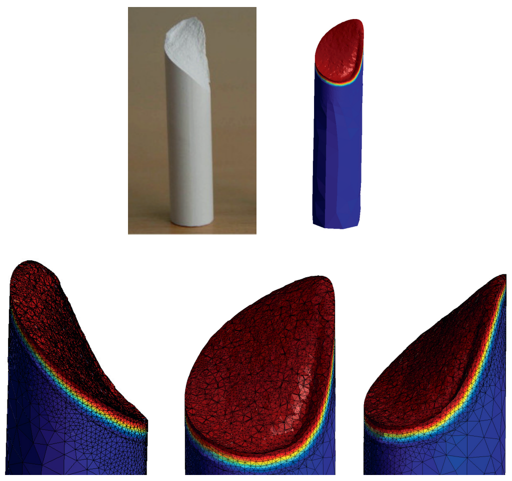


Figure VI.7: Chalk comparison of the final state between simulation and experiment presented in Bordas et al. (2008) (Reprinted from Bordas et al. (2008) Copyright (2007) with permission from Elsevier).

forward manner as both sides are more or less at the same y position (vertical of picture). Note that the merging of Γ_0 fronts does not imply the automatic merging of Γ_c fronts. Figure VI.4 illustrates this scenario.

The final crack location is given in figure VI.5 after 386 load steps. The cube is split into two components.

VI.2 Chalk under torsion

This test case was studied in Bordas et al. (2008) with a meshfree method. A cylindrical chalk bar is twisted with two opposite torques at its ends. Geometry and loading are presented in figure VI.6. The loading is applied along the tangential direction of the chalk surface. To follow Bordas et al. (2008), the loading magnitude depends on the angle θ defined in figure VI.6.

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

Table VI.1: Three-point bending test of a notched beam (dimension in mm).

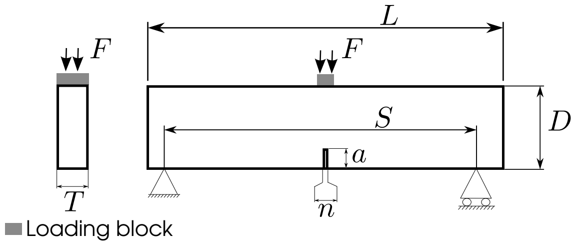


Figure VI.8: Three-point bending test of a notched beam: generic geometry and boundary conditions.

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 Γ_0 small sphere with a center over the chalk surface at L/2 along the cylinder axis. It allows reducing the time consumption by using a mesh with a refined slice where a crack will start and is expected to develop. Figure VI.7 presents the result of the simulation after 103 load steps, when the chalk is completely split into two parts.

One can observe a good agreement of the crack shape between the experiment and the simulation: the development of a helicoid (bottom-left and middle points of view) is followed by a blunt finish (bottom-right point of view) where the crack lips shape is straighter.

VI.3 Quantitative comparison : Three-point bending test of a notched beam

In this chapter, we try to provide some insight into the quantitative quality of the TLS method by making a comparison with an experimental test. We choose one of Hoover et al. (2013) campaign which provides data for a size effect study on a concrete beam under three points bending conditions. The general beam geometry and loading are depicted in figure VI.8. Two (out of 18) size combinations are chosen from Hoover et al. (2013) campaign : the Bc and Cb cases (dimensions given in table VI.1).

The force displacement curves given in figure VI.9 show the experimental response. The CMOD (crack mouth opening displacement) is measured at the bottom of the beam between two points symmetrically located ac-

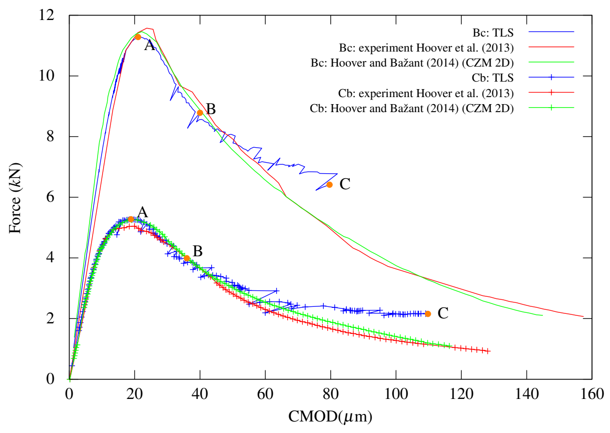


Figure VI.9: Load displacement curves for notched beam case Bc and Cb.

cording to the notch (and separated by 137 mm and 59mm in the test case, respectively Bc and Cb.).

The Cohesive Zone Model (CZM) simulations corresponding to the test case are given in Hoover and Bažant (2014) and are reproduced in figure VI.9.

The idea is to compare the TLS simulations with these two results sets (experiment and CZM). In order to model properly the concrete behavior, the brutal softening regime of figure II.3 can not be used. In order to adopt a different softening regime, we consider the work of Parrilla Gómez et al. (2015) where Y_c now depends on d :

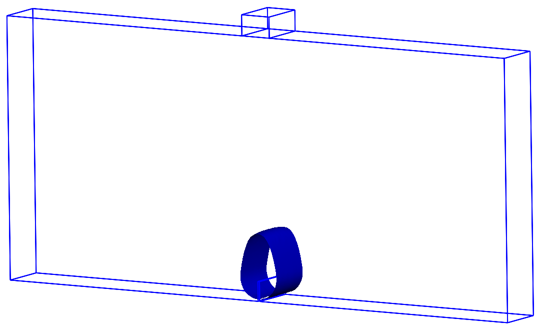
$$Y_c(d) = Y_c^0\ h(d) \tag{VI.1}$$

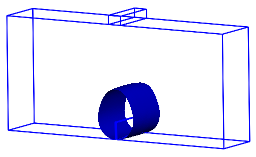
where Y_c^0 is a constant.

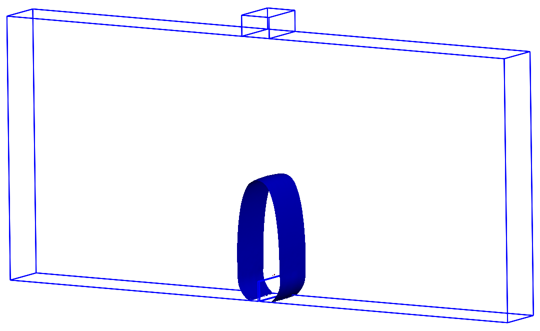
Using Parrilla Gómez et al. (2015), the CZM Hoover and Bažant (2014) parameters are transformed into TLS parameters ($h(d)$ and Y_c^0).

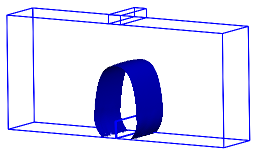
The results given in figure VI.9 show that the TLS simulations are rather close to the test and CZM simulations. The test peak load is obtained with an error of 2.5% and 4.5%, for the Bc (top) case and for the Cb (bottom) case, respectively (Note that the CZM simulations do give errors of 1% (Bc) and 4.1%(Cb) with respect to the test peak load). The last part of the post-peak is in discrepancy with the test. In Parrilla Gómez et al. (2015), the load-CMOD curves on another test case show a more accurate post-peak region. In Parrilla Gómez et al. (2017), a full comparison with Hoover et al. (2013) campaign in 2D, have investigated this aspect and the authors observed better coherence.

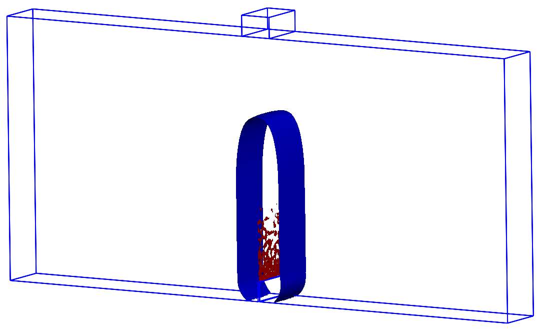
In figure VI.10, Γ_0 and Γ_c are presented for points A, B and C. Those points, shown in figure VI.9, correspond to the curve peak (A), the first appearance


(a) A


(b) A


(c) B


(d) B


(e) C

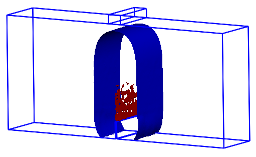

(f) C

Figure VI.10: Notched beam case Bc (a,c,e) and Cb (b,d,f) : iso surface at points A B and C of figure VI.9, Γ_0 in blue, Γ_c in red.

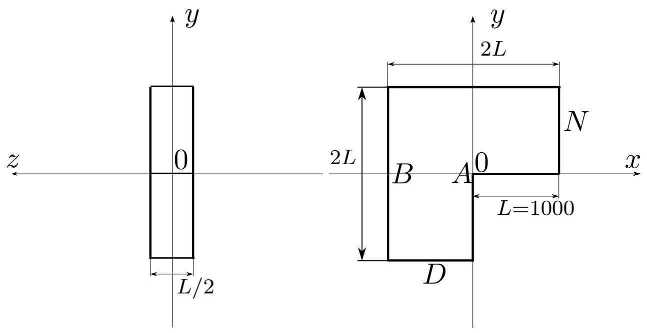
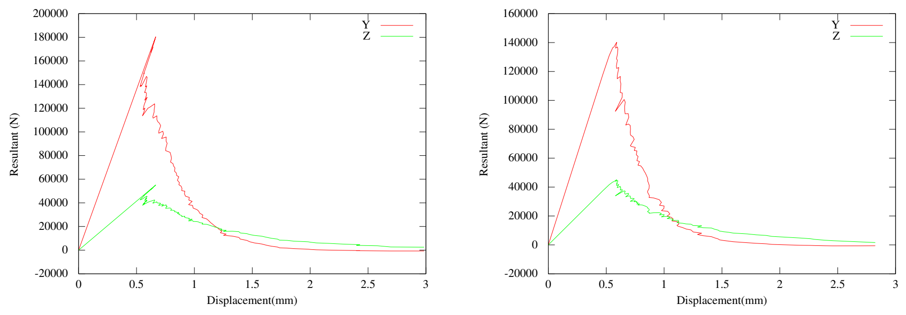


Figure VI.11: L shape: geometry (dimension in mm) and boundary conditions, D face with displacements fixed to 0 in any direction, N face with displacements imposed along y and z directions.



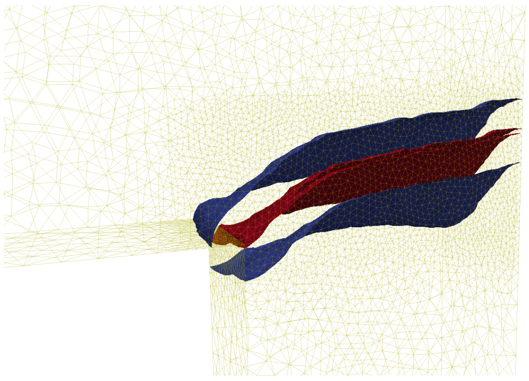
(a) automatic damage initiation (b) forced damage initiation

Figure VI.12: L shape: force-displacement curves (along Y and Z) with automatic (a) or forced (b) damage initiation.

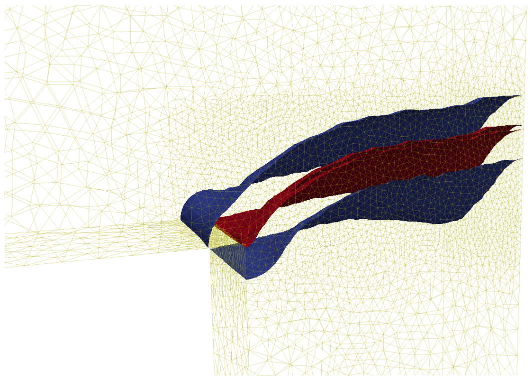
of Γ_c (B) and the end of simulation (C). One can observe that the point B, for both cases, happens quite late in the process. The damage first develops in a very long process zone. In VI.10e, we see that Γ_c appears in a non-uniform manner. It is related to the lengthy process zone where, on the skeleton of ϕ, the damage is almost 1. The transition to a damage equal to 1 with an automatic introduction of Γ_c discontinuity is then hard to softly achieve because a large part of the skeleton can vary abruptly. In those simulations, we chose to slow down the front advance (which gives this non-uniform intermediate state). However, the new TLS model version introduced in Lé et al. (2018) is certainly a better solution to this problem and could also have some effects in the last part of the post-peak curves too.

VI.4 L shape - mode I+III

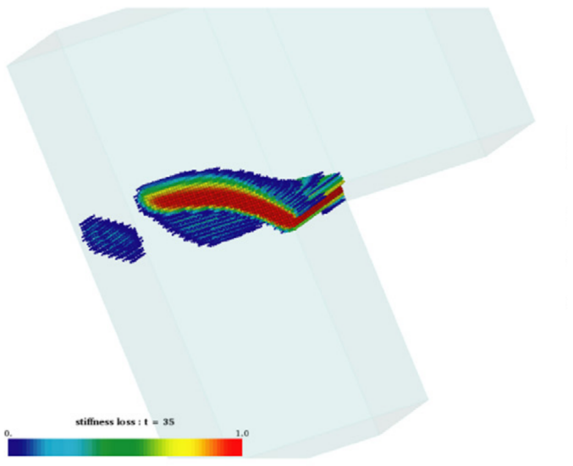
In Lorentz and Godard (2011), the authors modify the standard L Shape - mode I test case by introducing an additional lateral imposed displacement.



(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 compare to this paper

Figure VI.13: L shape: TLS results (a and b), gradient method results (c). TLS results are iso surface at the end of computation, Γ_0 in blue, Γ_c in red.

Consequently, the mode III is activated and the crack path is slightly modified. The geometry and loading are presented in figure VI.11. The mesh is refined around the corner A up to the edge B.

Two types of computations have been conducted: one with an automatic damage initiation and the other one with a forced damage initiation (along the corner, setting an initial cylindrical ϕ of radius $1.05 \times l_c$). This second case is needed to improve the comparison with Lorentz and Godard (2011) results where the authors obtain, at the corner, a rather straight damage distribution. The force-displacement curves are given in figure VI.12 and the final situations are presented in figure VI.13.

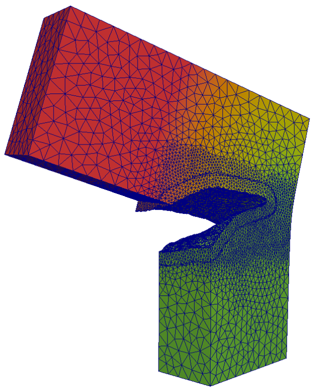


Figure VI.14: L shape: displacement field (scaled) at load step 120 (with automatic damage initiation).

Looking at figure VI.12b, one can see that the forced damage initiation removes the initial steep snap back. The load-displacement curves are similar in shape to those found in Lorentz and Godard (2011). A less brutal softening mechanism such as 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 initial steep snap back even with an automatic damage initiation.

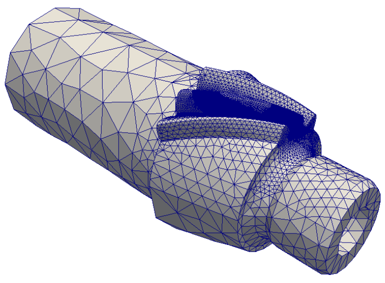
The comparison in figure VI.13 shows that there is little to no difference between the final crack shapes between the two damage initiation strategies, except in the corner vicinity. Compared to Lorentz and Godard (2011), the crack is twisting the same way but the simulation was conducted further in this work as compression is considered in the free energy density (II.5) in Ω_+. This test illustrates the interest of an automatic damage initiation which here allows obtaining a more complex crack path in the corner vicinity.

The displacement field at load step 120 in figure VI.14 illustrates a correct enrichment and representation of Γ_c. The opening of the crack faces is evident.

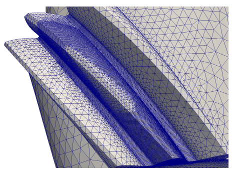
In conclusion, this simulation shows that the TLS model gives equivalent results when compared to another numerical tool which uses a damage gradient method.

VI.5 A spiral bevel gear

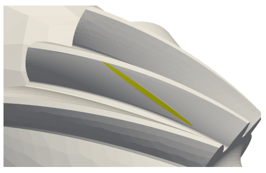
This test case mimics an industrial study on a spiral bevel pinion gear of a helicopter transmission system. It first appeared in a Nasa report (Spievak et al. (2000)) and was published in Spievak et al. (2001). Later Ural et al. (2005) again use this problem with a new computational technique. Those studies were conducted to help the pinion gear designers by giving them a crack



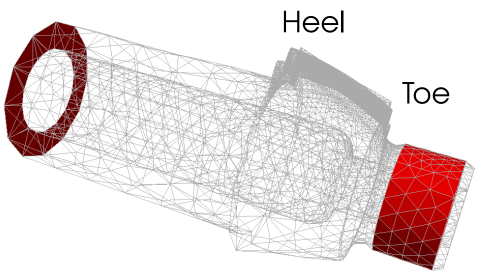
(a) Mesh general view



(b) Central tooth mesh view



(c) Applied loading zone (yellow)



(d) Boundary conditions (red)

Figure VI.15: Spiral bevel pinion gear: mesh, boundary conditions and loading.

path coming from the simulation. The purpose of this test case is to show that the TLS method, with a complex geometry, gives right information about the crack shape without specific mesh refinement and 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 any varying complex loading is done since the testing implementation only deals with a quasi-static loading. The load location corresponds to the highest point of a single tooth contact (HPSTC see Spievak et al. (2000)). An ellipse E on an arbitrary CAD plane P [1] is projected on the tooth to follow the Hertz contact shape. In figure VI.15c, it appears as a yellow zone where the mesh over the surface is following its boundary.

[1]not shown here but roughly located in front of the studied tooth, locally parallel to its face

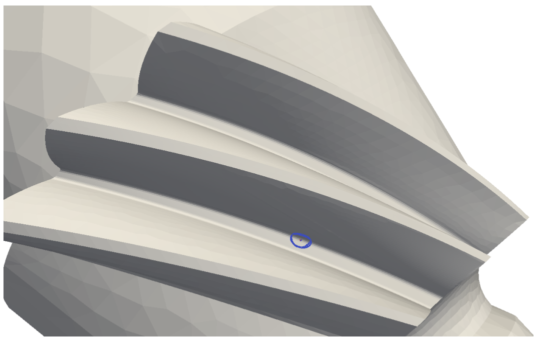


Figure VI.16: Spiral bevel pinion gear at load step 9: in blue Γ_0 and the little red spot is the initial crack location.

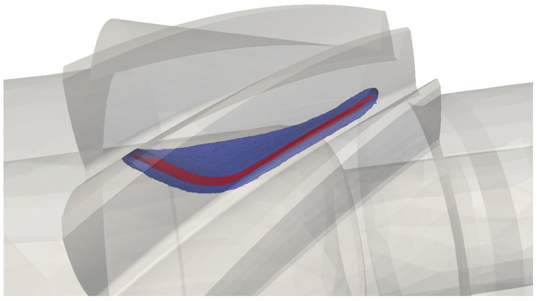


Figure VI.17: Spiral bevel pinion gear at load step 246: Iso surfaces Γ_0 (blue) and Γ_c (red).

The loading magnitude F in this zone is computed with the following expression:

$$F = H_c.F_0$$

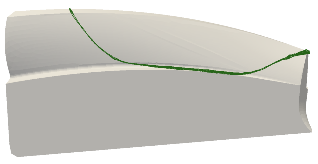
where:

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

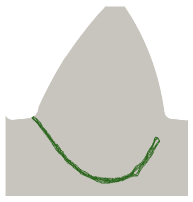
  $$H_c = \sqrt{1 - \left(\frac{x}{a}\right)^2 - \left(\frac{y}{b}\right)^2}$$

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

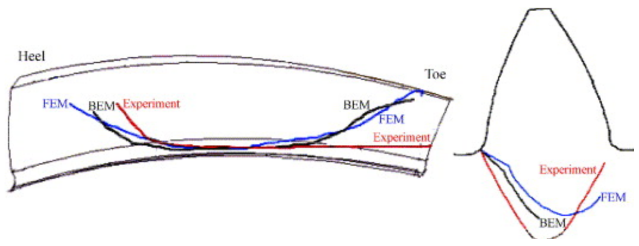
Figure VI.15d shows, in red, the clamped face (ring at the end of the long shaft) and the connection sliding pivot (cylinder over the surface of the smaller shaft). Figures VI.15a and VI.15b show general and focus views of the mesh. The mesh is overall much finer on the central tooth. The element size in the refined zone is at least $\frac{l_c}{4}$. The mesh contains 453 824 nodes.



(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

Figure VI.18: Spiral bevel pinion gear at load step 385: comparison with Ural et al. (2005).

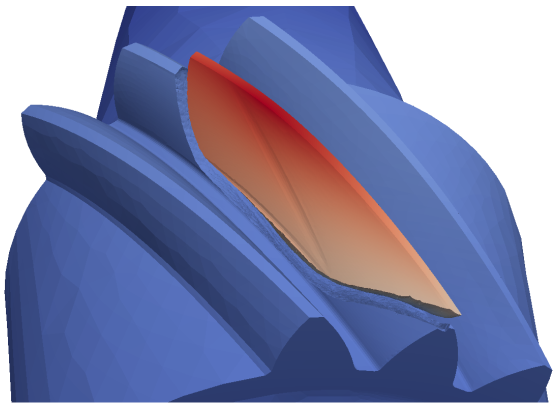


Figure VI.19: Spiral bevel pinion gear at load step 380: displacement field.

This simulation produces the following results:

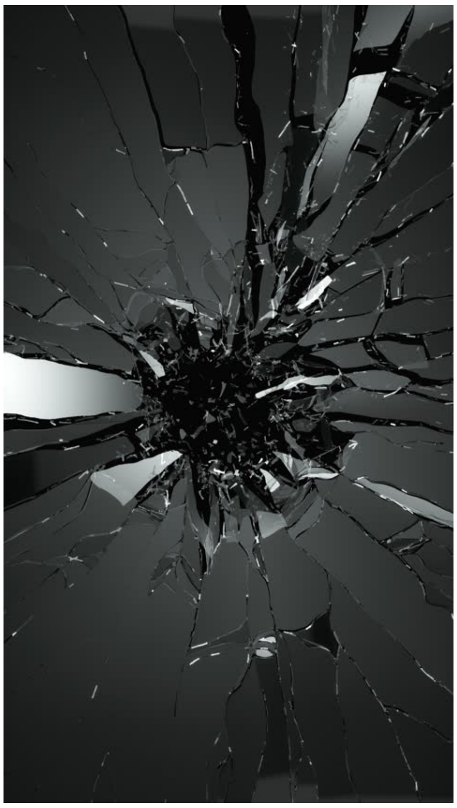
- The (automatic) damage initiation is found in the concave part of the fillet at almost mid-distance (8% shift) from the toe and heel sides (see VI.15d for sides location). It is where Ural et al. (2005) put the crack. Only a small damaged zone is put at this location which makes no assumption about the future crack path.
- The first loading step extends this zone, and at the 9^{th} load step a crack

(Γ_c) is automatically put. It emanates from the damage initiation location (figure VI.16).

- The crack shape is consistent with the experimental results given in Ural et al. (2005) and reported in figure VI.18.

The general situations after 246 and 380 load steps are depicted respectively with iso-surfaces in figure VI.17 and with the displacement field in figure VI.19.

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



Chapter VII Conclusions and perspectives

VII.1 Thesis assessment

The simulation of cracks initiation and propagation in 3D is nowadays still hard to tackle. On the one hand, the mechanical models have their pros and cons regarding the accuracy and the features they provide in a such context. So making the right choice is not simple. On the other hand, computational consumption remains, most of the time, an issue for 3D simulation. In this thesis, we have addressed both concerns.

This thesis first demonstrates that the TLS model is well-designed for the quasi-static simulations of quasi-brittle material in 3D. The presented simulations confirm that the versatility and quality of the model are well-kept in 3D. The new "double cut" algorithm added in this work participates in this quality, giving the right representation of the crack lips without any extra discretization cost.

Regarding the computational performance, this thesis again demonstrates that TLS model is well-designed for 3D simulations since it imposes a very localized constraint for the problem spatial discretization. It has been exploited in the mesh adaptation strategies, proposed in this work, to reduce the problem size and consequently its computational cost. Despite those efforts, the linear system resolution that arises from the non-linear problem treated in the TLS algorithm remains a computational bottleneck. Different strategies have been suggested in this work to overcome this problem. All of them lead to consider parallelism as a natural remedy. In particular, the message passing protocol has been adopted for its performance and hardware flexibility. It induces in all those strategies an element-based partitioning with a distribution of the problem on all available processes. Among the three proposed solutions, this thesis shows that the two-scale strategy offers the best potential in terms of scalability and algorithmic integration within the TLS solver.

Note also that regarding computational throughput, the "double cut" algorithm implementation has exhibited excellent performance thanks to its cutting database. Moreover, a distributed mesh, which implies distributed integration and assembly, has also greatly contributed to reduce the elapsed computation time.

VII.2 Perspectives

An obvious perceptive is to finish the TLS integration in the distributed two-scale method. For that, one will need to determine the right enrichment to use at the TS global level to obtains the best results convergence. Then, one will have to optimize the scale loop integration in the staggered algorithm 1.

All this will be possible only if all components of this algorithm are well parallelized (in particular if the new modal concept proposed in Moreau et al. (2017) is adopted). Regarding the parallelism, hybrid MPI/multi-threaded paradigm should be benchmarked. Finally, the scale size jump, load balancing and SuperPatch update (in particular, the tracking of small isolated damaged zone will deserve some attention) will have to be addressed. On top of all previous steps, one may wish to adapt the SuperPatch mesh using the proposed coarsening strategy of chapter IV.2.

This will meet the mesh adaptation perspectives. On this topic, the crack tip vicinity identification proposed in this work has to be consolidated. The tool that provides Γ_c might give this additional topological information in the future. The offsetting concept should also be improved. As far as the coarsening is concerned, this type of adaptation has been possible, thanks to the "double cut" algorithm. However, some transport issue has appeared in this adaptation, and needs to be addressed in the future. Finally the unstructured mesh adaptation tool created in this work will have to be enhanced to provide better throughput and more functionalities.

In the end, as far as valorization is concerned, continuing the transfer in eXlibris of the code developed in this thesis (distributed "double cut" algorithm version, distributed TS implementation, ...) is expected. If the TLS model integration in the distributed TS method comes to an end, publishing an article on the subject would definitively value the preliminary work proposed in this thesis.

Chapter VIII
Annexes

A
Double cut algorithm details

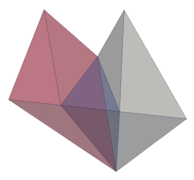
A.1 Element Cutting

Following the scheme given in chapter III.3.2, using Hert and Schirra (2013) for the convex hull computation and Hachenberger and Kettner (2013) for the subtractions, one can obtain, in 3D, with tetrahedron element, 17 reference patterns of element cut as shown in figure A.2. A cut pattern corresponds to a set of edge cut points and a tetrahedron node sign: it leads to a unique topological element cut. The 17 patterns do not take into account the specific metric (single cut are put at the edge middle and double cuts at one-third and two-thirds of the edge).

We can group the 17 patterns into four different categories:

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

In table A.1, the number of possible permutations from each pattern is given. Without close node treatment, a total of 111 permutations has to be handled. With a close node treatment, some polytopes change and are not giving anymore null sub-element for the integration. See figure A.1b for an illustration where one of the negative domain parts reduces to a single tetrahedron producing 1 instead of 3 tetrahedrons for the integration. This is interesting when some specific extra computation is made with those sub-elements (such as some fluid flow in the crack) and when the null volume is an issue. For this work we could have used those null sub-element. However, creating a tool to handle double cut algorithm, we chose to be a little more general from the start, and eliminate null volume sub-elements. It leads to handle 8399 permutations with the close node treatment. From an implementation point of view, a natural choice is to use a database to hold all those patterns. The cutting procedure is then reduced to cut the edges and query the database with the edge cut pattern as a search key. Database


(a) Tetrahedron negative domain polytope splitting: out of initial domain, 3 convex polytopes are created for pattern 13 of figure A.2

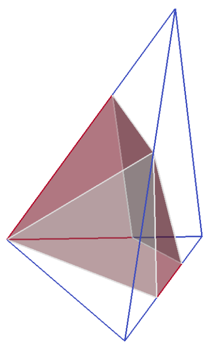

(b) Pattern 5 of figure A.2 with 2 cut nodes from the same edge collapsing and one cut node for two edges collapsing on one node of the tetrahedron.

Figure A.1: Convexity and close node treatment illustrations.

gives then the integration cell and the topological relation for the enrichment identification (number of independent parts of supports).

A.2 Geometrical comparison between single and double cut algorithms

To illustrate how single and double cut algorithms perform, we consider a hammer head shark surface (from Lutz Kettner home page at Max Planck institute: https://people.mpi-inf.mpg.de/ kettner/proj/obj3d/) which is discretized with triangles and plunged into unstructured meshes of increasing number of elements (figure A.3). For each mesh, a level set and a signed vector distance function are computed from the shark surface. An associated cutting algorithm is then used. Fins appear sooner with the double cut algorithm, as expected, since this algorithm detects layers even if they are

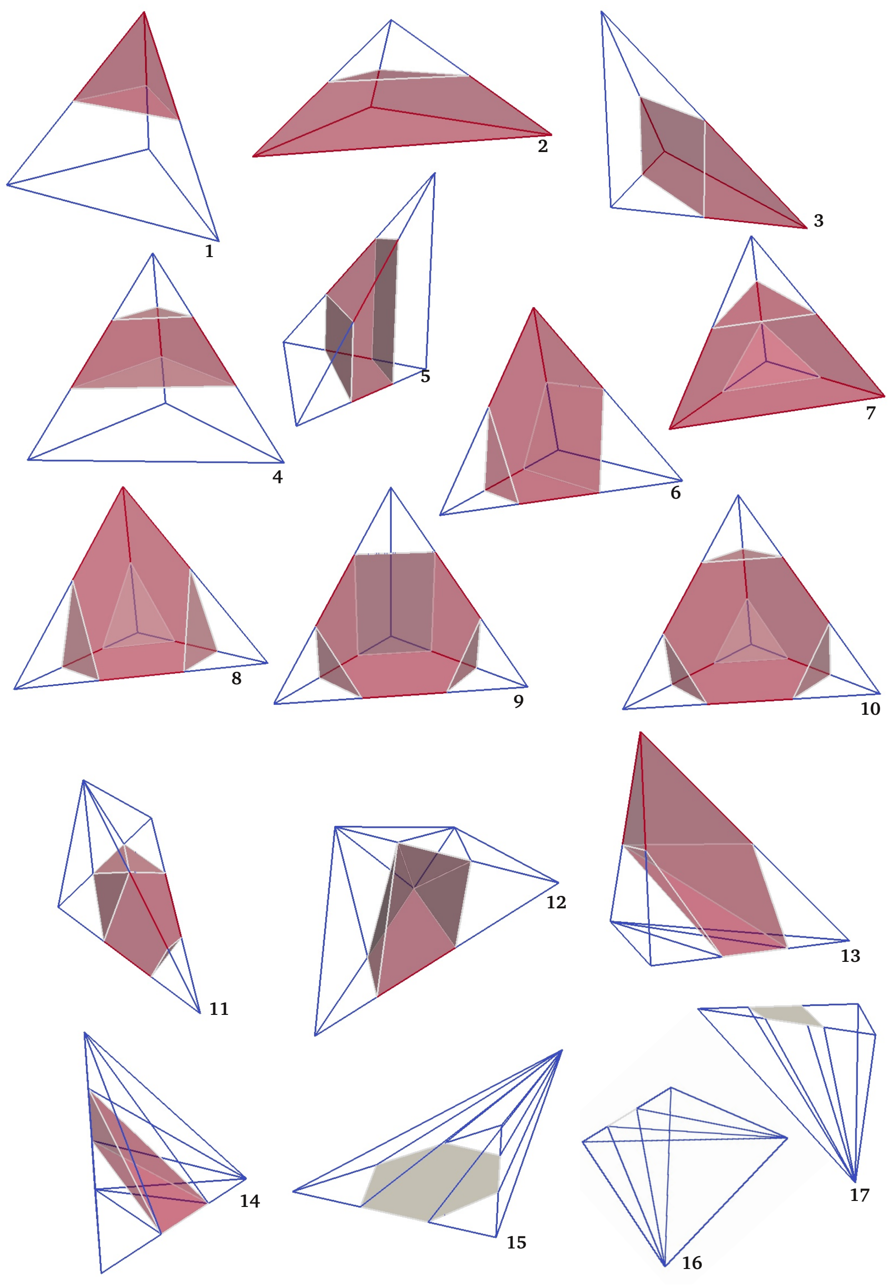


Figure A.2: Tetrahedron topological cut patterns: the red zone corresponds to a positive domain. On each edge, the red segments are in the positive domain, the blue segments are in the negative domain and the grey segments are in the iso-l_c.

	Number of possible permutations with or without close node treatment	
Patterns	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

Table A.1: Number of the possible topological permutation obtained using elemental topological rotation and symmetry.

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

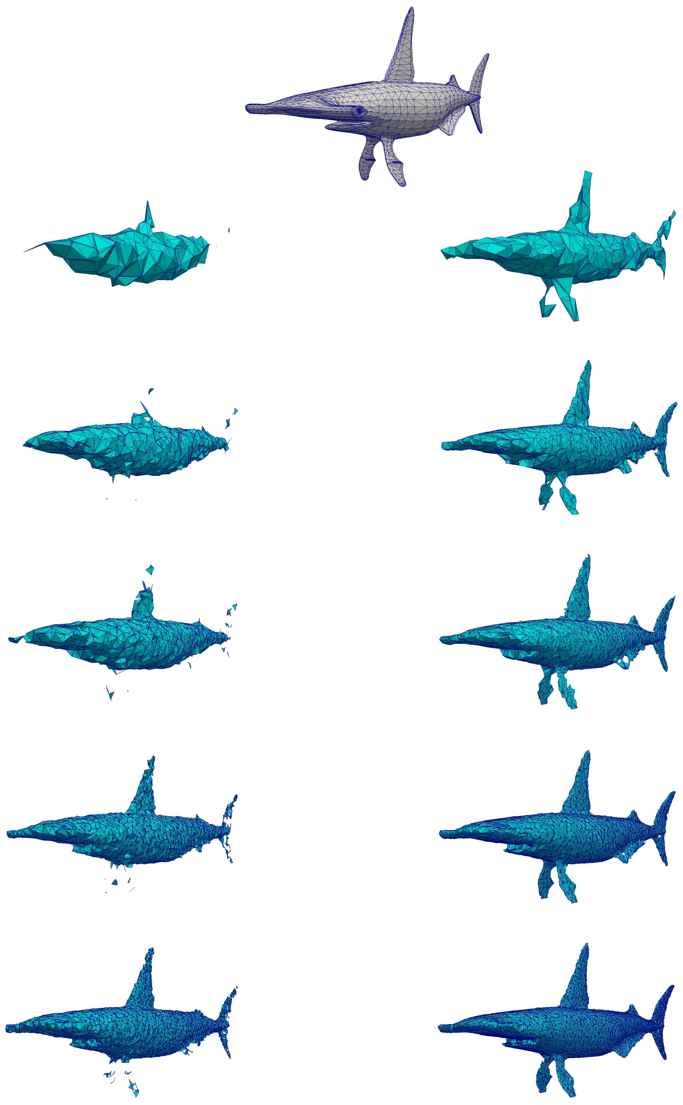
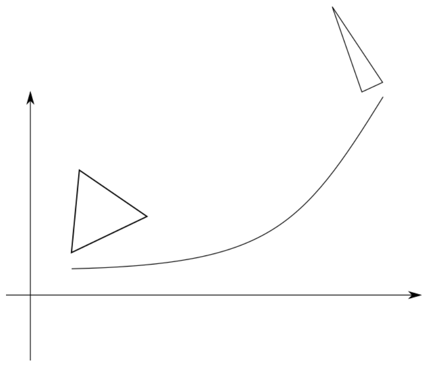


Figure A.3: Iso-contours obtained with a single (left) or double (right) cut algorithm for a hammer head shark shape (reference shape on the top). Meshes used are more and more refined from top to bottom.



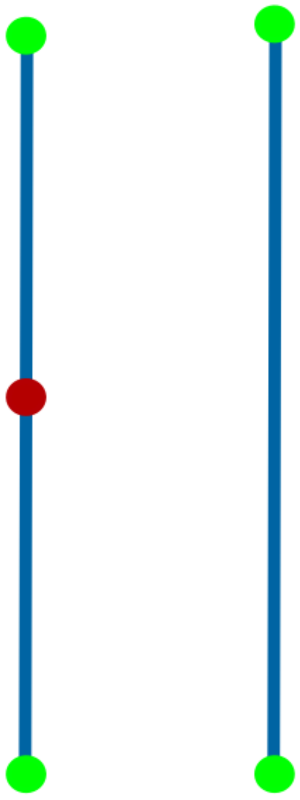
B
Aspect ratio

B.1 Criteria

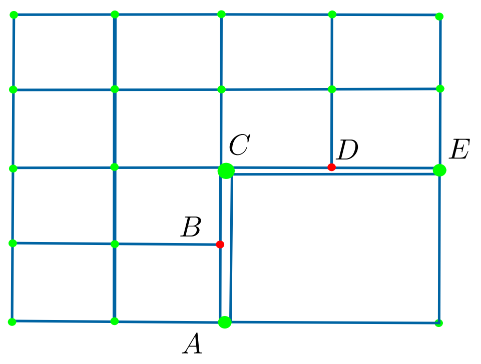
In this work, the used aspect ratio criterion is :

$$\delta = \frac{\max\limits_{i=1,6}\left(length_{edge_i}\right)}{\min\limits_{j=1,4}\left(height_j\right)}\sqrt{\frac{2}{3}}$$

where $length_{edge_i}$ is the length of the ith edge of the tetrahedron and $height_j$ is the tetrahedron height from the jth corner. For a regular tetrahedron of edge length a, the height is $a\sqrt{\frac{2}{3}}$ and then the criterion is 1. The higher this criterion, the worse the quality of the elements (element is rather flat, having edges of very different sizes).



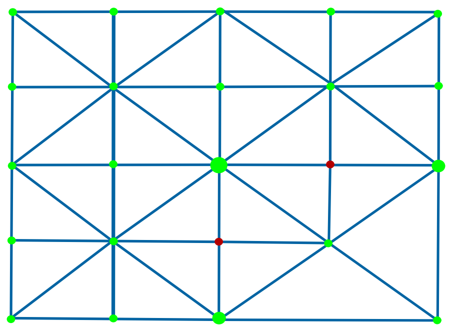
C
Hanging nodes



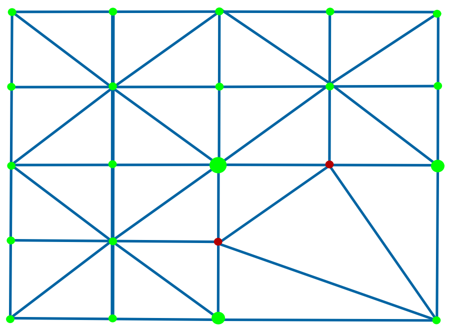
(a) Hanging situation.The edges are shifted for viewing purposes but normally they are superimposed

$$\overrightarrow{U_B} = \frac{1}{2}\left(\overrightarrow{U_A} + \overrightarrow{U_C}\right) \quad \overrightarrow{U_D} = \frac{1}{2}\left(\overrightarrow{U_C} + \overrightarrow{U_E}\right)$$

(b) solution with cinematic constrains (order 1 displacement field example)



(c) Simplex solution with extra center node



(d) Simplex solution without center node

Figure C.1: Hanging example in 2D with hanging nodes on edges

C.1 Origin

If a mesh must be adapted, the element splitting process implies adding nodes in the middle of its edges. It is done independently for each mesh element. However, mid-nodes are merged for edges common to split elements. A hanging node appears when in an edge adjacency, there exists, at least, one element split and one not split. In this case, the mid-node of this edge is said to be hanging. This node is only related to some mesh elements, but not to all the adjacency, which leads for example, to a discontinuous displacement field in mechanics. In 2D, only node related to edge are involved as depicted in figure C.1a. The term hanging edge is then commonly used to talk about an edge that is facing two edges connected to a hanging node.

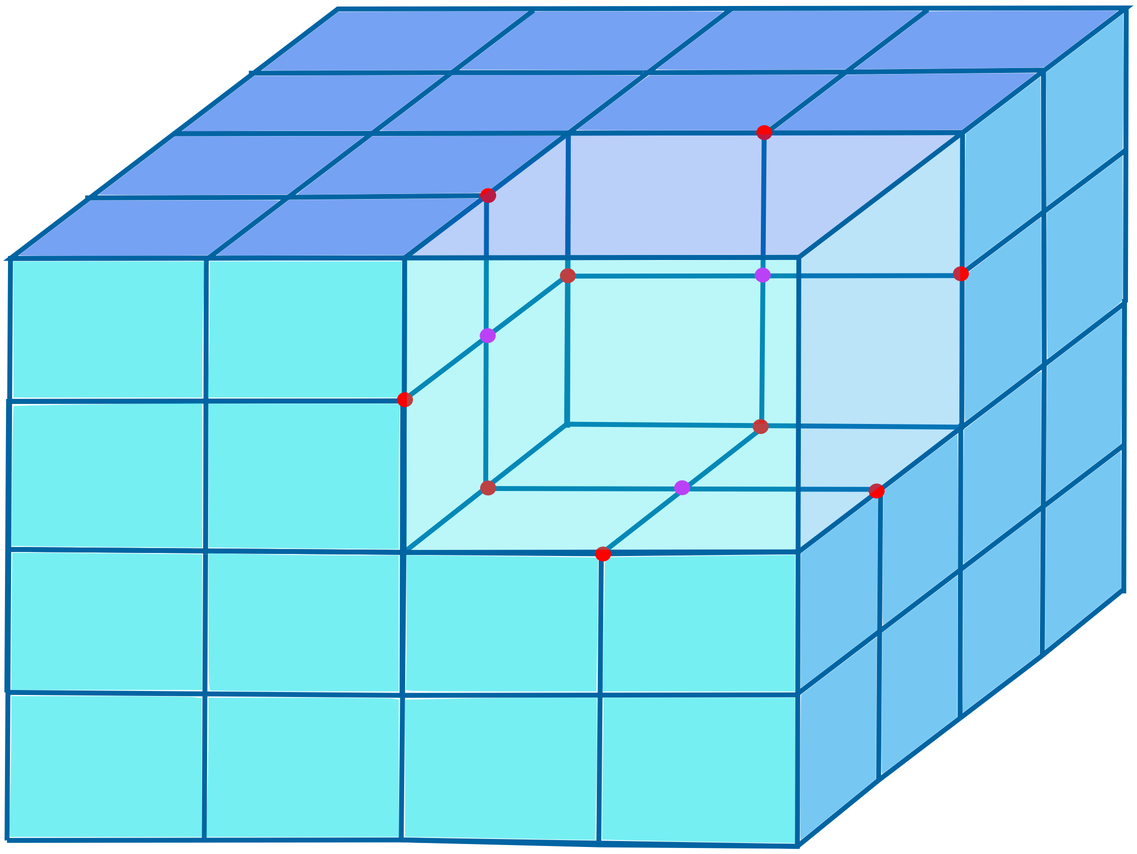


Figure C.2: Hanging situation example in 3D with hanging nodes on edges and faces

In 3D as presented in figure C.2 faces are also designated as hanging when the connected faces to its vertex are related to a node in its middle.

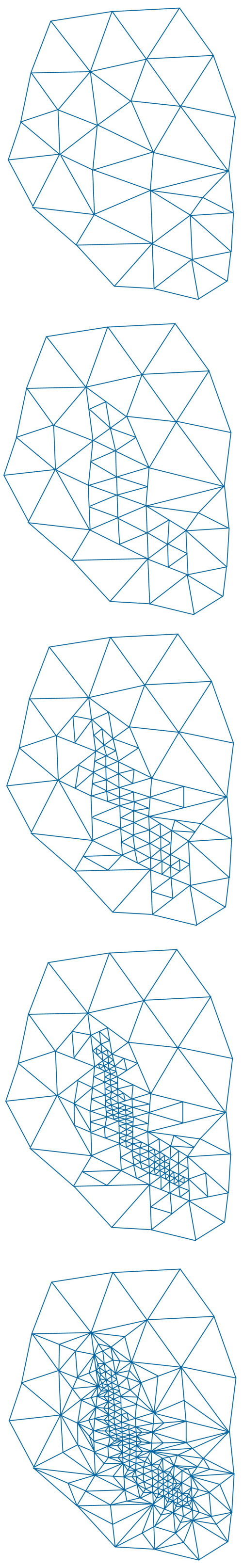
C.2 Treatment

To get a continuous field description, the hanging nodes have to be eliminated.

Kinematic linear relation is a solution that can be applied in all contexts (octree, unstructured, ...). For example, when computing the mechanical displacement field of figure C.1a in 2D, the use of additional equation, given in figure C.1b, eliminates the displacements of B and D. This solution has the following disadvantages: It is costly when elementary matrices are assembled, the final system to solve is densified which increases its resolution cost and it is less obvious to use with higher-order field.

When using a simplex mesh, two solutions can be used, at least. Both are using the simple principle of meshing elements that do have hanging edge or face to connect the hanging nodes with extra elements. The former illustrated in figure C.1c adds a node at the center of the element for that. Then, a simple loop on edges (or faces in 3D) creates one triangle (or 2 tetrahedrons in 3D) or two triangles (or 8 tetrahedrons in 3D) if a hanging node is present. The pros are that the meshing algorithm is simple and works perfectly well in 3D. The cons are the addition of many elements (compared to the second solution). Moreover, in an unstructured context, it gives elements with less interesting aspect ratio. The latter illustrated in figure C.1d uses an ad hoc meshing adapted to the hanging situation. The pros are that it gives fewer elements with a better aspect ratio than the first solution. The cons are that treating all possible cases in 3D is difficult. In Schroeder et al. (2004), they were forced to use a specific constrained Delaunay procedure to achieve this ad hoc meshing.

D
Distributed unstructured mesh adaptation



D.1 General concept

The inputs given by the user to this adaptation tool are a distributed tetrahedron mesh, an optional transport function set and a function that returns true or false if the element given as an argument must be split or not. Split an element in 3D is the action of:

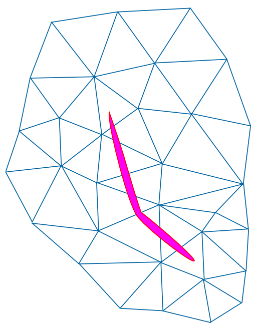
- Creating nodes on the middle of its edges (if they are not already created)
- Replace the tetrahedron by eight smaller tetrahedrons joining original vertex and mid-edge nodes. The four tetrahedrons at the middle (not connected to original vertex) can be created in three different ways depending on the new internal edge choice. An aspect ratio criterion (see B) is used to select the best solution among the three.
- A set of actions to handle the hanging nodes treatment and their associated edges or faces.

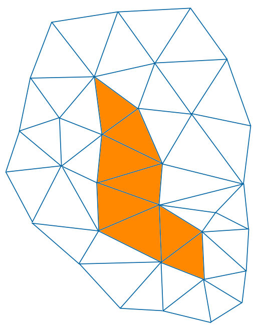
The same kind of actions are possible in 2D with triangles. The user provides the transport function set if he needs to transport information to the newly created element. For example, an initial mesh coloring can be propagated to the new refined embedded mesh by such transport function.

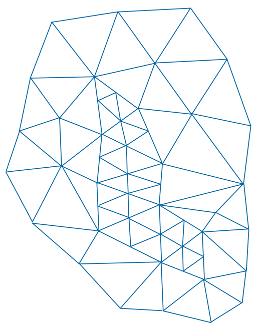
In figure D.1, the global idea of the algorithm in a 2D sequential context, is described with a small example (figure D.1a presents the initial mesh and the fictitious refinement location). The adaptation is made by successive passes that split the elements fulfilling the criterion. The first pass selects all initial mesh elements fulfilling the criterion requirement (in orange in figure D.1b). Then those elements are split and pass 1 gives the mesh in figure D.1c. The algorithm enters a second pass in figure D.1d with the mesh created from the pass 1. It selects the elements based on the criterion (in orange) and the elements that need to be split to fulfill the 2:1 constraint (in yellow). These elements are split and give the mesh in figure D.1e. Same thing in pass 3, selection in figure D.1f and splitting in figure D.1g. Then, in this example, the pass 4 does not detect any new element fulfilling the criterion which stops the pass loop. The adaptation ends, selecting all elements having, at least, one hanging node (figure D.1h) and uses a particular splitting scheme to obtain the final unstructured refined mesh (figure D.1i).

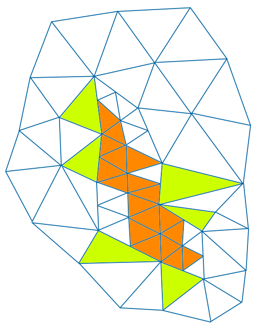
D.2 Parallel algorithm

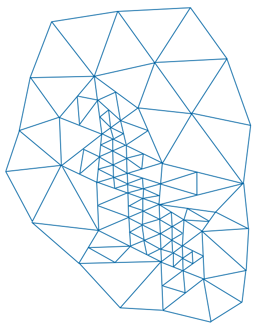
The algorithm 5 as mentioned in D.1 works by successive passes. Compared to the sequential vision given in figure D.1, several communications have to be added to propagate the 2:1 constraint and the hanging situation. For

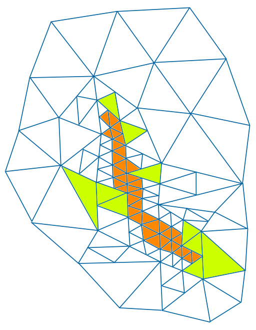

(a) Initial mesh with refinement location

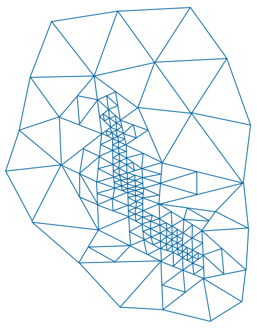

(b) 1st pass select

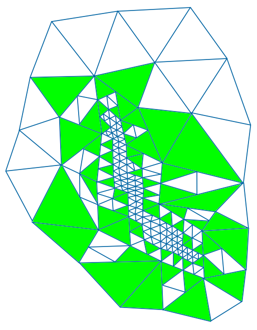

(c) 1st pass: split


(d) 2d pass: select


(e) 2d pass: split


(f) 3d pass: select


(g) 3d pass: split


(h) Removing hanging nodes: select

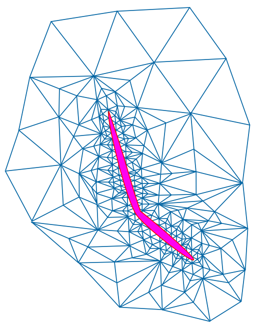

(i) Final mesh with refinement location

Figure D.1: Unstructured mesh adaptation illustration in 2D sequential context. Elements are split because they fulfill user criterion, 2:1 constraint or hanging node treatment. The fictitious refined location is represented in pink.

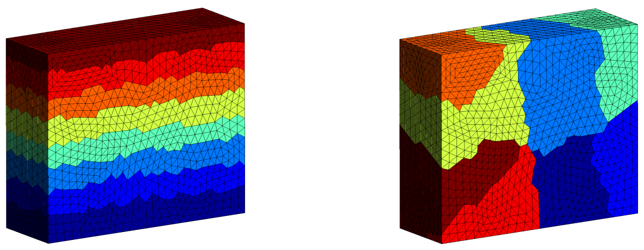
each pass, or level, a set of element ($\mathcal{S}$) is created based on the criterion (user function) that can have been itself updated by the new mesh definition from the previous pass. This set is filled with the elements that need to be split

Algorithm 5 Adaptation algorithm.

```
Starting from a distributed mesh with a set of elements $\mathcal{M}$ for each process
$Nbpass = 0$
repeat
    update criterion against new distributed mesh
    Let $\mathcal{S} \subset \mathcal{M}$ be the initial ordered set of element verifying criterion
    repeat
        repeat
            Let $\mathcal{O} \subset \mathcal{M} \setminus \mathcal{S}$ be the set of elements that must be split to respect one level rule if elements of $\mathcal{S}$ are split.
            $\mathcal{S} = \mathcal{S} \cup \mathcal{O}$, $\mathcal{O}$ being placed first in $\mathcal{S}$
        until $\mathcal{O} \neq \varnothing$
        Let $\mathcal{Q} \subset \mathcal{M} \setminus \mathcal{S}$ be the set of elements that must be split to respect one level rule if elements of remote $\mathcal{S}$ are split.    ▷ communication
        $\mathcal{S} = \mathcal{S} \cup \mathcal{Q}$, $\mathcal{Q}$ being placed first in $\mathcal{S}$
    until $\mathcal{Q} \neq \varnothing$
    Split elements of $\mathcal{S}$ following $\mathcal{S}$ order.
    Create $Nbc$ the maximum number of elements refined across all process    ▷ communication
    Clean and update distributed mesh and $\mathcal{M}$    ▷ communication
    $Nbpass = Nbpass + 1$
until $Nbc \neq 0$ and $Nbpass < MaxNbPass$
Remove hanging node by specific splitting    ▷ communication
```

to respect the 2:1 constraint if element of $\mathcal{S}$ are split. Two loops do it. The first one, local to each process, adds a new element to $\mathcal{S}$ via an intermediate set $\mathcal{O}$ filled by inspecting the neighbor of $\mathcal{S}$.The drained elements from $\mathcal{O}$ are added first in $\mathcal{S}$ so that elements are treated in reverse order of dependence. The first split elements are then not concerned by criterion but their treatment allows splitting those that are concerned, without breaking the 2:1 constrain. This first loop ends when no more elements are in $\mathcal{O}$. The second loop which embedded the first one, works at inter-process level. On process boundary, the elements that are put in $\mathcal{S}$, must inform their adjacent remote element that they will be split. After the communication, a set $\mathcal{Q}$ of new elements to be split in order to respect 2:1 constraint on process boundary, is created. It is again drained and added first in $\mathcal{S}$ to respect the dependency. This second loop will end if $\mathcal{Q}$ is empty for all processes.

After setting $\mathcal{S}$, the effective splitting is done following $\mathcal{S}$ order to re-



(a) Initial strip (b) Initial ParMetis

Figure D.2: Test mesh dispatching on 8 process. Each color corresponds to a process owning this part of the mesh

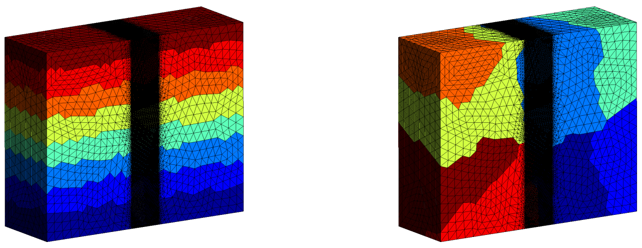
spect the dependency. Some communication is mandatory to clean the newly modified mesh and count Nbc, the maximum number of split elements among all processes. If Nbc is null, it means that criterion did not identify any new element to split and this level ends the pass loop. Another test is added to avoid an infinite loop if the user gives a wrong criterion function: The number of passes is limited by a $MaxNbPass$ value that may be changed by the user.

Then, all hanging nodes are removed using a the central node strategy. The idea is to split the connecting elements (elements that have, at least, one hanging edge or face) by adding in their center of gravity, a node used to construct the internal tetrahedrons appropriately connected to each of their faces. Unfortunately, this is not preserving the mesh quality and a case by case algorithm (where all cases are optimized so that tetrahedrons can fill them without the addition of an extra central node) may give better elements. In this thesis, we did not push further investigations on this topic (see also annex C).

The sentence "update criterion against new distributed mesh" in pass loop of algorithm 5 means that the user must provide an extra function to cope with the elements newly created in the mesh. Depending on what is used by the criterion function of the user a criterion transport can be mandatory for newly created elements from a previous pass. Otherwise, the criterion function can not give the right answer for those elements. It can be a source of problem if the criterion is not easily transportable.

D.3 Performance

To test the performance of algorithm 5, a parallelepiped is meshed with tetrahedrons. It has the same geometry as the case presented in figure H.1. A uniform refinement would have had no interest. No unbalancing would oc-



(a) Refined strip (b) Refined ParMetis

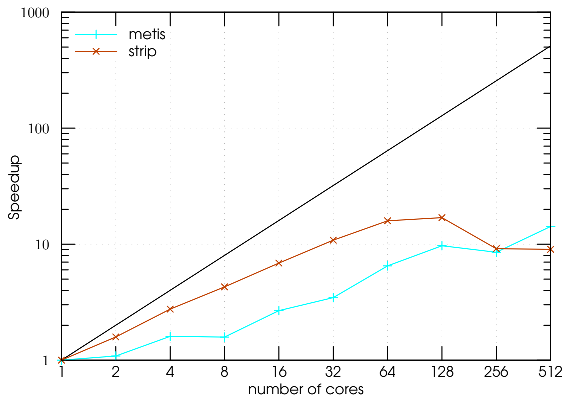
Figure D.3: Refined mesh from both dispatching on 8 process. Each color corresponds to a process owning this part of the mesh

cur. All processes if the mesh was distributed following a ParMetis partitioning, would have had roughly the same number of element to splits. No 2:1 constraint would have appeared. A quite perfect scaling would have been expected. So, the chosen refined zone is a strip passing through the middle (along the y-axis) of the plate. The user criterion function forces the elements to be refined 4 times at maximum in the band. With this refinement, the scalability is intuitively less obvious. Some processes can be offloaded (no element to split) and some others overloaded. It depends on the initial mesh distribution.

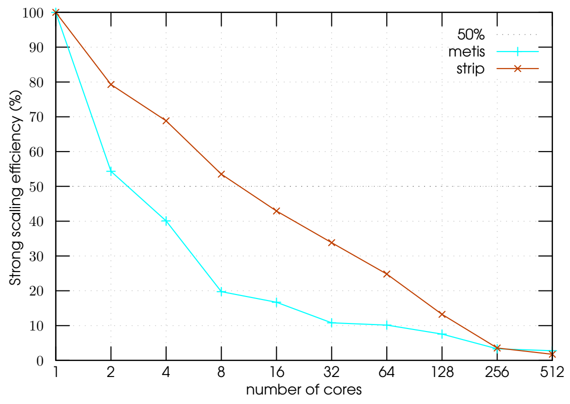
In this test, two different initial distributions are used. Figure D.2 presents those two distribution in the context of 8 processes. In figure D.2b the initial mesh is distributed using the conventional ParMetis partitioning. In figure D.2a, the initial mesh is distributed by a horizontal roughly equal strip (basic algorithm far from perfect). This last case is expected to give a better load balancing. The refined zone, perpendicular to those strips, is expected to be dispatched in all processes.

The obtained results are presented in figure D.3. Figure D.4 presents the strong scaling results for both initial distributions. The elapsed time curves presented in figure D.4c show that adding process does reduce the time consumption up to a point where the consumption increase. The striped distribution gives a more regular decrease than the Parmetis distribution. In figure D.4a and D.4b, the speedup and the efficiency confirm that the Parmetis distribution gives bad results which is quite natural. The load is not balanced in this case. We can check this aspect in figure D.4d and D.4e that presents the initial (before refinement) and final (after refinement) number of elements per process. The curves in those figures correspond to the averaged value across all processes. It turns to be the same for both distributions, because it corresponds to the global numbers of elements divided by the number of processes. More interesting is this quantity range (minimum and maximum). In figure D.4d, the initial min/max zone is close to the curve with the ParMetis distribution. ParMetis is correctly doing its job by giving a quite uniform dispatching of elements. For the strip distribution, up to 32 processes, the dispatching is correct. However, for 64 process, the stripes are incorrectly created (some process have no element). In figure D.4e, the ParMetis distribution exhibits its weakness with a large distribution unbalancing of refined element.This imbalance begins immediately upon the use of 2 processes. With 8 processes the unbalancing grows explaining a complete loss of efficiency at this point. For the strip distribution, the unbalancing appears later with 32 processes. However, in fact with 16 processes the stripes are already not perfect. It explains why the efficiency is maintained up to 40% until 16 processes are used. A better strip dispatching based on a ParMetis weighted partitioning may give a better efficiency.

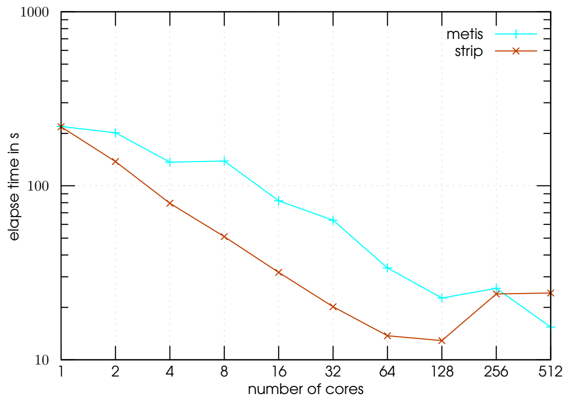
In conclusion, as shown by those curves, the right strong scaling of algorithm 5 is difficult to obtain. Nevertheless, communication is not an issue. With 512 processes, the elapsed time remains in order of magnitude of the best obtained values. The intrinsic algorithmic aspects are certainly to be revisited. However, an appropriate initial mesh distribution choice is essential from a performance point of view for this kind of tool.



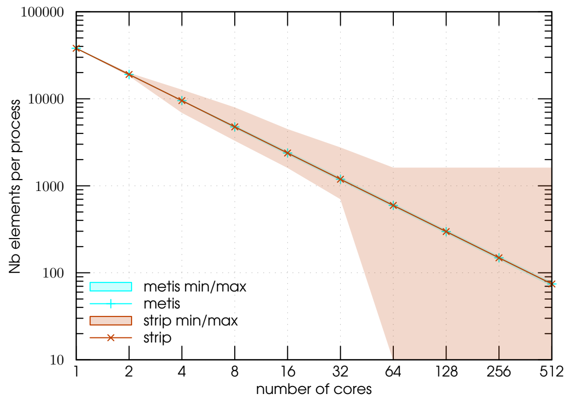
(a) Speedup



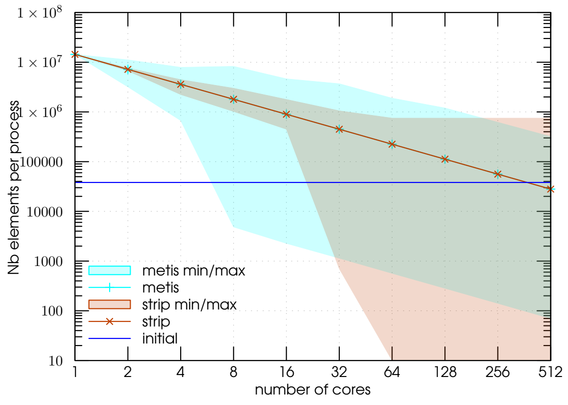
(b) Efficiency



(c) Time

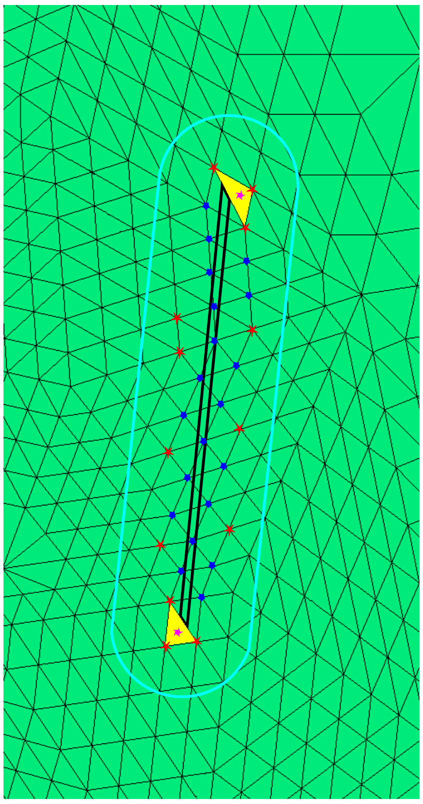


(d) Initial number of elements per process

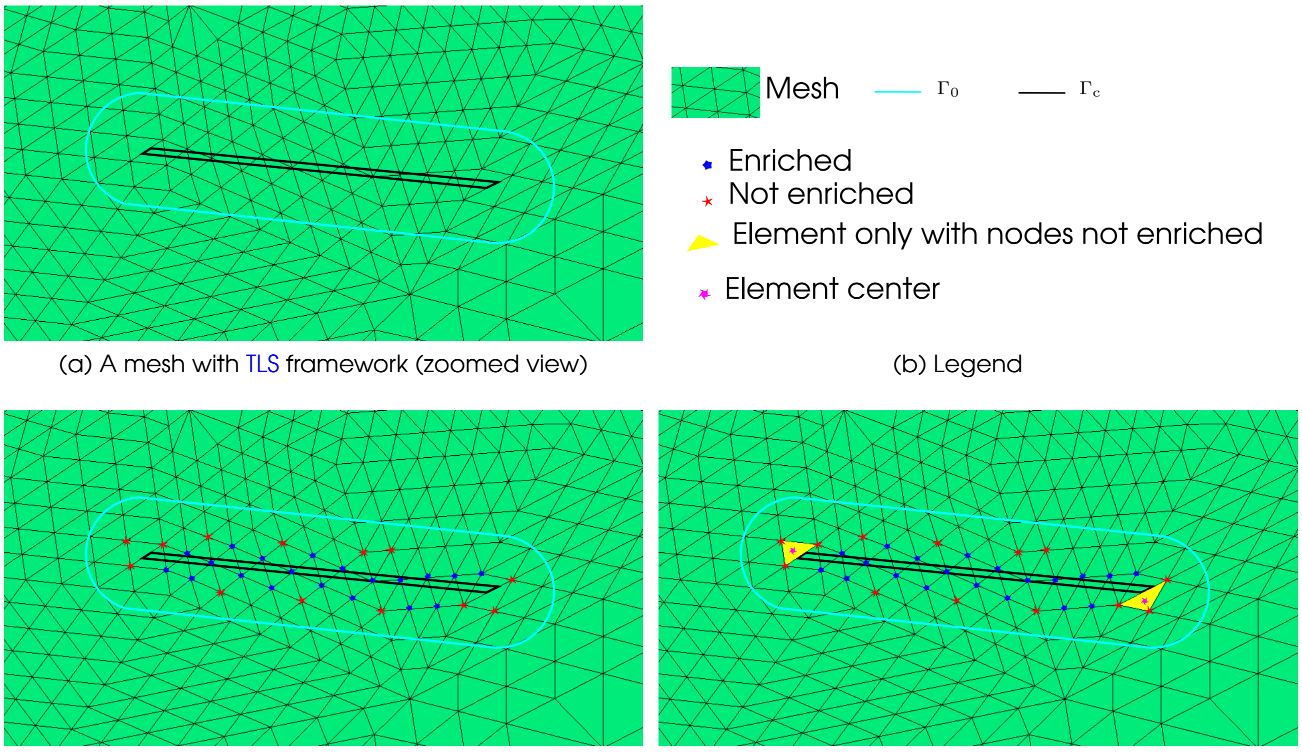


(e) Final number of elements per process

Figure D.4: Plate refinement strong scaling curves



E
Crack tip vicinity



(a) A mesh with TLS framework (zoomed view)

(b) Legend

(c) Enriched node identification

(d) Points creation

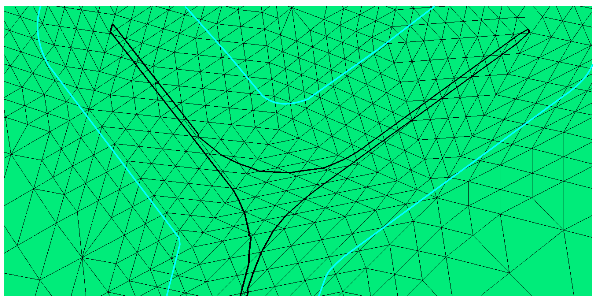
Figure E.1: Simple example of points localization in 2D

E.1 General concept

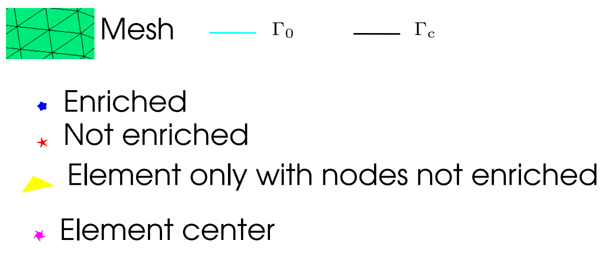
The damaged band zoning is used in chapter IV.2 and chapter V.3. In both cases the crack tip vicinity has to be located. Figure E.1 introduces, in a 2D context, the steps to obtain the coordinates of the points close to a crack tip. The principle is to use a previously adapted mesh where the TLS computation has been done (figure E.1a). The nodes, with a support that touches or intersects Ω_c, are classified into two categories. Nodes that have support separate in two or more parts by Ω_c are designated as enriched nodes. The others, designated as not enriched, are those whose support is not divided by Ω_c (only touched or reduced). This classification is presented in figure E.1c. The main idea is to consider that at the tips of the crack, it always exists, at least, one element made only of non-enriched nodes (shown in yellow in figure E.1d). This property is related to DCA which almost always stops a crack on an edge (chapter III). When it is not the case, the 3D mesh generally provides a close location where it is the case. In a simple crack wake, the elements usually have, at least, one node enriched. So, this classification brings the appropriate topological information to identify the crack tip. The final step, presented in E.1d, creates the points at the gravity center of the identified element. All those steps are summarized in algorithm 6.

Algorithm 6 Algorithm used to track close crack tip points.

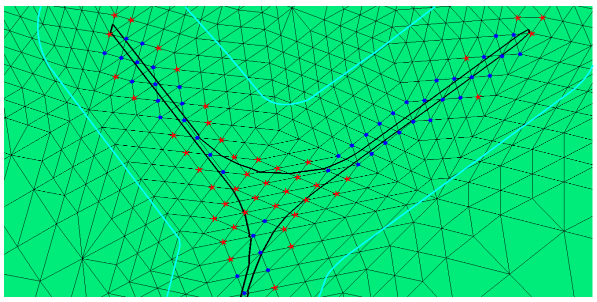
```
for each element $e$ topologically cut by $\Gamma_c$ do
    $a = 0$
    for each vertex $v$ of $e$ do
        compute $v$ support distinct parts number $k$
        if $k = 1$ then
            $a = a + 1$
        end if
    end for
    if $a$ equals $e$ number of vertex then
        create and store $e$ gravity center point
    end if
end for
```



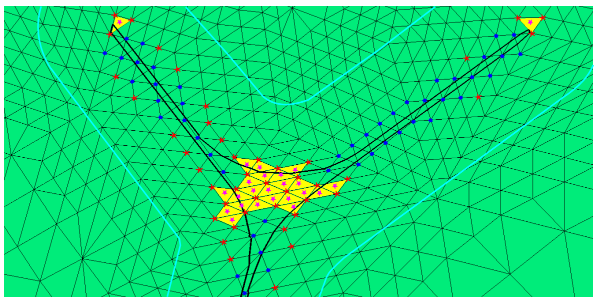
(a) A mesh with TLS framework (zoomed view)



(b) Legend



(c) Enriched node identification

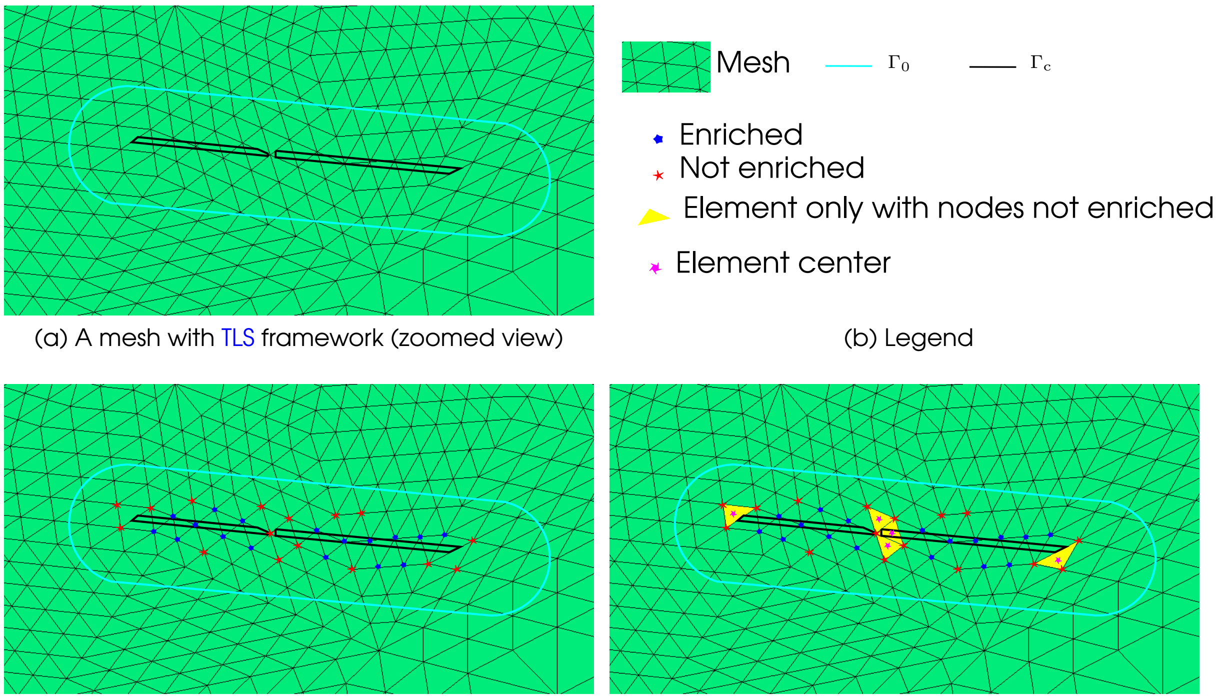


(d) Points creation

Figure E.2: Complex example of points localization in 2D

E.2 Limitations

In figure E.2 a more complicated case is presented. It is a forking situation where Ω_c has grown up to the point of fully embedding elements. In this case, as it can be seen in E.2d, the elements not at the crack tip are marked. They all do have their nodes not enriched but do not correspond to the tip location. It can be a limitation or not, depending on which application is using this algorithm. For the chapter IV.2 coarsening strategy, it is positive since



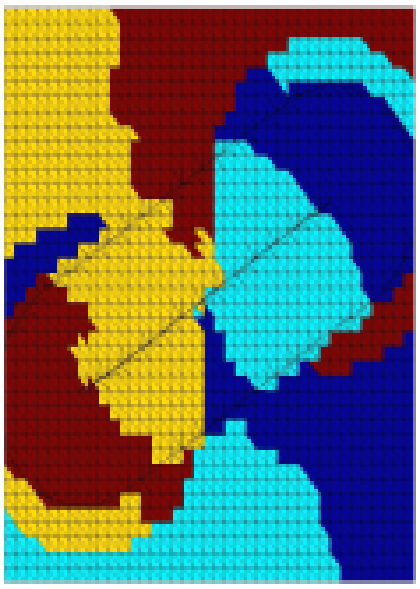
(a) A mesh with TLS framework (zoomed view) (b) Legend

(c) Enriched node identification (d) Points creation

Figure E.3: Problematic example of points localization in 2D

the refinement must be maintained in such a forking zone. For the chapter V.3 active zone strategy, there is no need to integrate such a region in the AZ group. However, that is not a stumbling block. It will just slow down the resolution.

Figure E.3 illustrates a case with a problem that as been observed during the simulations. It corresponds to the presence of a zone of not completely detached crack lips. In this case, the algorithm locates few elements in this zone (E.3d). Generally, those locks break as the crack progress but it can take some load steps. During this period, things are not optimal. For chapter IV.2 strategy, it can alleviate the coarsening itself. It is not perfect and those connections may (numerically) disappear with larger elements. For chapter V.3 active zone strategy, it is far from perfect as it slows down the resolution during all the periods where they are present.



F The load balancing strategy of the Active zone

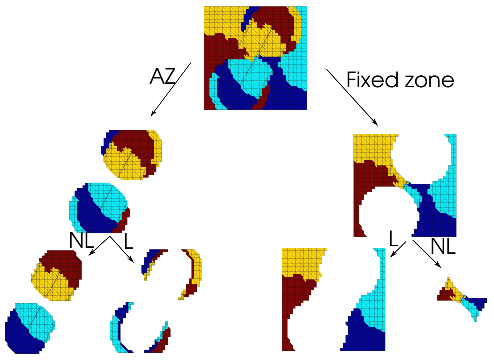
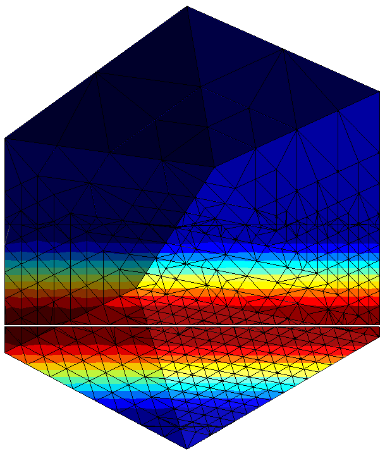


Figure F.1: Partitioning: a 4 processes example in 2D, each color represents a set of elements treated by a process.

In current AZ implementation, MUMPS is used with its distributed entries capability and is also tuned to use all processes for Schur matrix block-cyclic distribution. As MUMPS does its own matrix distribution for the factorization, all that has to be considered for the load balancing is the distributed matrix creation. It basically comes down to balancing the assemblies. So, with algorithm 2, four assemblies appear.The AZ and the fixed group are separated in linear (in Ω_{-}) and nonlinear (in Ω_{+}) elements group. Those four groups of elements are then partitioned to obtain a appropriate load balancing for the assembly computations.

A group partitioning is obtained with ParMetis. Figure F.1 depicts the process with a 2D example. On the right, the fixed zone is only treated when a new AZ is computed. The linear and nonlinear assemblies are done separately, so they need an independent partitioning. On the left, the AZ, here made by 2 circles around the tips of the oblique crack , is also split into linear and non-linear independent assemblies. So, an independent partitioning is also conducted for those two groups. During the load steps, those 2 partitioning sets are kept unchanged considering that the re-partitioning costs more than the possible degradation of the load balancing. When ϕ evolves some elements can pass from the linear group to the non-linear group and then, the partitioning may become less optimal. However, this is expected to be of sufficient quality during all load steps until the next AZ creation.

A more ad hoc partitioning would be to start from a partitioned Γ_0 and by some coloring algorithm, expand this initial partitioning to the AZ and the fixed zone. It would give a priori a more equilibrated partitioning during a load step. It also opens the AZ zone to a natural domain decomposition.



G
Illustration of TLS computation with distributed TS

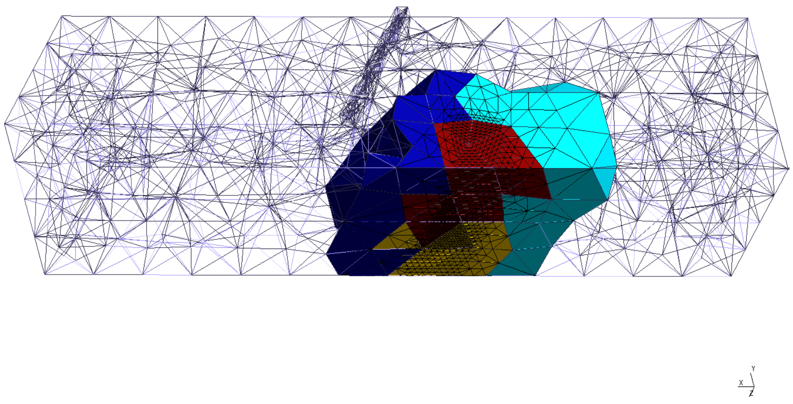
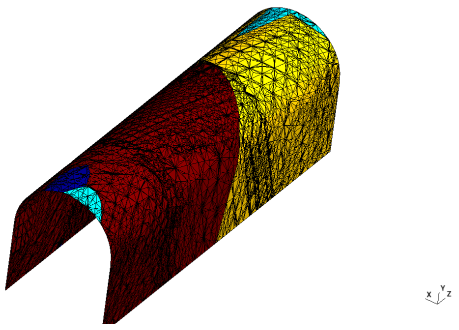
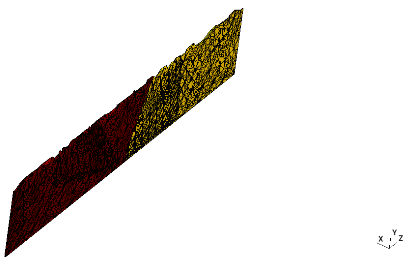


Figure G.1: SuperPatch mesh (colored per processes id) plunged in a global-scale mesh.



(a) Γ_0



(b) Γ_c

Figure G.2: Iso-zero and iso-l_c surface created in SuperPatch and colored per processes Id.

This chapter is only providing some images to illustrate what has been introduced in chapter V.4. It corresponds to a three points bending simulation of a beam. It is using the symmetric version of II.5 (i.e. $\beta = 1.$) and a fixed ϕ (no evolution). The crack is positioned outside median symmetrical plane of the beam. This simulation is following the principles of algorithm 3 but the scale loop is forced to 10 arbitrary iterations. The loop is entered with a global-scale non enriched simulation as the first boundary condition for the fine-scale problems. At the global level, only the TS enrichment (the version (V.5)) is used (no ramp Heaviside).

Figure G.1 shows the mesh of each scale: the global-scale mesh (i.e. the whole beam) and the union of all patches mesh (i.e the SuperPatch). The dis-

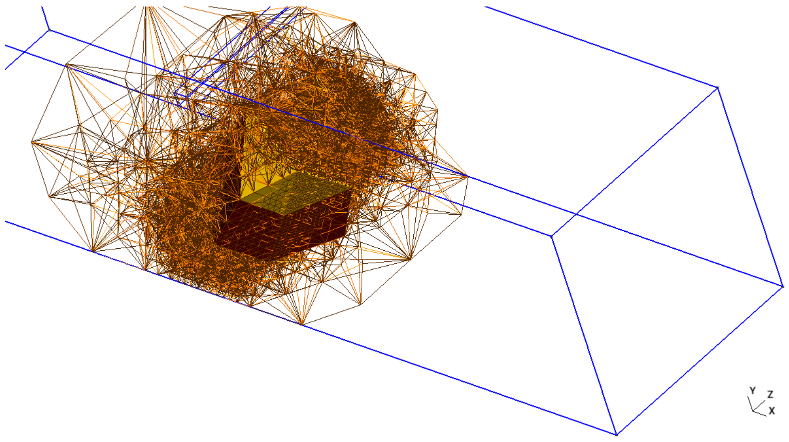


Figure G.3: Node 106 patch mesh colored by processes id in SuperPatch mesh context.

tribution of the problem on 4 processes is illustrated for the SuperPatch by one color per process. The distributed DCA is run on this last mesh and provides Γ_0 and Γ_c as presented in figure G.2 where the color represents the process id.

One patch corresponding to the node (106) at the bottom of the beam is chosen. It is presented in figure G.3, colored by process id (it is a distributed patch), in the SuperPatch and beam contexts. The damaged field associated with this patch is presented in figure G.4. Its boundaries are showed in figure G.5. The process of computation of the TS enrichment function with this patch is presented in figure G.6 for the beginning and ending of the scale loop.

Finally, the results at the global scale are presented in figure G.7: G.7a for the global scale without enrichment used as a boundary condition for the fine-scale computation at beginning of the scale loop, G.7b for the first TS enriched displacement field solution of the scale loop, G.7c for the last TS enriched displacement field solution of the scale loop and G.7d for the simple TLS displacement solution (frontal strategy also on 4 processes) on almost the same refined mesh as the TS resolution.

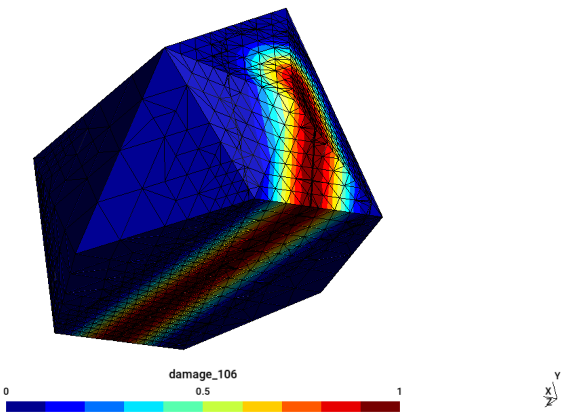
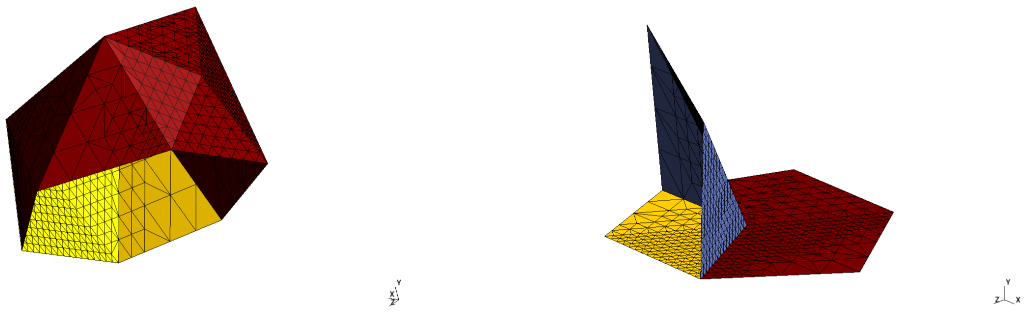


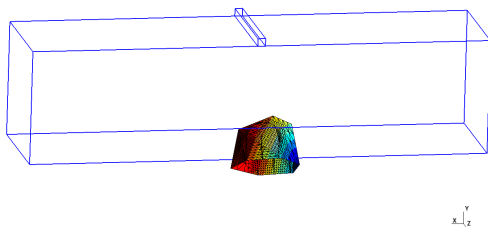
Figure G.4: Damage field for patch 106.



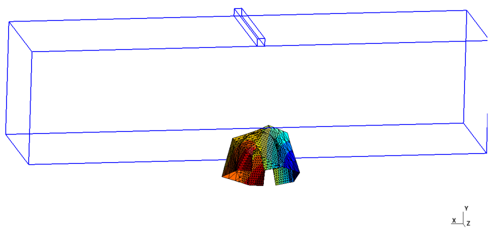
(a)

(b)

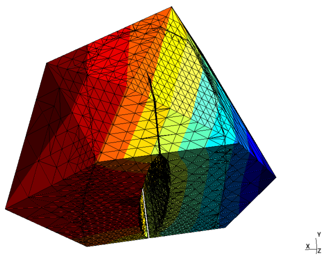
Figure G.5: Boundary of patch 106: G.5a with SuperPatch colored per process id, G.5b from part boundary colored per process id with inter-process boundary in blue.



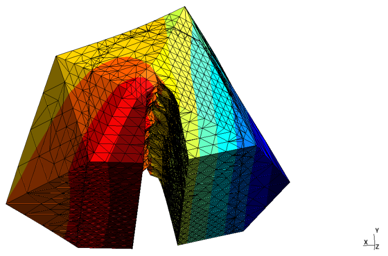
(a)



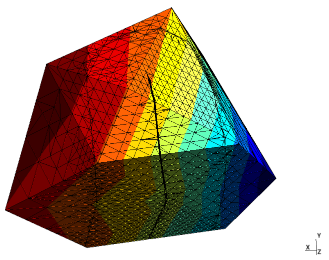
(b)



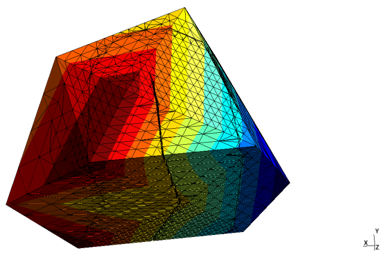
(c)



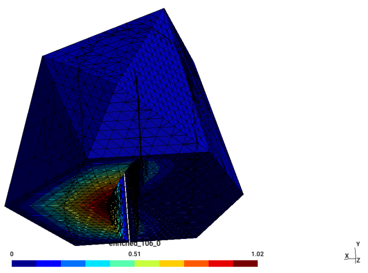
(d)



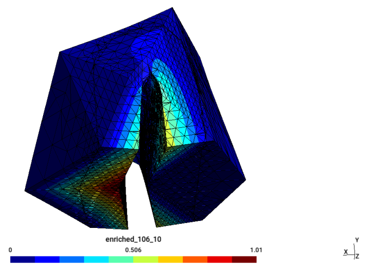
(e)



(f)

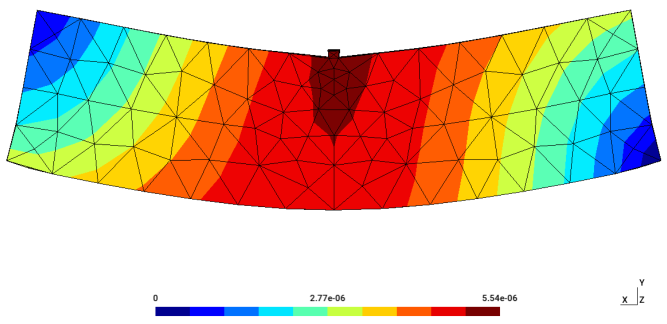


(g)

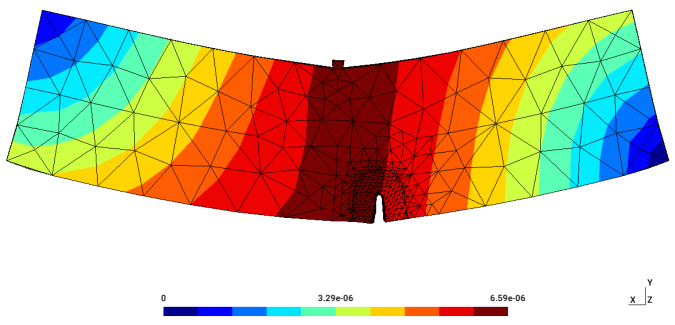


(h)

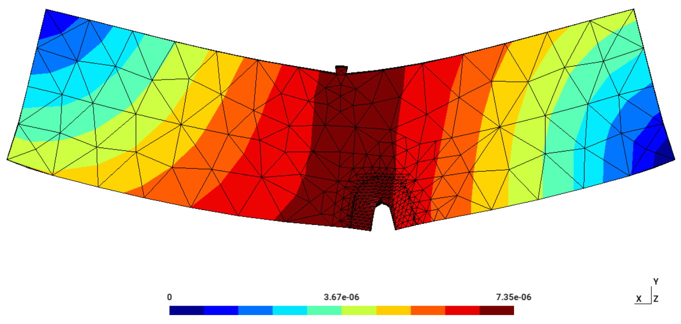
Figure G.6: patch 106 computation at begin (left) and end (right) of the scale loop: G.6a/G.6b imposed displacement (Robin boundary conditions), G.6c/G.6d patch displacement field solution, G.6e/G.6f patch displacement field projection on global-scale approximation, G.6g/G.6h scaled TS enrichment function (see (V.5)).



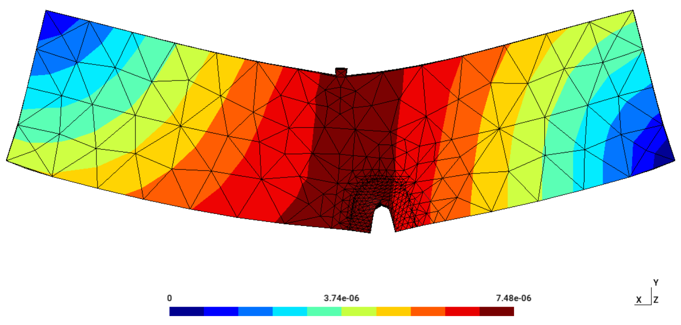
(a)



(b)



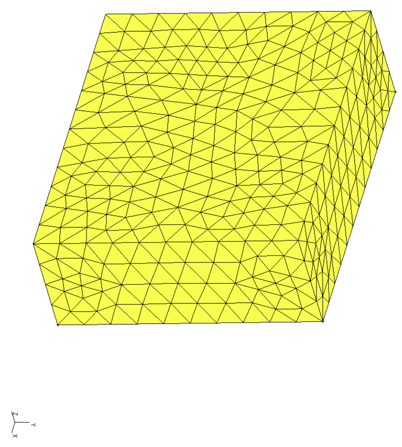
(c)



(d)

Figure G.7: Global-scale displacement field solutions.

H
Uniform test case



H.1 Description

The test case is a 3D plate with the geometry depicted in figure H.1. We can consider that it represents the tip crack zone in terms of computational effort (number of dofs). The coarser mesh called level zero or L0 (figure H.2a) is unstructured, with a maximal aspect ratio of 3.47 (figure H.3). Different refined meshes are used. At each refinement step, all edges are split into two edges and the tetrahedrons are split accordingly. Each step is called level and named Lx, where x denotes the level number ranged from 0 (no refinement) to 5 (initial element size is divided by 2^5, i.e, 32).

The mechanical properties are those of a simple uniform elastic isotropic material. The approximation order is 1 for all fields.

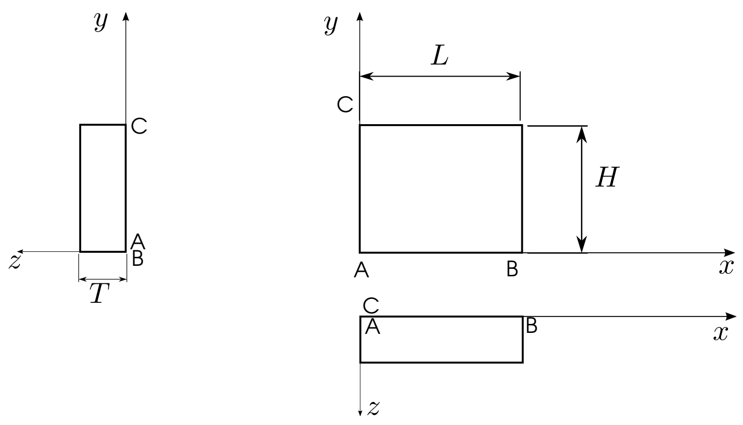


Figure H.1: Benchmark test: simple plate under volume and surface loading (see (H.7)). A, B, and C are the corner points used to fix the rigid body modes.

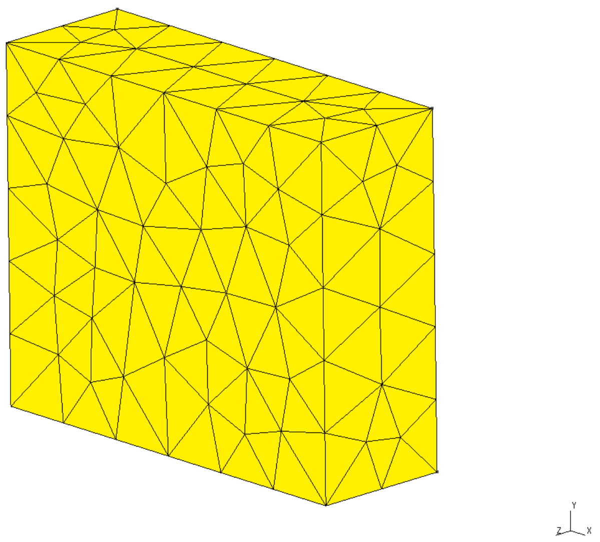
H.2 Analytical formula

We want to test a known cubic displacement field with this plate in order to obtain an accurate error computation (i.e. comparison with an analytic formula). The following $\overrightarrow{u}(x,y,z)$ displacements are used as analytic reference:

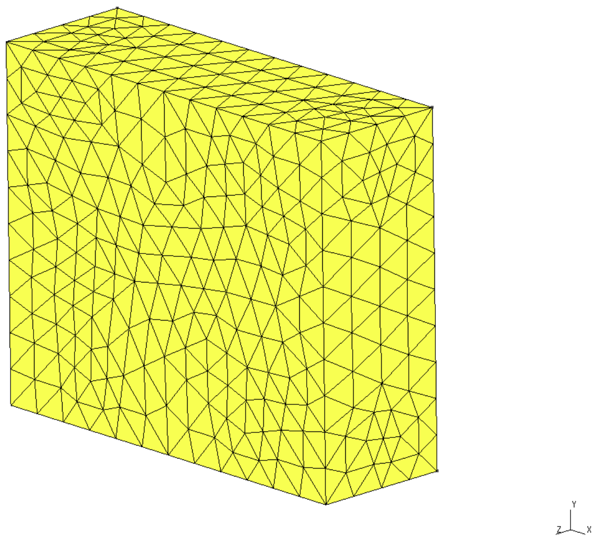
$$\overrightarrow{u}(x,y,z) = \frac{(\nu+1)(1-2\nu)F}{E}\begin{bmatrix} (x^2(\frac{L}{2}-\frac{x}{3}) + 2\nu(y^2-z^2)) \\ -4\nu xy \\ 4\nu xz \end{bmatrix} \tag{H.1}$$

where :

- E is the Young's modulus
- ν is the Poisson's ratio



(a) Initial mesh, level zero or L0: 188 nodes, 595 tetrahedrons



(b) Mesh called level one or L1: 1110 nodes, 4760 tetrahedrons

Figure H.2: Plate meshes

- F is a scalar corresponding to a force
- L is the plate length (figure H.1)

Dirichlet boundary conditions, imposed in this test case, are compatible with those fields since the point A is clamped in all direction ($\overrightarrow{u}(0,0,0) = \overrightarrow{0}$), the point B is clamped at least in z and y direction($\overrightarrow{u}(L,0,0).\overrightarrow{e_y} = \overrightarrow{u}(L,0,0).\overrightarrow{e_z} = 0$) and the point C is clamped at least in z direction ($\overrightarrow{u}(0,H,0).\overrightarrow{e_z} = 0$)

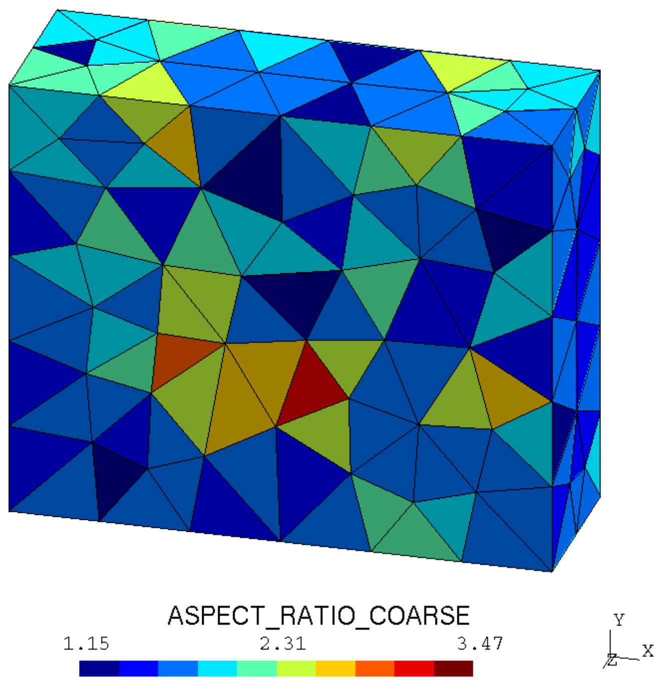


Figure H.3: Coarse mesh for plate with aspect ratio for each element

Considering small displacements, the strain is given by :

$$\epsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right) \tag{H.2}$$

and with the displacement field coming from (H.1), we have:

$$[\epsilon(x,y,z)] = \frac{(\nu+1)(1-2\nu)Fx}{E}\begin{bmatrix} (L-x) & 0 & 0 \\ 0 & -4\nu & 0 \\ 0 & 0 & 4\nu \end{bmatrix} \tag{H.3}$$

Hooke's law gives the stress tensor:

$$\sigma(x,y,z) = \mathbf{C} : \epsilon(x,y,z) \tag{H.4}$$

where $\mathbf{C}$ is the elasticity tensor corresponding to an isotropic material defined by E and ν.

It results in the following expressions using $\mathbf{C}$ and (H.3):

$$[\sigma(x,y,z)] = xF\begin{bmatrix} (\nu-1)(x-L) & 0 & 0 \\ 0 & \nu(L-x-4(1-2\nu)) & 0 \\ 0 & 0 & \nu(L-x+4(1-2\nu)) \end{bmatrix} \tag{H.5}$$

The equilibrium equations are:

$$\begin{cases} \overrightarrow{div}\ \sigma(x,y,z) + \overrightarrow{f_v} = 0 \ on\ \Omega \\ \sigma(x,y,z).\overrightarrow{n} = \overrightarrow{f_s}\ on\ \partial\Omega \end{cases} \tag{H.6}$$

From (H.6) and (H.5) we obtain the load to impose on the test case that gives the (H.1) displacement field :

$$\begin{cases} \overrightarrow{f_v} = \begin{bmatrix} (1-\nu)F(2x-L) \\ 0 \\ 0 \end{bmatrix} \\ \overrightarrow{f_s}_{(x=0)} = \overrightarrow{f_s}_{(x=L)} = \overrightarrow{0} \\ \overrightarrow{f_s}_{(y=H)} = \begin{bmatrix} 0 \\ xF\nu(L-x-4(1-2\nu)) \\ 0 \end{bmatrix} \\ \overrightarrow{f_s}_{(y=0)} = -\overrightarrow{f_s}_{(y=H)} \\ \overrightarrow{f_s}_{(z=T)} = \begin{bmatrix} 0 \\ 0 \\ xF\nu(L-x+4(1-2\nu)) \end{bmatrix} \\ \overrightarrow{f_s}_{(z=0)} = -\overrightarrow{f_s}_{(z=T)} \end{cases} \tag{H.7}$$

The analytic equation (H.3) is used as a reference for the numerical computations of the energy error associated with the different meshes and/or methods:

$$error(\overrightarrow{u}_{computed}) = \sqrt{\frac{\int_\Omega (\epsilon(\overrightarrow{u}_{computed}) - \epsilon(x,y,z)) : \mathbf{C} : (\epsilon(\overrightarrow{u}_{computed}) - \epsilon(x,y,z))\ \mathrm{d}\Omega}{\int_\Omega \epsilon(x,y,z) : \mathbf{C} : \epsilon(x,y,z)\,\mathrm{d}\Omega}} \tag{H.8}$$

H.3 Results

In this benchmark, the plate (described in annex H.1 using boundary conditiond from equation (H.7)) has to be considered as a fictitious 3D crack tip with a far simpler strain field. In this context, all the strategies of chapter V have been compared, with for each of them:

- Only the creation, assembly and resolution of the mechanical problem system have been monitored.
- The crack wake, and overall part definition have been neglected (it corresponds to different approximations for each strategy described in chapter V.5).
- The hybrid parallelism is almost never used in the simulations for a fair comparison. For example, using Mumps solver in some approaches with OMP capability and multi-threaded Blas, is unfair compared to none multi-threaded TS strategy. Moreover, multi-threading would have also complexify the creation of the curves (how many threads are needed per MPI process for each strategy, how to consider MKL Blas that adapts the number of threads when computing speedup,...).

- For a given number of processes, the coarse mesh is partitioned on them and splits uniformly up to the target refinement level. This refined mesh is used in the TS strategy (V.4) for the fine scale level. The same partitioned refined mesh is used for the comparison of the frontal (V.2) and active zone (V.3) strategies.

- The simulations have been conducted on the Liger computer (annex J.2) without interference from any other applications at node level (it is not completely the case for the network usage).

- For TS strategy, the benchmark follows the simple algorithm 3. Nevertheless, instead of a test based on the relative variation of $\vec{U}_s$ to stop the loop, the energy error given by (H.8) is compared to the one obtained with the frontal strategy or the modified AZ strategy. The loop stops as soon as TS energy error is less than the one obtain with other strategies.

This test provides a rough estimate of what can be expected from all strategies if they were finalized. It is made without extra complexity (the mesh, the mesh partitioning, ... are the same between the strategies). Recalling the naming convention of chapter V.5, hereafter, "frontal" will designate the frontal strategy, "domain" will designate the modified AZ strategy and "two-scale" the TS strategy.

H.3.1 Strong scaling

Figure H.4, H.5, H.6, H.7 and H.8 present strong scaling results respectively for the final refined level L1,L2,L3,L4 and L5. For each figure we have:

- In (c), the curves of elapsed time for mechanical problem resolution (in second). It corresponds to an average elapse time across processes. For the frontal results, it counts field declaration, system assembly and resolution. For domain results, it counts local domain field declaration and associated system assembly and condensation, preconditioner factorization, iterative resolution of the dense system,and decondensation phase. For TS results, it counts patch field creation, patch boundary condition integration, patch resolution, global enriched field declaration, global enriched system assembly and resolution, and for most of the same items their accumulated contribution while looping on scales.

- In (a), the speedup curves based on the elapsed time (c). For each approach, the sequential reference time is the one corresponding to its strategy, except for the domain results based on the frontal computation (with one process, there is only one domain without any condensation; it is therefore not possible to obtain a result with our implementation). The ideal speedup is plotted in black (if we use x processes we have to be x times faster than with a single process).

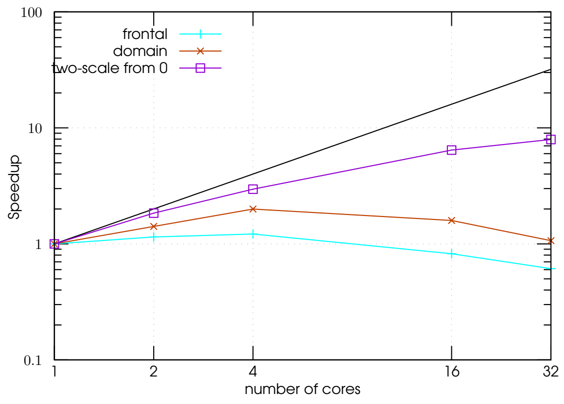
- In (b) the curves of strong scaling efficiency. They correspond to the speedup curves (a) divided by the number of used process and transformed in %. 100% corresponds to an ideal speedup (if we use x processes the gain obtained, compared to sequential time, is the maximum expected).
- In (d), the dofs curves. They show the solved system dimension for all strategies and their different involved scale. For the frontal, coarse and TS global enriched results, the dimensions remain constant when the number of processes changes. With the TS strategy "per patch" gives the average size of the fine-scale problems. Due to a uniform refinement, all patches got quite the same size on average and the parallelism does not change the situation. It roughly leads to a constant size. With the domain results, two curves are given: the first one ("domain(Schur)") indicates the size of the condensed system. The second one ("per domain") gives the average size of the domain problems.
- In (f) the information curves about the TS patch number. "total" gives the global number of patches for the problem. "per proc" gives the averaged number of patches treated by a process. "dist per proc" gives the averaged number of distributed patches treated by a process.
- In (e), the curves give the ratio between the TS or domain and the frontal elapsed times for a given number of processes. If this ratio is 1, the considered strategy has the same performance as the frontal approach. If lower than 1, it is slower. If greater than 1, it is faster. With the TS strategy, the "iter" curves represent the ratio if we only consider one scale iteration without counting the patch creation. This situation is expected when TS is used with algorithm 4.
- All the TS results got a "from x" indication that indicates from which global level, the method starts. "x" correspond to Lx. "from 1", in L3 figure H.6, means that the global level corresponds to the mesh L1 (coarser L0 refined once) and that the fine level is obtained by refining this mesh twice which provides an equivalent L3 mesh (3=1+2).
- It is hard to ensure that the presented curves are representing a continuous processing task increase. Take the example when going from 16 to 32 cores. On the one hand (16 tasks), the inter-process messages are exchanged inside a node (shared memory mechanism) and do not pass through the network. On the other hand (32 tasks), some messages are remaining in the node and some are passing through the network layers since a Liger node is limited to 24 cores. It forces, at least, to use 2 nodes for the computation. A fair comparison, but non-realistic from

a practical point of view, is to use a distribution with one process per node. In this context, increase the number of tasks would systematically raise the network load. No additional simulation has been conducted to clarify this point.

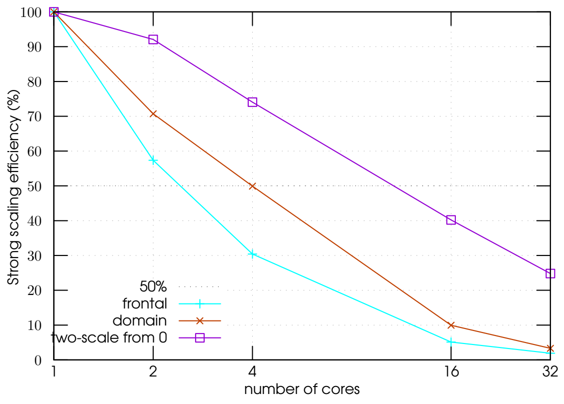
Of the L5 results, only those that have been successfully executed are shown in figure H.8. With this kind of problem size (9965229 dofs), memory becomes an issue. Liger computer permits only 128GB per node. Some observations show that the frontal strategy needs memory between 1Tb to more than 3TB (mainly due to the system factorization). For the domain approach, as only one domain is considered per process, the memory consumption remains high with few processes. For the TS method, only around 300GB is needed. It explains why it was impossible to obtain a sequential solution for any approach[1]. With the TS method, the smallest possible number of processes to be used is $\frac{300}{127} \sim 2.37$ which corresponds to 4 processes in power of 2 progression. To obtain a speedup and an efficiency, a perfect scaling between 1 and 4 processes has been considered. It allows extrapolating the elapsed time of the sequential computation. And it explains why we have a perfect strong scaling efficiency with 4 processes. For the other strategies, the first possible computation starts with 32 processes (1 core per node). Extrapolation for the elapsed time of the sequential computation is too hazardous in this case. The speedup and efficiency are then not provided. For the TS method, extra speedup and efficiency curves are provided with a name ending by "ideal global". They correspond to data where the elapsed time of the global level problem resolution has been kept constant to its minimal value (with 128 processes) when minimum has been reached (i.e. for 256, 512 and 984 processes).

As mentioned above, increasing the number of MPI process with the same network loading is not easy to achieve, especially for these L5 results. The first curve points, due to memory consumption, imply a rather natural increase of nodes per computation. However, when reaching 900 cores, the amount of work per core starts to be dominated by communication. Then the results, when passing from 984 cores on 41 nodes (41x24) to 1024 cores on 43 nodes, are too much perturbed by the network load increase to be relevant to compare the methods. This assertion comes from a test, not presented here, where the same 984 cores are used for the computation on 41, 42 and 43 nodes. A drastic drop in performance is observed. It leads to using a maximum of 984 cores on 41 nodes.

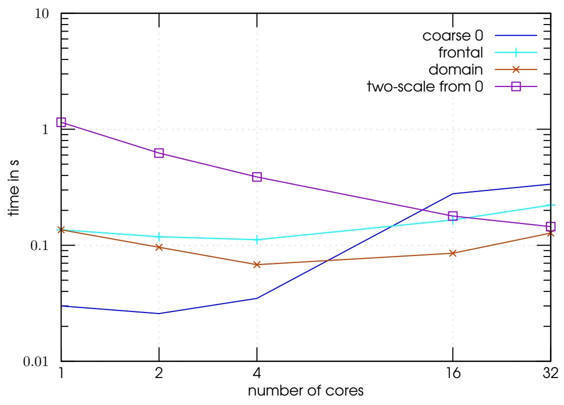
[1]solver out of core capability has not been considered. The speedup, in this case, would mainly have measured hardware capability instead of the algorithmic performance



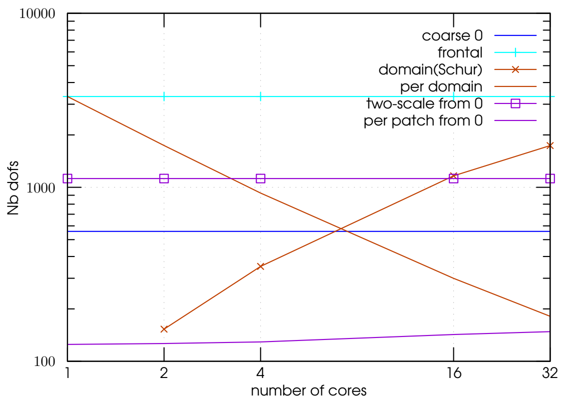
(a) Speedup



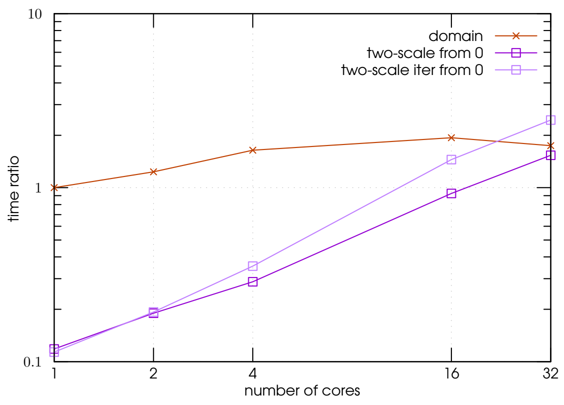
(b) Efficiency



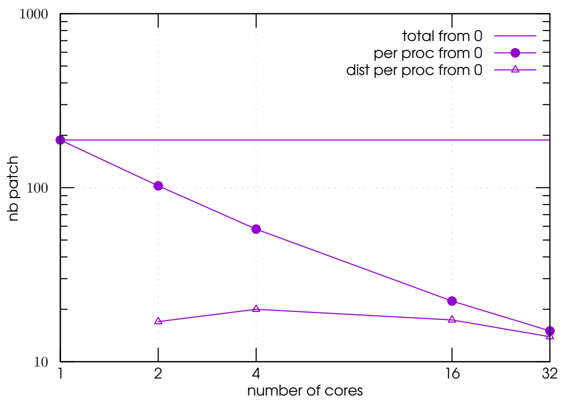
(c) Time



(d) Dofs

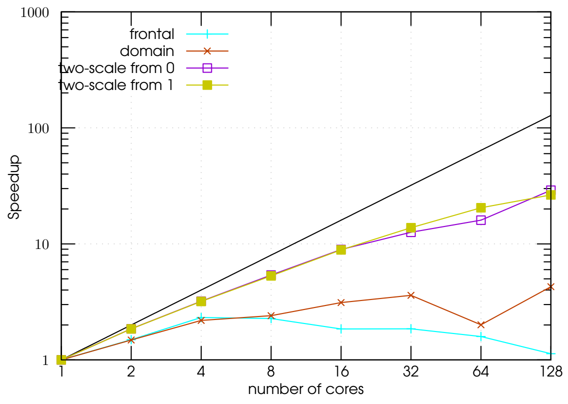


(e) Comparison with frontal

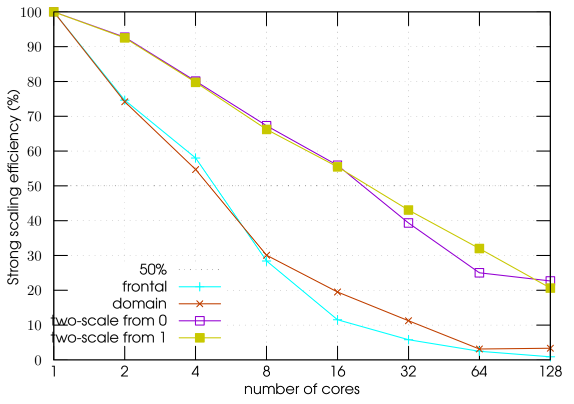


(f) Patch

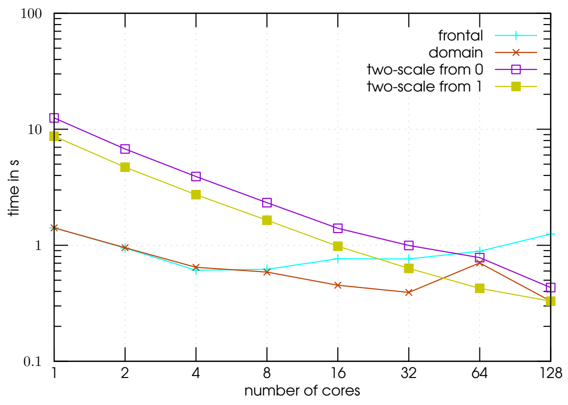
Figure H.4: Cubic field, Level 1, strong scaling



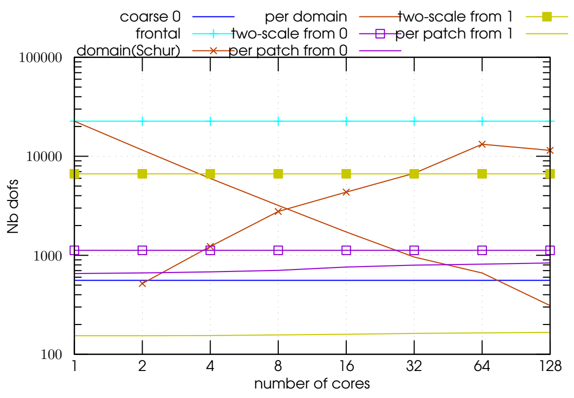
(a) Speedup



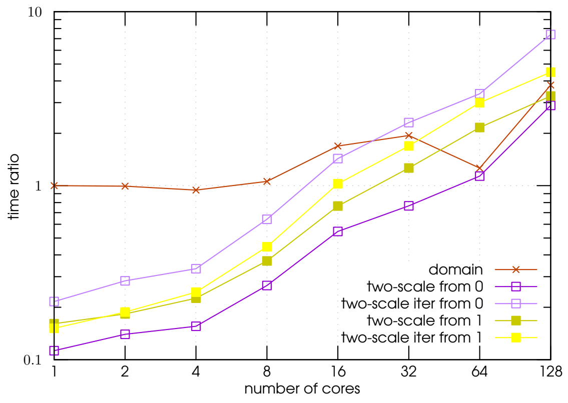
(b) Efficiency



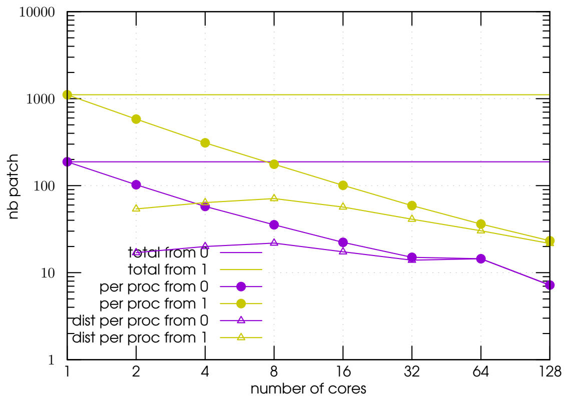
(c) Time



(d) Dofs

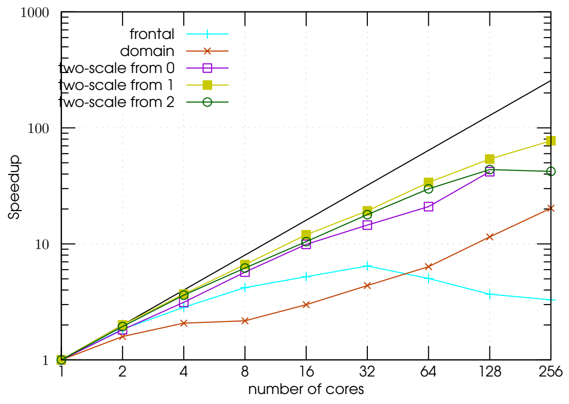


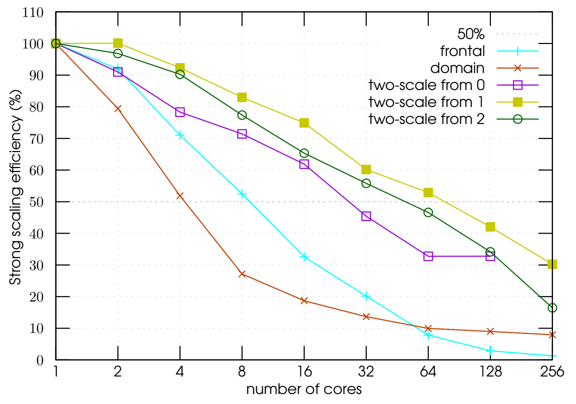
(e) Comparison with frontal

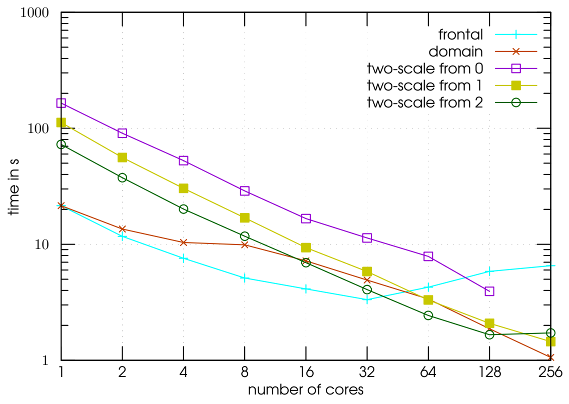


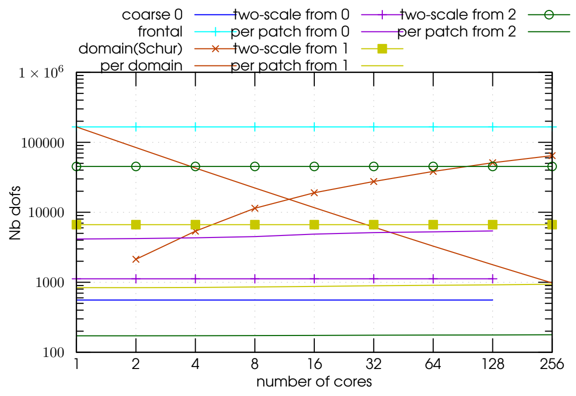
(f) Patch

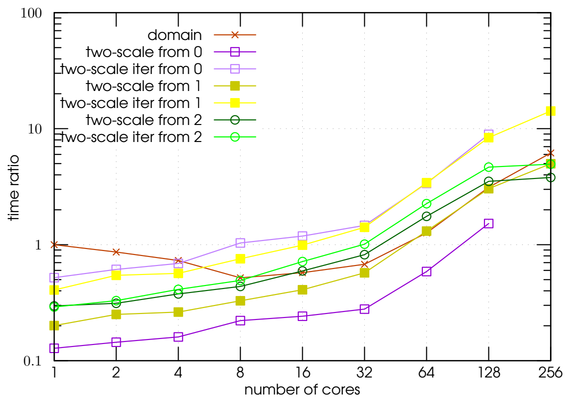
Figure H.5: Cubic field, Level 2, strong scaling.


(a) Speedup


(b) Efficiency


(c) Time


(d) Dofs


(e) Comparison with frontal

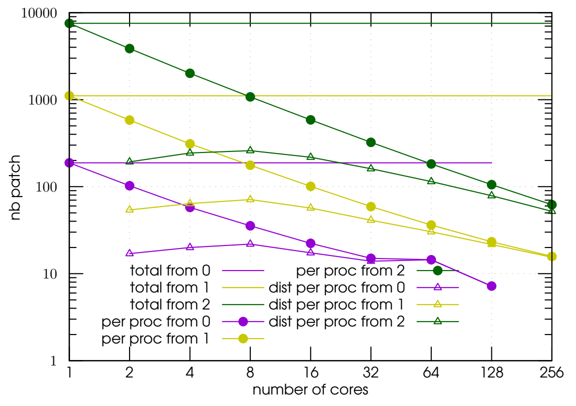
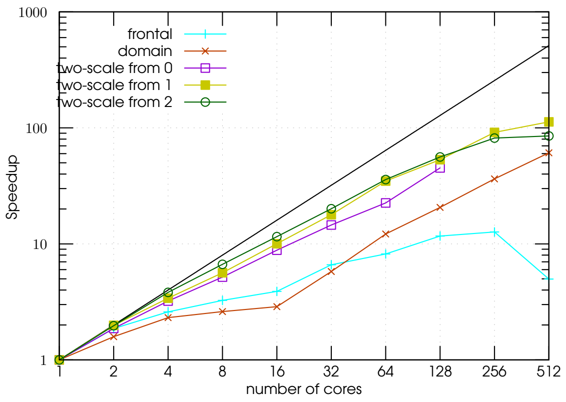
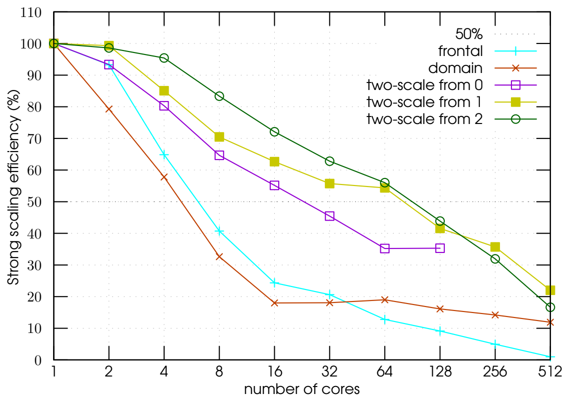

(f) Patch

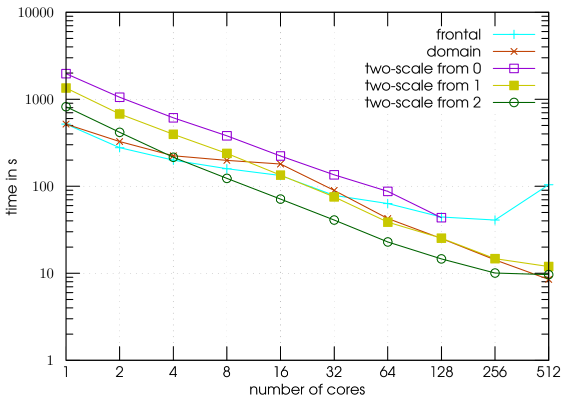
Figure H.6: Cubic field, Level 3, strong scaling .



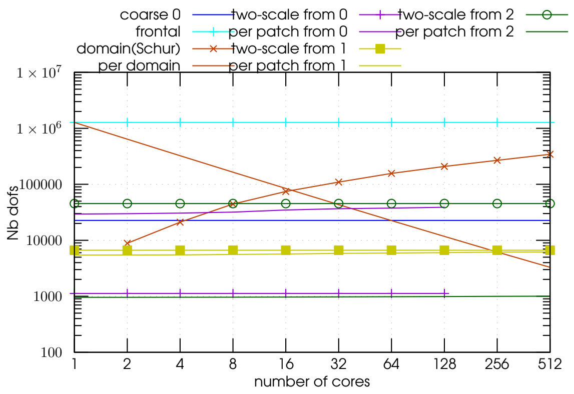
(a) Speedup



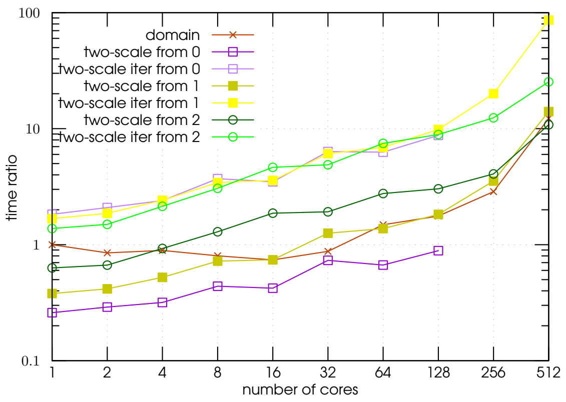
(b) Efficiency



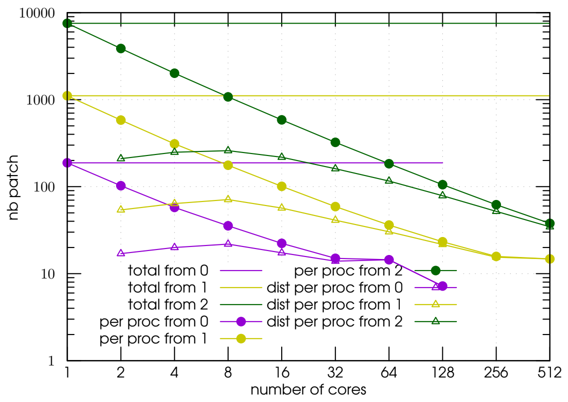
(c) Time



(d) Dofs

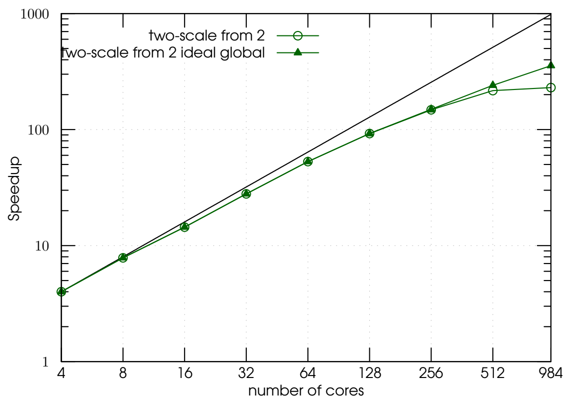


(e) Comparison with frontal

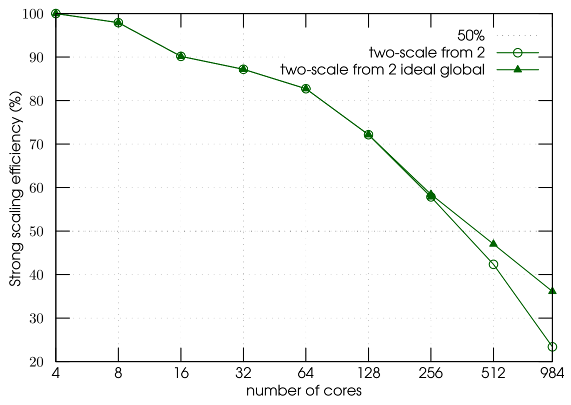


(f) Patch

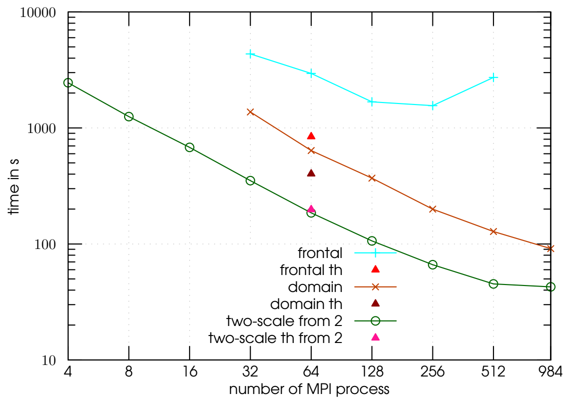
Figure H.7: Cubic field, Level 4, strong scaling.



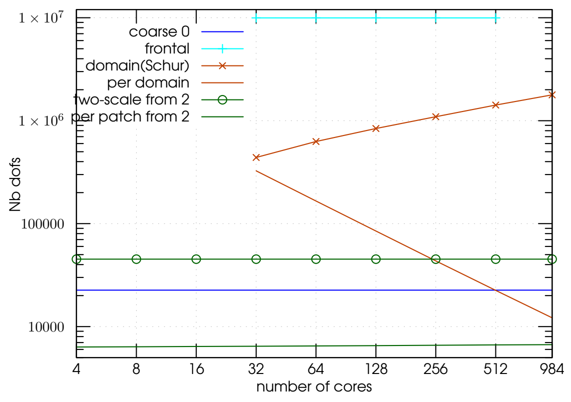
(a) Speedup



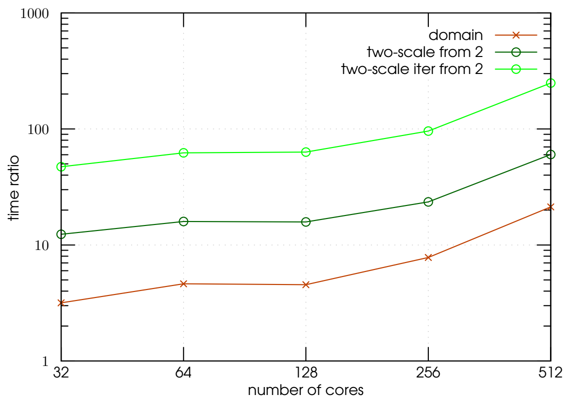
(b) Efficiency



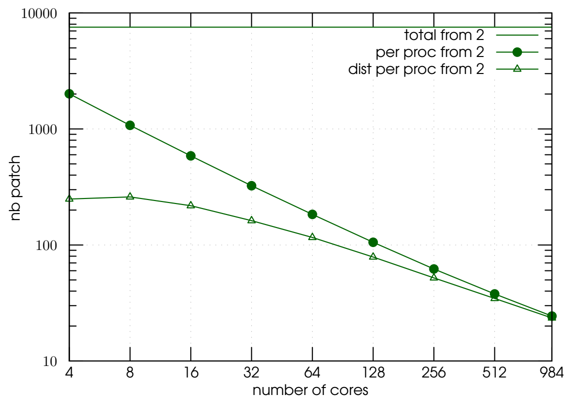
(c) Time



(d) Dofs



(e) Comparison with frontal



(f) Patch

Figure H.8: Cubic field, Level 5, strong scaling.

Finally, as the L5 results are concerned, note that a multi-threaded Blas (with MKL on-demand free strategy) has been tested with 64 MPI processes (named with "th"). It explains why in H.8e, the abscissa is given in number of MPI processes and not in number of cores. With the frontal and domain results, the Mumps multi-threaded capability has also been used with 4 threads per MPI process.

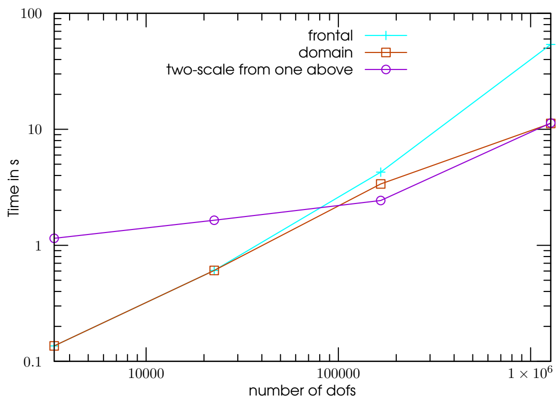
H.3.2 Weak scaling

Figure H.9 presents the "frontal", "domain" and "two-scale" weak scaling results. Three initial refinement Levels are considered (L1,L2,L3). For each initial level, two sets of curves are given: one corresponding to the computational elapsed time in seconds, and one corresponding to the weak scaling. The abscissa is the problem size in number of dofs. The points on those curves correspond to different problem sizes maintaining a constant number of dofs per process. The starting point corresponds to the initial level size treated in sequential. That is why all the weak scaling curves start at 100% since this scaling is the ratio of elapsed time in sequential divided by the elapsed time in parallel. For the frontal and domain results, nothing can be changed except the starting point. For the TS, the global level may change."two-scale from X above" curves (with X=1, 2, or 3) represents the problem size obtained by always starting from a global level X level above it. I.e. for X=1 with L1, the computed points are:

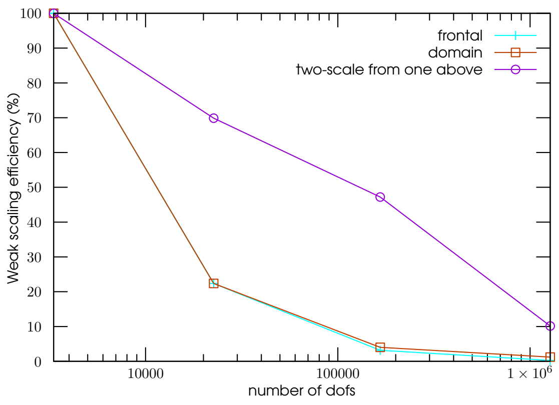
- L1(3324 dofs) from L0 with 1 process
- L2 (22611) from L1 with 7 processes ($\frac{22611}{7} = 3230.14 \sim 3324$)
- L3 (166185) from L2 with 50 processes ($\frac{166185}{50} = 3323.7 \sim 3324$)
- L4(1273173) from L3 with 383 processes ($\frac{1273173}{383} = 3324.21 \sim 3324$)

The other curves "two-scale from X" (with X=1 or 2) represent the obtained problem size by always starting from the same global level X. For example, for X=1 with L2, the computed points are:

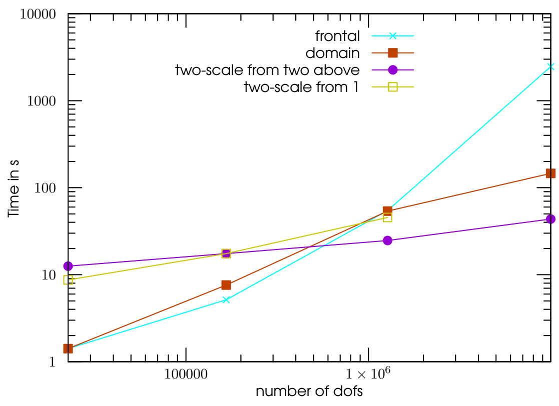
- L2(22611 dofs) from L1 with 1 process
- L3 (166185) from L1 with 7 processes ($\frac{166185}{7} = 23740.71 \sim 22611$)
- L4 (1273173) from L1 with 56 processes ($\frac{1273173}{56} = 22735,23 \sim 22611$)



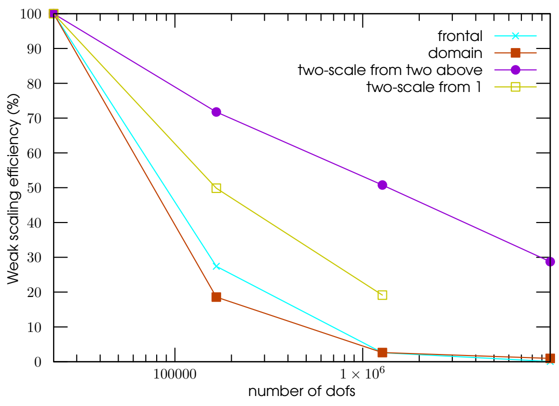
(a) Time L1



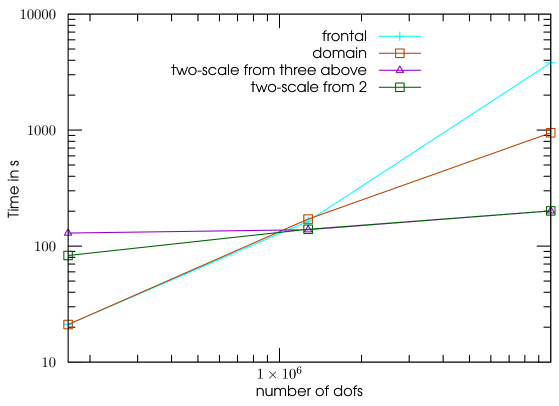
(b) Efficiency L1



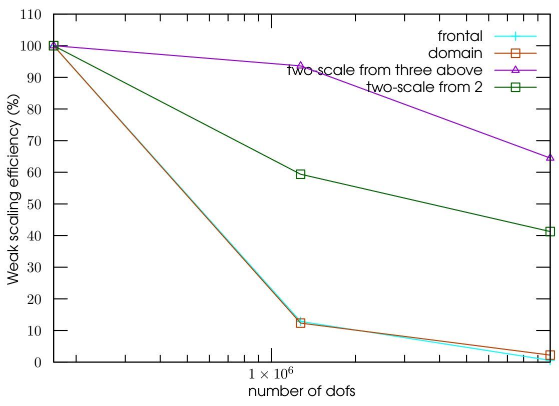
(c) Time L2



(d) Efficiency L2



(e) Time L3



(f) Efficiency L3

Figure H.9: Cubic field, weak scaling for different starting problem sizes (L1,L2 and L3).

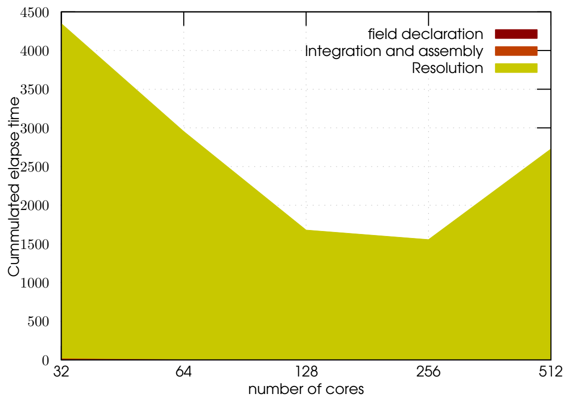
H.3.3 Task dispatching

Figure H.10 provides some information about the origin of the computational costs associated with the "frontal", "domain" and "two-scale" results for L5 simulation. The elapsed time for each task is accumulated. For each strategy, the identified task covers different items:

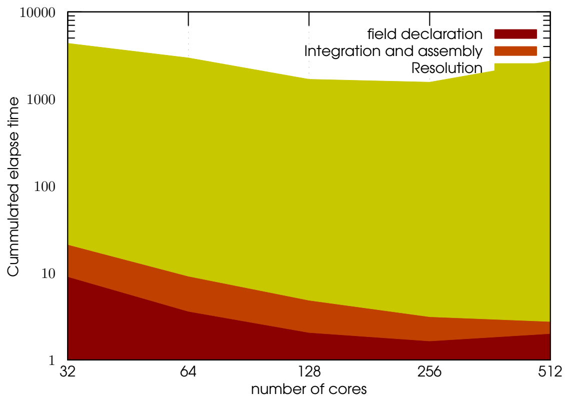
frontal: "field declaration" represents the dofs numbering (involving communication), the creation of the local matrix graph and the elimination of the local Dirichlet boundary conditions. "integration and assembly" corresponds to the local elementary integration and the elementary matrix assembly to the local part of matrix A. "Resolution" is the parallel direct sparse solver factorization and the resolution phases.

domain: "field declaration" represents the domain dofs numbering (no communication involved since the domains are restricted to only one process), the Schur dofs numbering (involving communication), the creation of the domain matrix graph and the elimination of the local Dirichlet boundary conditions. "integration and assembly" corresponds to the local elementary integration and the elementary matrix assembly to domain matrices. "Resolution (dense)" is the parallel iterative resolution of the distributed Schur problem. "Resolution " is the incomplete factorization of the direct sparse solver for each domain (creating their Schur contribution), the block Jacobi preconditioner factorization, the right-hand side b condensation and the de-condensation of the solution.

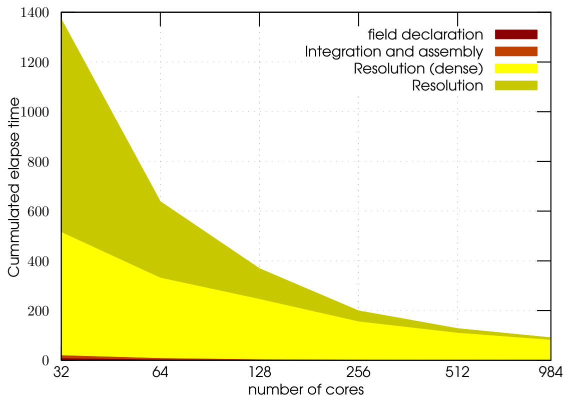
two-scale: "field declaration" represents the patch dofs numbering, the treatment of the patch boundary conditions, the integration and assembly (in sequential or parallel) of the patches, the sequential or parallel resolution of the patches using a direct sparse solver, the global level dofs numbering (involving communication) and the elimination of the Dirichlet boundary conditions. "field declaration update" only covers the patch boundary conditions integration and the sequential or parallel resolution of the patches with an already factorized matrix. It is the update phase of the enrichment functions that happen during the scale loop. "integration and assembly" corresponds to a local elementary integration of SuperPatch on coarse elements and an elementary matrix assembly in distributed local part of matrix A. "Resolution" is the parallel direct sparse solver factorization and the resolution phases at global scale.



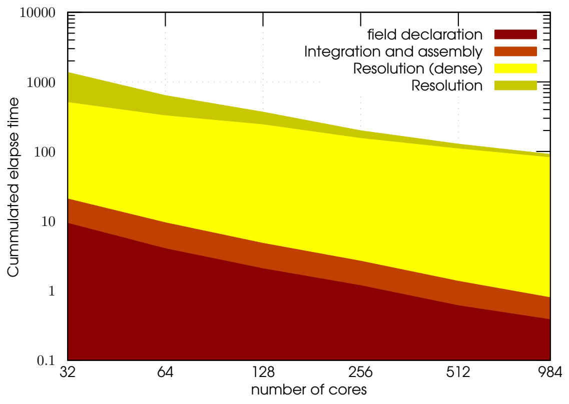
(a) Frontal



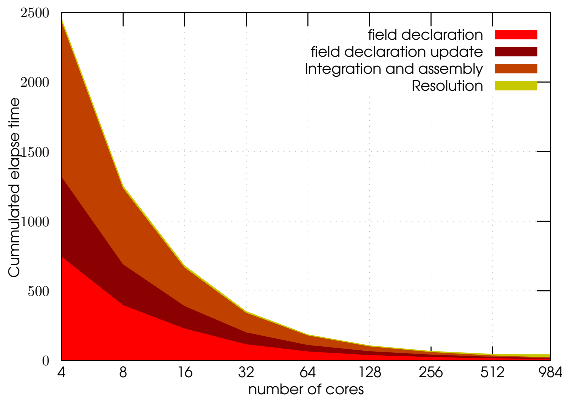
(b) Frontal (log scale)



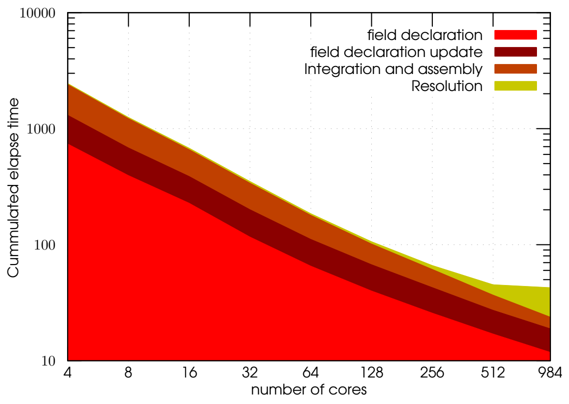
(c) Domain



(d) Domain (log scale)



(e) Two-scale



(f) Two-scale (log scale)

Figure H.10: Cubic field, Level 5, Elapse time dispatching

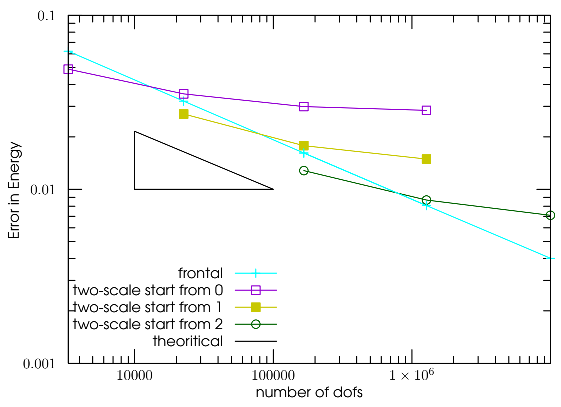


Figure H.11: Cubic field, error in energy. Frontal strategy being the reference for TS strategy.

H.3.4 Computational precision

In figure H.11, the energy error (from (H.8)) is plotted for the frontal strategy and the theoretical slop is given. The TS curves correspond to the initial error obtained using a global non-enriched field as boundary conditions for the patches. These curves give, for the TS resolution, the error of the "starting point". The scale iteration reduces those errors to be less or equal to the frontal one.

I
The computational performance of the chapter VI test cases

	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

Test case	Elapsed time	Number of processes
L shape (VI.4)	6.17h	16
Chalk (VI.2)	5.43h	24
Spherical Holes (VI.1)	9.25h	24
Spiral bevel gear (VI.5)	1.6d (up to step 246)	48

Table I.1: Test elapsed time in hours (h) or days (d) (run on Titan cluster, see annex J.2)

Item	Elapsed time	% of total
Total	37.93	100
Mechanical problem resolution Algorithm 1 II	23.68	62.42
TLS update Algorithm 1 I+III	13.12	34.59
AZ creation	0.32	0.84
Other	0.82	2.16

Table I.2: Computation times dispatching, in hours, of the spiral bevel pinion gear (the 246 first load steps).

In terms of performance in computation time, the state-of-the-art about the elapsed times for all the test cases of chapter VI is given in Table I.1, except for notched beams tests[1] (chapter VI.3). Note that the older Spherical Holes test case was not re-computed with AZ. The elapsed times given in the table correspond to a pre-AZ version without condensation.

The AZ was very useful for the spiral bevel gear test case where the model is large (453 824 nodes) from the start. Notably, in the case of the first load steps, the gain is high thanks to the small AZ envelope. As mentioned in annex E.2, the AZ determination can be problematic in some cases, such as in this simulation. Instead of obtaining a reduction in the size of the envelope at the end of the computation as damage front is reaching domain boundaries (like in figure V.2 with the chalk example), the AZ is very large due to several spurious "close crack tip points" detection. The times given in table I.1 correspond to the computation up to load step 246 (figure VI.17) where the AZ envelope is almost the one expected. Additional 4.4 days were used to reach loading step 385. Once more, here, the AZ is not ideal during those last load steps and future work on crack tip detection will hopefully reduce those time consumption.

To have an insight on which part of the computation is consuming CPU

[1] For this notched beams test the AZ envelop is not well computed. This problem combined with too small elements size ($h_t \approx l_c/20$), makes irrelevant the usage of the AZ implementation. Therefore, the elapsed times are not representative.

Item	elapsed time	% of total	% of task
Mechanical problem resolution Algorithm 1 II	23.68	62.42	100
System creation (Integration + assembly)	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 I.3: Detailed computation times in hours for mechanical problem resolution of the spiral bevel pinion gear(the 246 first load steps).

Item	elapsed time	% of total	% of task
TLS update Algorithm 1 I+III	13.12	34.59	100
$\bar{Y}$ computation chapter III.4	5.24	13.8	39.9
Update domain (Γ_0 and Γ_c creation)	2.72	7.17	20.74
Update ϕ and other	5.16	13.61	39.35

Table I.4: Detailed computation times in hours for the TLS update of the spiral bevel pinion gear (the 246 first load steps).

time in a rather intensive simulation, the test case of the spiral bevel pinion gear was profiled. The results for 246 first load steps are given in table I.2, I.3 and I.4.

The first table presents an overall distribution of the computation time. First, regarding the TLS update task, which is detailed in table I.4, it participates for 34.59 % of the total simulation elapsed time, which is not negligible. The actual main consumers are the computation of $\bar{Y}$ and ϕ update. Both of those tasks are not well parallelized and correspond to a rather old algorithm. More recent fast marching techniques used in eXlibris will replace those computations and in particular, the algebraic system resolution attached to them.

One can note that the new double cut algorithm proposed in this thesis does not consume much computation time. It is embedded in the update domain item, which represents only 7.17 % of the total simulation elapsed time without being parallelized in these tests.

Secondly, regarding the overall time distribution, the efforts of reducing the mechanical problem resolution via the parallelism and AZ techniques, are well justified because the corresponding task remains to be the most important consumer. As detailed in table I.3, it is mostly penalized by the algebraic resolution (78.91 %: factorization, condensation, solving, ...). The matrix creations (elementary creation and assembly), are not greedy in terms of time consumption despite the use of sub-elements for integration (XFEM technique). The parallelism associated with a proper load balancing (annex F.1) kept the consumption low anyway. Remaining sequential computation such as dof updates will have to be also treated in parallel in the future.

	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

Table I.5: Elapsed times computation (in hours) up to 80^{th} load step for the spiral bevel pinion gear (chapter VI.5). Comparison between the sequential and parallel versions with or without AZ. The ratio corresponds to a comparison per column, line and diagonal.

To conclude, the AZ performances are again illustrated by the spiral bevel pinion gear test case which is an intensive simulation. In table I.5, the elapsed times after 80 load steps are given for different configurations. Upper left results correspond to a sequential computation with no AZ nor 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 the one hand, the introduction of AZ and condensation (upper right) reduces the time consumption by 6.35. It validates by itself the interest of this technique. On the other hand, applying the parallel computation on an intensive task (lower left) reduces the time consumption by 8.56. Now applying the parallel computation on the AZ technique (lower right) divides the time consumption by 3.25. If the CPU time of mechanical problem computation is extracted from the global time, it is a ratio of 8.08, which is in fact obtained in this case. It illustrates the right choice of the association of parallel technique with the dense condensed resolution. We can add that the memory consumption becomes an issue with the condensation if no parallel distribution of the dense system is made. In parallel, the use of the AZ technique reduces the time consumption by 2.41, which is less than sequential ratio but still interesting. Compared to a non-AZ sequential version, adding both strategies as proposed in this work, reduces time consumption by 20.64.

J Material characteristics and simulation parameters

Test case	L shape (VI.4)	Chalk (VI.2)	Spherical Holes (VI.1)	Spiral bevel gear (VI.5)	Tree point bending (VI.3)
Material	PMMA	Chalk	PMMA	Steel AISI9310	Concrete
Elastic Parameter :					
Young Modulus (N/mm^2)	25 850	2 000	2 800	206 844	41 240
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	$Y_c^0.h(d)$
traction/compression parameter β	0	0	1	0	0
TLS parameter :					
l_c (mm)	100.	4	10.	0.7	20.
$d(\phi)$	$\begin{cases} 0 \; for \; \phi < 0 \\ \left(\frac{\phi}{l_c}\right)^2 \left(3 - 2\frac{\phi}{l_c}\right) \; for \; 0 \leqslant \phi \leqslant l_c \\ 1 \; for \; l_c < \phi \end{cases}$				$\begin{cases} 0 \; for \; \phi < 0 \\ \frac{\phi}{l_c}\left(2 - \frac{\phi}{l_c}\right) \; for \; 0 \leqslant \phi \leqslant l_c \\ 1 \; for \; l_c < \phi \end{cases}$

Table J.1: Simulation parameter set.

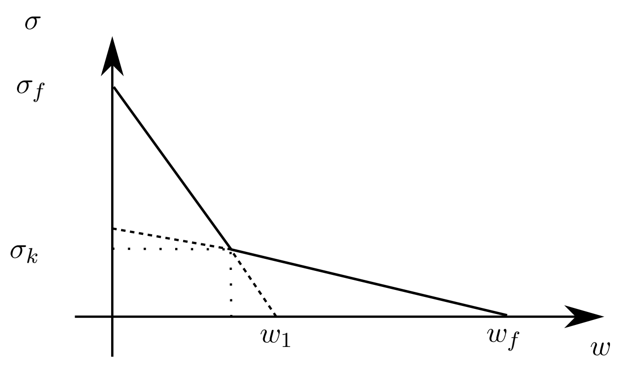


Figure J.1: Bilinear cohesive zone softening law in terms of stress versus opening displacement.

J.1 Material characteristics and parameters

The material characteristics and parameters of the model used in chapter VI are presented in table J.1. For the test case in VI.3, the critical damage energy release rate is a function of the damage. This function is built using an equivalence with a bi-linear cohesive zone model, as stated in Parrilla Gómez et al. (2015). The four extra information mandatory to describe $Y_c^0\ h(d)$ are coming from the description of the bi-linear softening law given in terms of stress versus opening displacement (figure J.1). Table J.2 gives the values fitted by Hoover and Bažant (2014) and used in this paper.

	Values
σ_f	3.92MPa
σ_k	0.588MPa
w_1	25.3μm
w_f	94.8μm

Table J.2: Bilinear cohesive zone softening law coefficients from Hoover and Bažant (2014).

J.2 Simulation condition

Two computers were mainly used for this thesis. They are named Titan and Liger.

Titan is an AMD based computer made of 5 nodes, with 32 cores (4 x 8 cores Opteron 6328 at 3.2MHz) per nodes. These nodes use raid 0 local hard disk for I/O and between 200 to 380 GB of memory. The network is an InfiniBand 4x DDR. The software used on this platform are eXlibris (2016), GNU g++ compiler from GCC 4.8.2 , MVAPICH2 2.0, Mumps 4.10.0, Scalapack 1.8.0, ParMetis 3.1.1, CGAL 4.2, lapack 3.4.2, openBlas 0.2.11.

Liger is an INTEL based computer made of 252 nodes, with 24 cores (2 x 12 cores Xeon E5-2680v3 at 2.5GHz) per nodes. Thses nodes use Gpfs network disk for I/O and 128GB of memory. The software used on this platform are eXlibris (2018), Intel icpc (using GCC 4.9) compiler, Intel MPI, Mumps 5.1.2, Intel MKL Scalapack, ParMetis 4.0.3, CGAL 4.2, intel MKL lapack, Intel MKL blas. All Intel software are coming from parallel studio xe 2015 update 3 suites.

The linear system resolutions use mainly Mumps (Amestoy et al. (2001) and Amestoy et al. (2006)).

Glossary

A | C | D | E | G | H | M | P | R | S | T | W | X

A

aspect ratio

Criterion that gives the finite element quality (annex B).

AZ

Active zone: Solving strategy using condensation techniques presented in chapter V.3.

C

core

Basic computational unit of the CPU: A core can only run one program or thread of execution at a time (except with a hardware multi-threaded core like Intel chipset).

CPU

Central processing unit also called the main processor.

crack tip

Crack location where both lips are connected. It is also called the crack front since the crack grows from it. It corresponds to points (in 2D) or lines (in 3D). See figure I.1b .

crack wake

Crack location where both lips are not connected. It is assimilated to the crack path. See figure I.1b .

CZM

Cohesive Zone Model.

D

DCA

Double Cut Algorithm.

dof

Degree Of Freedom.

E

efficiency

Measure of the scalability: for the strong scaling, it corresponds to the ratio of the speedup by the numbers of used cores. For the weak scaling it corresponds to the ratio of the program execution time ran with one core by the program execution time ran with N cores (equivalent to the speedup but in the context of a constant amount of work per process).

eXlibris

Set of C++ libraries providing FEM, XFEM, TLS, linear algebra, cutting algorithm, tensors, ... tools. This thesis relies on those libraries.

G

Γ_0

Damage front.
iso-0 level set of the signed distance function ϕ.

Γ_c

Crack lips.
iso-l_c level set of the signed distance function ϕ.

GFEM

Generalized Finite Element Method.

H

hanging node

In Quadtree or Octree context, the hanging nodes exist in a cell boundary when they have no counterpart in neighborhood cells due to cell scale change. See annex C.

h_o

Mesh size outside the TLS damage band.

h_t

Mesh size in the crack tip vicinity.

h_w

Mesh size in the crack wake region.

M

MPI

Message Passing Interface: normalized parallel protocol that uses independent process on eventually independent computers which communicate by exchanging messages.

P

patch

In the TS method, we will define a patch of an entity as the independent fine-scale problem which is embedded into this entity support.

R

ρ

Characteristic length associated with the curvature of a crack path . See figure I.1b.

S

scalability

The scalability (also referred as the scaling efficiency) indicates how efficient an application is, when using an increasing number of parallel processing elements.

SGFEM

Stable Generalized Finite Element Method.

speedup

The speedup of a parallel program can be defined as the ratio of this program execution time ran with one core by the program execution time ran with N cores.

sSGFEM

Scaled Stable Generalized Finite Element Method.

strong scaling

The strong scaling of a parallel program can be defined as its scalability when starting from a given problem size, we increase the numbers of process used. It represents the application capacity to face a huge CPU-time consumption.

SuperPatch

In the TS strategy, it corresponds to a mesh that groups all patches fine-scale elements.

support

The support of an entity is the set of elements over which the approximation function associated with this entity is non-zero..

SVDF

Signed Vector Distance Function.

T

thread-safe

A program or a function is thread-safe if, when it is launched in a multi-threaded environment, it does not do concurrent uncontrolled writing access to a unique memory location.

TLS

Thick Level Set.

TS

Two-scale or GFEM^{gl} method.

W

weak scaling

The weak scaling of a parallel program can be defined as its scalability when starting from a given problem size, we increase both the numbers of process and the problem size so that the amount of work (dofs in FEM) per computing unit remains constant. It represents the application capacity to face huge memory consumption.

X

XFEM

Extended Finite Element Method.

List of table and figures

L.1 List of Figures

L.2 List of Tables

Bibliography

P. Amestoy, I. Duff, J. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001. doi: 10.1137/S0895479899358194. URL https://doi.org/10.1137/S0895479899358194. 152

P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J. L'Excellent, and C. Weisbecker. Improving multifrontal methods by means of block low-rank representations. *SIAM Journal on Scientific Computing*, 37(3):A1451–A1474, 2015. doi: 10.1137/120903476. 54

P. R. Amestoy, A. Guermouche, J.-Y. LExcellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing*, 32(2):136 – 156, 2006. ISSN 0167-8191. doi: https://doi.org/10.1016/j.parco.2005.07.004. URL http://www.sciencedirect.com/science/article/pii/S0167819105001328. Parallel Matrix Algorithms and Applications (PMAA04). 152

Z. P. Bažant and M. Jirasek. Nonlocal Integral Formulations of Plasticity and Damage : Survey of Progress. *Journal of Engineering Mechanics*, 128 (November):1119–1149, 2002. 15

T. Belytschko, Y. Lu, and L. Gu. Crack propagation by element-free Galerkin methods. *Engineering Fracture Mechanics*, 51(2):295–315, may 1995. ISSN 00137944. doi: 10.1016/0013-7944(94)00153-9. 14

H. Ben Dhia. Problemes mecaniques multi-echelles: La methode Arlequin. *Comptes Rendus de l'Academie de Sciences - Serie IIb: Mecanique, Physique, Chimie, Astronomie*, 326(12):899–904, 1998. ISSN 12874620. doi: 10.1016/S1251-8069(99)80046-5. 53

H. Ben Dhia and O. Jamond. On the use of XFEM within the Arlequin framework for the simulation of crack propagation. *Computer Methods in Applied Mechanics and Engineering*, 199(21-22): 1403–1414, apr 2010. ISSN 00457825. doi: 10.1016/j.cma.2009.11.014. URL http://dx.doi.org/10.1016/j.cma.2009.11.014https://linkinghub.elsevier.com/retrieve/pii/S0045782509003892. 53, 61

P.-E. Bernard, N. Moës, and N. Chevaugeon. Damage growth modeling using the thick level set (tls) approach: Efficient discretization for quasi-static loadings. *Computer Methods in Applied Mechanics and Engineering*, 233-236(0):11 – 27, 2012. ISSN 0045-7825. doi: http://dx.doi.org/10.1016/j.cma.2012.02.020. URL http://www.sciencedirect.com/science/article/pii/S004578251200062X. 16, 19, 21, 22, 24, 27, 34, 41, 74, 149

J. E. Bolander and S. Saito. Fracture analyses using spring networks with random geometry. *Engineering Fracture Mechanics*, 61(5-6):569–591, 1998. ISSN 00137944. doi: 10.1016/S0013-7944(98)00069-1. 12

S. Bordas, T. Rabczuk, and G. Zi. Three-dimensional crack initiation, propagation, branching and junction in non-linear materials by an extended meshfree method without asymptotic enrichment. *Engineering Fracture Mechanics*, 75(5):943–960, Mar. 2008. ISSN 00137944. doi: 10.1016/j.engfracmech.2007.05.010. URL http://linkinghub.elsevier.com/retrieve/pii/S0013794407002536. 14, 78, 162

F. Cazes and N. Moës. Comparison of a phase-field model and of a thick level set model for brittle and quasi-brittle fracture. *International Journal for Numerical Methods in Engineering*, 2015. doi: 10.1002/nme.4886. 21

T. Coupez. Metric construction by length distribution tensor and edge based error for anisotropic adaptive meshing. *Journal of Computational Physics*, 230(7):2391–2405, 2011. ISSN 00219991. doi: 10.1016/j.jcp.2010.11.041. URL http://dx.doi.org/10.1016/j.jcp.2010.11.041. 47, 49

H. S. M. Coxeter. *Regular Polytopes, 3^d ed.* Dover Publications, 1973. ISBN 9780486614809. 33

P. A. Cundall and O. D. L. Strack. A discrete numerical model for granular assemblies. *Géotechnique*, 29(1):47–65, mar 1979. ISSN 0016-8505. doi: 10.1680/geot.1979.29.1.47. 12

C. A. Duarte and D. J. Kim. Analysis and applications of a generalized finite element method with global-local enrichment functions. *Computer Methods in Applied Mechanics and Engineering*, 197(6-8):487–504, 2008. ISSN 00457825. doi: 10.1016/j.cma.2007.08.017. 53, 59

I. Fracture Analysis Consultants. FRANC3D, 2019. URL http://fracanalysis.com. 13

M. Frémond and B. Nedjar. Damage, gradient of damage and principle of virtual power. *International Journal of Solids and Structures*, 33(8):1083–1103, 1996. ISSN 00207683. doi: 10.1016/0020-7683(95)00074-7. 15

I. Gargantini. An effective way to represent quadtrees. *Commun. ACM*, 25(12):905–910, Dec. 1982. ISSN 0001-0782. doi: 10.1145/358728.358741. 42

B. Giovanardi, A. Scotti, and L. Formaggia. A hybrid XFEM - Phase field (Xfield) method for crack propagation in brittle elastic materials. *Computer Methods in Applied Mechanics and Engineering*, 320:396–420, 2017. ISSN 00457825. doi: 10.1016/j.cma.2017.03.039. URL http://dx.doi.org/10.1016/j.cma.2017.03.039. 53, 61

J. Gomes and O. Faugeras. The vector distance functions. *International Journal of Computer Vision*, 52:161–187, 2003. 26, 27, 29

A. A. Griffith. The Phenomena of Rupture and Flow in Solids. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 221(582-593):163–198, jan 1921. ISSN 1364-503X. doi: 10.1098/rsta.1921.0006. 12

V. Gupta, C. Duarte, I. Babuška, and U. Banerjee. Stable GFEM (SGFEM): Improved conditioning and accuracy of GFEM/XFEM for three-dimensional fracture mechanics. *Computer Methods in Applied Mechanics and Engineering*, 289(FEBRUARY):355–386, 2015. ISSN 00457825. doi: 10.1016/j.cma.2015.01.014. URL `http://linkinghub.elsevier.com/retrieve/pii/S0045782515000341`. 67

P. Hachenberger and L. Kettner. 3D Boolean operations on Nef polyhedra. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.3 edition, 2013. URL `http://doc.cgal.org/4.3/Manual/packages.htmlPkgNef3Summary`. 97

S. Hert and S. Schirra. 3D convex hulls. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.3 edition, 2013. URL `http://doc.cgal.org/4.3/Manual/packages.htmlPkgConvexHull3Summary`. 97

C. G. Hoover and Z. P. Bažant. Cohesive crack, size effect, crack band and work-of-fracture models compared to comprehensive concrete fracture tests. *International Journal of Fracture*, 187(1):133–143, 2014. ISSN 1573-2673. doi: 10.1007/s10704-013-9926-0. URL `http://dx.doi.org/10.1007/s10704-013-9926-0`. 80, 151, 152, 165

C. G. Hoover, Z. P. Bažant, J. Vorel, R. Wendner, and M. H. Hubler. Comprehensive concrete fracture tests: Description and results. *Engineering Fracture Mechanics*, 114:92 – 103, 2013. ISSN 0013-7944. doi: http://dx.doi.org/10.1016/j.engfracmech.2013.08.007. 79, 80

G. Irwin. Analysis of stresses and strains near the end of a crack traversing a plate. *Journal of Applied Mechanics*, 24:361–364, 01 1957. 12

J.A. Sethian. *Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, Cambridge Monograph on Applied and Computational Mathematics edition, 1999. 14

A. Karma, D. A. Kessler, and H. Levine. Phase-field model of mode III dynamic fracture. *Physical Review Letters*, 87(4):45501–1–45501–4, 2001. ISSN 10797114. doi: 10.1103/PhysRevLett.87.045501. URL `http://link.aps.org/doi/10.1103/PhysRevLett.87.045501`. 15

D. J. Kim, C. A. Duarte, and N. A. Sobh. Parallel simulations of three-dimensional cracks using the generalized finite element method. *Com-*

putational Mechanics, 47(3):265–282, 2011. ISSN 01787675. doi: 10.1007/s00466-010-0546-5. 53, 64, 66, 72

B. Lé, N. Moës, and G. Legrain. Coupling damage and cohesive zone models with the Thick Level Set approach to fracture. *Engineering Fracture Mechanics*, 193:214–247, 2018. ISSN 00137944. doi: 10.1016/j.engfracmech.2017.12.036. 82

H. Li and C. A. Duarte. A two-scale generalized finite element method for parallel simulations of spot welds in large structures. *Computer Methods in Applied Mechanics and Engineering*, 337:28–65, 2018. ISSN 00457825. doi: 10.1016/j.cma.2018.03.030. 60, 63

E. Lorentz and V. Godard. Gradient damage models: Toward full-scale computations. *Computer Methods in Applied Mechanics and Engineering*, 200 (21-22):1927–1944, May 2011. ISSN 00457825. doi: 10.1016/j.cma.2010.06.025. 15, 82, 83, 84

J. Mazars. A description of micro- and macroscale damage of concrete structures. *Engineering Fracture Mechanics*, 25(5-6):729–737, 1986. ISSN 00137944. doi: 10.1016/0013-7944(86)90036-6. 18, 21

C. Miehe, F. Welschinger, and M. Hofacker. Thermodynamically consistent phase-field models of fracture: Variational principles and multi-field FE implementations. *International Journal For Numerical Methods in Engineering*, 2010. doi: 10.1002/nme.2861. 15

N. Moës, J. Dolbow, and T. Belytschko. A finite element method for crack growth without remeshing. *International Journal for Numerical Methods in Engineering*, 46(1):131–150, 1999. ISSN 00295981. doi: 10.1002/(SICI)1097-0207(19990910)46:1⟨131::AID-NME726⟩3.0.CO;2-J. URL http://www.researchgate.net/publication/51992357_A_finite_element_method_for_crack_growth_without_remeshing/file/32bfe5134c0d103772.pdf. 14

N. Moës, C. Stolz, and N. Chevaugeon. Coupling local and non-local damage evolutions with the thick level set model. *Advanced Modeling and Simulation in Engineering Sciences*, 1(1):16, 2014. doi: 10.1186/s40323-014-0016-2. URL http://dx.doi.org/10.1186/s40323-014-0016-2. 19, 77

K. Moreau, N. Moës, D. Picart, and L. Stainier. Explicit dynamics with a non-local damage model using the thick level set approach. *International Journal for Numerical Methods in Engineering*, 102(3-4):808–838, apr 2015. ISSN 00295981. doi: 10.1002/nme.4824. URL http://doi.wiley.com/10.1002/nme.4824. 24

K. Moreau, N. Moës, N. Chevaugeon, and A. Salzman. Concurrent development of local and non-local damage with the Thick Level Set approach: Implementation aspects and application to quasi-brittle failure. *Computer Methods in Applied Mechanics and Engineering*, 2017. ISSN 00457825. doi: 10.1016/j.cma.2017.08.045. 24, 92

A. Parrilla Gómez, N. Moës, and C. Stolz. Comparison between Thick Level Set (TLS) and cohesive zone models. *Advanced Modeling and Simulation in Engineering Sciences*, 2015. doi: 10.1186/s40323-015-0041-9. URL http://dx.doi.org/10.1186/s40323-015-0041-9. 21, 80, 84, 151

A. Parrilla Gómez, C. Stolz, N. Moës, D. Grégoire, and G. Pijaudier-Cabot. On the capability of the Thick Level Set (TLS) damage model to fit experimental data of size and shape effects. *Engineering Fracture Mechanics*, 184:75–87, 2017. ISSN 00137944. doi: 10.1016/j.engfracmech.2017.07.014. 80

R. H. J. Peerlings, R. de Borst, W. A. M. Brekelmans, and M. G. D. Geers. Gradient-enhanced damage modelling of concrete fracture. *Mechanics of Cohesive-Frictional Materials*, 3(4):323–342, 1998. ISSN 10825010. doi: 10.1002/(SICI)1099-1484(1998100)3:4⟨323::AID-CFM51⟩3.0.CO;2-Z. 15

J. P. A. Pereira, D.-J. Kim, and C. A. Duarte. A two-scale approach for the analysis of propagating three-dimensional fractures. *Computational Mechanics*, 49(1):99–121, Aug. 2011. ISSN 0178-7675. doi: 10.1007/s00466-011-0631-4. 53, 60, 61

G. Pijaudier-Cabot and Z. P. Bažant. Nonlocal damage theory. *Journal of Engineering Mechanics ASCE*, 113:1512–1533, 1987. 15

J. Plews and C. Duarte. Generalized finite element approaches for analysis of localized thermo-structural effects. *International Journal for Numerical Methods in Engineering*, 104(6):408–438, nov 2015. ISSN 00295981. doi: 10.1002/nme.4942. 60

Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition, 2003. doi: 10.1137/1.9780898718003. URL https://epubs.siam.org/doi/abs/10.1137/1.9780898718003. 52

A. Salzman, N. Moës, and N. Chevaugeon. On use of the thick level set method in 3d quasi-static crack simulation of quasi-brittle material. *International Journal of Fracture*, pages 1–29, 2016. ISSN 1573-2673. doi: 10.1007/s10704-016-0132-8. URL http://dx.doi.org/10.1007/s10704-016-0132-8. 19, 27, 75

A. Salzman, N. Chevaugeon, N. Moës, and G. Legrain. On performances of the thick level set method in 3d quasi-static fracture simulation of quasi-brittle material. In CFRAC2017, editor, *BOOK OF ABSTRACTS*, volume 1, pages 37 – 38. Fifth International Conference on Computational Modeling of Fracture and Failure of Materials and Structures, GeM-ECN-Eccomas, 6 2017. URL https://cfrac2017.sciencesconf.org/. 64

A. G. Sanchez-Rivadeneira and C. A. Duarte. A stable generalized/eXtended FEM with discontinuous interpolants for fracture mechanics. *Computer Methods in Applied Mechanics and Engineering*, 345:876–918, 2019. ISSN 00457825. doi: 10.1016/j.cma.2018.11.018. URL https://doi.org/10.1016/j.cma.2018.11.018. 69

W. Schroeder, B. Geveci, and M. Malaterre. Compatible triangulations of spatial decompositions. In *IEEE Visualization 2004*, pages 211–217. IEEE Comput. Soc, 2004. ISBN 0-7803-8788-0. doi: 10.1109/VISUAL.2004.15. URL http://ieeexplore.ieee.org/document/1372199/. 107

Siemens. Simcenter samcef, 2019. URL https://www.plm.automation.siemens.com/global/en/products/simcenter/. 13

A. Sillem, A. Simone, and L. J. Sluys. The Orthonormalized Generalized Finite Element Method-OGFEM: Efficient and stable reduction of approximation errors through multiple orthonormalized enriched basis functions. *Computer Methods in Applied Mechanics and Engineering*, 287:112–149, 2015. ISSN 00457825. doi: 10.1016/j.cma.2014.11.043. URL http://dx.doi.org/10.1016/j.cma.2014.11.043. 68

L. E. Spievak, P. a. Wawrzynek, and A. R. Ingraffea. Simulating Fatique Crack Growth in Spiral Bevel Gears. Technical Report May 2000, NASA/CR, Glenn Research Center, 2000. 84, 85

L. E. Spievak, P. a. Wawrzynek, A. R. Ingraffea, and D. G. Lewicki. Simulating fatigue crack growth in spiral bevel gears. *Engineering Fracture Mechanics*, 68(1):53–76, Jan. 2001. ISSN 00137944. doi: 10.1016/S0013-7944(00)00089-8. URL http://linkinghub.elsevier.com/retrieve/pii/S0013794400000898. 84, 85

A. Ural, G. Heber, P. A. Wawrzynek, A. R. Ingraffea, D. G. Lewicki, and J. B. C. Neto. Three-dimensional, parallel, finite element simulation of fatigue crack growth in a spiral bevel pinion gear. *Engineering Fracture Mechanics*, 72: 1148–1170, 2005. doi: 10.1016/j.engfracmech.2004.08.004. 84, 85, 87, 88, 162

F. P. Van Der Meer and L. J. Sluys. The Thick Level Set method: Sliding deformations and damage initiation. *Computer Methods in Applied Mechanics and Engineering*, 285(285):64–82, 2015. doi: 10.1016/j.cma.2014.10.020. URL www.elsevier.com/locate/cmahttp://dx.doi.org/10.1016/j.cma.2014.10.020. 77

Titre : Développement de l'approche « Thick Level Set » dans un cadre 3D parallèle

Mots clés : TLS, fracture, parallèle, two-scale, adaptation de maillage

Résumé : La simulation en quasi statique de fissures complexes en 3D avec des matériaux quasi-fragiles est encore difficile à aborder de nos jours. De nombreuses méthodes et modèles proposent des solutions partielles à ce problème. Le modèle Thick Level Set (TLS), utilisant une approche combinant la mécanique de l'endommagement et la représentation explicite des fissures, permet, dans la simulation, l'initiation de fissures et leurs croissances complexes (coalescence ou ramification en suivant des chemins sinueux). Dans cette thèse, nous montrons que la mise en œuvre de ce modèle dans un contexte 3D parallèle fournit un outil précis, polyvalent et avec un bon potentiel d'adaptabilité au parallélisme.
En ce qui concerne la précision, un nouvel outil appelé « algorithme de double coupe », laisse passer une zone complètement endommagée dans un élément de maillage sans condition de taille. Cet outil a amélioré la mise en œuvre existante de la TLS et apporte également un moyen d'optimiser la discrétisation en grossissant le maillage dans le sillage des fronts de fissure. Cette adaptation réduit la taille du problème mécanique discrétisé et, par conséquent, les efforts de résolution du système linéaire algébrique associé.
En ce qui concerne l'adaptabilité au parallélisme, le goulot d'étranglement est le temps de résolution du système algébrique et la consommation de mémoire associée. La stratégie développée dans cette thèse, pour la résolution parallèle de ces systèmes, commence par une approche de base. Ensuite, pour améliorer l'adaptabilité au parallélisme, des méthodes liées à la décomposition de domaine et au multi-échelle sont investiguées. Le reste des tâches de résolution de la TLS sont, quant à elles, partiellement parallélisées. La principale préoccupation est d'obtenir un outil qui sera globalement plus rapide ou capable de traiter des problèmes plus importants, si on lui fournit plus d'unités de traitement.
Enfin, certains cas tests illustrent les résultats obtenus avec une implémentation 3D parallèle.

Title : Thick Level Set model implementation in 3D parallel context

Keywords : TLS, fracture, parallel, two-scale, mesh adaptation

Abstract :
A complex 3D fracture simulation of quasi-brittle material in quasi-static is still hard to tackle nowadays. Many methods and models propose partial solutions to this problem. The Thick Level Set (TLS) model, which uses an approach mixing damage mechanics and explicit crack representation, provides an easy fracture initiation, a complex crack growing capability (coalescing or branching) and an accurate tortuous fracture path. In this thesis we will demonstrate that the implementation of this model in a parallel 3D context provides an accurate and versatile tool that potentially scales.
Regarding the accuracy, a novel tool called the “double cut algorithm” has enhanced the existing TLS implementation by letting pass a straight fully damaged zone in a mesh element without conditions on its size. This tool also brings a way to optimize the discretization by coarsening the mesh in a crack front wake. This adaptation reduces the size of the discrete mechanical problem and therefore the effort for the linear algebra resolution.
As far as the scaling is concerned, the bottleneck is the linear algebra resolution time and its associated memory consumption. The parallel solving strategy developed in this thesis to tackle this problem starts first with a basic approach. Then by switching to a method close to domain decomposition and later to a two-scale method, it permits increasing scalability. The other TLS tasks are also partially parallelized. The principal concern is to obtain a tool that either runs faster or can treat a more significant problem if we provide much more computational units.
Finally, some test cases illustrate the obtained results with a parallel 3D implementation.