**République Algérienne Démocratique et Populaire**

**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**

**Université Echahid Cheik Larbi Tébessi - Tébessa**

**Faculté des Sciences et de la Technologie**

**Département de Génie civil**

# THÈSE

Pour l'obtention du diplôme de

## Doctorat LMD

**En :** Génie Civil

**Spécialité :** Structures et matériaux avancés

**Par :**

**REBADJ CHABANE**

**THÈME**

# Étude et modélisation des discontinuités dans le domaine statique linéaire et non linéaire pour différents types de structures en utilisant la méthode des Éléments Finis Étendus

Présentée et soutenue, le **07 / 07 / 2026**, devant le jury composé de :

| | | |
|---|---|---|
| M. HARKATI El Hadi | Professeur / Université de Tébessa | Président |
| M. BELGASMIA Mourad | Professeur /Université de Sétif 1 | Rapporteur |
| M. KEBICHE Khelifa | Professeur / Université de Sétif 1 | Examinateur |
| M. LAOUAR Mohamed Salah | Professeur / Université de Tébessa | Examinateur |
| M. DEGHBOUDJ Samir | Professeur / Université de Tébessa | Examinateur |
| M. LABED Abderrahim | MCB / Université de Tébessa | Invité |

# Remerciements

*En premier lieu, Je tiens à exprimer ma gratitude envers ALLAH clément et miséricordieux pour la force et la persévérance qu'il m'a accordées tout au long de ce travail modeste.*

*Je souhaite témoigner ma sincère reconnaissance à toutes les personnes, qui par leurs soutiens et leurs contributions ont rendu possible l'aboutissement de cette thèse. Ma première pensée va à mon directeur de thèse Monsieur* ***BELGASMIA Mourad****, Professeur à l'Université Ferhat Abbas, Setif1, pour son encadrement précieux, sa constante disponibilité, sa bienveillance et ses conseils perspicaces, qui ont été mes repères tout au long de ce parcours.*

*J'adresse également mes remercîments les plus sincères au professeur* ***HARKATI El Hadi****, directeur du laboratoire des mines de l'université Larbi Tebessi de Tébessa, et chef du comité de formation doctorale (CFD), pour m'avoir accueilli au sein de son laboratoire et pour l'appui administratif et scientifique essentiel qu'il m'a apporté.*

*Je suis très honoré et reconnaissant envers les membres de jury : Monsieur* ***KEBICHE Khelifa****, Professeur à l'Université Ferhat Abbas, Setif1, Monsieur* ***LAOUAR Mohamed Salah****, Professeur à l'Université de Larbi Tebessi, Tébessa, Monsieur* ***DEGHBOUDJ Samir****, Professeur à l'Université de Larbi Tebessi, Tébessa qui ont accepté d'évaluer ce travail. Je remercie également Monsieur* ***LABED Abderrahim*** *enseignant à l'Université de Larbi Tebessi-Tébessa, d'avoir accepté d'être l'invité d'honneur de la soutenance.*

*Je tiens à aussi à exprimer ma profonde gratitude envers l'équipe de la recherche, formée de Monsieur* ***BELGASMIA Mourad*** *et Madame* ***MOUSSAOUI Sabah*** *de l'Université Ferhat Abbas, Setif1, pour leurs collaboration dans le travail scientifique, l'analyse des données et la rédaction des articles scientifiques.*

*Ma gratitude va également à ma famille. Je remercie mes chers parents,* ***REBADJ Mahmoud*** *et* ***KERMICHE Bornia,*** *pour leurs soutient permanant. Je remercie mon épouse,* ***AMRANI Amel****, professeure à l'Université de Constantine, pour ses conseils et sa présence. Je remercie mes précieux enfants,* ***Salah Eddine****,* ***Hanine*** *et* ***Abderrahmane*** *; et leur souhaite pleine réussite dans leurs études. Je remercie enfin mes* ***frères*** *et* ***sœurs*** *et leurs familles pour leur affection et leur soutient.*

*Je remercie également toutes les personnes qui, par leur enseignement, leurs conseils ou leur bienveillance ont contribué à ce travail et ont marqué mon parcours.*

*REBADJ CHABANE* ***07/07/2026***

# Résumé :

Malgré les avancées significatives de la mécanique linéaire élastique de la rupture (**LEFM**), la modélisation numérique des fissures et l'évaluation précise des facteurs d'intensité de contrainte (stress intensity factors, SIFs)- demeurent des défis majeurs, notamment pour les configurations, de chargements en mode mixtes. Cette thèse propose une formulation de la méthode des éléments finis étendus (**XFEM**) dédiée à l'analyse quasi-statique de structures fissurées, appliquée à une plaque élastique comportant une fissure inclinée soumise à différents chargements uniaxiaux et biaxiaux, en régime linéaire et non linéaire.

L'approche repose sur l'enrichissement et l'extension d'une bibliothèque de calcul orientée objet en **C++**, **OpenXFEM++**, dont l'architecture modulaire facilite l'intégration des enrichissements **XFEM**, les champs **Level Set** et de nouvelles lois de comportement. L'un des principales contributions de cette recherche réside dans la mise en œuvre d'une plateforme numérique extensible permettant une modélisation robuste et flexible des discontinuités ainsi que l'intégration cohérente de nouveaux modèles physiques. La formulation est validée par une série de benchmarks académiques couvrant les modes de rupture I, II et mixte, montrant une excellente concordance avec les solutions analytiques de référence, les résultats de la littérature et les simulations réalisées sous Abaqus, tout en présentant une convergence indépendante du maillage.

Une étude paramétrique met également en évidence l'influence des conditions de chargement et des paramètres géométriques sur l'évolution des facteurs d'intensité de contrainte en mode mixte. Enfin, la plateforme est étendue au domaine élastoplastique grâce à l'implémentation d'une la loi de comportement de Von Mises associé à un algorithme de retour radial et une matrice tangente consistante, démontrant ainsi sa capacité de manière unifiée des problèmes avancés de mécanique de la rupture.

**Mots clés :**

XFEM, facteurs d'intensité de contrainte en mode mixte (MMSIFs), fissure centrale inclinée, modèles de chargement, programmation orientée objet en C++.

# Abstract:

Despite significant advances in linear elastic fracture mechanics (**LEFM**), the numerical modelling of cracks and the accurate evaluation of **stress intensity factors** (**SIFs**) remain major challenges, particularly for mixed-mode loading configurations.

This **PhD** thesis proposes an Extended Finite Element Method (**XFEM**) formulation dedicated to the quasi-static analysis of cracked structures, applied to an elastic plate containing an inclined crack subjected to various uniaxial and biaxial loading conditions under both linear and nonlinear regimes.

The proposed approach is based on expanding and enhancing **OpenXFEM++,** a C++ object-oriented **C++** computing library, whose modular architecture facilitates the integration of **XFEM** enrichments, Level Set functions and constitutive models.

One of the main contributions of this research is the development of an extensible numerical platform providing a robust and flexible framework for modeling discontinuities while enabling the seamless integration of new physical models.

The proposed formulation is validated through a series of the benchmarks problems covering the mode I, mode II and mixed-mode fracture, showing excellent agreement with analytical reference solutions, published numerical results and finite element simulations performed using **Abaqus**, together with mesh-independent convergence.

A comprehensive parametric study further highlights the influence of loading conditions and geometric parameters on the evolution of mixed-mode stress intensity factors.

Finally, the platform is extended to elastoplastic analysis through the implementation of a Von Mises constitutive model combined with a radial return algorithm and consistent tangent stiffness matrix, demonstrating its capacity to provide a unified and efficient computational framework for advanced fracture mechanics problems.

**Keywords:**

**XFEM**, mixed-mode stress intensity factor (**MMSIFs**), inclined central crack, loading patterns, C++ object-oriented programming.

# ملخص:

على الرغم من التقدم الكبير الذي حققته **ميكانيكا الكسر المرنة الخطية (LEFM)**، فان النمذجة العددية للشقوق و التقييم الدقيق **لمعاملات شدة الاجهاد** (**stress intensity factors , SIFs**) لا يزالان يمثلان تحديين رئيسيين، ولا سيما في مجال **التحميل المختلط**.

تقدم هذه الأطروحة صياغة **لطريقة العناصر المحدودة الممتدة (XFEM)** مخصصة للتحليل الشبه الساكن للهياكل المتشققة ، وتطبق على صفيحة مرنة تحتوي على شق مائل خاضعة لأنواع مختلفة من التحميل أحادي وثنائي المحور، في المجالين الخطي وغير الخطي.

يحتوى النهج المقترح على تطوير و توسيع المكتبة الحسابية **OpenXFEM++** المبنية بلغة **C++** وفق مبادئ البرمجة كائنية التوجه ، حيث تتيح بنيتها المعمارية دمج إغناءات **XFEM**، وحقول **Level Set** وقوانين السلوك الميكانيكي بكفاءة عالية. و تمثل احدى المساهمات العلمية لهذه الاطروحة في تطوير منصة عددية قابلة توفر اطارا مرنا و فعالا لنمذجة الانقطاعات، مع إمكانية دمج نماذج فيزيائية جديدة بصورة متسقة.

وقد جرى التحقق من صحة الصياغة المقترحة من خلال مجموعة من المسائل المرجعية التي تغطي أنماط الكسر من النمط الأول (mode I) والنمط الثاني (mode II) والنمط المختلط (mixte)، حيث أظهرت النتائج توافقا ممتازا مع الحلول التحليلية المرجعية ونتائج الأدبيات العلمية، ومحاكاة العناصر المحدودة المنجزة باستخدام البرنامج (**Abaqus**)، بالإضافة الى تحقيق تقارب مستقل عن كثافة الشبكة. كما ابرزت دراسة برامترية شاملة تأثير ظروف التحميل والمعلمات الهندسية على تطور عوامل شدة الاجهاد في نمط الكسر المختلط.

وأخيرًا، تم توسيع المنصة الى المجال مرن-لدن (**élastoplastic**) من خلال دمج نموذج فون ميزس (**Von Mises**) مع خوارزمية العودة الشعاعية (**radial return algorithm**) والمصفوفة المماسية المتسقة (**consistent tangent matrix**)، مما يبرهن على قدرة المنصة على توفير اطار حسابي موحد و فعال لمعالجة المشاكل المتقدمة في ميكانيكا الكسر.

**الكلمات المفتاحية:**

**XFEM**، عامل شدة الإجهاد في الوضع المختلط (**MMSIF**) ، الشق المركزي المائل، أنماط التحميل، البرمجة كائنية التوجه بلغة **C++**.

# Table des matières

Remerciements……I

Résumé……III

Abstract……IV

ملخـــــص……V

Table des matières……VI

Liste des figures……XI

Liste des tableaux……XV

Liste des symboles et abréviations ……XVI

**Introduction générale**……2

**Chapitre 1 : Théorie de la mécanique linéaire de rupture**

1 Introduction ..... 5

2 Généralité en Elasticité Linéaire ..... 6

2.1 Analyse énergétique ..... 7

2.2 Asymptotique Isotrope ..... 8

3 Facteur d'Intensité de Contrainte (SIF) ..... 10

3.1 L'intégrale J ..... 10

3.2 Intégrale interaction ..... 13

4 Etat de l'art des méthodes de simulation des fissures ..... 16

4.1 Méthodes classiques basées sur les éléments finis ..... 17

4.1.1 La méthode des éléments finis (FEM) ..... 18

4.1.2 Le relâchement de nœuds ..... 18

4.1.3 Le modèle de zone cohésive (CZM) ..... 19

4.1.4 La technique d'extension virtuelle de fissure (VCET) ..... 21

4.1.5 La technique de fermeture virtuelle de fissure (VCCT) ..... 21

4.1.6 Les méthodes de remaillage autour de la fissure ..... 23

4.1.7 L'élimination d'élément (EDM) ..... 23

4.2 Méthodes non basées sur les éléments finis ..... 23

4.2.1 Méthodes des éléments frontières (BEM) ..... 23

4.2.2 Méthodes sans maillage (Méthodes particulaire) ..... 24
4.3 Méthodes basées sur la partition de l'unité ..... 25
4.3.1 La partition de l'unité ..... 25
4.3.2 Méthode des éléments finis étendus (EFEM) ..... 26
5 Conclusion ..... 27

## Chapitre 2 : Principes de la méthode des éléments finis étendus (XFEM)

1 Introduction ---------- 29
2 Modèles existants de la discontinuité ---------- 29
2.1 Une discontinuité forte ..... 30
2.2 Une discontinuité faible ..... 30
3 Méthode des éléments finis étendus (XFEM) ---------- 31
3.1 Introduction ..... 31
3.2 Généralités ..... 31
4 Modélisation des fissures par XFEM ---------- 34
4.1 Types d'enrichissement ..... 34
4.1.1 La fonction Heaviside ..... 34
4.1.2 Fonctions asymptotiques pour singularité ..... 34
4.2 Stratégies d'enrichissement ..... 35
4.3 Formulation de l'approximation XFEM ..... 36
4.4 Discrétisation et opérateur gradient ..... 36
4.4.1 Expression discrète du champ de déplacement ..... 36
4.4.2 Opérateur gradient discrétisé ..... 37
4.4.3 Procédure éléments finis classique ..... 37
4.5 Construction des matrices de rigidité ..... 38
4.5.1 Expression de la matrice de rigidité enrichis ..... 38
4.5.2 Conditionnement de la matrice de rigidité ..... 39
4.6 Modélisation d'une fissure en 1D ..... 39
4.7 Modélisation d'une fissure en 2D ..... 42
5 Conclusion ..... 45

## Chapitre 3 : Implémentation XFEM par l'approche orientée objeen en C++

1. Introduction ---------- 47
2. Les caractéristiques de l'approche Orientée Objet ---------- 47
3. Aperçu sur le langage C++ ---------- 48

4. Implémentation orientée objet en C++ par éléments finis enrichis---------- 50
4.1. Classe Domain ..... 53
4.2. Classe FEM Component ..... 57
4.3. Classe Element ..... 58
4.3.1. ComputeTangentStiffnessMatrix() ..... 59
a) ComputeConstitutiveMatrix() ---------- 60
b) PlusProduct (b,db,dV) ---------- 60
c) ComputeBmatrixAt()---------- 60
d) ComputeVolumeAround() ---------- 60
4.3.2. IsEnriched() ..... 61
4.3.3. Les fonctions « assembleLhsAt( ) » et « assembleRhsAt( ) » ..... 62
4.4. Classe Gausspnt ..... 64
4.5. Classe GeometryEntity ..... 65
4.5.1. Piecewiselinear ..... 66
a) La méthode getListOfVertices() ---------- 66
b) La méthode FindSegmentClosestTo() ---------- 66
c) La méthode computeSignedDistanceOfPoint()---------- 67
d) La méthode givePositionComparedTo() ---------- 68
e) Les méthodes intersects(Triangle* t) et intersects(Segment* s) ---------- 69
f) Les méthodes intersection(Triangle* t) et intersection(:Segment* s) ---- 69
4.5.2. Vertex ..... 69
a) La méthode getCoordinates()---------- 69
b) La méthode booléenne interactsWith()---------- 69
c) Les méthodes de prédicat (méthodes booléennes) ---------- 70
4.6. Classe EnrichmentItem ..... 71
a) getGeometry() et giveMyGeo() ---------- 71
b) GiveEnrFuncVector() ---------- 73
4.6.1. CrackInterior ..... 74
a) Getmytips() et Givemytips()---------- 75
b) TreatMeshGeoInteractionForMyTips() ---------- 75
c) GiveCrackLength() et computeCrackLength() ---------- 78
d) ResolveLinearDependency() ---------- 78
e) UpdateMyGeometry()---------- 81
4.6.2. CrackTip ..... 82

a) ComputeLocalCoordOf(Mu::Point* p) ---------- 82
b) ComputePolarCoordOf(Mu::Point *p) ---------- 82
c) GiveRadiusOfDomainIntegration(), makeBall(), buildIntegrationDomain1() , buildIntegrationDomain2() . ---------- 82
d) ComputeSIFs() et computeK_eq() ---------- 85
e) computeSIFMultiplierForOneMat().---------- 86
f) computeInteractionIntegral() ---------- 86
g) UpdateMyGeometry() ---------- 90
h) UpdateEnrichment() ---------- 92
4.7. Classe enrichmentdetector ..... 93
4.8. Enrichmentfunction ..... 94
4.8.1. Discontinuousfunction ..... 95
4.8.2. HomogeneousElastCrackAsymp ..... 96
4.9. IntegrationRule ..... 98
4.9.1. SplitGaussLegendreQuadrature ..... 98
4.9.2. StandardGaussLegendreQuadrature ..... 103
4.10. Classe Load ..... 104
5 Conclusion ---------- 107

**Chapitre 4 : Validation et discussion des résultats numériques**

1. Introduction ..... 108
2. Fissure centrale inclinée sous chargement en traction ..... 108
3. Effet de l'orientation de la fissure inclinée sur sifs : ..... 111
4. Deux fissures centrales inclinées similaires ..... 115
5. Fissure centrale inclinee dans une plaque soumise à un chargement biaxial ..... 117
6. Conclusion ..... 122

**Chapitre 5 : Modélisation non-linéaire par XFEM**

1. Introduction ..... 123
2. Equations fondamentales de la mécanique des milieux continus : ..... 124
2.1. Equations d'équilibre : ..... 124
2.2. Conditions aux limites ..... 125
2.3. Relations cinématiques ..... 126

2.4. Lois de comportement élastique linéaire ..... 126
3. Fondements de la théorie de la plasticité ..... 127
3.1. Décomposition additives des déformations ..... 127
3.2. Lois de comportement incrémentale ..... 127
3.3. Surface de charge et critères d'écoulement ..... 128
3.4. Loi d'écoulement plastique ..... 128
3.5. Condition de Khun-Tucker : ..... 129
3.6. Condition de consistance ..... 129
3.7. Représentation conceptuelle de la surface de charge ..... 130
4. Critère de von Mises ..... 131
5. Itégration numérique : algorithme de retour radial ..... 136
6. Matrice constitutive élasto-plastique ..... 140
7. Application numérique ..... 148
8. Conclusion ..... 155
**Conclusion générale** ---------- 157
**Référence bibliographiques** ---------- 161

# Liste des figures

## Chapitre 1 :
## Théorie de la mécanique linéaire de rupture

Figure 1-1 Mode de rupture…….……..…7

Figure 1-2 Paramétrage de la pointe de fissure…….……8

Figure 1-3 Schéma de l'intégral J au fond de fissure…….……..12

Figure 1-4 Schéma de l'intégrale d'interaction au fond de fissure…….……..15

Figure 1-5 Fonction de pondération sur les éléments…….……..16

Figure 1-6 Méthode de relâchement des nœuds…….……..19

Figure 1-7 Méthode des éléments cohésifs…….……..20

Figure 1-8 Techniques de fermeture virtuelle de fissure pour élément bidimensionnel…..…22

## Chapitre 2 :
## Principes de la méthode des éléments finis étendus (XFEM)

Figure 2 1 Différents types de discontinuités……..……..31

Figure 2-2 Discrétisation d'un problème de joint pour a) un modèle XFEM avec un maillage structuré b) un modèle FEM……..……33

Figure 2-3 Coordonnées polaires dans la base locale en fond de fissure……..……..35

Figure 2 4 Modélisation d'une discontinuité de champ de déformation dans une barre unidimensionnelle……..….40

Figure 2-5 Modélisation d'une discontinuité de champ de déformation dans deux éléments quadrilatères bidimensionnels……..…..…43

Figure 2 6 Subdivision des éléments quadrilatères de types Q4 en sous-domaines triangulaires……..…..43

## Chapitre 3 :
## Implémentation XFEM par l'approche orientée en C++

Figure 3-1 Organigramme d'héritage de la classe FEM Component……..…..52

Figure 3-2 Schéma d'appel de la méthode instanciateYourself……..…..54

Figure 3-3 Diagramme d'activité de la méthode solveFractureMechanicsProblemAt…..…57

Figure 3-4 Graphique d'appel de la méthode ComputeTangentStiffnessMatrix……..61

Figure 3-5 Diagramme de collaboration pour la classe Elément……..64

Figure 3-6 Organigramme d'héritage de la classe GeometryEntity…….…....66

Figure 3-7 Schéma de détermination d'un segment…….….67

Figure 3-8 Graphique d'appel de la méthode intersection…….…..70

Figure 3-9 Organigramme d'héritage de la classe EnrichmentItem……..…...72

Figure 3-10 Graphique d'appel de la méthode giveEnrFuncVector()……..…73

Figure 3-11 Graphique d'appel de la méthode giveEnrFuncVector ()……..75

Figure 3-12 Graphique d'appel de la méthode UpdateMyGeometry……81

Figure 3-13 Graphique d'appel de la méthode giveRadiusOfDomainIntegration.…….....84

Figure 3-14 Éléments intersectés par le cercle d'enrichissement …….…….85

Figure 3-15 Graphique d'appel de la méthode computeSIFs…….……...…88

Figure 3-16 Graphique d'appel de la méthode UpdateMyGeometry……..……..91

Figure 3-17 Diagramme d'héritage de la classe EnrichmentFunction……..…….94

Figure 3-18 Graphique d'appel de la méthode EvaluateYourSelfAt……..…...96

Figure 3-19 Graphique d'appel de la méthode EvaluateYourGradAt…….…..97

Figure 3-20 Technique de partition élémentaire…….…….102

Figure 3-21 Technique de partition des points de Gauss……..…...104

Figure 3-22 Diagramme d'héritage de la classe Load…….……105

Figure 3-23 Diagramme de collaboration et d'héritage de la classe Load……..…….106

Figure 3-24 Points de Gauss élémentaires pour le calcul numérique……..……106

## Chapitre 4 :
## Validation et discussion des résultats numériques

Figure 4-1 Géométrie d'une fissure centrale inclinée dans une plaque infinie soumise à une traction uniaxiale……..……...108

Figure 4-2: Valeurs normalisées des facteurs d'intensité de contrainte en modes I et II (calculées / Abaqus) en fonction de l'angle d'inclinaison β…….…..110

Figure 4-3 Valeurs normalisées du SIF mode I en fonction de l'angle d'inclinaison " β " pour une plaque infinie comportant une fissure centrale inclinée sous traction uniaxiale……....113

Figure 4-4 Valeurs normalisées du SIF mode II en fonction de l'angle d'inclinaison " β " pour une plaque infinie comportant une fissure centrale inclinée sous traction uniaxiale……..…114

Figure 4-5 Distribution des points de Gauss pour les éléments affectés par une fissure centrale inclinée dans une plaque infinie soumise à une traction uniaxiale…….……...114

Figure 4-6 Les déplacements selon XX et YY d'une plaque infinie carrée soumise à une traction uniaxiale munie d'une fissure centrale inclinée…….……...115

Figure 4-7 Géométrie d'une double fissure inclinée dans une plaque rectangulaire soumise à un chargement de traction uniaxiale……..116

Figure 4-8 Champ de contrainte associés à une configuration à double fissures inclinées avec un angle d'inclinaison β=α=20°…….……...117

Figure 4-9 Géométrie d'une fissure centrale inclinée dans une plaque carrée soumise à un chargement de traction biaxial……..……..…118

Figure 4-10 Normalized SIFs mode I values vs horizontal loading controlled by factor λ and angle of inclination "β" for an infinite plate with an inclined center crack under vertical tensile loading…….119

Figure 4-11 La valeur des SIFs normalisés mode II en fonction du facteur de charge horizontale λ et de l'angle d'inclinaison « β » pour une plaque infinie avec une fissure centrale inclinée sous charge répartie……...120

Figure 4-12 discrétisation d'une fissure centrale inclinée par Abaqus……..……..…121

Figure 4-13 discrétisation d'une fissure centrale inclinée dans une plaque carrée soumise à un chargement de traction biaxial par Abaqus……121

Figure 4-14 Le déplacement U2 d'une plaque carrée soumise à un chargement de traction pure par Abaqus……..122

## Chapitre 5 :
## Modélisation non-linéaire par XFEM

Figure 5-1 L'état du problème…….126

Figure 5-2 surfaces d'écoulement dans le plan déviatorique……..130

Figure 5-3 La surface d'écoulement de Von Mises dans l'espace de contrainte et dans le plan déviatorique……137

Figure 5-4 Incrément de contrainte…….140

Figure 5-5 La contrainte de retour dans le cas de Von Mises : vue dans l'espace de contrainte…….141

Figure 5-6 Géométrie d'une fissure centrale inclinée dans une plaque soumise à une traction uniaxiale…….154

Figure 5-7 Champ de contrainte Von Mises d'une plaque carrée soumise à une traction uniaxiale munie d'une fissure centrale inclinée……..……...155

# Liste des tableaux

Tableau 4-1 : Convergence des SIF normalisés pour une fissure centrale inclinée sous chargement en traction.……..……..…108

Tableau 4-2 : KI normalisé pour diverses discrétisations et tailles de domaine……..…109

Tableau 4-3 : KII normalisé pour diverses discrétisations et tailles de domaine….……...109

Tableau 4-4 : Convergence des SIF normalisés pour une fissure centrale incliné……..111

Tableau 4-5 : Convergence des facteurs d'intensité de contrainte mixte normalisées……....115

Tableau 4- 6 : Convergence des facteurs d'intensité de contrainte mixte normalisées…..…115

Tableau 4-7 : Convergence des SIFs normalisées en mode I pour une fissure centrale inclinée sous chargement biaxial……..117

Tableau 4-8 : Convergence des SIFs normalisées en mode II pour une fissure centrale inclinée sous chargement biaxial……..117

# Liste des symboles et abréviations

## Abréviations :

**MEF** : méthode des éléments finis.

**XFEM** : méthode des éléments finis étendus.

**PUFEM**: méthode de la partition de l'unité

## Symboles :

$u^h(x)$ : Champ de déplacement.

$N_I(x)$, $N_J(x)$, $N_K(x)$ : Fonctions de forme standard des éléments finis.

$I$ : L'ensemble des nœuds standards.

$J$ : l'ensemble des nœuds enrichis par la fonction de Heaviside .

$K$ : L'ensemble des nœuds enrichis par les fonctions asymptotiques de fond de la fissure.

$u_I$ : Degré de liberté classique du nœud i.

$a_J$ : Degré de liberté enrichi pour la fonction de Heaviside.

$b^a{}_K$ : Degré de liberté enrichis pour les fonctions asymptotiques.

$H(x)$ : Fonction de Heaviside modifiée

$F_a(x)$: Fonctions asymptotiques associées au champ singulier en pointe de la fissure..

$F(x)$ : Fonction de niveau

$(r,\theta)$ : Coordonnées polaires

$\Psi_n(x), \Psi_t(x)$ : Fonctions de distances.

$u^h(x)$ : Champ de déplacement.

$U^e$ : Vecteur des degrés de liberté élémentaire (globaux)

$\varepsilon(x)$ : Tenseur de déformation

$\sigma$ : Tenseur de contrainte

B(x) : Opérateur gradient discrétisé

$B^{std}(x)$ : Opérateur gradient discrétisé partie classique

$B^{enr}(x)$ : Opérateur gradient discrétisé partie enrichis.

$(\xi, \eta)$ : Coordonnées de l'espace parent

$(x, y)$: Coordonnées de l'espace physique.

$J$ : Matrice Jacobienne.

$D$ : Matrice de comportement constitutif

$K^e$ : Matrice de rigidité.

$K_{uu}$ : Matrice de rigidité standard.

$K_{uu}$ : Matrice de rigidité Heaviside.

$K_{uu}$ : Matrice de rigidité des fonctions asymptotique.

$K_{ua}$ , $K_{ub}$ ; $K_{ab}$: Termes de couplage.

$D^{el}$ : Tenseur d'élasticité.

$\varepsilon^{el}$ **:** Déformation élastique.

$\varepsilon^{p}$ **:** Déformation plastique.

$D^{ep}$ : Matrice constitutive tangente élastoplastique.

$n(\sigma)$ : Vecteur d'écoulement plastique.

$d\gamma$ : Multiplicateur plastique.

H : Paramètre d'écrouissage

# Introduction générale

## Introduction générale :

La fissuration constitue l'un des mécanismes de dégradation les plus critiques des structures de génie civil, affectant de manière significative leur durabilité, leur sécurité et leur capacité portante. La compréhension et la prédiction du comportement des structures fissurées relèvent du domaine de la mécanique linéaire de la rupture, qui fournit un cadre théorique rigoureux pour l'analyse de l'initiation et de la propagation des fissures dans les matériaux quasi-fragiles couramment utilisés en génie civil.

Dans ce contexte, les méthodes numériques, en particulier la méthode des éléments finis (**MEF**), se sont imposées comme des outils incontournables pour l'analyse des structures fissurées. Toutefois, la **MEF** classique présente des limites majeures lorsqu'il s'agit de modéliser la propagation des fissures, en raison de la nécessité d'un remaillage successif et coûteux sur le plan du calcul. Afin de dépasser ces contraintes, la méthode des éléments finis étendus (**XFEM**), introduite à la fin des années 1990, a constitué une avancée majeure en permettant une modélisation des discontinuités indépendamment du maillage.

Grâce à l'enrichissement des fonctions de forme classiques par des fonctions discontinues ou singulières, telles que la fonction Heaviside ou les fonctions asymptotiques au voisinage de fond de fissure, la **XFEM** permet de présenter avec précision les champs de déplacement et de contrainte induits par la présence de fissures. Cette approche a ouvert la voie à de nombreuses applications dans le cadre de la mécanique linéaire de la rupture, puis s'est progressivement étendus à des problématiques plus complexes, incluant les matériaux hétérogènes, les géométries tridimensionnelles, ainsi que le comportement non linéaire.

Parallèlement à ces développements méthodologiques, l'évaluation précise des facteurs d'intensité de contrainte (**SIF**), paramètres fondamentaux de la mécanique linéaire de rupture, gouvernant l'amorçage et la propagation de la fissure, demeure un enjeu central. De nombreuses techniques ont été proposées, telles que les intégrales de contour et les intégrales d'interaction, et ont été intégrées dans des cadres **XFEM**. Cependant, la complexité croissante des configurations de fissuration, notamment en mode mixte et sous chargement varié, continue de poser des défis théoriques et numériques importants.

Malgré les progrès considérables réalisés au cours des dernières décennies, plusieurs verrous scientifiques subsistent. En particulier, la modélisation précise de fissures inclinées sous des chargements de traction et de compression, ainsi que l'interprétation physique des facteurs d'intensité de contrainte en mode mixte, restent insuffisamment explorées dans la littérature, notamment dans le cadre de la programmation orientée objet pleinement extensible.

De plus, bien que des bibliothèques XFEM orientées objet aient été développées afin d'améliorer la modularité et la maintenabilité des codes numériques, leur exploitation pour l'étude systématique de l'influence des conditions de chargement sur les facteurs d'intensité de contrainte en mode mixte demeure limitée. Ces constats soulignent la nécessité de proposer des implémentations numériques avancées, capables de combiner rigueur théorique, flexibilité algorithmique et pertinence physique.

L'objectif principal de cette thèse est de contribuer au développement et à l'application de la méthode des éléments finis étendus pour l'analyse des structures fissurées en génie civil, dans un cadre théorique rigoureux et orienté ingénierie. Plus précisément, ce travail vise :

- à améliorer et à enrichir une implémentation **XFEM** basée sur une architecture de programmation orientée objet en langage **C++**, en s'appuyant sur la bibliothèque **OpenXFEM++**.
- à intégrer de nouveaux types de chargements mécaniques, conformément aux principes fondamentaux de l'ingénierie logicielle (héritage, polymorphisme, extensibilité).
- à évaluer avec précision les facteurs d'intensité de contrainte (**SIF**) en mode mixte pour les fissures inclinées dans des plaques bidimensionnelles soumises à divers scénarios de chargement.
- à analyser l'influence des paramètres géométriques et mécaniques, tels que la longueur et l'angle d'inclinaison de la fissure, sur la réponse en rupture.
- et enfin, à étendre l'analyse vers la modélisation élastoplastique des structures fissurées, afin de mieux représenter le comportement réel des matériaux en génie civil.

La thèse est structurée en cinq chapitres complémentaires :

**Le chapitre 1** est consacré aux fondements théoriques de la mécanique linéaire de rupture, avec un accent particulier sur les concepts des facteurs d'intensité de contrainte.

**Le chapitre 2** présente la modélisation des fissures par la méthode des éléments finis étendus, en détaillant la formulation mathématique et les fonctions d'enrichissement.

**Le chapitre 3** traite la programmation orientée objet appliquée à la **XFEM** et décrit en détail l'architecture et les extensions apportées à la bibliothèque **OpenXFEM++**.

**Le chapitre 4** est dédié à présentation et à la discussion des résultats numériques obtenus pour différentes configurations de fissures et de chargements.

Enfin, **le chapitre 5** étend l'approche **XFEM** à la modélisation élastoplastique des structures fissurées, ouvrant ainsi des perspectives vers des analyses plus réalistes du comportement des ouvrages de génie civil.

# Chapitre 1 : Théorie de la mécanique linéaire de rupture

## 1 Introduction

L'augmentation du nombre d'accidents dans les bâtiments, causés par des dimensions ou des conceptions incorrectes, ainsi que la présence de défauts primaires dans la structure, tels que des fissures source de l'amorçage de la défaillance, ont conduit les chercheurs à envisager une nouvelle discipline de l'ingénierie appelée mécanique de la rupture, qui permet de décrire la propagation de la fissure afin d'éviter la destruction de l'élément fissuré.

On peut considérer la mécanique de la rupture sous deux perspectives en fonction du type de sollicitation exercée sur la structure : la propagation quasi-statique qui est liée à la fatigue, tandis que la propagation dynamique peut être causée soit par un chargement dynamique, soit par une rupture brutale du matériau.

Il existe trois domaines spécialisés dans la mécanique de la rupture qui étudient les modèles de défaillance des matériaux dans diverses conditions, en mettant l'accent sur les mécanismes et les comportements propres à ces matériaux :

-La mécanique élastique linéaire de la rupture, également connue sous le nom de mécanique de la rupture fragile, se caractérise par le comportement élastique linéaire global de la structure fissurée. Le comportement non linéaire reste restreint au niveau du fond de fissure, car les contraintes à une pointe de fissure sont infinies et qu'une certaine forme d'inélasticité se produit à proximité immédiate de la pointe de fissure. Il est possible de formuler une telle problématique dans cette zone en se basant sur les hypothèses simples de la mécanique élastique linéaire.

-La mécanique de la rupture ductile traite les problèmes où le comportement est particulièrement non linéaire dans son ensemble, c'est-à-dire qu'elle concerne les matériaux qui subissent une déformation plastique significative avant la rupture. Ces matériaux ont la capacité d'absorber plus d'énergie avant de se fissurer, ce qui les rend plus résistants à la propagation des fissures.

-Alors que le domaine de la mécanique d'endommagement se focalise sur la détérioration progressive des matériaux en raison de charges répétées, de fatigue, de corrosion ou d'autres processus de transformation. Elle prend en compte la progression des microfissures et des microcavités qui aboutissent finalement à la rupture macroscopique.

L'objectif de la mécanique de la rupture est donc de pouvoir évaluer le caractère critique d'une fissure déjà existante et éventuellement prédire sa propagation.

Les conséquences statiques de la fissuration avec des ruptures fragiles (loi de comportement de Hooke) sont mises en avant dans cette thèse. S'appuyant sur des idées théoriques issues de la propagation quasi-statique.

## 2 Généralité en élasticité linéaire

On affirme que le matériau se fracture lorsque la force de traction ou l'extension maximale d'un corps dépasse un seuil spécifique.

Dans un milieu continu, une fracture peut provoquer une fragilisation plus ou moins grave.

La fissure peut se propager de manière cyclique (fatigue) ou brutale sous l'action des forces extérieures, conduisant à sa destruction complète ou à celle de la rupture.

Ainsi, nous avons la possibilité de donner diverses définitions de la fissure, selon plusieurs points de vue, en particulier :

-Du point de vue de la modélisation, il s'agit de la séparation irréversible d'une surface interne S d'un milieu continu en deux surfaces opposées $\Gamma$ + et $\Gamma$ − après avoir été soumis à des sollicitations externes. Cela entraîne une rupture visuelle de la structure géométrique.

Ces deux lèvres $\Gamma$+ et $\Gamma$− symbolisent la fissure. On définit le front de la fissure par la courbe $\Lambda$ = $\Gamma$+ ∩ $\Gamma$−. En général, on suppose que les surfaces $\Gamma$+ et $\Gamma$− sont libres.

-Sur le plan mécanique, on peut définir une fissure comme une fracture dans le champ de déplacement le long du plan S.

Effectivement, dans un milieu continu, le champ de déplacement est toujours continu, tandis qu'en cas de fissure, elle est interrompue et discontinue au passage de S.

$$[u] = u^{+} - u^{-} \qquad (1\text{-}1)$$

Le saut de déplacement permet de différencier et de définir les trois types d'ouvertures, ainsi que les trois types de sollicitation d'une fissure, illustrés dans la figure 1.

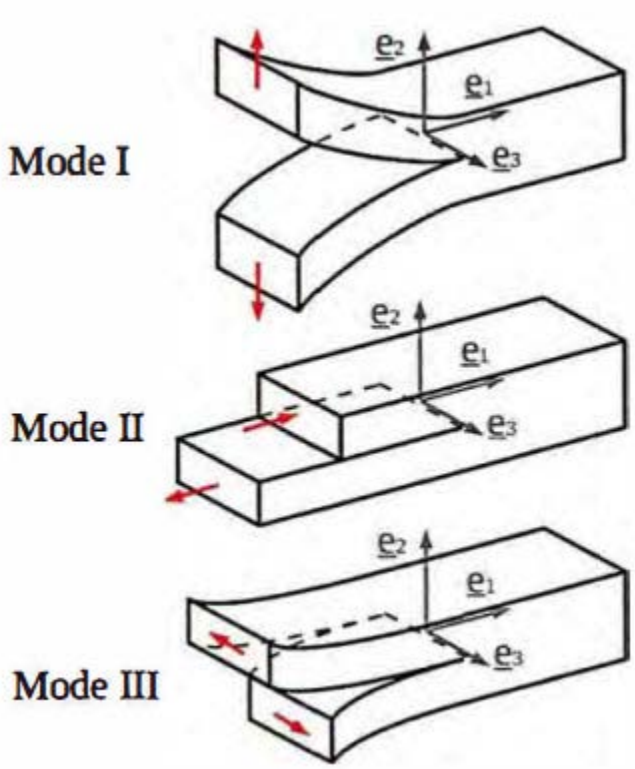



**Figure 1-1** Mode de rupture

-Le mode I : le mode d'ouverture pur où le saut de déplacement est perpendiculaire à la structure. $[\![u_1 = 0]\!], [\![u_2 \neq 0]\!], [\![u_3 = 0]\!]$

-Le mode II : le mode de cisaillement plan où le déplacement est parallèle au plan de la fissure et normal au front. $[\![u_1 \neq 0]\!], [\![u_2 = 0]\!], [\![u_3 = 0]\!]$

-Le mode III : le mode de cisaillement anti-plan où le saut de déplacement est parallèle au plan de la fissure et au front. $[\![u_1 = 0]\!], [\![u_2 = 0]\!], [\![u_3 \neq 0]\!]$

Il existe deux analyses distinctes pour représenter la présence d'une fissure.

Le premier est énergétique, ce qui entraîne la détermination du taux de restitution d'énergie. Tandis que le deuxième consiste à étudier le comportement singulier de la solution en pointe de fissure dans le cadre de l'élasticité linéaire, ce qui détermine les facteurs d'intensité de contrainte.

### 2.1 Analyse énergétique

En 1920, Griffith [1] est l'auteur des premiers travaux sur la propagation de fissures. Il a exploré le problème de la rupture, d'un point de vue énergétique, dans un milieu élastique fissuré, où il associe la diminution de l'énergie potentielle Π, d'un système mécanique à l'avancement d'une fissure en élasticité linéaire.

Pour une fracture fragile, Griffith définit le taux de restitution d'énergie (également connu sous le nom de G) comme la quantité d'énergie consommée par la structure afin de permettre l'accroissement de la surface fissurée (également appelée A).

$$G = -\frac{d\Pi}{dA} \qquad (1\text{-}2)$$

Ainsi, le comportement des fissures se manifeste par la transformation de l'énergie potentielle $\Pi$ de la structure en énergie de décohésion (définie par $\gamma$) au voisinage de la pointe de fissure.

Il s'agit donc de comparer la restitution d'énergie G calculée à sa valeur critique, où trois situations sont envisagées :

- $G < 2\gamma$ → la fissure reste stable (il n'y a pas de propagation de fissure).
- $G = 2\gamma$ → la fissure est sur le point de se propager.
- $G > 2\gamma$ → il y a une propagation instable de la fissure.

### 2.2 Asymptotique isotrope

En 1957, Irwin [2] a étendu la théorie de Griffith aux matériaux ductiles en effectuant une analyse asymptotique en pointe de fissure.

Cette analyse a permis de déduire les champs asymptotiques de contrainte et de déplacement en pointe de fissure, ce qui permet d'avoir une vision globale du processus de fissuration en étudiant le bilan énergétique associé à la propagation de la fissure.

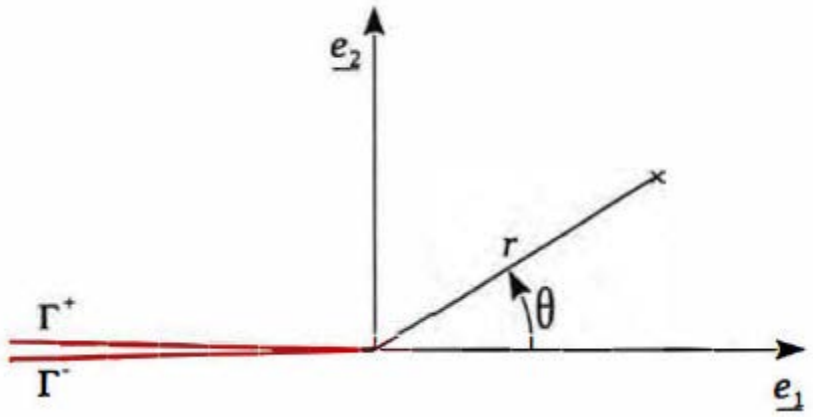



**Figure 1-2** Paramétrage de la pointe de fissure

S'appuyant sur les travaux de Westergaard [3], qui a analysé les champs des déplacements et de contraintes élastiques près de l'extrémité d'une fissure sous chargement donné , Irwin [2] a donné une définition des nouveaux grandeurs scalaires qui "mesurent" l'intensité de la singularité des contraintes en pointe de fissure pour chacun des trois modes d'ouverture de fissure , appelés **facteurs d'intensité de contrainte** (KI, KII et KIII), et qui dépendent uniquement de la géométrie et du chargement.

Leurs expressions pour chaque mode élémentaire de rupture sont données par l'équation :

$$\begin{cases} K_I(r,\theta) = \lim_{r\to 0}\sqrt{2\pi r}\ \sigma_{22}(\theta=0) = \lim_{r\to 0}\dfrac{\mu}{\kappa+1}\sqrt{\dfrac{2\pi}{r}}\ [\![u_2(\theta=\pi)]\!] \\ K_{II}(r,\theta) = \lim_{r\to 0}\sqrt{2\pi r}\ \sigma_{11}(\theta=0) = \lim_{r\to 0}\dfrac{\mu}{\kappa+1}\sqrt{\dfrac{2\pi}{r}}\ [\![u_1(\theta=\pi)]\!] \\ K_{III}(r,\theta) = \lim_{r\to 0}\sqrt{2\pi r}\ \sigma_{33}(\theta=0) = \lim_{r\to 0}\dfrac{\mu}{\kappa+1}\sqrt{\dfrac{2\pi}{r}}\ [\![u_3(\theta=\pi)]\!] \end{cases} \quad (1\text{-}3)$$

Où *r* et *ϴ* sont les coordonnées cylindriques à partir du repère centré sur le fond de fissure.

$\kappa$ : constante de Kolosov définie comme suit $\begin{cases} \dfrac{3-\nu}{1+\nu} & \text{en contraintes plane} \\ 3-4\nu & \text{en déformations plane} \end{cases}$

Lorsque l'un des facteurs d'intensité de contrainte $K_i$ atteint sa valeur critique, cela entraîne une fracture instable. La ténacité à la fracture, également connue sous le nom de $K_{ic}$, correspond à la capacité du matériau à résister à un champ de contrainte spécifique à la pointe d'une fissure.

En se basant sur l'approche de Westergaard [3], Williams [4], Irwin [2], et d'autres tels que Gdoutos [5], Tada et al [6] et Bui[7], nous intégrons ces facteurs dans l'expression analytique en pointe de fissure pour chaque mode de rupture en élasticité plane. Les déplacements à la pointe s'écrivent comme suit :

$$\begin{cases} u_1(r,\theta) = \dfrac{1}{2u}\sqrt{\dfrac{r}{2\pi}}\left(K_I\cos\dfrac{\theta}{2}(\kappa-\cos\theta) + K_{II}\sin\dfrac{\theta}{2}(\kappa+\cos\theta+2)\right) \\ u_2(r,\theta) = \dfrac{1}{2u}\sqrt{\dfrac{r}{2\pi}}\left(K_I\sin\dfrac{\theta}{2}(\kappa-\cos\theta) + K_{II}\cos\dfrac{\theta}{2}(\kappa+\cos\theta-2)\right) \\ u_3(r,\theta) = \dfrac{1}{u}\sqrt{\dfrac{r}{2\pi}}K_{III}\sin\dfrac{\theta}{2} \end{cases} \quad \textbf{(1 – 4)}$$

Dans le cas du mode mixte, il est évident d'additionner les expressions du tenseur des contraintes des trois modes :

$$\sigma_{ij}(r,\theta)=\frac{1}{\sqrt{2\pi r}}\left(K_I f_{ij}^{I}(\theta)+K_{II} f_{ij}^{II}(\theta)+K_{III} f_{ij}^{III}(\theta)\right) \qquad \textbf{(1- 5)}$$

## 3 Facteur d'intensité de contrainte (SIF)

Le facteur d'intensité de contrainte (**Stress Intensity Factor - SIF**) qui a été introduit par Irwin [2] en 1957, est un coefficient scalaire très important en mécanique de la rupture. Et qui dépend de la configuration géométrique et des conditions de chargement appliquées au corps.

La détermination de ces facteurs décrivent l'évolution de la résistance du matériau à la propagation des fissures, en caractérisant la résistance des singularités de contrainte près de la pointe de la fissure, C'est-à-dire qu'il n'existe qu'une seule solution asymptotique à énergie finie et que le champ de contrainte est singulier en $1/\sqrt{r}$ de la pointe de fissure.

Plusieurs techniques ont été employées pour évaluer les facteurs d'intensité de contrainte. On peut les classer de la manière suivante :

1. Théorique (méthode de Westergaard semi-inverse et méthode des potentiels complexes).
2. Numérique, on peut utiliser différentes fonctions telles que Green, pondération, ainsi collocation des limites, méthode alternée, transformées intégrales, dislocations continues et méthodes d'éléments finis.
3. Expérimentales (photoélasticité, moiré, holographie, caustiques et combinaisons de ces méthodes).

En général, les approches théoriques se restreignent aux plaques d'étendue infinie avec des configurations géométriques basiques de fissures et de conditions aux limites.

Dans des cas plus complexes, il est nécessaire de faire appel à des techniques numériques ou expérimentales.

Dans cette étude, parmi les méthodes numériques employées pour calculer les paramètres de fracture tels que les facteurs d'intensité de contrainte en mode mixte, on peut observer la méthode intégrale de domaine (the domain integral method) en collaboration avec la méthode des intégrales d'énergie d'interaction (interaction energy integrals method) .

### 3.1 L'intégrale J

La séparation des lèvres fissurées n'est plus la seule raison qui conduit à une dissipation d'énergie, comme c'était le cas dans le cas élastique linéaire, mais une partie de cette énergie est en réalité utilisée dans le processus de plasticité.

Il ne faut donc pas considérer le matériau comme élasto-élastique, mais plutôt comme élastique non linéaire, comme le souligne Cherepanov [8, 9] et Rice qui ont fondé [10, 11] le concept

d'intégrale ***J***, qui permet d'évaluer le taux de récupération d'énergie **G** des matériaux non linéaire, selon l'équation (1-1) on peut écrire :

$$J = G = -\frac{d\Pi}{dA} = 2\gamma \qquad \textbf{(1-6)}$$

La détermination de cette intégrale, qui entoure la pointe de fracture (Figure 2), est relativement facile expérimentalement, elle permet de déterminer l'énergie par unité de surface de la fracture en utilisant les paramètres des éléments.

Le caractère important de l'intégrale J réside dans son indépendance du contour Γ tant qu'il n'y a pas de décharge possible, et dans sa capacité à être liée au déplacement d'ouverture de la pointe de la fissure.

En raison de ces propriétés caractéristiques, ***J*** a été proposé comme candidat intéressant pour un critère de rupture [2, 7]. Sous chargement en mode d'ouverture, le critère d'initiation de fissure prend la forme $J = Jc$, où $Jc$ est une propriété du matériau pour une épaisseur donnée dans les conditions spécifiées [12].

Dans des conditions des déformations planes, la valeur critique de $J$, $J_{Ic}$ est liée à la ténacité à la rupture par déformation plane $K_{Ic}$

$$J_{Ic} = \frac{1-\upsilon^2}{E} K_{Ic}^2 \qquad \textbf{(1- 7)}$$

Le vecteur intégral ***J*** est défini par l'intégration le long d'un chemin dans un plan bidimensionnel :

$$J = \int_{\Gamma} \left( W n_1 - \sigma_{jk} n_j \frac{\partial u_k}{\partial x_1} \right) d\Gamma \qquad \textbf{(1 – 8)}$$

Où :

« $\Gamma$ » est une ligne de contour fermée entourant la pointe de la fissure du bas vers le haut de la fissure dans le sens inverse des aiguilles d'une montre, dans laquelle « n » est son unité normale sortant de $\Gamma$.

« $W$ » est la densité d'énergie de déformation ( $W = \int_0^{\varepsilon_{ij}} \sigma_{ij} d\varepsilon_{ij}$ )

«$\sigma_{jk}.n_j$» est le vecteur de traction perpendiculaire à Γ dans la direction extérieure,

« $u_k$ » est le vecteur de déplacement et « $d\,\Gamma$ » est un élément différentiel de longueur d'arc le long du contour fermé $\Gamma$.

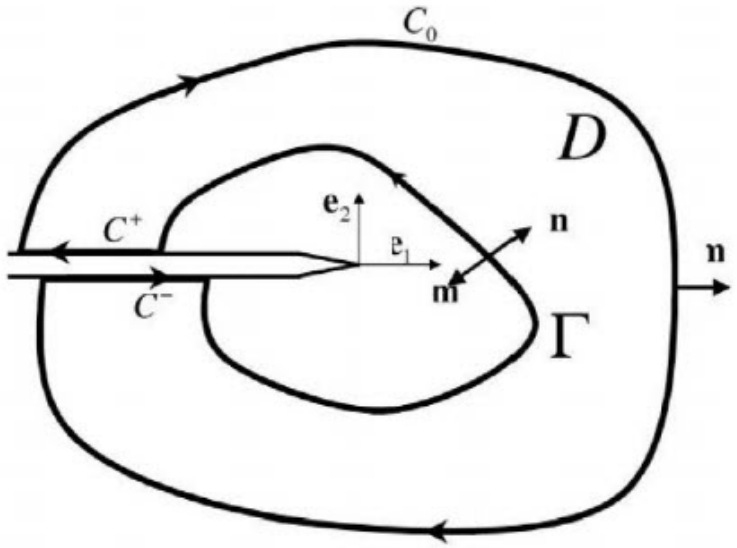



**Figure 1-3** Schéma de l'intégral J au fond de fissure

L'intégrale ***J*** obtenue à partir de l'équation (1-8), est calculée en suivant un chemin "$\Gamma$" dans le sens inverse des aiguilles d'une montre, à partir d'un point donné sur la partie plane de la surface inférieure de la fissure et se terminant à un point donné sur la partie plane de la surface supérieure la fissure qui est la région R délimitée par la surface fermée.

L'absence de singularité dans le contour AB $\Gamma_1$ CD $\Gamma_2$ implique que l'intégrale J est nulle sur ce contour.

Alors :

$$J_{AB\,\Gamma 1\,CD\,\Gamma 2} = J_{B\,\Gamma 1\,C} + J_{CD} + J_{D\,\Gamma 2 A} + J_{AB} = 0 \qquad \textbf{(1 – 9)}$$

Lorsque les lèvres de fissure AB et CD qui sont en parallèles à l'axe z sont exemptent de traction (libre de chargement) $n_1 = 0$ et $\sigma_{jk}.n_j = 0$, cela signifie que

$J_{CD} = J_{AB} = 0$

En remplaçant (9) dans l'équation (1-8), ce qui donne :

$J_{B\,\Gamma 1\,C} + J_{D\,\Gamma 2 A} = 0$

Ou : $J_{B\,\Gamma 1\,C} = J_{A\,\Gamma 2 D}$

Pour les problèmes de mécanique de la rupture élastique linéaire en 2D, l'intégrale de contour « J » est égale au taux de restitution d'énergie « G ».

La relation entre l'intégrale de contour « ***J*** » et les facteurs d'intensité de contrainte mode I et II (KI et KII) pour les problèmes mixtes en 2D peut être exprimée comme ci-dessous [13]:

$$J = G = \frac{1}{E^*}(K_I^2 + K_{II}^2) \qquad \textbf{(1 – 10)}$$

Ou $E^*$ est définit comme

$$E^* = \begin{cases} \dfrac{E}{1-\upsilon^2} & \text{pour l'état de déformation plane} \\ E & \text{pour l'état de contrainte plane} \end{cases}$$

### 3.2 Intégrale interaction

Cette technique, introduit par Sih et Asaro [14] exploite l'intégrale du contour « ***J*** » et la solution asymptotique en pointe de la fissure

Deux états sont pris en compte dans le fond de fissure, l'état réel représenté (1) et l'état auxiliaire représenté (2), qui correspond à la solution asymptotique pour les modes I et II.

La somme des deux états (1+2) est représentée par l'intégrale J, ce qui donne :

La somme des deux états d'un corps fissuré (1 et 2) avec des caractéristiques $(\sigma_{ij}^{(1,2)}, \varepsilon_{ij}^{(1,2)}, u_{ij}^{(1,2)})$ , est représentée par l'intégrale J, ce qui donne :

$$J^{(1,2)} = \int_{\Gamma}\left(\frac{1}{2}(\sigma_{ij}^{(1)} + \sigma_{ij}^{(2)})((\varepsilon_{ij}^{(1)} + \varepsilon_{ij}^{(2)})\delta_{1j} - (\sigma_{ij}^{(1)} + \sigma_{ij}^{(2)})\frac{\partial(u_i^{(1)} + u_i^{(2)})}{\partial x_1}\right)n_j d\Gamma \qquad \textbf{(1-11)}$$

Après avoir développé et réorganisé les termes

$$J^{(1,2)} = J^{(1)} + J^{(2)} + I^{(1,2)} \qquad \textbf{(1-12)}$$

Ou $I^{(1,2)}$ est appelée intégrale d'interaction pour les états 1 et 2, qui est égale à :

$$I^{(1,2)} = \int_{\Gamma}\left[W^{(1,2)}\delta_{1j} - \sigma_{ij}^{(1)}\frac{\partial u_i^{(2)}}{\partial x_1} - \sigma_{ij}^{(2)}\frac{\partial u_i^{(1)}}{\partial x_1}\right]n_j d\Gamma \qquad \textbf{(1- 13)}$$

Ou $W^{(1,2)}$ est l'interaction d'énergie de déformation et égale à :

$$W^{(1,2)} = \sigma_{ij}^{(1)}\varepsilon_{ij}^{(2)} = \sigma_{ij}^{(2)}\varepsilon_{ij}^{(1)} \qquad \textbf{(1-14)}$$

L'intégrale J équation (1- 12) pour la somme des états 1 et 2 en déformation plane, donne :

$$J^{(1,2)} = J^{(1)} + J^{(2)} + \frac{2}{E^*}(K_I^{(1)}K_I^{(2)} + K_{II}^{(1)}K_{II}^{(2)}) \qquad \textbf{(1-15)}$$

Avec $$I^{(1,2)} = \frac{2}{E^*}(K_I^{(1)}K_I^{(2)} + K_{II}^{(1)}K_{II}^{(2)}) \qquad \textbf{(1-16)}$$

Donc, le choix de l'état auxiliaire (2) comme champs asymptotiques s'explique pour les deux modes comme suit :

-Lorsque l'état 2 est choisi comme champs asymptotiques du mode I pur avec KI $^{(2)}$ = 1, KII $^{(2)}$ = 0, cela donne le facteur d'intensité de contrainte mode I, en termes d'intégrale d'interaction.

$$K_I^{(1)} = \frac{E^*}{2} I^{(1,Mode\ I)} \qquad \textbf{(1-17)}$$

-Lorsque l'état 2 est choisi comme champs asymptotiques du mode II pur avec KII $^{(2)}$ = 1, KI $^{(2)}$ = 0, cela donne le facteur d'intensité de contrainte mode II en termes d'intégrale d'interaction.

$$K_{II}^{(1)} = \frac{E^*}{2} I^{(1,Mode\ II)} \qquad \textbf{(1-18)}$$

La réécriture de l'intégrale de contour (1-15), $I^{(1,2)}$, est réalisée en utilisant une intégrale de domaine plus adaptée aux calculs éléments finis.

Et comme l'intégrale de contour [9] n'est pas bien adaptée aux calculs éléments finis, il est donc utile de le reformuler sous une forme de domaine équivalente en multipliant l'intégrante par une fonction de pondération suffisamment lisse q(x) qui prend une valeur d'unité sur un ensemble ouvert contenant la pointe de fissure et s'annule sur un contour extérieur prescrit $\Gamma_0$.

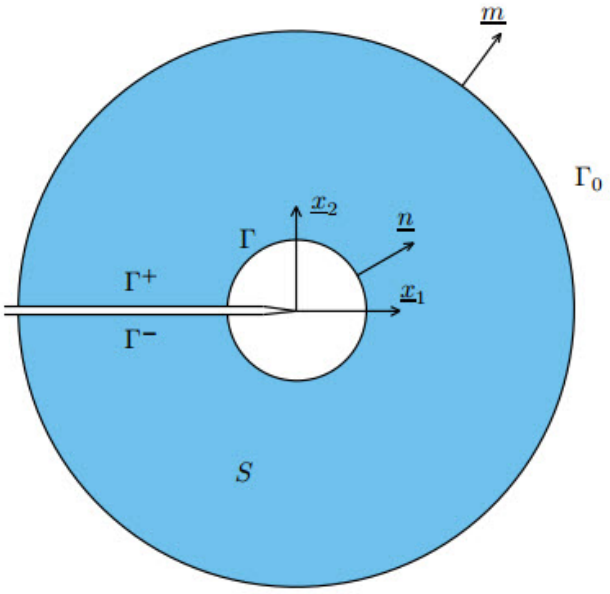



**Figure 1-4** Schéma de l'intégrale d'interaction au fond de fissure

L'intégrale d'interaction pour les états 1 et 2 peut s'écrire (voir Figure 4) :

$$I^{(1,2)} = \lim_{\Gamma \to 0} \left\{ \int_{\Gamma \cup \Gamma_0 \cup \Gamma^+ \cup \Gamma^-} \left[ W^{(1,2)} \delta_{1j} - \sigma_{ij}^{(1)} \frac{\partial u_i^{(2)}}{\partial x_1} - \sigma_{ij}^{(2)} \frac{\partial u_i^{(1)}}{\partial x_1} \right] q m_j ds \right\} \qquad \textbf{(1-19)}$$

Dans la figure 1.4, soit le contour $C$ l'union des courbes $\Gamma$, $\Gamma_0$, $\Gamma^+$ et $\Gamma^-$ et soit $m_j$ la normale extérieure au domaine $A$.

Remarque : la frontière du domaine A est $\partial A = S = \Gamma \cup \Gamma_0 \cup \Gamma^+ \cup \Gamma^-$. La normale extérieure $m_j$ au domaine A est :

$$m = \begin{cases} -n & sur\ \Gamma \\ +n & sur\ \Gamma_0, \Gamma^+ \ et\ \Gamma^- \end{cases} \qquad \textbf{(1-20)}$$

En utilisant le résultat de (18), et le théorème de flux-divergence pour le contour $dA$, l'intégrale d'interaction peut être simplifiée et l'équation (1- 17) devient

$$I^{(1,2)} = \int_{\Gamma} \left[ -W^{(1,2)} \delta_{1j} + \sigma_{ij}^{(1)} \frac{\partial u_i^{(2)}}{\partial x_1} + \sigma_{ij}^{(2)} \frac{\partial u_i^{(1)}}{\partial x_1} \right] \frac{\partial q}{\partial x_j} dA \qquad \textbf{(1-21)}$$

Lorsqu'on calcule cette intégrale, le domaine S est composé de tous les éléments finis situés dans un rayon rd autour de la pointe de fissure.

Dans les faits, $r_d$ est sélectionné comme le double de la racine carrée de l'aire de l'élément touché par la pointe de fissure.

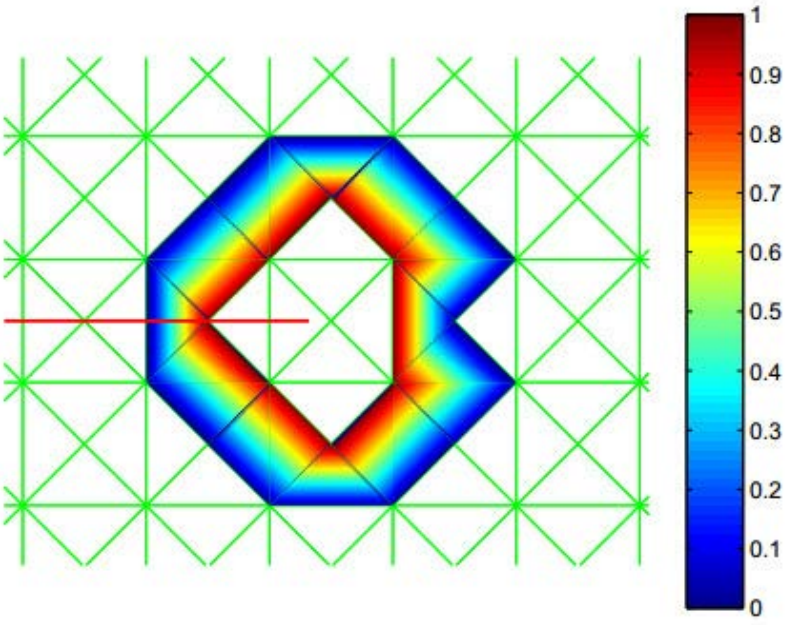



**Figure 1-5** Fonctions de pondération « q » sur les éléments

La figure 1.5 représente un domaine S qui comprend des éléments finis et la fonction q qui lui est associée. Cette fonction vaut 1 pour les nœuds situés à l'intérieur de S et 0 pour les nœuds situés à l'extérieur de S.

L'interpolation de la géométrie implique l'interpolation de cette fonction q sur les fonctions de forme éléments finis.

Il est important de noter que cette définition de q implique que seuls les éléments du bord de S, pour lesquels le gradient de q n'est pas nul, contribuent au calcul de l'intégrale d'interaction $I^{(1,2)}$

Concernant les techniques de calcul des facteurs d'intensité de contrainte par des intégrales de contour ou de domaine, on pourra aussi se référer aux travaux de Stern [15, 16], M. Lorentzon,[17], Sih GC [18] et Murakami Y [19].

## 4 Etat de l'art des méthodes de simulation des fissures:

En utilisant les concepts de la mécanique de la rupture élastique linéaire (LEFM), l'analyse de la durée de vie en fatigue présente certaines limites significatives, comme la nécessité d'un défaut initial dans la ligne de liaison, l'ignorance de la durée de vie d'initiation de la fatigue et

l'impossibilité de prendre en considération l'effet du comportement linéaire et non linéaire des matériaux.

Par conséquent, de nombreuses méthodes pour simuler l'ouverture et la propagation des fissures ont été mises au point au cours des dernières années : telles que les méthodes analytiques, semi-analytiques et numériques.

On utilise principalement les méthodes numériques pour réaliser des analyses successives, qui permettent de résoudre des équations aux dérivées partielles représentant le comportement d'un solide fracturé prenant en compte les charges appliquées et les déplacements imposés, afin d'obtenir une estimation des déplacements, déformations et contraintes en tout point de ce solide.

Certains grandeurs caractéristiques telles que les trois facteurs d'intensité de contrainte (K) dans la mécanique élastique linéaire de la rupture sont utilisées pour ramener l'intensité et la variation de ces champs à proximité du fond de fissure, dans le modèle de fissuration choisi, afin d'obtenir la direction et la longueur de propagation de la fissure.

Il existe plusieurs méthodes de simulation qui ont été élaborées afin de mettre en place des modèles d'analyse des dommages dans un logiciel d'analyse numérique.

Dans cette partie, nous aborderons brièvement les trois catégories des méthodes d'analyse numérique les plus fréquemment employées pour simuler la propagation de fissure

## 4.1 Méthodes classiques basées sur les éléments finis:

### 4.1.1 La méthode des éléments finis (FEM):

La méthode classique des éléments finis (**FEM**) est une méthode numérique largement utilisée au cours de la dernière décennie, elle a montré son efficacité et sa puissance pour résoudre de nombreux problèmes de mécanique des solides ainsi que des problèmes d'ingénierie complexe.

Effectivement, pour l'utilisation de cette méthode dans la mécanique élastique linéaire de la rupture, il est nécessite de subdiviser le domaine en sous-domaines (connus sous le nom de maillage) dont les frontières internes, qui varient d'une analyse à l'autre, doivent correspondre aux frontières du domaine d'origine, cela implique de répéter ce processus de maillage à chaque étape, au moins dans la zone entourant la formation des nouvelles frontières, ce qui la rend chronophage. Elle est donc moins adaptée pour réaliser les analyses successives du problème de fissuration.

Ray Clough [20] est généralement considéré comme l'un des fondateurs du MEF car il a défini la méthode et introduit le concept de « méthode des éléments finis » dans un article de 1960, par la suite, grâce à des chercheurs tels que J. H. Argyris [21], et O. C. Zienkiewicz [22], des progrès majeurs ont été réalisés dans la théorie mathématique de la FEM, établissant les fondements théoriques rigoureux nécessaires pour appliquer la FEM à une variété de problèmes physiques.

Différents modèles et techniques ont été appliqués dans les codes de calculs en utilisant le **MEF**, Nous les mentionnons à titre d'illustration :

### 4.1.2 Le relâchement des nœuds :

la méthode de relâchement de nœuds (**Node Relaxation Method**) est une méthode numérique basée sur une technique relativement simple et très répandue consiste à bloquer via des conditions aux limites en déplacement (conditions de Dirichlet), des nœuds du maillage situés sur le trajet de fissuration, et ensuite, à relâcher des nœuds correspondant lorsqu'un critère de rupture est satisfait, en supprimant brutalement une ou plusieurs relations entre deux pas de propagation pour déduire une nouvelle position de la fissure .

Ce relâchement instantané est considéré comme l'un des inconvénients de la méthode, car il risque de produire des chocs numériques notamment en dynamique [23] Ce qui ouvre la voie à l'émergence d'un phénomène des oscillations non physique de la surface libre de la fissure ainsi qu'en pointe.

Ce dernier est traité par le relâchement progressif d'un nœud par l'application d'une force nodale [24] qui se diminue linéairement de manière à maintenir le facteur d'intensité de contrainte K constant durant le pas de propagation.

Cette méthode permet de simuler des ruptures en 2D, mais très difficile à mettre en œuvre en 3D, elle a été initialement utilisée par Kobayashi [25, 26] pour analyser des essais de rupture fragile, De nombreuses littératures scientifiques ont également été publiées, notamment : Vaezzadeh et al [27] ont donné une analyse dynamique tridimensionnelle à grands déplacements des structures de câbles lâches (**Slack Cables**) soumises à des charges dynamiques en utilisant la méthode relâchement de nœuds. Fang et al [28] ont proposé une nouvelle méthode appelée méthode de relaxation dynamique des nœuds qui a été mise en avant pour simuler le processus de soudage multicouche et multipasse (the multilayer and multipass welding process) afin de réduire la distorsion du maillage,

Cette méthode de calcul présente certains inconvénients, notamment la nécessité de connaitre préalablement le trajet de fissure, la nécessité d'imposer à la pointe de fissure de se situé sur un nœud ainsi qu'une difficulté pour simuler une fissure non rectiligne.

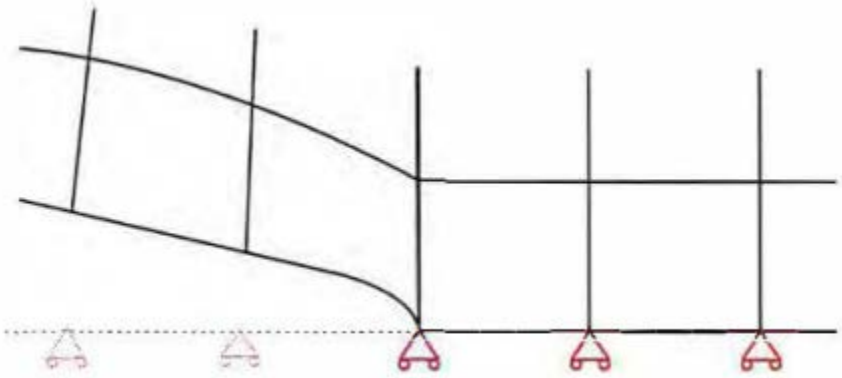


**Figure 1-6** Méthode de relâchement des nœuds

Afin d'éviter ce problème, une autre approche a été proposée : la méthode de séparation des nœuds. Lorsqu'un nœud satisfait au critère de rupture, il est dédoublé. Le relâchement local des contraintes en ce nœud permet d'ouvrir une fissure à cet endroit. [29].

### 4.1.3 Le modèle de zone cohésive (Cohesive Zone Model – CZM) :

peut être considérée comme la version la plus sophistiquée de la méthode de relâchement de nœuds, elle été proposé par Dugdale [30] en 1960 pour étudier les matériaux ductiles, et Barenblatt en 1962 [31].

Dans cette méthode, au lieu d'appliquer une force aux nœuds que l'on souhaite relâcher et de les faire décroître de façons arbitraire, elle modélise la zone proche de la pointe de la fissure, avec une zone dans laquelle les forces de cohésion agissent pour résister à la séparation, c.-à-d. faire relier en pointe de fissure, l'ouverture (donc le saut de déplacement) à la contrainte normale aux lèvres [32, 33].

La zone cohésive est souvent définie le long des surfaces potentiellement fissurées et subit une loi de traction-séparation qui traduit la dégradation progressive des éléments situées dans cette zone.

Cette technique présente des avantages comme la possibilité d'éviter les singularités tout en étant faciles à utiliser dans divers programmes numériques tels qu'**ANSYS**, **Abaqus**, **COMSOL**…..etc. Un inconvénient majeur a également été constaté, à savoir que le chemin d'écrasement doit être prédéfini.

La **CZM** a été utilisée dans de nombreux travaux scientifiques, notamment par Y. Kang [34] qui a présenté une analyse de fracture quasi-statique d'une poutre en béton et une analyse de fracture dynamique du problème de Kalthoff par la combinaison entre la méthode des particules finies (**FPM**) et le modèle de zone cohésive (**CZM**) .

Un nouveau modèle de zone cohésive dépendant du temps (**CZM**) est introduit par H. Cui [35] avec N modèles Maxwell (**N Maxwell models**) qui sont connectés en parallèle et laissent la traction d'entrée indépendante du temps se transformer en une traction dépendante du temps Un test d'étirement à vitesse constante, un test de chargement cyclique et un test de relaxation ont été appliqués dans un simple élément cohésif 2D.

J.A. Nairn et Y. E. Aimene [36] ont démontées une nouvelle méthode de mise en œuvre des zones cohésives appelée modélisation de la zone cohésive de résistance, qui peut modéliser de manière réaliste la pré-fissuration alors que les méthodes antérieures génèrent des prédictions irréalistes.

Celle-ci montre que la mixité des modes, qui change pendant la croissance des fissures, dépend des propriétés de la zone cohésive et de l'extrémité de la zone de processus cohésif qui est interprétée comme la pointe de la fissure. M. Linke et R. Lammering [37] ont présenté une étude paramétrique qui a permis d'élargir la littérature existante en adaptant la solution numérique à la solution analytique grâce à l'étalonnage de la résistance à la cohésion en utilisant des critères plus appropriés et plus généralement utilisés.

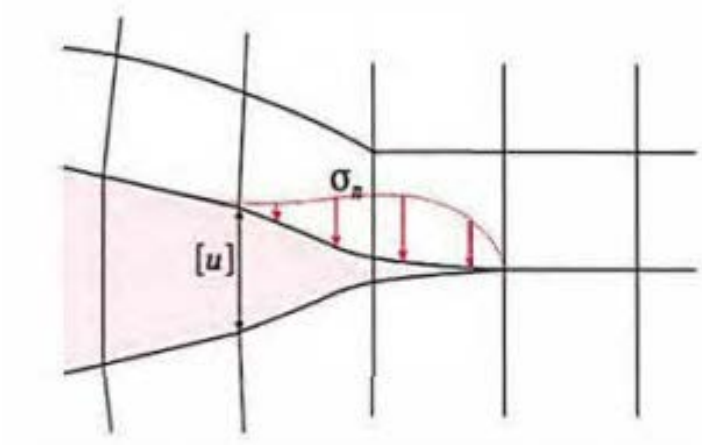



**Figure 1-6** Méthode des éléments cohésifs

### 4.1.4 La technique d'extension virtuelle de fissure (Virtual Crack Extension Technique-VCET) :

C'est une technique numérique utilisée pour déterminer les trois facteurs d'intensité de contrainte et simuler la propagation des fissures dans les matériaux avec l'introduction d'une extension virtuelle de fissure en termes de rigidité et déplacement

Cette technique consiste à calculer le changement d'énergie potentielle totale du système afin d'évaluer le taux de restitution d'énergie **G**, en tenant compte du fait que la fissure se propagera lorsque G atteint une valeur critique.

Cette technique a été suggérée par Park [38] et Helen [39] en 1974, puis a été perfectionnée par De Lorenzi en 1982[40, 41].

La **VCET** présente l'avantage de pouvoir évaluer de manière précise la valeur de G. Ses inconvénients comprennent : l'obligation de choisir une pré-fissure, avec un maillage fin à proximité du fond de fissure [42].

Étant donné l'obligation de remaillage, une valeur de G nouvelle est calculée après chaque étape de croissance de la fissure.

plusieurs analyses 2D et 3D sont effectuées en utilisant cette technique comme D. R. Tamayo [43] qui a développé et démontré cette technique pour les problèmes de fracture thermoélastique évitant l'utilisation d'intégrales de conservation d'énergie telles que l'intégrale « $J$ » .

Montoya A. [44] a fait étendre la **VCET** aux matériaux non linéaires subissant une déformation plastique à l'aide de modèles de matériaux basés sur la déformation et la théorie incrémentale de la plasticité et en utilisant un cycle de chargement/déchargement.

H. Millwater [45] a proposé une nouvelle version de méthode d'extension de fissure virtuelle pour calculer le taux de libération d'énergie, ce qui conserve la simplicité conceptuelle de la différenciation numérique et élimine les problèmes numériques concernant la perturbation de la taille de la fissure

### 4.1.5 La technique de fermeture virtuelle de fissure (Virtual Crack Closure Technique-VCCT) :

Le concept de fermeture virtuelle des fissures (**VCCT**) (appelé aussi modified crack closure method) a été introduit par Rybicki et Kanninen [46] en 1977 afin de représenter les fissures en 2D, puis a été étendu pour inclure les fissures en 3D grâce aux travaux Shivakumar et al [47].

En se basant sur l'hypothèse d'Irwin [2] selon laquelle l'énergie requise pour fermer virtuellement une fissure est équivalente à l'énergie requise pour la propager sur une longueur est $\Delta l$ .

la **VCCT** vise à évaluer le taux de restitution d'énergie pour chaque mode de rupture en tenant compte des forces et des déplacements nodaux des éléments autour du front de fissure.

Donc, **VCCT** est une méthode d'énergie globale similaire à la **VCET**, où le taux de restitution d'énergie G peut être déterminé de manière facile, à condition que la fissure soit préalablement définie et que le remaillage soit obligatoire après chaque étape de croissance de la fissure.

La **VCCT** a été utilisée dans de nombreux travaux scientifiques, notamment par Y.P. Liu et T.J. Li [48] qui ont calculé le facteur d'intensité de contrainte (**SIF**) à l'aide d'éléments finis tétraédriques. E. Farkash et al [49] ont proposé une nouvelle approche est appelée Multi-VCCT en modélisant des fissures d'interface bidimensionnelles puis étendue aux problèmes de fissures d'interface tridimensionnelles comprennent deux matériaux homogènes et isotropes différents, ainsi que deux matériaux transversalement isotropes.

De plus, L. Zhou et al [50] ont exposé une combinaison entre la méthode des éléments finis enrichis (**EFEM**) et la technologie de fermeture virtuelle de fissures (**VCCT**) en introduisant la fonction de couverture d'interpolation, afin de résoudre les problèmes d'initiation et de croissance des fissures dans divers matériaux et conditions de chargement.

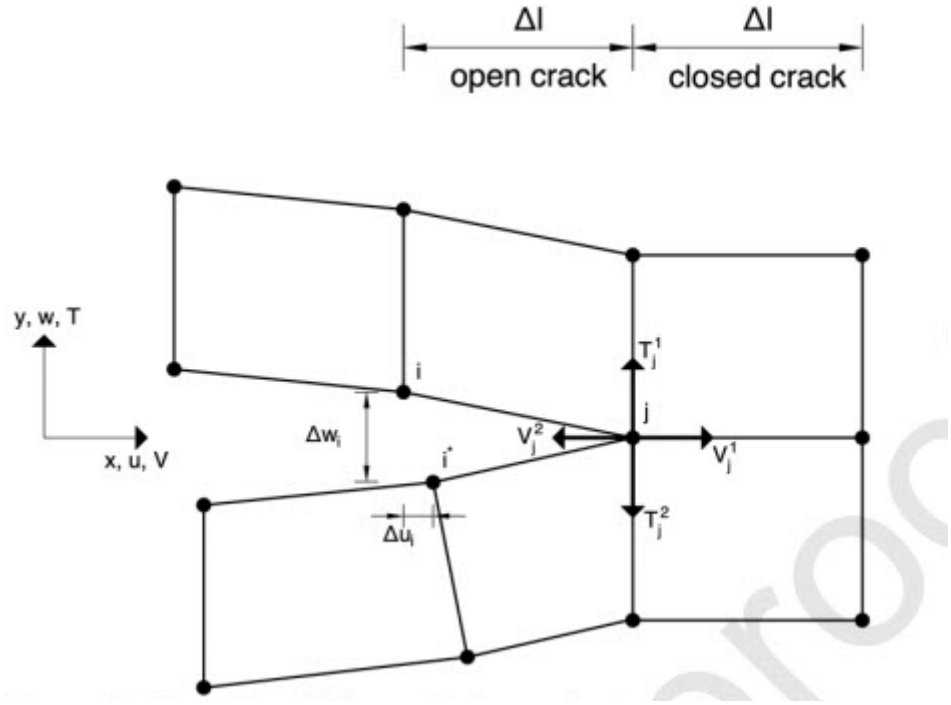



**Figure 1-7** Techniques de fermeture virtuelle de fissure pour élément bidimensionnel

### 4.1.6 Les méthodes de remaillage autour de la fissure :

Ces méthodes ont été décrites pour la première fois par Swenson et al [51] en 1988. Elles sont des techniques cruciales couramment utilisées en mécanique de fracture pour simuler précisément la propagation des fissures dans les matériaux pour mieux capturer les gradient de contrainte et de déformation Parmi eux le remaillage adaptif, le remaillage incrémental à chaque étape d'incrémentation, le raffinement adaptif du maillage ….etc.

Parmi les inconvénients, on retrouve l'utilisation d'outils de génération automatique de maillages, qui sont souvent lourds, afin de faire apparaître un remaillage successif suffisamment petit autour du fond de fissure tout au long du processus de propagation. C'est le cas, par exemple, quand on tente de représenter des géométries complexes et quand le trajet de fissuration est tridimensionnel.

### 4.1.7 L'élimination d'élément (Element Deletion Method) :

L'élimination d'élément est une technique peu couteuse et largement utilisée et implémentée dans de nombreux code commercial de simulation par élément finis telle-que le code américain LS-DYNA.

Le concept de cette technique est basé sur un calcul répétitif après chaque suppression d'un ou plusieurs éléments dans lesquels une grandeur mécanique dépasse un critère définis par exemple la contrainte calculée dépasse la ténacité d'un élément [52].

Ces éléments supprimées qui forment un vide, représentent la fissure, ce qui peut introduire une discontinuité artificielle dans le maillage ainsi l'enregistrement d'une diminution de la masse globale ce qui implique une imprécision [29]

## 4.2 Méthodes non basées sur les éléments finis :

### 4.2.1 Méthodes des éléments frontières (Boundary Element Method) :

Contrairement à ce qui est conventionnel en **FEM**, qui nécessite un maillage du domaine étudié en éléments discrets, la méthode des éléments de frontière appelée (boundary element method - **BEM**) exprime le problème de représentation des fissures de façon intégrale, ou elle nécessite uniquement le maillage de la frontière du domaine concerné.

Cette méthode, et conformément aux certaines conditions aux limites, les équations aux dérivées partielles sont converties en équations intégrales sur la frontière du domain discrétisé via une certaine solution de base de ces équations aux dérivées partielles dite fonctions de Green) [53],

dans lesquelles seul le bord (et les lèvres de la fissure) sont maillés et discrétisés, ce qui réduit considérablement la charge du travail de remaillage lors de sa propagation. [54–56]. Celle-ci est donc considérée comme le principal avantage de la méthode, sans compter qu'elle permet d'avoir une bonne précision à la détermination du facteur d'intensité de contrainte (**SIF**) et donne aussi de bons résultats en analyse vibratoire [57].

Plusieurs limitations ont également enregistré pour cette méthode numérique dont les problèmes de conditionnement de matrice, le stockage des données pour chaque intégration et la nécessité de mailler les zones plastiques de manière conventionnelle pour le traitement du cas des problèmes élasto-plastiques, ainsi que certains problèmes d'analyse transitoire du fait d'un décalage de la solution fondamentales en espace et en temps qui doit être effectuée à chaque pas de temps.[54, 58]

Parmi les améliorations apportées à cette méthode, on retrouve la méthode des éléments finis à limites d'échelle appelé (scaled boundary finite element method -**SBFEM**) proposée par Wolf et Song [59], Qui combine les avantages du **BEM** et du **FEM**, ce qui est devenu possible de réduire la zone discrète, comme dans le cas de la méthode **BEM**, et d'éviter de calculer la solution fondamentale, comme dans le cas de la méthode **FEM**. [60, 61].

### 4.2.2 Méthodes sans maillage (Méthodes particulaire) :

Depuis les débuts des éléments finis, le problème de la distorsion du maillage est apparu, ce qui entraîne un manque important de précision dans les simulations. Par conséquent, des méthodes sans maillage (Meshless methods) ont été créées spécifiquement pour tenter de contrecarrer ce manque de précision.

La méthode **EFG** (Element Free Galerkin) est la plus utilisée parmi les nombreuses variantes de méthodes dites sans maillage ( Meshless methods ) qui sont initialement proposées aux problèmes de mécanique de la rupture par T. Belytschko [62–64] pour résoudre les problèmes de la fissuration quasi-statique, puis elles sont évoluées pour inclure les problèmes de fissuration dynamiques [65–67] , les fissurations en 3D [63, 68] et ensuite les fissurations du béton [65]

la résolution de la forme faible des équations aux dérivées partielles à l'aide de la méthode Galerkin, rend la méthode **EFG** plus proche de la méthode des éléments finis (**FEM**) que la méthode des éléments de frontière (boundary element method -**BEM**), de ce fait, un couplage avec la **MEF** a également été proposé pour diminuer la zone sans maillage à une zone proche de la fissure [69].

Cette méthode facilite la modélisation de la croissance arbitraire des fissures car elle ne nécessite pas de remaillage du domaine pour l'approximation du champ de déplacement ainsi que la propagation des fissures est modélisée en étendant les surfaces des fissures.

La méthode **EFG** ne raisonne plus sur un découpage de la géométrie en éléments, mais sur une représentation du volume par une densité de nœuds ayant des domaines d'influence, remplaçant ainsi la notion d'éléments des méthodes **FEM**.

Ces derniers sont munis de fonctions poids non nulles sur une zone d'influence qui peut contenir d'autres nœuds, alors que la propagation d'une fissure dans cet ensemble de nœuds s'effectue par la diminution du poids de certains nœuds dans l'approximation du déplacement en un certain point si la ligne joignant ces nœuds et ce point est coupée par la fissure. [67-68].

Cette méthode est très efficace lorsque l'on traite de très grandes déformations, et donne des résultats intéressants en mécanique linéaire et non-linéaire de la rupture [70, 71], sauf qu'elle possède un certain nombre d'inconvénients, tel que la lourdeur des calculs mis en œuvre (calcul des voisins), la taille du domaine d'influence et surtout la difficulté d'imposer des conditions aux limites cinématiques.

### 4.3 Méthodes basées sur la partition de l'unité :

#### 4.3.1 La partition de l'unité :

L'idée d'enrichir l'approximation de la méthode des éléments finis par des fonctions A priori proche de la solution prévue pour mieux décrire le champ de déplacement d'un problème considéré, représente la base du concept de partition de l'unité (Partition of Unity Method-**PUFEM**), qui a été introduit par Babuška et Melenk [72, 73].

Cette idée qui devient pertinente dans le contexte de la mécanique de la rupture, s'applique particulièrement pour tous les problèmes présentant une singularité du champ de déplacement au fond de fissure ou une discontinuité au niveau des lèvres de fissure,

Parmi les avantages de cette méthode, on mentionne qu'il n'est pas nécessaire de représenter explicitement la fissure par le maillage, et la modélisation la propagation des fissures sans remaillage du domaine.

Mathématiquement, la partition de l'unité c'est l'ensemble de fonctions de forme des éléments finis classiques dans un espace $\Omega$ qui vérifient ce qui suit :

$$\sum_i N_i(\underline{x}) \equiv 1 \quad \forall \underline{x} \in \Omega \qquad \textbf{(1-22)}$$

Alors le champ de déplacement après l'enrichissement de l'espace d'approximation avec une fonction d'enrichissement $\psi(x)$ s'écrit comme suit :

$$u(x) = \underbrace{\sum_i N_i(\underline{x})\, u}_{\text{approx EF classique}_i} + \underbrace{\sum_i N_i(\underline{x}) . \psi(\underline{x})\, a_i}_{\text{enrichisse;ent}} \qquad \textbf{(1-23)}$$

On peut considérer que l'enrichissement tel que décrit dans (12), et qui est global, est plus coûteux et inefficace car le nombre de degrés de liberté enrichis est directement proportionnel au nombre total de nœuds du maillage, ce qui constitue l'un des inconvénients de cette méthode.

Ensuite quelques améliorations ont été apportées à la propriété de la méthode de partition de l'unité, basées sur l'enrichissement des sous-domaines locaux, en prenant en considération une meilleure prise en compte de phénomènes locaux tels que la singularité ou les discontinuités [74, 75]

Ces modifications qui permettent d'enrichir l'espace des fonctions de forme dans la zone d'intérêt en ayant une connaissance préalable des propriétés de résolution de problèmes sont considérées comme la base de la méthode des éléments finis généralisés (**GFEM**) et de la méthode des éléments finis étendue (**XFEM**) [76], qui se distinguent l'une de l'autre par la caractéristique de localisation d'enrichissement .

### 4.3.2 Méthode des éléments finis étendus (Extended Finite Element Method) :

La méthode des éléments finis étendus (eXtended Finite Element Method - **XFEM**) qui est une idée originale de Belytschko et Black [77, 78], repose principalement sur la flexibilité offerte par la partition de l'unité (**PUFEM**)..

Elle se présente comme un outil polyvalent pour les problèmes de propagation de fissures qui présente d'énormes avantages par rapport aux autres méthodes numériques mentionnées ci-dessus

La méthode **XFEM**, initialement conçu pour résoudre les problèmes quasi-statiques, vise à éviter de réaliser des maillages conformes aux fissures. Elle consiste à enrichir une approximation classique éléments finis en ajoutant différentes fonctions, ce qui représente parfaitement la discontinuité et la singularité d'une fissure dans un environnement matériel.

Au cours des deux dernières décennies, cette méthode a subi de nombreux progrès. Nous examinerons plus en détail cette méthode et ses utilisations dans le chapitre 2.

## 5 Conclusion :

Ce chapitre a exposé les principes théoriques de mécanique de la rupture linéaire élastique, accompagné des principaux paramètres permettant de décrire le comportement des fissures. Une analyse comparative des principales approches numériques a révélés les limites des méthodes classiques et l'avantage de la méthode des éléments finis étendus (XFEM), qui se présente comme une alternative performante pour la modélisation des discontinuités sans remaillage. Ces fondements théoriques fournissent la base scientifique requise pour le développement et l'implémentation de la formulation XFEM, qui seront détaillés dans le chapitre suivant.

# CHAPTRE 2 : PRINCIPES DE LA METHODE DES ELEMENTS FINIS ETENDUS (XFEM)

# 1 Introduction

Les problématiques associées à la présence des fissurations, aux phénomènes de la propagation des discontinuités matérielles ou d'interfaces constituent un défi major en mécanique numérique. Bien que les méthodes analytiques offre un cadre théorique rigoureux, et dans certains cas, des solutions d'une grandes précession, leurs domaine d'applicabilité demeure limite face à la complexité géométrique et à la non linéarité de ce phénomène. Ces restrictions soulignent la nécessité de recourir à des méthodes numériques avancées capables de traiter efficacement des configurations réalistes et d'en capturer fidèlement les mécanismes physiques.

La méthode des éléments finis (**FEM**) est imposée comme un outil de référence pour la modélisation des structures continues, et dans certaine mesure, des configurations présentant des discontinuités. Néanmoins, elle présente une limite fondamentale dans le traitement de ces derniers : le maillage doit impérativement être conforme aux interfaces et aux fronts de rupture. Cette contrainte impose la génération d'un maillage strictement conforme à la géométrie des discontinuités. Ce processus de remaillage répété à chaque mise à jour régulière lors de l'évolution et de la propagation des discontinuités s'avère rapidement très coûteux, tant en temps de calcul qu'en effort de prétraitement, notamment pour les configurations complexe, ou les discontinuités de évoluent de manière aléatoire
Dans ce contexte, la Méthode des Elements Finis Etendus (**XFEM**) [77, 78] a vu le jour à la fin des années 90, proposant un changement de paradigme. Son principe fondamental repose sur la séparation de la géométrie du maillage et celle des discontinuités. Plutôt que de contraindre le maillage à suivre la géométrie d'une discontinuité, cette méthode permet d'enrichir localement un espace d'approximation dans les éléments traversés par celle-ci.
Cet enrichissement repose sur le principe de la partition de l'unité (**PUFEM**)[72, 73], permettant l'intégration de fonctions prédéfinies dans la base discrète, tels que les fonctions de sauts afin de représenter les fortes discontinuités (déplacement) et des fonctions de singularités pour illustrer précisément champ de contrainte autour du fond de la discontinuité.

# 2 Modèles existant de la discontinuité

Le concept de discontinuité est une notion de mécanique de la rupture qui concerne la réaction des matériaux aux contraintes, en particulier en ce qui concerne les imperfections et les défauts présentes dans le matériau.

Ce concept est en relation avec les modifications soudaines des caractéristiques mécaniques du matériau liées à ces défauts.

Les discontinuités de déplacement sont généralement classées en deux catégories principales, à savoir

## 2.1 Une discontinuité forte

Une forte discontinuité est un terme qui désigne un problème discontinue où les propriétés mécaniques (telles que la résistance, la rigidité) du matériau changent brusquement au niveau de la discontinuité, elle survient précisément lorsque le champ primaire est discontinue, et elle se manifeste particulièrement dans les fissures et les vides.

Une forte discontinuité se caractérise par un saut abrupt dans le champ de déplacement

En d'autres termes, Une discontinuité forte peut être définie comme une discontinuité dans lesquelles les champs de déplacement et de déformation sont tous deux discontinus

Notamment, il est important de préserver le principe fondamental de la **XFEM**, qui consiste à enrichir l'approximation standard des éléments finis par des fonctions adéquates.

## 2.2 Une discontinuité faible

La discontinuité faible est un terme qui désigne un problème continu ou les propriétés mécaniques du matériau varient de manière continue mais non uniforme autour de la discontinuité, il se manifeste particulièrement dans les interfaces entre deux matériaux, les inclusions de matériaux ou entre deux phases solides,

Une solution est dite faiblement discontinue lorsqu'elle présente un champ de déplacement continu et que son premier dérivé est discontinue, c.-à-d. le champ de déformation discontinue[79].

En d'autres termes, une fonction d'interpolation qui présente un coude ou un pli le long des discontinuités faibles, c'est-à-dire qu'elle représente une grandeur physique qui est continue mais avec un gradient de la grandeur physique qui est discontinu (un saut dans sa dérivée)

Dans ce cas, on utilise des fonctions d'enrichissement de nature différente de celles désignées pour une forte discontinuité (voir la figure 2-1).

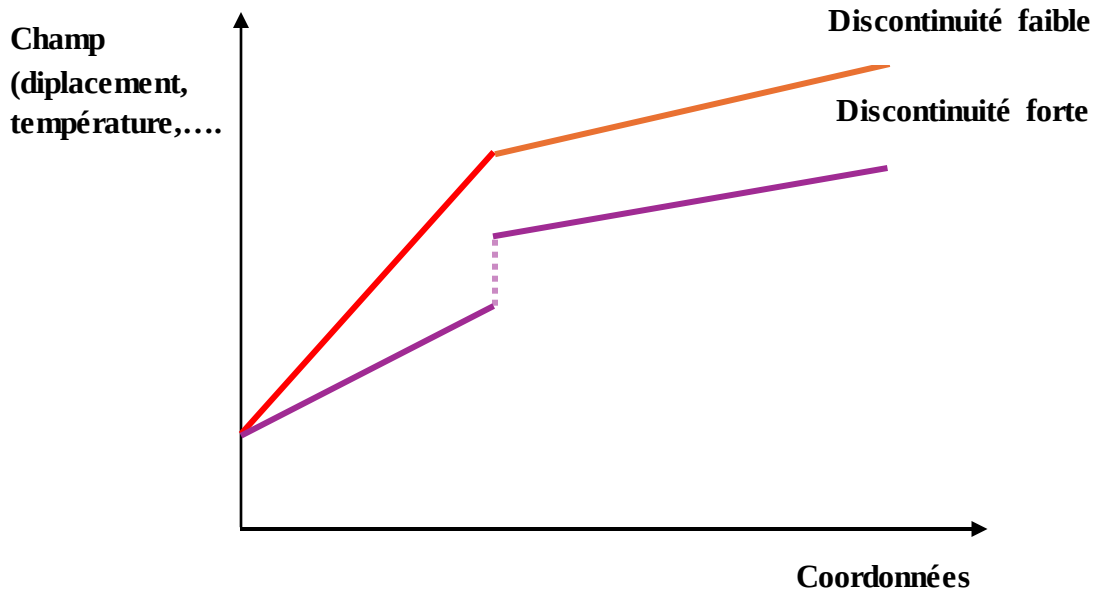



**Figure 2-1** Différent types de discontinuités

# 3 Méthode des éléments finis étendus (XFEM)

## 3.1 Introduction

Ce chapitre se concentre sur une analyse succincte de la méthode des éléments finis étendue (**XFEM)**. Il débutera par une brève revue bibliographique, son émergence et son développement remarquable, ainsi que sur ses différents domaines d'application.

Une introduction à la méthode des éléments finis étendue (**XFEM**) et les outils mathématiques employés pour cette méthode, ainsi que la notion d'enrichissement, sont examinées afin de mettre en évidence l'avantage que cette méthode apporte en enrichissant la règle classique des éléments finis.

Ensuite, La formulation du problème avec les conditions aux limites, le partitionnement à l'aide de la méthode **XFEM**, les règles d'intégration numérique seront expliquées.

Ce chapitre se conclut par des illustrations simples de discrétisation en 1D et 2D en utilisant la méthode **XFEM**, avant d'être appliquée de manière plus étendue à divers problèmes dans le chapitre suivant.

## 3.2 Généralités

Les algorithmes de calcul robustes et stables sont essentiels pour prédire correctement la charge de rupture et le comportement après le pic en dépassant la résistance du matériau. Ils doivent être capables de traiter un ensemble d'équations gouvernantes hautement non linéaires afin d'éviter l'apparition du processus de défaillance progressive dans les structures d'ingénierie.

Ces défaillances sont souvent initiées par des fissures de surface ou proches de la surface, ce qui réduit la résistance du matériau. Il est donc essentiel de réaliser une modélisation précise et une analyse minutieuse de la structure pour évaluer la résistance réelle du corps

La méthode des éléments finis (**FEM**), considérée comme l'un des outils numériques les plus courants pour trouver des solutions approximatives aux équations aux dérivées partielles, a été appliquée dans de nombreux domaines des sciences de l'ingénierie pour étudier, modéliser et prédire le comportement des structures.

La méthode des éléments finis s'appuie sur les propriétés d'approximation des polynômes, elle nécessite donc souvent des solutions lisses afin d'obtenir une précision optimale.

Cependant, il existe un certain nombre de cas où la méthode des éléments finis classique impose des restrictions sur son application effective, en conséquence de quoi la solution contient un comportement non lisse, tel que les fortes discontinuités dans le champ de déplacement comme dans le cas de corps fissurés.

Ainsi, l'utilisation de la méthodologie **FEM** dans les structures contenant des fissures et qui nécessite la conformité du maillage à la géométrie de la fissure ainsi que sa mise à jour à chaque fois que la fissure se propage, devient onéreuse et laborieuse en termes de calcul afin d'obtenir une convergence optimale de la solution, et cela entraîne également une perte de la précision lorsque les données sont transférées de l'ancien maillage au nouveau maillage.

La méthode des éléments finis étendus, également connue sous le nom de **XFEM**, qui a connu une croissance significatif au cours des deux dernières décennies, a été présentée comme une alternative à la méthode classique des éléments finis **FEM** comme solution à ce problème.

Donc, La méthode des éléments finis étendus (**XFEM)** qui est un procédé numérique développé par [78] et Moës [77], est une extension de la méthode des éléments finis classique **FEM**, utilisée lorsque des discontinuités se produisent dans l'espace étudié afin de s'affranchir des difficultés de maillage et de remaillage

La méthode **XFEM** à exploiter la flexibilité offerte par la méthode de la partition de l'unité (**PUFEM)** développée par Babuška et Melenk [72, 73], utilisant un support maillé pour construire l'espace d'approximation pour ne plus avoir à réaliser des maillages conformes aux fissures et aux frontières, ce qui permet de rendre les maillages plus simples pour des problèmes avec des discontinuités.

Cette caractéristique a l'avantage de rendre la méthode des éléments finis étendus (**XFEM**) plus flexible afin d'éviter la dépendance du maillage qui sert de support aux fonctions de forme

l'approximation du champ de déplacement discontinu dans La méthode des éléments finis étendue (**XFEM)** repose sur la création des fonctions de forme spécifiques en utilisant des fonctions

d'enrichissement ajoutées localement , C'est-à-dire là où il faut enrichir le domaine, cela conduit à l'enrichissement de la base des fonctions de forme afin de représenter des discontinuités au sein même d'un élément de maillage.

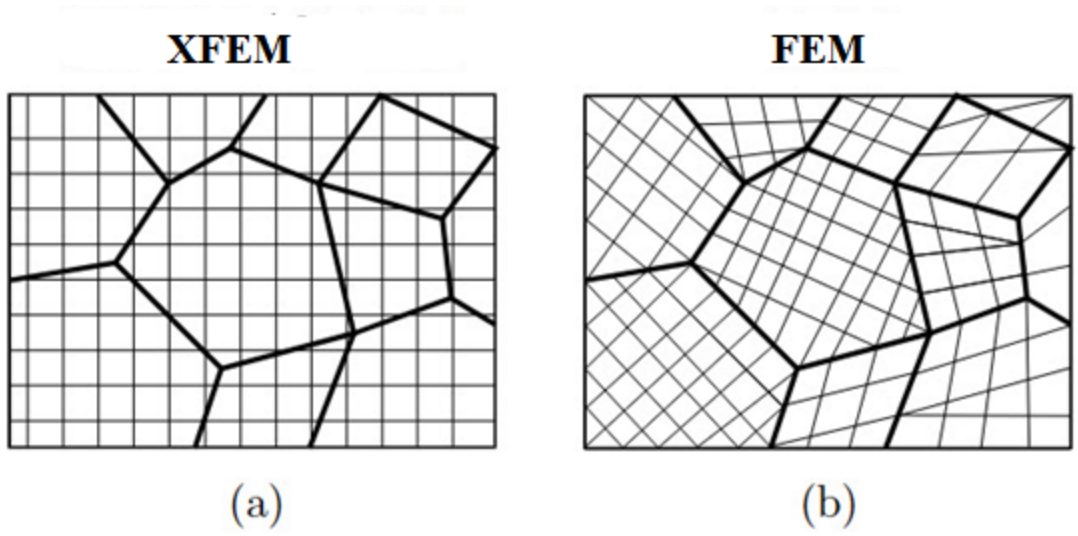



**Figure 2-2** Discrétisation d'un problème de joint pour : a) un modèle XFEM avec un maillage structuré b) un modèle FEM avec un maillage conforme à la géométrie de la discontinuité

Donc l'ajout des degrés de liberté virtuels indépendants, conduit à un système d'équations algébriques constitué de deux catégories d'inconnues, les degrés de liberté classiques et les degrés de liberté enrichis, dont la solution permet de définir les limites des fissures et d'évaluer approximativement le champ de déplacement globale.

L'enrichissement se caractérise d'une part par un enrichissement discontinu, où la tâche de remaillage est écartée lors de la propagation de la fissure, car le déplacement d'une discontinuité ne sera qu'une mise à jour des fonctions d'enrichissement.

Et d'autre part par un enrichissement singulier qui permet d'obtenir de bonnes approximations près de la singularité même sur un maillage relativement grossier, contrairement à la méthode des éléments finis classiques où il est nécessaire d'avoir un maillage conforme aux fissures et de réduire la taille des éléments finis

En effet, la frontière physique du domaine modélisée par **XFEM** ne sera plus définie par la topologie du maillage et sera donc donnée de manière séparée, permettant à la frontière des géométries d'être indépendante des éléments Par conséquent, la divergence du champ d'approximation sera mise en évidence par les divergences dans les fonctions de forme.

Les différentes caractéristiques mentionnées ci-dessus confèrent à La méthode des éléments finis étendue (**XFEM)** des avantages distincts par rapport à la méthode des éléments finis classique **FEM** pour la modélisation de discontinuités arbitraires.

# 4 Modélisation des fissures par XFEM

Cette méthode est basée sur la notion de partition de l'unité (**PUFEM**) (indiquée ci-dessus), qui implique un enrichissement local des nœuds proches de la discontinuité afin de décrire le saut dans le domaine de déplacement discontinu à travers la fissure.

## 4.1 Types d'enrichissement

Dans le cadre de la méthode des éléments finis étendus (**XFEM**), la représentation correcte des discontinuités du champ du déplacement et des singularités repose sur deux familles fondamentales de fonctions :

### 4.1.1 La fonction Heaviside

Cette fonction [77] est introduite afin de modéliser les discontinuités fortes du champ de déplacement au sein des éléments finis totalement traversées par une fissure. Elle permet de représenter le saut cinématique existant entre les deux lèvres d'une fissure sans tenir compte du maillage, évitant ainsi toute contrainte d'alignement entre la géométrie de la fissure et les frontières élémentaires. Elle est définit généralement à partir d'une fonction de niveau $\Phi(x)$, utilisée pour décrire la position et l'orientation de la fissure dans le domaine, donc :

$$\Phi(x) = H(f(x)) = \begin{cases} +1 & si \quad f(x) > 0 \\ -1 & si \quad f(x) < 0 \end{cases} \qquad \textbf{(2-1)}$$

Avec $H(f(x))=0$ définissant la surface de discontinuité.

### 4.1.2 Fonctions asymptotiques pour singularité

Ces fonctions [80] sont introduites afin de représenter avec exactitude le comportement singulier du champ de déplacement et le champ des contraintes au voisinage immédiat du fond de fissure, comme le prévoit la mécanique de rupture élastique. Ces fonctions appelées les fonctions de Williams sont donnée comme suit :

$$F(r,\theta)=\left\{\sqrt{r}\sin\frac{\theta}{2},\sqrt{r}\cos\frac{\theta}{2},\sqrt{r}\sin\frac{\theta}{2}\sin\theta,\sqrt{r}\cos\frac{\theta}{2}\sin s\theta\right\} \quad \textbf{(2-2)}$$

Ces fonctions sont issue des solutions analytiques de types Westergaard et sont exprimées en coordonnées polaires $(r,\theta)$, centrées aux fond de fissure, elles capturent explicitement la singularité en $1/\sqrt{r}$ des contraintes et l'évolution angulaire du champ de déplacement, offrant ainsi une présentation précise de la mécanique locale de la rupture.

Ces coordonnées peuvent être s'exprimer en fonction des coordonnées globales via les fonctions de niveau qui sont des fonctions de distance, comme :

$$\begin{cases} \mathrm{r}(\mathrm{x})=\sqrt{\Psi_{\mathrm{n}}(\mathrm{x})^2+\Psi_{\mathrm{t}}(\mathrm{x})^2} \\ \theta(\mathrm{x})=\arctan\left(\dfrac{\Psi_{\mathrm{n}}(\mathrm{x})}{\Psi_{\mathrm{t}}(\mathrm{x})}\right) \end{cases} \quad \textbf{(2-3)}$$

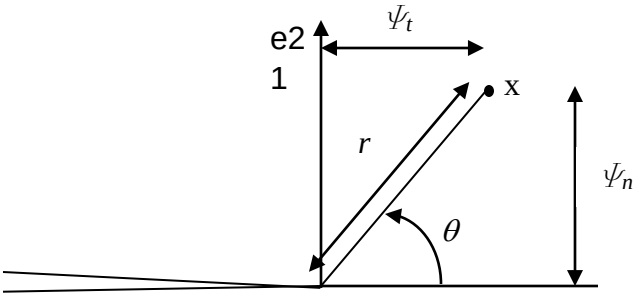



**Figure 2-3** Coordonnées polaires dans la base locale en fond de fissure

## 4.2 Stratégies d'enrichissement

La précision et le cout du calcul sont affectés par la sélection des nœuds à enrichir. Concernant la zone de fond de fissure, une stratégie d'enrichissement géométrique est privilégiée. On intègre tous les éléments situés dans un cercle $C(x_0,r)$, dont le centre est positionné à l'extrémité de la fissure et qui a pour rayon « *r* ». Ces éléments sont généralement enrichis par les fonctions asymptotiques [81, 82].

L'emploi combiné de ces enrichissement (Heaviside et asymptotique) confrère à **XFEM** une capacité accrue à représenter simultanément le saut de déplacement au niveau de la fissure et la singularité locale en pointe, tout en conservant la structure initiale et évitant les processus de remaillage

## 4.3 Formulation de l'approximation XFEM

Pour un problème bidimensionnel de mécanique de solide avec une fissure, l'approximation du champ de déplacement par **XFEM** est donnée par l'expression générale suivante :

$$u^h(x) = \underbrace{\sum_{I\in \mathrm{N}} N_I(x)u_I}_{\textit{FEM Standard}} + \underbrace{\sum_{J\in \mathrm{N}_{er}} \tilde{N}_J(x)H_j(x)a_J}_{\textit{Enrichissement Heaviside}} + \underbrace{\sum_{K\in N_{tip}} \tilde{N}_K(x) \sum_{\alpha=1}^{4} (F_\alpha(x)\, b_K^\alpha)}_{\textit{Enrichissement Asymptotique}} \qquad \textbf{(2-4)}$$

Où:

$N_I(x)$, $N_J(x)$, $N_K(x)$ : sont les fonctions de forme Standard des éléments finis.

$I$ : désigne l'ensemble des nœuds standards.

$J$ : correspond à l'ensemble des nœuds enrichis par la fonction de Heaviside (éléments entièrement traversées par la fissure).

$K$ : représente l'ensemble des nœuds enrichis par les fonctions asymptotiques de fond de fissure.

$u_I$ : degré de liberté classique du nœud i.

$a_J$ : degré de liberté enrichis pour la fonction Heaviside.

$b^a{}_K$ : degré de liberté enrichis pour les fonctions asymptotiques.

$H(x)$ : est la fonction de Heaviside modifiée

$F_\alpha(x)$: sont les fonctions asymptotiques associées au champ singulier en pointe de la fissure..

## 4.4 Discrétisation et opérateur gradient

### 4.4.1 Expression discrète du champ de déplacement

Sous forme vectorielle compacte, l'approximation s'écrit :

$$u^h(x) = \mathrm{N}(x).\mathrm{U}^e \qquad \textbf{(2-5)}$$

Où:

$U^e$ : est le vecteur des degrés de liberté élémentaire (globaux)

N(x) : la matrice des fonctions de forme enrichis.

### 4.4.2 Opérateur gradient discrétisé

Le calcul de déformations $\mathbf{B}(x)=\left[\mathbf{B}^{std}(x)\middle|\ \mathbf{B}^{enr}(x)\right]$ nécessite la dérivation des fonctions de forme enrichis. Le tenseur de déformation s'exprime sous forme vectorielle, par ;

$$\varepsilon(x)=\begin{bmatrix}\varepsilon_{xx}\\ \varepsilon_{yy}\\ 2\varepsilon_{xy}\end{bmatrix}=\mathbf{B}(x).\mathbf{U}^{e} \qquad \textbf{(2-6)}$$

Ou, B(x) est l'opérateur gradient discrétisé. Celui-ci se décompose en

$$\mathbf{B}(x)=\left[\mathbf{B}^{std}(x)\middle|\ \mathbf{B}^{enr}(x)\right] \qquad \textbf{(2-7)}$$

$\mathbf{B}^{std}(x)$ correspond à la partie classique liée au fonctions de forme standard :

$$\mathbf{B}^{std}(x)=\begin{bmatrix}N_{I,x} & 0\\ 0 & N_{I,y}\\ N_{I,y} & N_{I,x}\end{bmatrix} \qquad \textbf{(2-8)}$$

Tandis que $\mathbf{B}^{enr}(x)$ regroupe les contributions dues aux fonctions d'enrichissement :

$$\mathbf{B}^{enr}(x)=\begin{bmatrix}N_{J,x}H & 0\\ 0 & N_{J,y}H\\ N_{J,y}H & N_{J,x}H\end{bmatrix} \qquad \textbf{(2-9)}$$

### 4.4.3 Procédure éléments finis classique

Les dérivées des fonctions de forme standard $N_i$ sont obtenues par l'approche isoparamétrique classique. Les relations entre les dérivés de l'espace parent $(\xi,\eta)$ et l »espace physique (x,y) font intervenir la matrice Jacobienne « J ».

$$\begin{bmatrix}\dfrac{\partial N_I}{\partial x}\\ \dfrac{\partial N_I}{\partial y}\end{bmatrix}=J^{-1}\begin{bmatrix}\dfrac{\partial N_I}{\partial \xi}\\ \dfrac{\partial N_I}{\partial \eta}\end{bmatrix} \qquad \textbf{(2-10)}$$

Sachant que la matrice Jacobienne « J » de transformation est définie par ;

$$J=\begin{bmatrix}\dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi}\\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta}\end{bmatrix}=\sum_{I=1}^{4}\begin{bmatrix}\dfrac{\partial N_I}{\partial \xi}x_1 & \dfrac{\partial N_I}{\partial \xi}y_1\\ \dfrac{\partial N_I}{\partial \eta}x_1 & \dfrac{\partial N_I}{\partial \eta}y_1\end{bmatrix} \quad \textbf{(2-11)}$$

Donc pour un élément quadrilatère bilinéaire Q4, les dérivées dans l'élément de référence sont :

$$\begin{cases}\dfrac{\partial N_1}{\partial \xi}=-\dfrac{1}{4}(1-\eta),\ \dfrac{\partial N_1}{\partial \eta}=-\dfrac{1}{4}(1-\xi)\\ \dfrac{\partial N_2}{\partial \xi}=\dfrac{1}{4}(1-\eta),\ \dfrac{\partial N_2}{\partial \eta}=-\dfrac{1}{4}(1+\xi)\\ \dfrac{\partial N_3}{\partial \xi}=\dfrac{1}{4}(1+\eta),\ \dfrac{\partial N_3}{\partial \eta}=\dfrac{1}{4}(1+\xi)\\ \dfrac{\partial N_4}{\partial \xi}=-\dfrac{1}{4}(1+\eta),\ \dfrac{\partial N_4}{\partial \eta}=\dfrac{1}{4}(1-\xi)\end{cases} \quad \textbf{(2-12)}$$

Cette démarche reste inchangée dans **XFEM**, la modification intervenant uniquement dans l'ajout des colonnes correspondantes aux ddl enrichis dans la matrice B.

## 4.5 Construction des matrices de rigidité

### 4.5.1 Expression de la matrice de rigidité enrichis

Pour un élément enrichi, la matrice de rigidité est obtenue en intégrant l'énergie de déformation sur le domaine de l'élément $\Omega_e$ :

$$K^e=\int_{\Omega_e} B^T.D.B.d\Omega \quad \textbf{(2-13)}$$

Où « *D* » est la matrice de comportement constitutif (lois de Hooke). En raison de degré de liberté supplémentaires, la matrice de rigidité élémentaire possède une structure par blocs :

$$K^e=\begin{bmatrix}K_{uu} & K_{ua} & K_{ub}\\ K_{au} & K_{aa} & K_{ab}\\ K_{bu} & K_{ba} & K_{bb}\end{bmatrix} \quad \textbf{(2-14)}$$

Où:

$K_{uu}$ : correspond à la rigidité standard (taille 8x8 pour un quadrilatère à 4 nœuds).

$K_{uu}$ : correspond à la rigidité Heaviside.

$K_{uu}$ : correspond à la rigidité des fonctions asymptotique.

$K_{ua}$ , $K_{ub}$ ; $K_{ab}$ : termes de couplage.

Donc la matrice de rigidité élémentaire atteint une dimension de (16x16)

### 4.5.2 Conditionnement de la matrice de rigidité

La méthode **XFEM** peut être sujette à des problèmes de conditionnement inapproprié de la matrice de rigidité globale. Ce phénomène se produit généralement lorsque le tracé de la fissure se s'approche considérablement d'un nœud ou est presque parallèle à une arrête d'un élément. Dans cette situation, le support de la fonction d'enrichissement relative à l'élément devient presque nul, entrainant des dépendances linéaires entre les colonnes de la matrice.

On utilise fréquemment des techniques de conditionnement ou des approches de modification géométriques locales (tel que la suppression de l'enrichissement pour les nœuds trop proches) dans le but de stabiliser la solution.

## 4.6 Modélisation d'une fissure en 1D

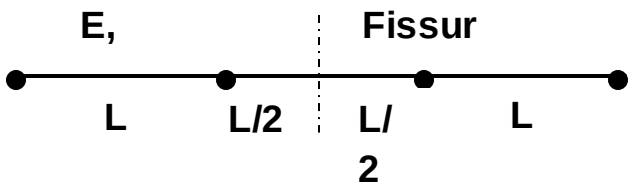
Dans cet exemple, on se propose d'illustrer une stratégie d'enrichissement permettant de modéliser la discontinuité de champ de déformation dans un bar unidimensionnelle. La barre considérée encastrée à une extrémité et libre de l'autre, ou une force axiale « *F* » est appliquée.

L'objectif est de mettre en évidence la capacité de la méthode **XFEM** à représenter une discontinuité interne sans recourir à un remaillage explicite

On suppose que la barre est divisée en trois éléments finis unidimensionnels, chacun étant définie par deux nœuds et associé à des fonctions de forme linéaires telles qu'illustré à la Figure 3.

Une discontinuité cinématique, assimilé à une fissure est introduite à l'intérieur du domaine, séparant la barre en deux domaine distinct.

Le champ de déplacement des deux parties de la barre est approché à l'aide d'une formulation **XFEM** enrichis, dans laquelle des fonctions d'enrichissement appropriées sont ajoutées aux fonctions de forme classique, afin de capturer la discontinuité du champ de déplacement ou de déformation



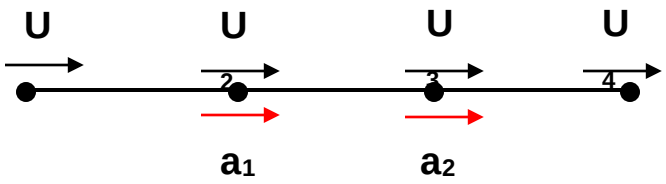


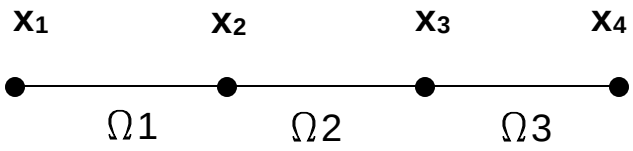

**Figure 2-4** Modélisation d'une discontinuité de champ de déformation dans un bar unidimensionnelle

Ou :

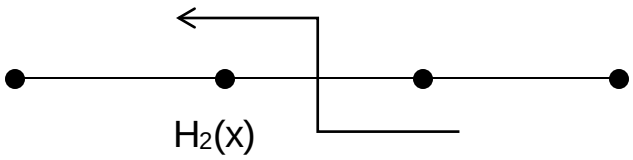
$u_i$ : standard degré de liberté

$a_i$ : degré de liberté enrichis

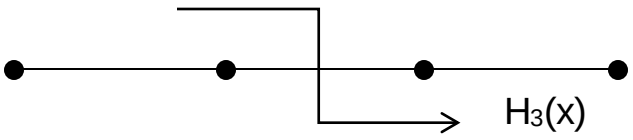
Domaine discrétisé




Fonction de saut sur le support de nœuds ($x_2$)




Fonction de saut sur le support de nœuds ($x_3$)




Ou :

$x_i$ : nœuds

$[x_2, x_3]$ : domain devisé par une fissure.

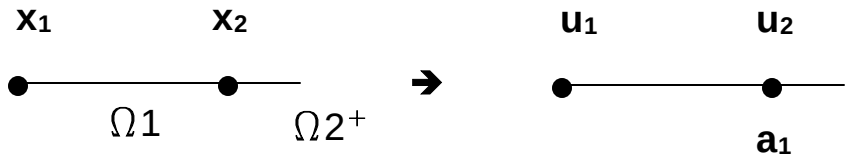
L'enrichissement se fait par step function

L'addition de deux degrés de liberté (4 standard + 2 enrichis)

Pour chaque élément, la matrice de rigidité en **XFEM** s'écrit comme suit :

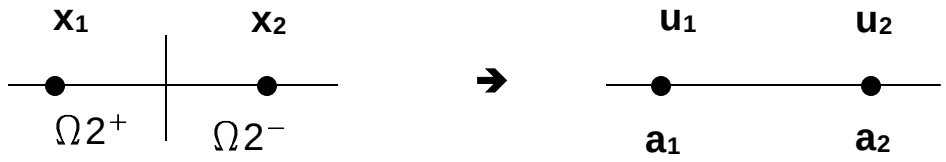
$$[K]^e = \begin{bmatrix} K_{uu} & K_{ua} \\ K_{au} & K_{aa} \end{bmatrix} \quad \text{Ou :} \quad \begin{cases} [K_{uu}] = \int_0^l (B_{std})^T D B_{std}\, dx \\ [K_{ua}] = \int_0^l (B_{std})^T D B_{enr}\, dx \\ [K_{au}] = \int_0^l (B_{enr})^T D B_{std}\, dx \\ [K_{aa}] = \int_0^l (B_{enr})^T D B_{enr}\, dx \end{cases}$$

Element 1 ($\Omega 1$) :




==> la matrice de rigidité globale d'élément « 1 » est : $[K_1] = \frac{EA}{L}\begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \end{bmatrix}$

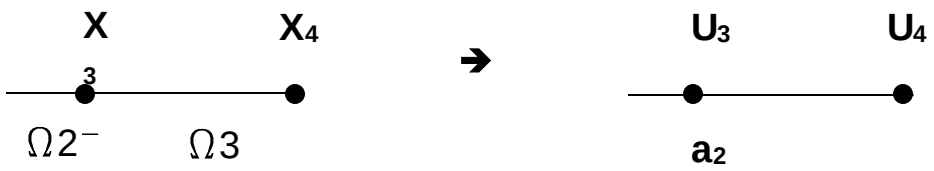
Element 2 ($\Omega 2$) :




===> la matrice de rigidité globale d'élément « 2 » est :

$$[K_2] = \frac{EA}{L}\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Element 3 ($\Omega 3$) :



===> La matrice de rigidité globale de la barre complète est :

$$[K_e]=[K_1]+[K_2]+[K_3]$$

$$[K_e]=\frac{EA}{L}\begin{bmatrix} 1 & -1 & 0 & 0 & -1 & 0 \\ -1 & 2 & -1 & 0 & 1 & 0 \\ 0 & -1 & 2 & -1 & 0 & -1 \\ 0 & 0 & -1 & 1 & 0 & 1 \\ -1 & 1 & 0 & 0 & 2 & -1 \\ 0 & 0 & -1 & 1 & -1 & 2 \end{bmatrix}$$

## 4.7 Modélisation d'une fissure en 2D

Dans cette section, l'objectif principal est d'illustrer de manière détaillée la mise en œuvre de la méthode des éléments finis étendus **XFEM** pour la modélisation d'une fissure traversant partiellement un domaine bidimensionnelle.

On considère deux éléments quadrilatères adjacentes de type Q4, soumises à un état de déformation plane, et caractérisés par un matériau élastique isotrope linéaire définie par un module d'élasticité E=210 GPa et un coefficient de poisson ν=0.3.

La géométrie du domaine représentée dans la figure 4 , est définie à partir des coordonnées nodales suivantes : 1(0.0, 0.0) ; 2(1.0, 0.0) ; 3(2.0, 0.0) ; 4(2.0, 1.0) ;5 (1.0, 1.0) ; 6(0.0, 1.0), tandis que la fissure est représenté par un segment reliant les points 7(0.0, 0.03) ; 9(1.3, 0.3), avec une intersection situé à l'intérieure commun de la frontière au point 8(1.0, 0.3)

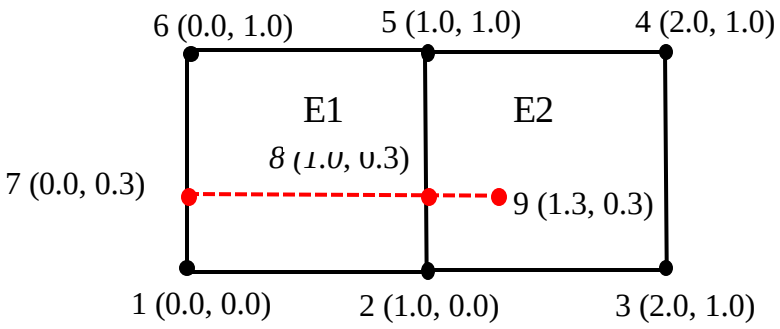



**Figure 2-5** Modélisation d'une discontinuité de champ de déformation dans deux élément quadrilatères bidimensionnelle

L'élément E1 est entièrement coupé par la fissure et est donc enrichi par la fonction Heaviside, alors que l'élément E2 contient le fond de la fissure et est donc enrichi par les fonctions Asymptotiques décrivant le champ singulier au voisinage de la pointe.

Afin de permettre une intégration numérique rigoureuse des termes de rigidité en présence de discontinuité, chaque élément quadrilatère est subdivisé en sous-domaines triangulaires.

Cette subdivision permet d'utiliser une formulation standard des fonctions de forme triangulaires afin de capturer correctement la géométrie de la fissure à l'intérieur de l'élément

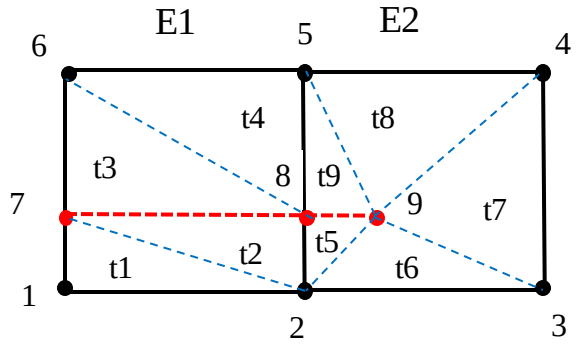



**Figure 2-6** Subdivision des éléments quadrilatères de types Q4 en sous-domaines triangulaires

1-Calcul de la matrice constitutive $D$ :

$$D = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \approx 10^9 \begin{bmatrix} 230.769 & 69.23 & 0 \\ 69.23 & 230.769 & 0 \\ 0 & 0 & 80.77 \end{bmatrix}$$

La matrice D relie déformation – contrainte pour un matériau linéaire isotrope en régime linéaire.

2-Matrice de rigidité **XFEM**:

a) Dans le cas de l'élément E1, qui est traversé complétement par la fissure, est donc enrichi par la fonction Heaviside seulement. La matrice de rigidité dont la dimension est de (12x12) s'écrit comme suit :

$$K^{e1} = \begin{bmatrix} K_{uu} & K_{ua} \\ K_{au} & K_{aa} \end{bmatrix}$$

Avec

- La formule de la matrice de rigidité classique en éléments finis dont la dimension est de (6x6) est : $K_{uu}^{E1} = \int_{\Omega e1} B^T . D . B . d\Omega$

- La formule de la matrice de rigidité de couplage déplacement continue/discontinue, dont la dimension est de (6x6) est : $K_{ua}^{E1} = \int_{\Omega e1} B^T . D . B^a . d\Omega$

- La formule de la matrice de rigidité de couplage déplacement continue/discontinue, dont la dimension est de (6x6) est : $K_{aa}^{E1} = \int_{\Omega e1} \left(B^a\right)^T . D . B^a . d\Omega$

D'où la définition de la matrice B enrichis :

$$\mathrm{B}^{enr}(x) = \begin{bmatrix} \frac{\partial(N_i H)}{\partial x} & 0 \\ 0 & \frac{\partial(N_i H)}{\partial y} \\ \frac{\partial(N_i H)}{\partial y} & \frac{\partial(N_i H)}{\partial x} \end{bmatrix}$$

b) Dans le cas de l'élément E2, qui contient le fond de fissure, est donc enrichis par les fonctions asymptotiques. La matrice de rigidité de dimension (30x30) s'écrit comme suit :

$$K^{e2} = \begin{bmatrix} K_{uu} & K_{ub} \\ K_{bu} & K_{bb} \end{bmatrix}$$

Avec :

- La formule de la matrice de rigidité classique en éléments finis dont la dimension est de (6x6) : $K_{uu}^{E2} = \int_{\Omega e1} B^T . D . B . d\Omega$

- La formule de la matrice de rigidité de couplage déplacement continue/discontinue, dont la dimension est de (6x30) est : $K_{ub}^{E2} = \int_{\Omega e1} B^T . D . B^b . d\Omega$

- La formule de la matrice de rigidité de couplage déplacement discontinue/discontinue, dont la dimension est de (6x30) est : $K_{bb}^{E2} = \int_{\Omega e1} \left(B^b\right)^T . D . B^b . d\Omega$

D'où la définition de la matrice B enrichis :

$$\mathrm{B}^{enr}(x) = \sum_{\alpha=1}^{4} \begin{bmatrix} \frac{\partial(N_i F_\alpha)}{\partial x} & 0 \\ 0 & \frac{\partial(N_i F_\alpha)}{\partial y} \\ \frac{\partial(N_i F_\alpha)}{\partial y} & \frac{\partial(N_i F_\alpha)}{\partial x} \end{bmatrix}$$

## 5 Conclusion

Ce chapitre a établi les bases théoriques et numériques de la méthode des éléments fins étendus (XFEM), appliquée à la modélisation des discontinuités en mécanique de la rupture. Après avoir présenté les principes d'enrichissement, la formulation mathématique et les stratégies d'intégration numérique, une application détaillée sur un exemple bidimensionnel a illustré la construction des matrices de rigidité enrichis et le traitement des éléments traversés par une fissure ou contenant son extrémité. Ainsi ce chapitre établit le cadre méthodologique sur lequel reposent le développement de la plateforme OpenXFEM++ et les analyses numériques réalisés dans les chapitres suivants.

# CHAPITRE 3 : IMPLEMENTATION XFEM PAR L'APPROCHE ORIENTÉE OBJET EN C++

## 1. Introduction

La programmation orientée objet abrégé en **POO**, qui a été inventée dans les années 1960, figure parmi les paradigmes de programmation les plus réputés et efficaces.

La **POO** est une approche ou méthode utilisée pour la rédaction d'un code visant à le rendre plus accessible. Elle repose sur l'organisation des codes source par la mise en place d'une structure dans laquelle les données seront stockées, afin de faciliter leur accès et leur modification

La programmation orientée objet (**POO**) offre aux programmeurs la possibilité de mieux modéliser le monde réel ou des problématiques complexes. Elle fournit une méthode structurée et systématique pour élaborer et implémenter des applications robustes et facilement évolutives. Pour structurer les programmes, La **POO** se focalise sur les notions de classes et d'objets les qui en dérivent, qui sont comparables aux objets du monde réel, servant à représenter les composants de l'application. Cela permet de segmenter le programme en unités indépendantes plus petites interagissant entre elles.

## 2. Les caractéristiques de l'approche Orientée Objet

La programmation orientée objet (**POO**) offre plusieurs bénéfices, y compris :

- Une meilleure compréhension : la classe est aisée à comprendre, car elle peut être aisément comprise par d'autres programmeurs.
- Une meilleure Structuration et organisation : le code peut être structuré de manière logique et structurale, et le programme peut être segmenté en sous-classes et petites unités faciles à gérer.
- Une meilleure réutilisation : la classe ainsi que son code peuvent être employés sur divers sites sans nécessiter une réécriture complète du code. Ceci permet d'économiser du temps et des efforts et contribue à améliorer la maintenance des logiciels
- Une meilleure Évolutivité : afin de permettre l'intégration aisée de nouvelles fonctionnalités au code source, Parce qu'une modification apportée à une structure de données ne touche que le code de la classe qui a été modifiée.
- Une meilleure Simplicité du test : le code source se prête aisément aux tests.

- Une meilleure protection des données : On ne peut accéder aux informations d'un objet qu'à travers les propriétés et les méthodes qui sont publiques (l'encapsulation et l'abstraction)

## 3. Aperçu sur le langage C++ :

**C++** est un langage de programmation orientée objet. Ce langage de programmation, compilé et à usage général, se distingue par son typage statique, sa sensibilité à la casse et sa syntaxe libre. Il est multi-paradigmatique, impératif, procédural, fonctionnel, générique et unifié.

Les experts ont des opinions divergentes quant à la catégorisation du langage **C++**, certains le considérant comme un langage de bas niveau en raison de sa proximité du matériel (langage machine), du fait qu'il offre à l'utilisateur un contrôle important sur la gestion de la mémoire, de meilleures performances et finalement l'élaboration de programmes robustes.

Quelques-uns le rangent parmi les langages de haut niveau, car ceux-ci dépendent du processeur, contrairement aux langages de bas niveau.

Cependant, la majorité le perçoit comme un niveau intermédiaire puisqu'il regroupe à la fois un ensemble de fonctionnalités linguistiques de haut et de bas niveau.

Le **C++** est souvent jugé performant en matière de mémoire et de rapidité comparé à d'autres langages de programmation tels que Java, Python, et ainsi de suite.

Sa genèse remonte au début de 1979, quand **Bjarne Stroustrup** [83], un informaticien danois travaillant à Bell Labs de Murray Hill, New Jersey, a eu l'idée de créer un nouveau langage de programmation suite à son expérience en programmation lors de sa thèse doctorale.. Il choisit de développer le langage C en raison de sa performance, son flexibilité et sa large portée, tout en intégrant les bénéfices du langage **Simula**67, notamment le concept de classe (avec les classes dérivées et les fonctions virtuelles).

La version initiale de ce langage novateur, dénommée « **C with Class** » [83], a été publiée en 1980, ou le nom faisait référence aux classes, étant donné qu'elles constituent l'une des avancées significatives de ce nouveau langage par rapport au C.

en 1982, Stroustrup a pu introduire de nouvelles fonctionnalités dans le langage "C with Class". En plus des classes et classes dérivées ajoutées précédemment, il a pu ajouter des fonctions virtuelles, la surcharge de noms de fonctions et d'opérateurs, l'utilisation de références et de

constantes, une allocation sécurisée de mémoire libre par type (new/delete), ainsi que des commentaires sur une ligne et des références.

Il a aussi également conçu un nouveau compilateur autonome, nommé **Cfront**.

En 1983, le langage « **C with Class** » a été renommé The **C++** Programming Language « **C++** » (prononcé « C plus plus »), ce nom a été inventé par Rick Mascitti, il signifie la nature évolutive des changements par rapport au C ; « ++ » est l'opérateur d'incrémentation du langage **C**.

En 1985, la première édition du « **C++** » a vu le jour, et rapidement devenue la référence incontournable pour ce langage à cause de l'absence d'un standard uniforme à l'époque. De plus, la première application commerciale utilisant C++ a été lancée en octobre de cette même année.

Au fil de temps, le **C++** a subi de nombreuses mises à jour et développements qui incluaient une diversité d'ajouts et de fonctionnalités variés pour répondre aux défis rencontrés par les utilisateurs, dont la plus notable :

- Le **C++ 2.0**, publié en 1989, a ensuite été suivi par sa seconde édition révisée en 1991. Cette version a introduit de nouvelles fonctionnalités telles que l'héritage multiple, les classes abstraites, les fonctions membres statiques, les fonctions membres constantes et les membres protégés.

- Le **C++98**, qui a vu le jour en 1998, constitue la première norme officielle normalisée par l'ISO et qui a été publiée le 1er septembre 1998.

Plusieurs mises à jour ont été effectuées au cours du nouveau millénaire, y compris le **C++03**, une mise à jour mineure lancée en 2003, la version **C++11**, considérée comme une mise à jour majeure qui a été adoptée et publiée par l'ISO en septembre 2011, le **C++14** publiée en août 2014, le **C++17** publiée en décembre 2017, le **C++20** publiée en février 2020 et enfin la plus récente, **C++23**, actualisée en octobre 2024.

De ce fait, il est donc envisageable que le langage **C++** soit relativement ancien, tout en demeurant pertinente, et il ne cesse pas de progresser.

Par ailleurs le langage **C++** demeure l'un des langages de programmation inégalée et populaire parmi les forums de programmation et les plus demandés, selon l'indice **PYPL** qui évalue la popularité des langages de programmation en fonction du nombre de recherches de cours sur

Google, **C/C++** occupe **la deuxième** position dans le classement global après le langage **Python** (janvier 2026). (**Voir…. https://pypl.github.io/PYPL.html**)

De nombreux langages de programmation ont tiré leur inspiration du **C++.** Comme c'est particulièrement le cas avec le langage **Java**, tout en laissant de côté certaines fonctionnalités de **C++** comme les pointeurs, la surcharge d'opérateurs et l'héritage multiple, afin de maintenir la simplicité du langage.

Ainsi que le langage **C#**, qui est basé sur la syntaxe **C++** et utilise des fonctionnalités clés comme les classes, en plus d'une variété de langages tels que **Perl**, **Falcon**, **PHP**, **Ada95** et le langage **D**. **C++** est compatible avec plusieurs plateformes, y compris Windows, Mac OS et les différentes variantes d'UNIX.

Le **C++**, étant un langage polyvalent, est largement adopté pour programmer une multitude de logiciels et est souvent la première option choisie pour les projets de grande envergure, qu'il s'agisse de développement mobile (système Symbian utilisé par Nokia,….), d'applications de bureau (Adobe Photoshop, Acrobat, Doxygen, Gmsh, Matlab, Microsoft Office, Autodesk Maya Interface graphique élégante KDE des bureau Linux,….), de Web (Chromium Browser, Firefox Browser, Dropbox, Mozilla Thunderbird, .. ) ou encore de systèmes d'exploitation.

De plus, de nombreux jeux le mettent en œuvre comme World of Warcraft, Civilization and the Jews etc….

## 4. Implémentation orientée objet en C++ par éléments finis enrichis

La programmation orientée objet a pour objectifs de fournir des codes à usage général, conçus pour être compréhensibles, facile à maintenir, extensible en permettant une incorporation beaucoup plus facile de nouveaux types d'éléments, de techniques de solution et d'autres fonctionnalités au fur et à mesure qu'ils deviennent disponibles.

Le code **OpenXFEM++** [84] , est un environnement open source écrit en langage **C++** et développé dans le cadre d'une approche de programmation orientée objet (**POO**), sur la base d'un code d'éléments finis existant **FEMOBJ** [85] , un package d'éléments finis orientée objet pour les applications non linéaires, statiques et dynamiques.

Ce présent code **OpenXFEM++** est conçu spécialement pour résoudre différents problèmes en mécanique de la rupture élastique linéaire, y compris le calcul des paramètres de rupture pour différents types de discontinuités ainsi que la modélisation de la croissance des fissures. Son

application a aussi été élargie pour la résolution des problèmes liés à la mécanique de la rupture élastique non linéaire.

La méthode des éléments finis étendus (**XFEM**) étant par nature un outil numérique modulaire, la programmation orientée objet facilite l'utilisation de ce module. Par conséquent, le code **OpenXFEM++** est évalué en termes d'efficacité numérique, et basé sur des mécanismes de programmation orientée objet.

**OpenXFEM++** englobe tous les aspects de la programmation orientée objet, y compris la conception d'applications basées sur ce paradigme. Cela implique la définition de classes et d'objets dotés de structures et d'attributs spécifiques, tels que les constructeurs, les destructeurs, les méthodes, les messages, l'encapsulation, le polymorphisme, l'héritage et les hiérarchies de classes... Il comprend de nombreuses classes et classes dérivées. Nous allons examiner ci-dessous certains types de classes courantes avec une description détaillée de chacune.

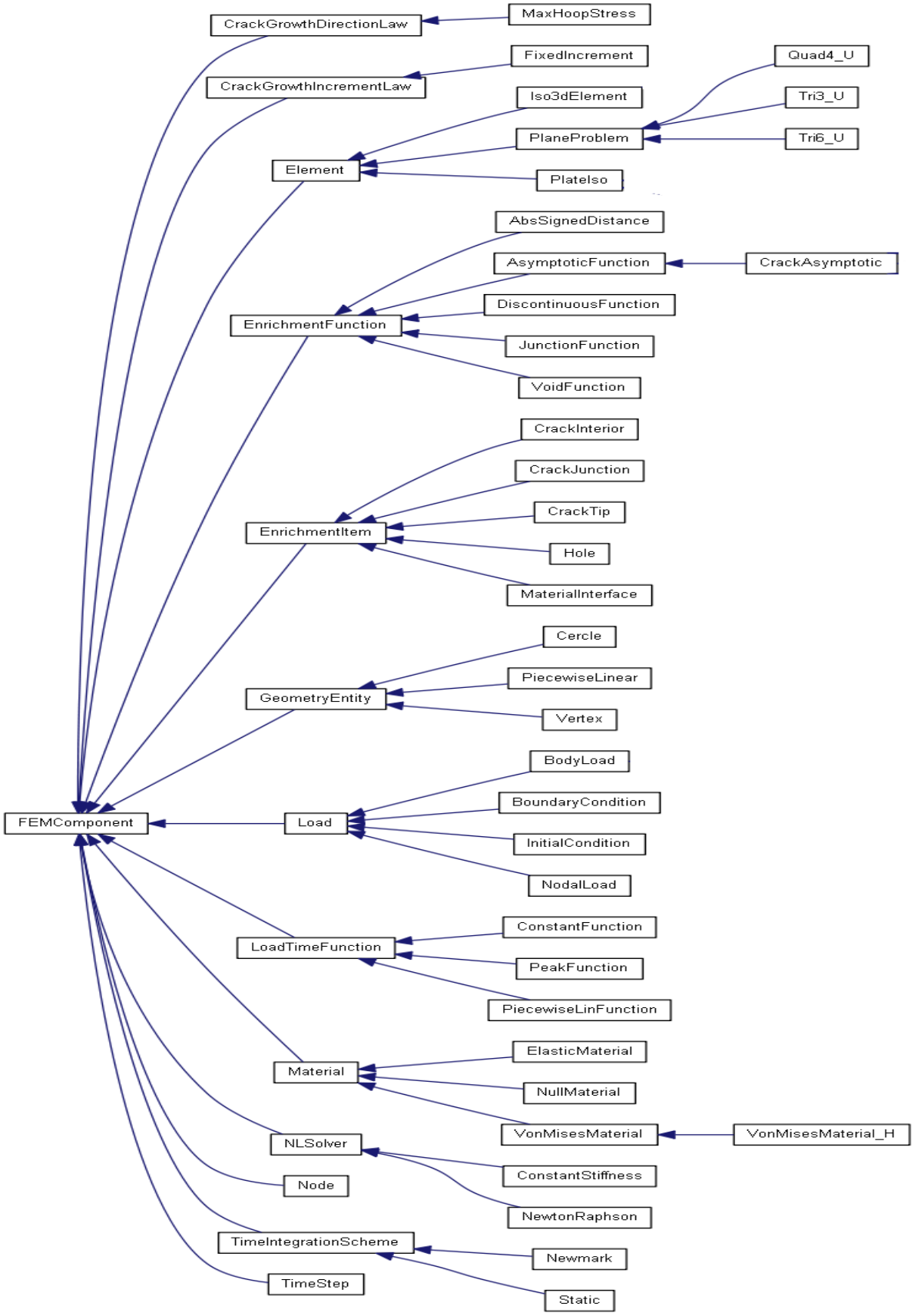



**Figure 3-1** Organigramme d'héritage de la classe FEM Component

## 4.1. Classe Domain

La classe « **domain** » qui représente le domaine physique du problème des éléments finis étendus à résoudre, est une classe abstraite, la superclasse de toutes les classes, elle est l'interlocuteur principal pour l'utilisateur par son interdépendance avec toutes les classes nécessaire à la simulation : **Node**, **Element**, **Material**, **Loads**, **EnrichmentItem**, **GeometryEntity**,…

C'est une classe centrale du solveur **FEM/XFEM**, qui joue un rôle fondamentale dans la Gestion du maillage (nœuds, éléments), et des entités **XFEM** (enrichissements, fissures, fonctions d'enrichissement), ainsi que sa responsabilité de l'instanciation, la résolution, l'assemblage et le post traitement du problème

Cette centralisation d'état est une approche très classique en éléments finis, ce qui permet d'avoir un point de contact unique où toutes les données et tous les processus de haut niveau sont coordonnés.

La classe « **domain** » utilise trois flux principaux, "**inputStream**", "**problem solving**" et "**outputStream**" pour bien déterminer le problème à étudier.

Dont le flux "**inputStream**" est le gestionnaire des entrées depuis le fichier « s00 » en créant un objet « **FileReader** » s'il n'existe pas. Ce flux qui se base principalement sur la méthode « **instanciateYoueself**() » , déclenche la lecture du fichier d'entrée en initialisant toute les listes des objets du problème étudier tel que **nodeList**, **elementList**, **materialList**, **enrichmentItemList**, **enrichmentFunctionList**, **geoEntityList**…..,

```
private :
  List*       elementList ;
  List*       nodeList ;
  List*       materialList ;
  List*       loadList ;
  List*       enrichmentFunctionList ;
  List*       enrichmentItemList
  List*       geoEntityList ;
```

```
List*        loadTimeFunctionList ;
```

la méthode « **instanciateYoueself**() » fait appel à de nombreuses méthodes de lecture du fichier d'entrée notamment les méthodes **'giveElement'**, **'giveLinearSystem'**, **'giveLoad'**, **'giveMaterial'**, **'giveNode'**, **giveFieldType'**, ..... etc)

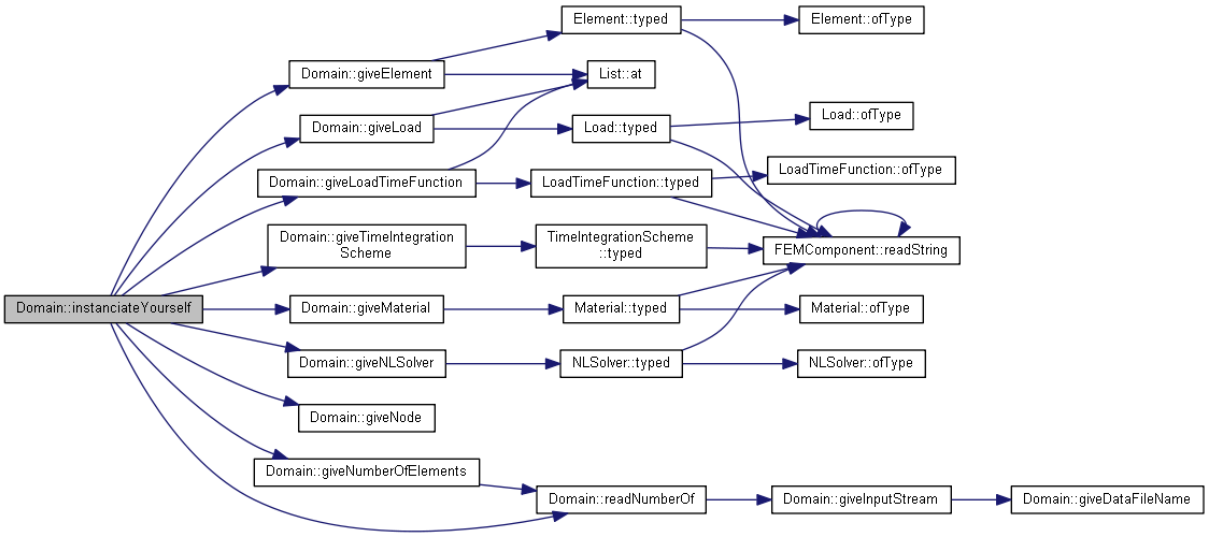



**Figure 3-2** Schéma d'appel de la méthode instanciateYourself

Tandis que le flux "**problem solving**" dépend principalement sur une méthode nommée **solveFractureMechanicsProblemAt(TimeStep)**, qui est considéré comme le cœur algorithmique du solveur **XFEM**, ceci est dû à son intégrité étroite à la structure de la classe « domain », ainsi que sa responsabilité de la modification dynamique du modèle en fonction des fissures, et de le résolution numérique **XFEM** d'un problème de mécanique de la rupture pour un pas de temps donné

```
void Domain :: solveFractureMechanicsProblem()
{   isXFEM = true ;
  TimeStep* currentStep ;
  this -> giveTimeIntegrationScheme() ;
  while (currentStep = timeIntegrationScheme->giveNextStep())
          this ->solveFractureMechanicsProblemAt(currentStep);
}
```

Cette méthode appelle directement ou indirectement presque toutes les sous-systèmes de la classe « **domain** » tel que la géométrie « **geoEntity** » , les éléments « **Element** », solveur « **NLSolver** » , les enrichissement « EnrichmentItem », ainsi que les sorties « **outputStream** » et fichiers « .m ».

La méthode **« solveFractureMechanicsProblemAt(TimeStep)** » peut être divisée en deux grandes scénarios selon le numéro du pas de temps (StepN->giveNumber()) :

La première phase (stepN->giveNumber() ==1) juste après l'enrichissement des nœuds **XFEM** , se décompose en quatre étapes différentes, dont la première est l'étape de détection des éléments interagissant avec les pointes de fissures ou les éléments coupées par la fissure utilisant la méthode « **treatMeshGeoInteractionPhase1**() » et aussi la méthode équivalente « **treatMeshGeoInteractionPhase2**() » qui a le même rôle mais après la propagation de la fissure.

La deuxième étape ou on fait appel à la méthode « **treatEnrichment**() » qui applique les fonctions d'enrichissement ( Heaviside, branch Function, etc.) aux nœuds enrichis pour l'ensemble du domaine ou la méthode « treatEnrichmentForBranchedCracks() » qui est utilisé pour les problèmes de fissures ramifiées pour trouver les nœuds enrichis par la jonction J(x).

La troisième étape est la résolution des conflits d'enrichissement par l'appel à la méthode «**resolveConflictsInEnrichment**() »,qui corrige le conflit de géométrie locale, c-à-d déterminer les éléments interagissant avec les discontinuités ainsi d'aligner les pointes sur les bords des éléments et la fissure sur ces bords, afin d'éviter qu'un nœud soit enrichis deux fois par le même type d'enrichissement.

Cette méthode appel à la méthode « **isSplitElement**() » qui assurer que les éléments coupés par la fissure (split element) sont géométriquement validées pour valider le calcul de la triangulation de **Delaunay**

```
this->resolveConflictsInEnrichment() ;
  for(size_t  i = 0 ; i < numberOfElements  ; i++)
  {       Element*  e = this->giveElement(i+1);
          if(e->isSplitElement())
            e->resolveConflictsInEnrItems();       }
```

Finalement la réduction des dépendances linéaires est la dernière étape, là où la méthode « **resolveLinearDependencyForEnrichment() »** sera utile, dont son rôle est de supprimer de la liste des nœuds enrichis par la fonction Heaviside H(x) tous les nœuds ne satisfaisant pas aux critères d'inclusion de la zone

le troisième et dernier flux "Out**putStream**" est un gestionnaire des fichiers de sortie crées tel que « .m » , « .dat » , « .s01 » , « .str » , ou l'exportation des données vers Matlab pour le post-traitement dans ce flux est assuré par la méthode principale « terminate » qui à son tour fait appel à des différentes méthodes telle que: **ExportMeshToMatlab()** , **exportEnrichedNodesToMatlab()** , **exportGaussPointsToMatlab()** , …..etc ,

```
public :
…..
  void exportMatlabPlotCommandsUndeformedMesh(char*  filename);
  void exportMatlabPlotCommands(char*  filename);
  void exportMatlabHeader(char*  filename);
  void exportMeshToMatlab(char*  filename);
  void exportCrackGeoToMatlab(char*  filename);
  void exportEnrichedNodesToMatlab(char*  filename);
  void exportDeformedMeshToMatlab(char*  filename);
  void exportNodeResultsToMatlab(char*  filename);
  void exportElementResultsToMatlab(char*  filename);
  void exportGaussPointsToMatlab  (char* filename);
  void exportConnectivityToMatlab  (char* filename);
……
```

En particulier, le processus de résolution est déclenché immédiatement lorsque nous rencontrons le message "**solveYourself**",

```
void  Domain  :: solveYourself()
{    isFEM = true ;
```

```
TimeStep* currentStep ;
this -> giveTimeIntegrationScheme() ;
while (currentStep = timeIntegrationScheme->giveNextStep())
        this -> solveYourselfAt(currentStep) ; }
```

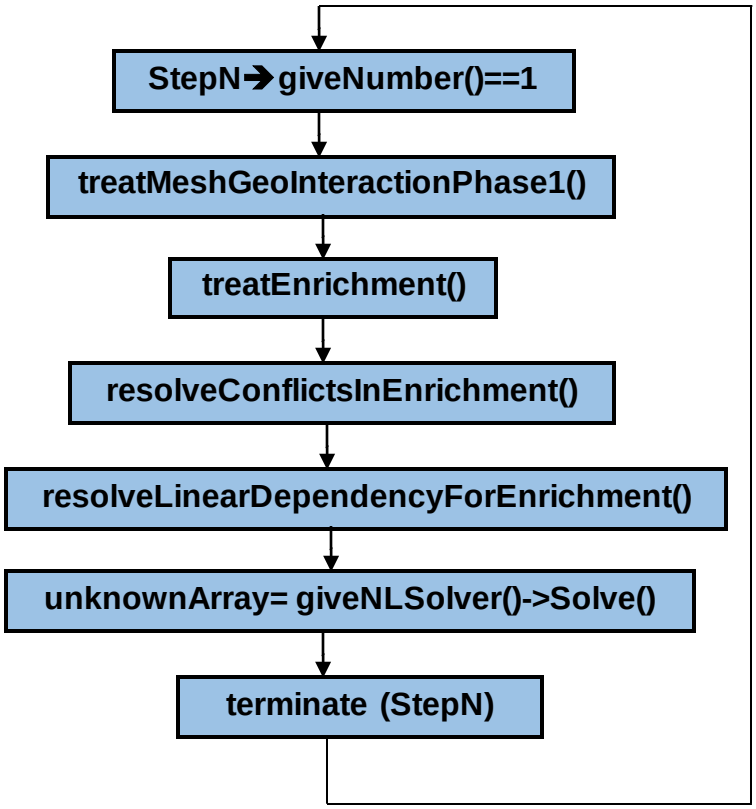



**Figure 3-3** Diagramme d'activité de la méthode solveFractureMechanicsProblemAt

## 4.2. Classe FEMComponent

la classe « **FEMComponent** » est une classe abstraire de base pour tous les composants physique dans un modèle élément finis étendus, c'est la superclasse de toutes les classes qui implémentent les composants d'un maillage : éléments, nœuds, pas de temps, matériaux, charges , fonctions charge-temps, les nœuds enrichis , les fonctions d'enrichissement…...etc.

Cette classe sert d'interface entre la représentation d'objet du modèle et les données d'entrées (fichier texte « **.inp** »). Elle fournit des méthodes tel que « **read** (char) », « **readInteger** (char data, int i) », « **readNumberOf** (char K) », « **readString** (char data, int i, char s) », …… ainsi que des fonctions comme « **isNumber** » et « **isInteger** » pour lire, interpréter et valider des données à chaque composant du modèle **XFEM**.

L'encapsulation et l'abstraction se manifestent clairement dans La classe « **FEMComponent** » par le passage obligatoire par la méthode de la classe « **domain** » (**domain->giveInputStream()**) pour accéder au fichier d'entrée, ce qui permet de séparer la logique de gestion des données du logique métier des composants **XFEM**. Ainsi que la méthode (**giveClassName**) ou (**giveKeyWord**) permet de gérer l'héritage et la polymorphie (exemple un matériau VonMises est traité comme « Material » dans le fichier de donnée.

A partir du fichier de donnée, la classe « **FEMComponent** » définit et utilise « **number »** et « domain » comme des attributs communs à toutes les classes de modèle étudié. Ces attributs sont généralement déclarés conventionnellement en C++ d'orientée objet sous forme protégés (protected) ou privés (private) dans cette classe, afin d'être accessibles par l'héritage pour toutes les sous classes :

```
{      protected :
     int      number ;
     Domain*   domain ;
//…..   } ;
```

L'attribut « **number »** à le rôle d'un identifiant unique pour chaque composant **XFEM** (élément, matériau, nœud, etc…), il est utilisé pour associer les données du fichier d'entrée à chaque composant lors de la lecture des données, par exemple en associant la ligne ou le bloc des données à cet composant dans le fichier d'entrée.

L'attribut **« domain »** son rôle principale est d'accéder au flux de données et d'assurer au composant de communiquer avec le modèle global (référence au modèle global), par exemple, un élément qui veut accéder à son matériau.

```
itos (number,n) ;
domain -> giveInputStream() -> read(keyword, n, data, i, value) ;
```

## 4.3.Classe Element

La classe « **Element** » est la classe la plus importante de la modélisation numérique du projet « **OpenXFEM++** », elle représente un élément finis dans un modèle de calcul par éléments finis étendue, elle permet de discrétiser un domaine continu en sous domaine plus simple (élément) pour résoudre des équations différentielles partielles.

Les responsabilités principales de la class « **Element** » sont principalement incarnés dans le stockage des propriétés topologiques et physiques de l'élément (matériau, nœud, charge, etc…), la réalisation des calculs locaux (matrice de rigidité, matrice de masse, le vecteur des forces internes, contraintes, déformations, etc…) ainsi que la participation à la résolution globale du système. En assurant l'interaction avec le reste du modèle via le domain globale « domain ».

Conformément aux conventions de la programmation orientée objet en C++, la class « **Element** » fait partie de ce paradigme, notamment en ce qui concerne l'héritage et l'encapsulation. Elle est conçue pour hériter la classe « FEMComponent », intégrant ainsi les deux des attributs fondamentales « number » et « domain », ce qui permettre à chaque d'extraire et lire ses propre données à partir du fichier d'entrée et d'interagir avec le reste du modèle. Quant à l'encapsulation, toutes les opérations de lecture, d'accès aux autre objets (matériaux, nœuds, charges, etc…) passent par des méthodes de la classe ou par le domaine (jamais directement par les données brutes)

La class « **Element** » referme diverses méthodes et fonctions pour la résolution globale du système **XFEM**, nous citerons certains d'entre-elles :

### 4.3.1. ComputeTangentStiffnessMatrix()

L'objectif général de cette méthode est le calcul de la matrice de rigidité tangente d'un élément fini, qui est utilisé dans les systèmes linéaires et non-linéaires du solveur. Cette méthode donne lieu à une résolution numérique de la formulation variationnelle du problème mécanique :

$$K_e = \int_{\Omega} B^T .D.B.dV = \sum_{gp} B^T .D.B.\underbrace{w_{gp}.\det(J)}_{dV} \qquad \textbf{(3-1)}$$

Ou

$B$ : Matrice de dérivées des fonctions de forme.

$D$ : Matrice constitutive.

$dV$ : Volume locale autour d'un point d'intégration

La matrice de rigidité reliée le déplacement nodal aux efforts internes, ou elle est utilisée dans le système global :

$$K_e u = f \qquad \textbf{(3-2)}$$

Cela implique que la **« ComputeTangentStiffnessMatrix() »** doit faire appel à d'autres méthodes provenant de divers classes du code notamment :

a) **ComputeConstitutiveMatrix()** de la classe « Material » , renvoie la matrice constitutive « D » qui décrit le comportement mécanique locale du matériau à chaque point de gauss donné (élastique, plastique , plan contrainte/déformation, etc…)

$$D = mat->ComputeConstitutiveMatrix(gp, this)$$

b) **PlusProduct (b,db,dV)** de la classe « Floatmatrix » et qui a le rôle de calculer et d'intégrer au récepteur le produit matricielle pondéré

c) **ComputeBmatrixAt( )** méthode qui construit la matrice des dérivées des fonctions de forme (matrice de déformation) « B » à chaque point de gauss $\frac{\partial N_i}{\partial x}, \frac{\partial N_i}{\partial y}$ , avec l'ajout des colonnes supplémentaire selon les fonctions d'enrichissement existant (fonction Heaviside ou fonction de singularité)

$$B = this->ComputeBmatrixAt(gp)$$

d) **computeVolumeAround( )** méthode virtuelle de la classe « Element » son rôle est le calcul du volume élémentaire local autour d'un point d'intégration, le volume arround est le poids d'intégration local donné par :

$$dV = this->computeVolumeAround(gp);$$

$$volume\ around(gp) = poids(gp).\det(J(gp))$$

Ou

*poids* : fourni par la règle de quadrature de Gauss.

$\det(J)$ : Déterminant du Jacobian de transformation (passage de l'espace de référence au domaine réel de l'élément.

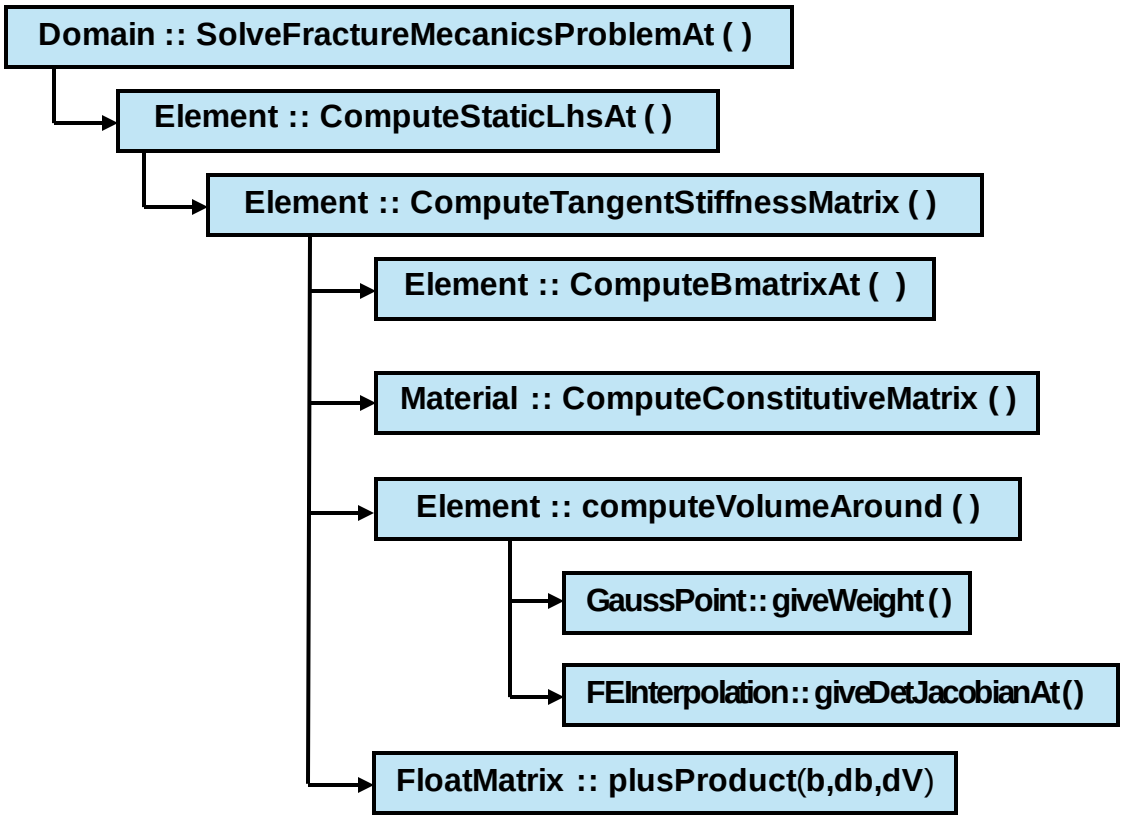



**Figure 3-4** Graphique d'appel de la méthode ComputeTangentStiffnessMatrix

### 4.3.2. IsEnriched() :

Si l'élément est coupé, touché par une fissure ou une interface, ou même contient la pointe de la fissure, le Jacobian peut être affecté et modifié, ceci est dû au fait que l'élément se subdivise localement en sous domaines (triangles dans l'élément coupé), ce qui rend nécessaire une intégration plus complexe.

Il est donc impératif d'invoquer la méthode booléen (bool Element :: **isEnriched**()) pour vérifier si un des nœuds de l'élément est enrichis par des fonctions d'enrichissement (Heaviside ou singularité), ce qui indique que ce nœud contient des degrés de libertés additionnels

Cette fonction « **isEnriched**() » constitue la passerelle entre l'interpolation effectuée par la méthode des éléments finis classique (FEM) et celle réalisé grâce à la méthode des éléments finis étendus (**XFEM**), en désignant ici les éléments finis enrichis.

Pour adapter le traitement de l'élément enrichis par rapport au standard FEM, elle est appelée dans des méthodes et fonctions critiques tels que :

- La fonction « **ComputeTangentStiffnessMatrix()** » qui assemble la matrice de raideur enrichis en ajoutant des blocs supplémentaires

$$K_e = \begin{bmatrix} K_{uu} & K_{ua} \\ K_{au} & K_{aa} \end{bmatrix} \qquad \textbf{(3-3)}$$

- La fonction « **ComputeBmatrixAt ( )** » qui ajoute des dérivés des fonctions de formes enrichis dans la matrice **B**

### 4.3.3. Les fonctions « assembleLhsAt( ) » et « assembleRhsAt( ) »

Ces fonctions intègrent respectivement les lignes et les colonnes enrichis dans l'assemblage de la matrice globale, ainsi que les termes enrichis dans le vecteur force

Cela implique l'appel des fonctions supplémentaires, notamment la fonction **Element:: giveXFEInterpolation()** qui renvoie l'interpolation finie utilisée pour approximer les fonctions discontinues( interpolation enrichie)

```
FloatMatrix* Element :: ComputeTangentStiffnessMatrix()
{
GaussPoint          *gp;
 FloatMatrix       *b,*db,*d;
 double            dV;
 //test ConstantStiffness?
 char nlSolverClassName[32] ;
 NLSolver* nlSolver     = this->domain->giveNLSolver();
 nlSolver  -> giveClassName(nlSolverClassName) ;
 if (stiffnessMatrix) {
        delete stiffnessMatrix;
 }
 stiffnessMatrix  = new FloatMatrix();
 Material *mat  = this->giveMaterial();
```

```
  for (size_t i = 0 ; i < this->giveNumberOfGaussPoints() ; i++)
  {
        gp = gaussPointArray[i];
        b  = this->ComputeBmatrixAt(gp);
        // compute D matrix
        if(! strcmp(nlSolverClassName,"ConstantStiffness"))
              d = this->giveConstitutiveMatrix()->GiveCopy();
        else
              d  = mat->ComputeConstitutiveMatrix(gp,this);
//d->printYourself();
        dV = this->computeVolumeAround(gp);
        db = d->Times(b);
        stiffnessMatrix->plusProduct(b,db,dV);
        delete d;
        delete db;
        delete b;
  }
  stiffnessMatrix->symmetrized();
  return stiffnessMatrix->GiveCopy();
}
```

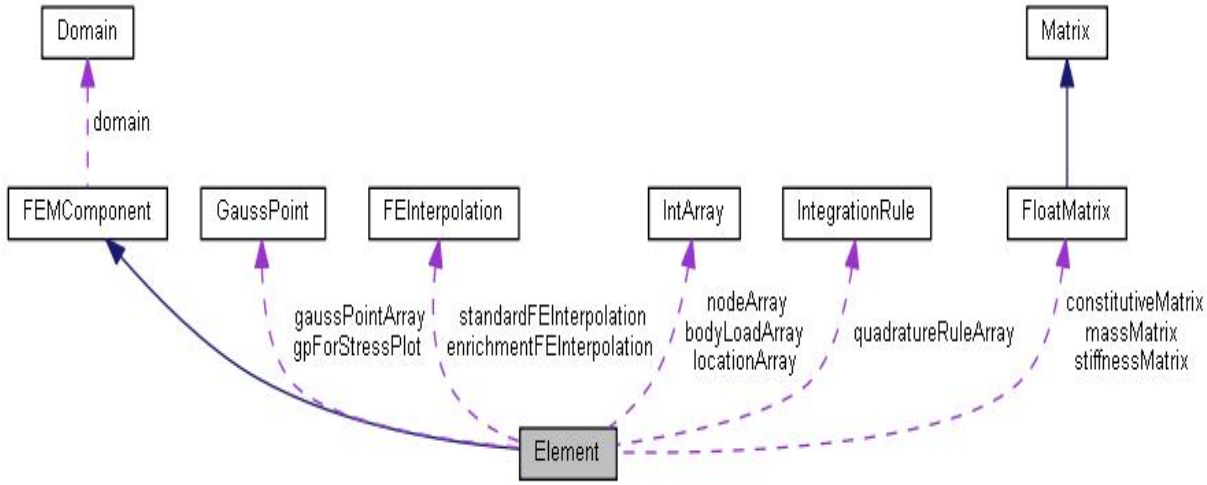



**Figure 3-5** Diagramme de collaboration pour la classe Elément

## 4.4. Classe Gausspnt

Cette classe joue un rôle fondamental dans la méthode des éléments finis étendus **XFEM**, elle encapsule non seulement les coordonnées et poids des points de gauss utilisés pour évaluer les intégrales sur les éléments en visant l'évaluation de la matrice de rigidité, mais aussi sont état mécanique (contraintes, déformations, déplacements, …). Elle constitue donc un nœud central entre la formulation mathématique de l'intégration numérique et la gestion à travers la programmation orientée objet.

Cette méthode consiste à initialiser pour chaque point de Gauss par le biais de son constructeur l'élément auquel il appartient (**element**), un identifiant local (number), ses coordonnées dans l'espace (**coordinates**) et son poids d'intégration (**weight**), ainsi que les vecteurs mécaniques associés (contraintes, déformations,….)

Elle repose sur une méthode principale **updateYourselfForXFEM()**,dont l'objectif est d'illustrer l'intégration avec l'enrichissement spécifiques liée à la fissuration. Elle montrant la gestion des états mécaniques internes (rollback des vecteurs mécaniques via les méthodes **letStrainVectorBe()** et **letStressVectorBe()**,ainsi que la remise à zéro entre itérations).

Dans cette classe, les concepts de la programmation orientée objet appliqués à la mécanique numérique sont démontrés de manière exemplaire. L'encapsulation est mise en avant lorsque la classe regroupe toutes les informations nécessaires pour le calcul local au point d'intégration telles que les coordonnées, poids d'intégration, contraintes, déformations,…etc.

En matière de polymorphisme, nous constatons que le point de Gauss ajuste son comportement selon le type d'analyse ou d'élément, par exemple la méthode **computestress()** est employée pour les nœuds non enrichis en se basant uniquement à la loi de **Hooke**. Pour les nœuds enrichis, cette méthode prend également en compte l'application de la loi de **Hooke**, et peut associer les fonctions d'enrichissement Heaviside et Williams selon le besoin.

## 4.5. Classe GeometryEntity

C'est une classe qui symbolise les entités géométriques abstraites interagissant avec les composants du maillage et déterminant la géométrie de la fissure. Elle repose sur :

**a)** Un constructeur polymorphe (**ofType**) qui offre la possibilité d'instancier dynamiquement les objets géométriques sous forme d'objets dérivés : **Vertex**, **PiecewiseLinear** et **Circle**,

```
GeometryEntity*  GeometryEntity :: ofType (char* aClass)
{
  GeometryEntity *newGeoEntity ;
  if (! strncmp(aClass,"PiecewiseLinear",5))
          newGeoEntity = new PiecewiseLinear(number,domain) ;
  else if (! strcmp(aClass,"Vertex"))
          newGeoEntity = new Vertex(number,domain) ;
  else if (! strcmp(aClass,"Circle"))
          newGeoEntity = new Cercle(number,domain) ;
}
```

**b)** Une méthode de description géométrique **giveMyGeoDescription()** qui a pour mission de déterminer l'option de représentation géométrique et mathématique des discontinuités en consultant le fichier d'entrée, soit une représentation standard (**StandardDescription** ) permettant d'identifier les nœuds enrichis via la méthode **StandardDescription()** de la classe **standarddescription**, ou une représentation par la méthode de l'ensemble de niveaux appliquée aux nœuds ( Level Set) via la méthode **LevelSetDescription()** de la classe **LevelSetDescription** destinée à représenter l'évolution des fissures en **XFEM**.

**c)** Des méthodes de calcul géométriques telles que la méthode booléenne **interactsWith()** initialisée dans la classe **GeometryDescription** , pour évaluer si une

entité géométrique interagit avec un élément finis donné, ainsi que la méthode **intersection**() qui est également présente pour déterminer les points d'intersection entre un entité géométrique et un élément finis, et stocke ces données dans un vecteur (Mu::Point). la gestion interne de ces intersections témoigne l'encapsulation adoptée par cette classe

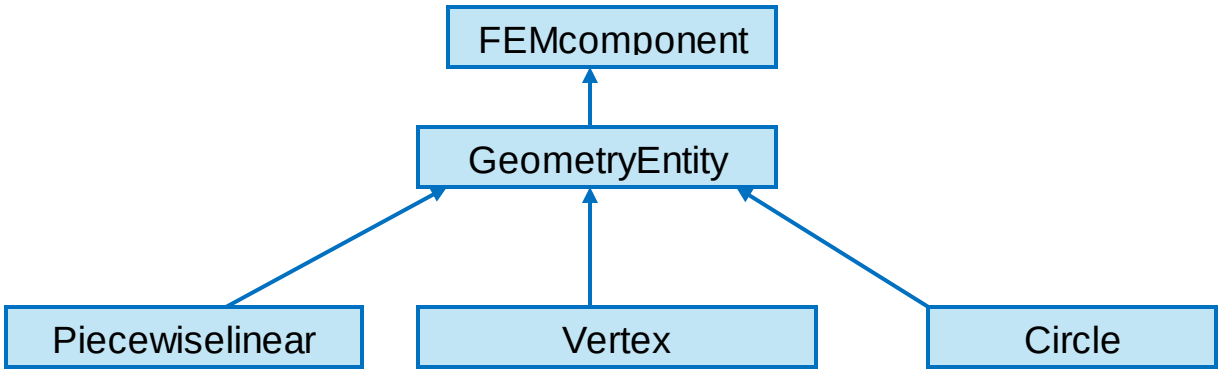



**Figure 3-6** Organigramme d'héritage de la classe GeometryEntity

### 4.5.1. Piecewiselinear

est une classe sous-classe de la classe mère **GeometryEntity,** qui symbolise une entité géométrique fondamentale dans l'architecture d'**OpenXFEM++**. Elle est définie par une série de segment connectée (polyline), généralement utilisée pour modéliser des fissures linéaire. Elle a un double rôle: d'une part, la représentation géométrique à travers le stockage et la manipulation des sommets (vertices) et segment constituant les polyline, et d'autre part, les capacités analytiques en proposant des fonctions de calcul géométriques ( distances, orientations, intersections,…).

Cette sous-classe comprend plusieurs méthodes regroupées comme suit :

**a) La méthode getListOfVertices()** (méthode de gestion des sommets et segments) qui lit la liste des sommets à partir du fichier d'entrée via la fonction **readInteger()** de la classe **FEMComponent** et construire la liste (**vertexList**) des points, ou chaque sommet est associé à un « vertex » .

**b) La méthode FindSegmentClosestTo()** qui détermine le segment le plus proche d'un point donné , supposons un point « P », le segment le plus proche est déterminer

en appliquons la formule de projection orthogonale de « P » sur la droite AB comme suit :

$$t = \left[(P-A).(B-A)\right] / \left\|B-A\right\|^2 \qquad \textbf{(3-4)}$$

- Si (t < 0) : le point le plus proche est A
- Si (t > 0) : le point le plus proche est B
- Si ($0 \leq t \leq 1$) : le point le plus proche est sur le segment

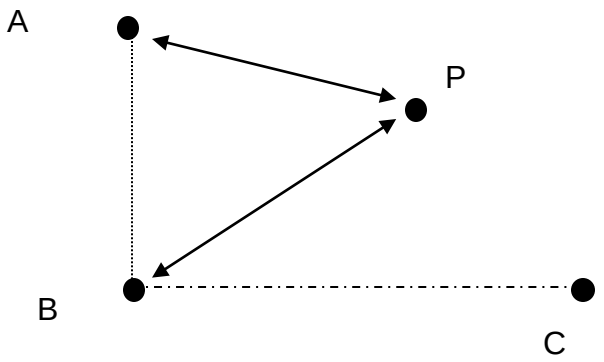



**Figure 3-7** Schéma de détermine d'un segment

**c) La méthode computeSignedDistanceOfPoint()** qui est chargée du calcul de la distance signée entre un point et une polyline. considérons un segment (AB), ou le vecteur associé est $v(x_2-x_1, y_2-y_1)$, et un point (P), qui correspond le vecteur (AP) représenté par $w(x_P-x_1, y_P-y_1)$,

A la Suite du calcul vectoriel effectué, on peut identifier trois scénarios géométrique pour déterminer la distance Euclidienne :

- la projection orthogonale de P sur la droite AB se trouve avant le point A, à condition que **(v**.w) $\leq 0$ , avec $v.w = \left\|v\right\|.\left\|w\right\|.\cos\theta$ , ou $\theta$ est l'angle entre AP et AB, (lorsque $\cos\theta \leq 0$ , alors $\theta \geq 90^o$ , donc la projection est en dehors du segment)
- la projection orthogonale de P sur la droite AB excède le point B, à condition que **(v.v)** ≤ **v.w** : ( $t = \left\|v\right\|.\left\|w\right\| / \left\|v\right\|^2$ ,c-à-d t >1)
- le cas final concerne la projection orthogonale de P sur le segment AB, ( $t = \left\|v\right\|.\left\|w\right\| / \left\|v\right\|^2$ ,pour vérifier cette condition , il faut que t ≤1) .

```
 Mu::Point  v(x2 - x1, y2 - y1);
 Mu::Point  w(coord->x - x1, coord->y - y1);
 First case:
double  c1 = w*v;
 if ( c1 <= 0 )
 return (w.norm())*sign;
second case:
 double  c2 = v*v;
     Mu::Point  w1(coord->x - x2, coord->y - y2) ;
      if ( c2 <= c1 )
              return (w1.norm())*sign;
third case
double  b = c1 / c2;
     Mu::Point  Pb(x1 + b * v.x, y1 + b * v.y) ;
     double  d = dist(Pb, (*coord)) ;
      return d*sign;
```

**d) La méthode givePositionComparedTo()** qui implémente le test d'orientation afin de définir la position relative d'un point par rapport à une polyline, ce qui simplifie la détection de côté par rapport à une fissure.

Cette classe implémente ce qui suit, Pour un segment (AB) et un point (P) :

```
double delta ;
 if(x2  > x1)
   delta = (x1-coord->x)*(y2-coord->y)  - (x2-coord->x)*(y1-coord->y) ;
 else
   delta = (x2-coord->x)*(y1-coord->y)  - (x1-coord->x)*(y2-coord->y) ;
 if (delta > EPSILON)
         return 1;
 else if (delta < -EPSILON     return -1;
 else     return 0
```

Ce qui est l'interprétation de la formule suivante (lorsque $x_B > x_A$) :

$$\Delta = (x_A - x_P)(y_B - y_P) - (x_B - x_P)(y_A - y_P) \qquad \textbf{(3-5)}$$

- Si (D> 0) : le point P se situe en dessus du segment AB (orientation trigonométrique positive)
- Si (D < 0) : le point P se situe en dessous du segment AB
- Si (D =0) : le point P se trouve en alignement avec le segment AB

De manière similaire, cette classe gère également l'orientation gauche/droite.

**e) Les méthodes intersects(const Mu::Triangle* t) et intersects(const Mu::Segment* s)** qui ont pour fonction de déterminer si le polyline qui est un union de segment stockées dans le vecteur **vector<Mu::Segment*>** , coïncide avec une entité géométrique soit triangle ou segment successivement

**f) Les méthodes intersection(const Mu::Triangle* t) et intersection(const Mu::Segment* s) :** qui calculent et déterminent les points d'intersections d'un polyline avec un triangle ou un segment respectivement.

### 4.5.2. Vertex

La classe Vertex dérivée de la classe parente **GeometryEntity**, incarne aussi une autre entité géométrique essentielle dans l'architecture d'**OpenXFEM++**. Elle dispose de fonctionnalités pour gérer ses coordonnées, ses relations et ses interactions avec les segments ainsi que les différents types d'éléments finis .

Cette sous-classe est comprend plusieurs methods, parmi lesquelles :

**a) La méthode getCoordinates()**

récupère les coordonnées à partir du fichier d'entrée, la méthode **giveCoordinate**() a pour mission de collecter et stocker ces coordonnées dans un tableau FloatArray, tandis que la méthode **getCoordinates**() est chargée de la mise à jour de ces coordonnées lors de la propagation de la fissure .par conséquent . ainsi, au-delà de ces trois méthodes qui établissent le lien entre l'entrée de utilisateur et la représentation interne, elles garantissent également l'encapsulation et la robustesse du code.

**b) La méthode booléenne interactsWith()**

Issue de la classe mère **GeometryEntity,** détermine si le vertex est relié ou interagit avec un élément finis à travers un test d'orientation basé sur le signe du déterminant suivant :

$$\Delta = (x_1 - x_0).(y_2 - y_0) - (x_2 - x_0).(y_1 - y_0) \qquad \textbf{(3-6)}$$

Les quatre méthodes de détections et de calculs des intersections entre un vertex (généralement un bout de fissure, un point de branchement de fissure, ou un point d'intersection avec les frontières de domaine) et des éléments géométriques ( triangles et segments) constituent des dispositifs précieux pour la détection de la propagation de la fissure et l'intégration sur les éléments coupées, ou l'on distingue :

**c) Les méthodes de prédicat (méthodes booléennes) :**

qui fournis une réponse binaire rapide et optimisé à la question « y'a-t-il une intersection » soit entre le segment contenant le vertex et un triangle via la méthode **intersects(const Mu::Segment * s)**, soit entre le segment contenant le vertex et un autre segment via la méthode **intersects(const Mu::Segment * s)**

Les méthodes de calcul géométrique complet des points d'intersection entre le segment contenant le vertex soit avec un triangle via la méthode **:intersection(const Mu::Triangle* t)** , soit avec un autre segment via la méthode **intersection(const Mu::Segment* s)** . Ces deux méthodes de calcul **intersection()** appelles systématiquement les méthodes de prédicat **intersects()** pour valider la nécessité de calcul .

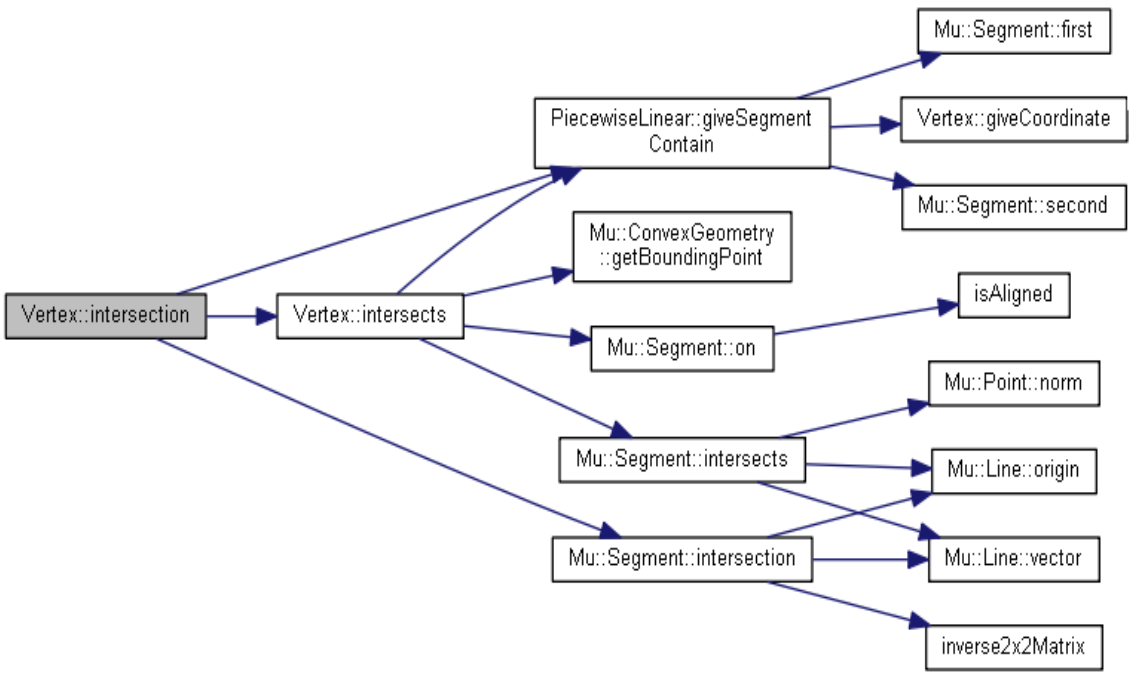



**Figure 3-8** Graphique d'appel de la méthode intersection

## 4.6. Classe EnrichmentItem

Cette classe illustre un objet d'enrichissement géométrique abstrait dans le contexte de la méthode des éléments finis étendus (**XFEM**), son objectif se résume à vérifier la définition, la détection et l'application des fonctions d'enrichissements pour modéliser les discontinuités forts et faibles dans un domaine donné.

Au-delà de fonctionner comme une classe de base polymorphe pour divers enrichissements concrets exposés comme sous forme d'objets, ou chacun ayant sa propre géométrie, la classe **EnrichmentItem** hérite aussi de la classe **FEMComponent** . cela lui permet d'avoir un numéro d'identification (**number**), d'accéder au domaine ( Domain* Domain ), et de lire les données grâce aux fonctions (read) , readInteger),….etc.

Pour garantir un fonctionnement optimale de la classe **EnrichmentItem**, celle-ci est dotée de plusieurs méthodes et fonctions comme :

### a) getGeometry() et giveMyGeo() :

Ces méthodes ont le rôle de collecter et d'associer l'objet géométrique à partir du fichier des données ( par le biais d'un attribut géométrique) , puis elles sollicitent le domaine (**domain -> giveGeoEntity(**geometryID)) pour crée une instance d'un objet **GeometryEntity.**

```
void EnrichmentItem::getGeometry()
{   if (geometryID  == 0)
          geometryID  = this  -> readInteger("geometry")  ;
  myGeometry  = domain  -> giveGeoEntity(geometryID)  ;  }
```

Ensuite, elle appelle la méthode **interactsWith**() pour tester si un élément finis est coupé ou infecté par la géométrie (par exemple coupé par une fissure) (**myGeometry->interactsWith(**)).cette dernière return les éléments qui doivent être enrichis, et les stockées dans des listes par la méthode **setListOfInteractedElements()**.

la **classe EnrichmentItem** s'appuie fortement sur la méthode de création (factory) nommée **ofType()** qui est utilisé lors de la lecture de fichier d'entrée pour générer les objets **XFEM** appropriés.

Cette dernière inclut quatre objets comme **CrackInterior**, **CrackTip**, **Hole** et **MaterialInterface**. Ces objets sont instanciés de façon dynamique en tant que sous-classe concrète.

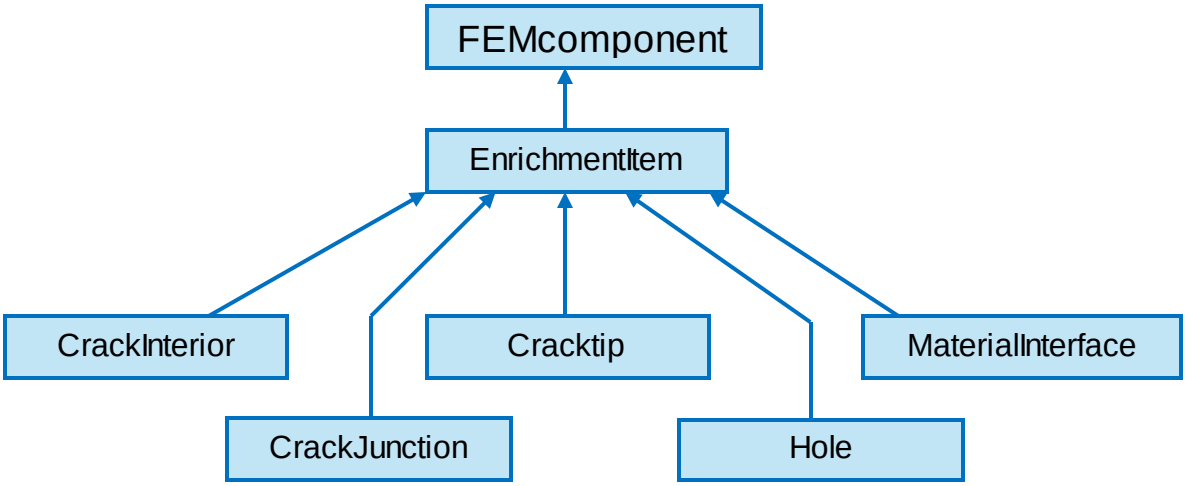



**Figure 3-9 Organigramme d'héritage de la classe EnrichmentItem**

```
EnrichmentItem*   EnrichmentItem :: ofType (char* aClass)
{
 EnrichmentItem*  newEnrichItem ;
 if (! Strcmp(aClass,"CrackInterior"))
        newEnrichItem  = new CrackInterior(number,domain) ;
 else if (! Strcmp(aClass,"CrackTip"))
        newEnrichItem  = new CrackTip(number,domain) ;
 else if (! Strcmp(aClass,"MaterialInterface"))
        newEnrichItem  = new MaterialInterface(number,domain) ;
 else if (! Strcmp(aClass,"Hole"))
        newEnrichItem  = new Hole(number,domain) ;
 }
```

Ce procédé démontre un modèle d'héritage vertical entre la classe **FEMComponent**, la classe **EnrichmentItem** (une base abstraite du **XFEM**) , ainsi que les classes **CrackInterior**, **CrackTip**,etc., ce qui symbolis des entités enrichissantes concrètes. Ce design respect le principe **Liskov de substitution,** selon lequel toutes les classes dérivées peuvent être manipulées via des pointeurs pointant vers la classe **EnrichmentItem,** ce qui favorise le polymorphisme, l'encapsulation et l'extensibilité de cette classe par l'intégration de nouveaux types de discontinuités dans le futur.

### b) GiveEnrFuncVector()

Selon la perspective de la programmation orientée objet, cette méthode renvoie un pointeur vers un vecteur qui contient des pointeurs vers des objets dérivés de la classe abstraite **enrichmentfunction.** Ces objets représentent l'ensemble des fonctions d'enrichissement liées à une entité géométrique enrichissante **(**objet de la classe **EnrichmentItem).** Cette méthode vise à initialiser de manière dynamique ces fonctions lors de leurs première invocation .Cela signifie qu'elle permet à chaque **EnrichmentItem** d'obtenir la listes des fonctions d'enrichissement qu'il doit appliquer

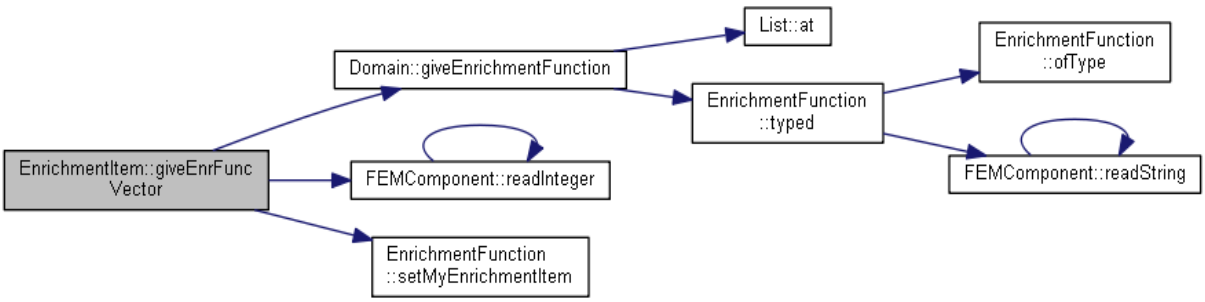



**Figure 3-10** Graphique d'appel de la méthode giveEnrFuncVector()

Cette méthode appelle plusieurs fonctions auxiliaires réparties dans différentes classes notamment la fonction **readInteger** de la classe **FEMComponent** pour lire le nombre des fonctions d'enrichissement dans le fichier des données, la fonction **giveEnrichmentFunction** pour obtenir un pointeur vers une fonction d'enrichissement spécifique, ainsi que la fonction **setMyEnrichmentItem** pour lier chaque fonction d'enrichissement à son objet parent **EnrichmentItem.**

Cette conception reflète un usage rigoureux de principe objet notamment la responsabilité unique (chaque classe a une tache bien définie)

```
vector<EnrichmentFunction*>*  EnrichmentItem::giveEnrFuncVector()
{
 EnrichmentFunction  *enrFn;
 if (!myEnrichFns)  // condition  vérifiant  si le vecteur  n'a pas encore été initialisé
                          // lazy  initialisation
 {
          myEnrichFns  = new vector<EnrichmentFunction*>  ;
          size_t  size  = this  -> readInteger("EnrichmentFuncs")  ;
          for  (size_t  i = 0 ; i < size  ; i++)
          {
   enrFn = domain->giveEnrichmentFunction(this->readInteger("EnrichmentFuncs",i+2));
                   enrFn->setMyEnrichmentItem(this)  ;
// Lie la fonction  avec l'objet  courant
 //via  enrFn->setMyEnrichmentItem(this)
                   myEnrichFns->push  back(enrFn) ;  // L'ajoute  au vecteur  myEnrichFns
          }
 }
 return  myEnrichFns  ;
}
```

### 4.6.1. CrackInterior:

Il s'agit d'une sous-classe de la classe abstraite **EnrichmentItem,** dont la principale tâche est de représenter une fissure interne dans un domaine simulé par éléments finis enrichis (**XFEM**) , elle offre la possibilité :

- d'identifier les éléments coupés par la fissure.
- d'enrichir les nœuds utilisant la fonction d'enrichissement Heaviside.

- De gérer l'évolution géométrique de la fissure
- De résoudre les dépendances linéaires….etc.

Cette sous-classe dispose des méthodes principales suivantes :

**a) Getmytips()** et **Givemytips() :**

sont deux méthodes publics interconnectées, leur rôle est de lire la liste des objets **enrichissants** identifiés du fichier des données grâce à la méthode hérite **readInteger**() de la classe **FEMComponent** et la méthode polymorphe **domain->giveEnrichmentItem(tip)**, cela indique le nombre des **CrackTip** qui sont associés à cette fissure. Ces objets sont explicitement convertis en **CrackTip** via l'appel de la méthode **setMyAssociatedCrackInterior**() de la classe **CrackTip**, puis stockés dans l'attribut **MyTips.** (le programmeur a utilisé l'hypothèse de type strict « **static_cast** » pour garantir que l'objet pointé est supposé être un **CrackTip**)

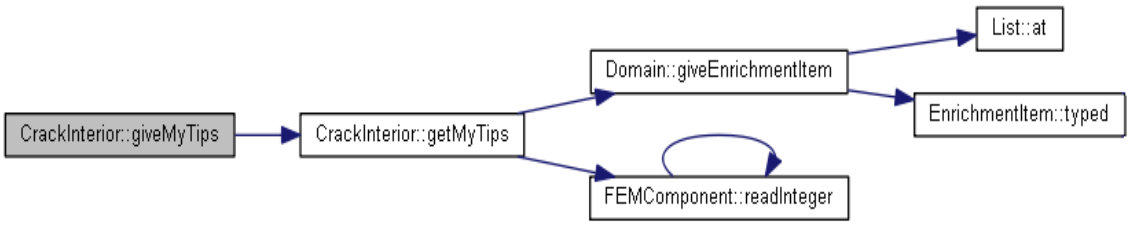



**Figure 3-11** Graphique d'appel de la méthode giveEnrFuncVector()

**b) TreatMeshGeoInteractionForMyTips() :**

cette méthode est indispensable dans la processus de traitement des fissures par **XFEM**, elle assure plusieurs taches notamment :

- Chercher l'élément contenant le Tip : Ou la méthode **giveElementsInteractWithMe()** de la classe **EnrichmentItem** est appelé pour renvois les éléments contenant le fond de la fissure , ensuite la méthode **giveMyTips**() initialise le vecteur **MyTips** déjà initialisé au sein de cette classe.

```
std::vector<Element*> *interactedElems = this->giveElementsInteractWithMe();
```

```
this->giveMyTips() ; // tips of the crack, maybe one or two.

Element *tipElem = NULL ;

CrackTip *tip ;
```

- b- Pour chaque Tip, nous cherchons l'élément qui le contient ou avec lequel il touche son bord . Nous utilisons la méthode **tip->interactsWith(*element)** de la classe **EnrichmentItem** pour tester si le Tip se trouve parmi les éléments déjà identifiés comme étant traversés par la fissure , si ce n'est pas le cas, nous procédons à une vérification de tous les éléments du domaine. L'utilisation de la méthode **IsOnEdge**() de la classe **Element** permet de vérifier que le fond de la fissure coïncide exactement avec un arrêt d'un élément. Si confirmé , la recherche prend fin et par la suite un objet **Mu::Point** qui contient les coordonnées du Tip seront créées de manière dynamique .ce dernier sera marqué comme étant un arrêt par la méthode **tip->setIsOnElementEdge**() de la classe **CrackTip**. Ce Tip sera enrichis par des fonctions d'enrichissement asymptotiques via la méthode **isEnrichedWith**(tip) de la classe **Element**. A la fin, la procédure sera lancée de nouveau pour un autre Tip s'il existe.

```
for(size_t i = 0 ; i < myTips.size() ; i++ )
 {      tip = myTips[i] ;
        for(size_t j = 0 ; j < interactedElems->size() ; j++ )
        {       if( tip->interactsWith((*interactedElems)[j]) )
                {       std::cout << " The tip is within element :) " << std::endl ;
                 tipElem = (*interactedElems)[j] ;
                 j = interactedElems->size() ;
                }
        }
        // not yet find out tip element, do more tests
        if(tipElem == NULL)
```

```
{for(size_t  k = 0 ; k < this->domain->giveNumberOfElements()  ; k++)
        {       Element  *e = this->domain->giveElement(k+1)  ;
         if(tip->interactsWith(e))
         {      tipElem  = e ;
                k = this->domain->giveNumberOfElements()  ;
         }
        }
}
// If the tips touch element edges then need check more ...
if(tipElem  == NULL)
{       std::cout  << " The tip touched element edge !!! " << std::endl  ;
        double  x = tip->giveTip()->giveCoordinate(1)  ;
        double  y = tip->giveTip()->giveCoordinate(2)  ;
        Mu::Point  *p = new Mu::Point(x,y)  ;
        for(size_t  k = 0 ; k < interactedElems->size()  ; k++ )
        {       if((*interactedElems)[k]->IsOnEdge(p))
         {
                std::cout  << " Ah, I found you !!! " << std::endl  ;
                tipElem  = (*interactedElems)[k]  ;
std::cout  << " Number of tip element  " << tipElem->giveNumber()  << std::endl ;
         k = interactedElems->size()  ;     // not to check more, Tri's help .
         }
        }
        tip->setIsOnElementEdge();
        bool found  ;
std::list<Element*>  *supportOfTipElem  = tipElem->giveNeighboringElements()  ;
```

```
                std::list<Element*>  :: iterator  j = supportOfTipElem->begin()  ;
                found  = false  ;
                while(  (!found)  && (j != supportOfTipElem->end())  )
                {          if((*j)->IsOnEdge(p))
                  {                tipElem  = *j ;
std::cout  << " Number of tip element   " << tipElem->giveNumber()  << std::endl  ;
                          found  = true ;
                  }
                  j++ ;
                }
                delete p ;
          }
          tipElem->isEnrichedWith(tip)  ;
          tip->setListOfInteractedElements(tipElem);
          tipElem  = NULL ;
  }
```

### c) GiveCrackLength() et computeCrackLength():

Ce sont des méthodes conçues pour calculer la longueur de la fissure en se basant sur les coordonnées des extrémités, puis stocker ces informations.

### d) ResolveLinearDependency() :

Cette méthode est élaborée pour détecter et corriger les dépendances linéaires peuvent survenir suite à l'enrichissement ( relations redondantes entre fonctions de forme enrichis) , ce qui rend le système linéaire singulier et instable d'un point de vue numérique . Elle ne concerne que les nœuds enrichis par la fonction d'enrichissement Heaviside dans les éléments que la fissure traverse. Cette méthode se manifeste comme suit :

- Retourne le vecteur des éléments géométriques affectées par la fissure utilisant la méthode **giveElementsInteractWithMe**() hérité de la classe **EnrichmentItem**

```
std::vector<Element*>*   elemList  = this->giveElementsInteractWithMe();
```

- Collecte des nœuds associés à ces éléments dans une liste **checkedNodes**, puis l'étrillage et la suppression des doublons dans les listes **sort**() et **unique**() , afin d'éviter les tests redondants. L'objet géométrique associé à la fissure sera récupéré via la méthode **giveMyGeo**() de la classe **GeometryEntity** ( souvent un objet **PiecewiseLinear** composé de segment). un objet **Mu::Point** qui contient les coordonnées des nœuds sera créé de manière dynamique, ce qui facilite le test de la position du point par rapport à la fissure via une boucle de la méthode **isStepEnriched**() qui à son tour de rôle dépend sur la méthode **givePositionComparedTo**() de la classe **GeometryEntity**. Cette dernière retourne la valeur « 1 » → lorsque le point se situe au-dessus de la fissure, la valeur « -1 » pour le cas contraire, et la valeur « 0 » qui indique que le point est sur la fissure. (Plus on se rapproche de la valeur « 0 » (ou le point n'est pas exactement sur la fissure), cela indique une situation géométrique instable):

```
void CrackInterior :: resolveLinearDependency()
{  std::vector<Element*>*   elemList  = this->giveElementsInteractWithMe();
 #ifdef
 std::cout<<  "Elements  split  by this CrackInterior  : " << std::endl  ;
 for(size_t  i = 0 ; i < elemList->size()  ; i++)
        std::cout  << (*elemList)[i]->giveNumber()  << " " ;
 std::cout  << std::endl  ;
#endif
 Element  *e  ;
```

```
std::list<Node*>  checkedNodes;
for(size_t  i = 0 ; i < elemList->size()  ; i++)
{       e = (*elemList)[i];
        for(size_t  j = 0 ; j < e->giveNumberOfNodes();  j++)
                checkedNodes.push_back(e->giveNode(j+1));
}
// get rid of repeated nodes ...
checkedNodes.sort();
checkedNodes. Unique();
GeometryEntity  *geoOfEnrItem  = this->giveMyGeo()  ;
if(isBranchedCrack  == false)
{       for(std::list<Node*>::iterator  it  =  checkedNodes.begin()  ;  it  !=
checkedNodes.end()  ; it++)
        {       if((*it)->isStepEnriched())
                {       Mu::Point  *p = (*it)->makePoint()  ;
                 int  check = geoOfEnrItem->givePositionComparedTo(p);
                 if(check  == 1 || check == -1)
                        (*it)  -> resolveLinearDependency(this);
                 delete  p ;
                }
        }
   }
}
```

### e) UpdateMyGeometry()

Cette méthode a pour mission de mettre à jour de la géométrie de la fissure de manière dynamique, en particulier lors de sa propagation, en se basant sur les nouvelles positions des extrémités (**Vertex**) et segments (**Segment**) de cette fissure . cela signifie l'initialisation des objets **MyTips**. La récupération de la nouvelle géométrie s'effectue via un pointeur myGeometry qui pointe vers une entité géométrique casté explicitement en **PiecewiseLinear** issue de la sous-classe **PiecewiseLinear**, sur l'hypothèse que **CrackInterior** utilise une géométrie polygonale . en présence d'une fissure interne avec deux CrackTip, il absolument nécessaire d'invoquer les méthodes **insertNewVertexInFront**(tip) et **insertNewSegmentInFront**(seg) pour instancier le fond et le segment de fissure gauche successivement, tandis que les méthodes **insertNewVertexAtBack** (tip) et **insertNewSegmentAtBack** (seg) sont utilisées pour générer le fond et le segment de fissure droite, .

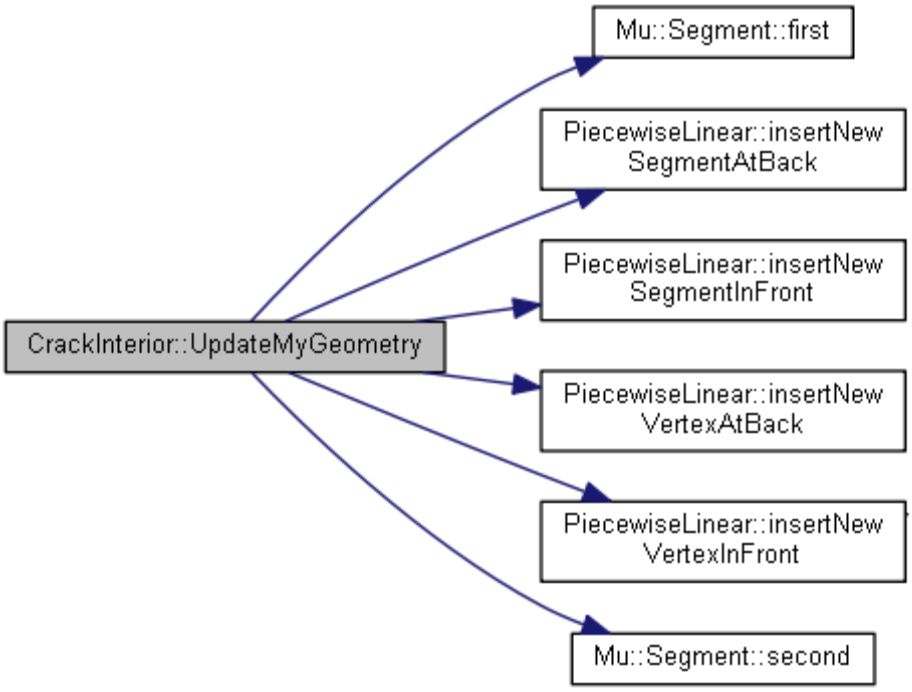



**Figure 3-12** Graphique d'appel de la méthode UpdateMyGeometry

### 4.6.2. CrackTip

Il s'agit d'une sous-classe de la classe abstraite **EnrichmentItem**, ce qui signifie qu'elle représente un objet d'enrichissement géométrique. Elle modélise l'extrémité d'un fissure (le Tip) dans une simulation de la mécanique de la rupture utilisant la méthode des éléments finis étendus (**XFEM**). Dans le code actuel, cette sous-classe joue un rôle crucial dans l'enrichissement local du champ de déplacement à l'aide des fonctions asymptotiques, le calcul du facteur d'intensité de contrainte, ainsi que la croissance de la fissure.

Cette sous-classe est centrale dans le traitement des singularités **XFEM**. Sa conception robuste qui respecte les principes de la programmation orientée objet repose sur la modularité et l'extensibilité. Elle incarnant une stratégie typique en matière d'objet dans les solveurs **XFEM** moderne, ou des opérations tel que la gestion de la géométrie, le traitement des singularités, le calcul des SIFs,…etc. sont des objets actifs capables de gérer leurs propres mise à jour.

Pour réaliser tous cela, la sous-classe CrackTip se base sur plusieurs méthodes, parmi lesquelles :

**a) ComputeLocalCoordOf(Mu::Point* p) :** L'objectif principal de cette méthode est de convertir les coordonnées globale (x,y) d'un point dans le domaine en coordonnées locale (x',y') aligné avec l'axe du segment de fissure en pointe , afin de permettre une représentation précise du champ mécanique (déplacement , contrainte).

Elle exploite la méthode **giveTipSegment**() qui est un membre de la même sous-classe **CrackTip** , et qui pointe sur l'objet géométrique **Mu::Segment** , par la suite elle interprète les coordonnées des deux extrémités du segment , ce qui simplifie le calcul de l'angle d'inclinaison ( en radian) du référentiel local par rapport à l'axe global x . cela facilite la transformation en coordonnées locales ( $x_{loc}$ est la direction du propagation de la fissure et $y_{loc}$ est la direction normale)

$$\begin{bmatrix} x_{loc} \\ y_{loc} \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{pmatrix} x - x_2 \\ y - y_2 \end{pmatrix} \qquad (3\text{-}7)$$

**b) ComputePolarCoordOf(Mu::Point *p) :**Cette méthode fait appel à la méthode précédente, et utilise les coordonnées locales (x',y') calculées et stockées dans

un tableau FloatArray pour extraire les coordonnées polaires $(r,\theta)$, puis retourne un pointeur vers un tableau contenant $(r)$ et $(\theta)$

Ou : $r = \sqrt{(x_{loc})^2 + (y_{loc})^2}$ → est la distance à l'origine locale

$\theta = arctan\left(\frac{x_{loc}}{y_{loc}}\right)$ → est l'angle entre l'axe de fissure et la position du pointe

**c) GiveRadiusOfDomainIntegration(),makeBall(),buildIntegrationDomain1() , buildIntegrationDomain2() :** Sont des méthodes utiles pour construire le domaine d'intégration , tel que **giveRadiusOfDomainIntegration(),** est une méthode qui fournis la taille d'un domaine circulaire (rayon d'intégration $r_d$) autour du fond de la fissure , connu sous le nom de zone d'intégration .elle est utilisée pour l'intégration des grandeurs énergétiques tel que l'énergie élastique et le facteur d'intensité de contrainte via l'intégrale de « J ».

L'identifiant de schéma d'enrichissement « **enrDetectorID »** sous forme entière et qui correspond à la valeur « **enrichScheme »** spécifiée dans le fichier d'entrée, permet au programme de déterminer quelle stratégie est adoptée pour le calcul du rayon d'intégration $r_d$ , ou peut ainsi identifier deux cas distincts :

- Cas schéma adaptative (**enrDetectorID** ==1), ou $r_d = fac.\sqrt{A_e}$ avec « $A_e$ » représente l'aire de l'élément qui englobe la pointe de fissure identifié par la méthode hérite **giveElementsInteractWithMe**() de la classe **EnrichmentItem** , tandis que « fac » est un facteur d'échelle empirique spécifiée dans le fichier d'entrée sous le nom de **domainIntFac** obtenue grâce à la méthode hérite **readInteger**() de la classe **FEMComponent**.
- Cas schéma fixe (**enrDetectorID** ==2), $r_d = valeur\ fixe$ , cette valeur tirée du fichier d'entrée sous le nom de **domainIntRad** est obtenue via à la méthode hérite **read**() de la classe **FEMComponent**

Suite à la détermination du rayon d'intégration rd, la méthode **makeBall**() automatise directement le domaine d'intégration centré sur la pointe de fissure, en créant géométriquement une bulle d'intégration circulaire. Cela facilite la sélection des éléments enrichis autour du tip, ce qui rend possible l'application de techniques d'intégration spécifiques pour l'évaluer les champs des déplacements et des contraintes ainsi que les facteurs d'intensités de contrainte.

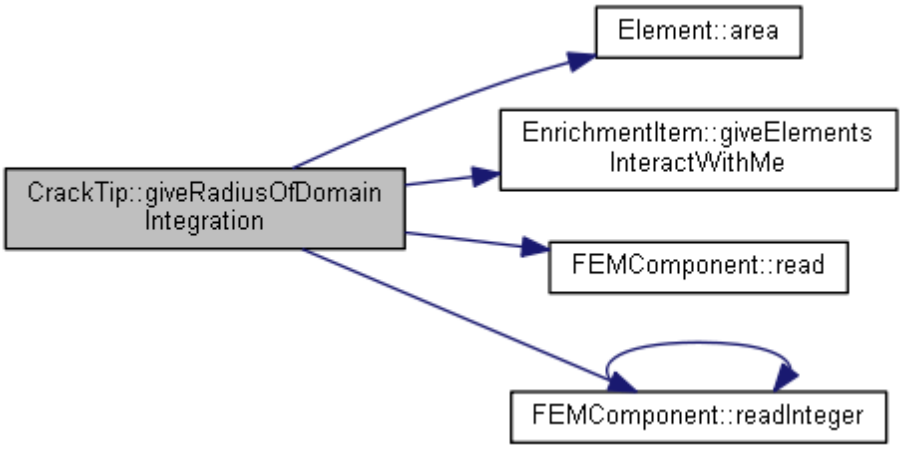



**Figure 3-13** Graphique d'appel de la méthode giveRadiusOfDomainIntegration

La méthode **makeBall()** créé de manière dynamique un objet de type **Mu::Circle (rad, center)** qui présente la zone d'influence locale du tip dans le maillage où **rad** définit son rayon et **center** indique les coordonnées du tip.

La méthode **buildIntegrationDomain1() :** exécute différentes tâches, elle collecte et stocke la liste qui contenant tous les éléments finis interagissant avec la bulle autour du tip grâce à une variable nommé **interactedElements** dans un vecteur de type **vector<Element*>** via à la méthode **giveElementsInteractWithMe**() qui est issue de la classe **EnrichmentItem**. cette dernière ne revoie qu'un seul élément **TipElem,** l'élément qui englobe le Tip , servant ainsi de point de départ pour l'identification des conflits géométriques.

La seconde tâche consiste à repérer les conflits géométriques c-à-d tous les éléments voisins qui se trouvent intégralement ou partiellement dans le domaine d'intégration via la méthode **conflicts(ball)** . cette dernière fait appel à la méthode **giveNeighboringElements()** qui proviennent toutes les deux de la classe **Element,**

alors que la méthode **buildIntegrationDomain2()** optimise La méthode **buildIntegrationDomain1()** est améliorée par l'intégration d'un filtre plus strict grâce à l'utilisation de la méthode booléen **intersects()** tirée de la classe **Element**. Ceci permet d'optimiser à la fois la précision numérique et l'efficacité computationnelle ( calcul du SIF). la méthode **intersects()** sert à vérifier si un élément géométrique est croise un cercle (Tip $(x_T, y_T)$ , R) préalablement définie , cette opération est réalisée en calculant la distance « $r_i$ » entre chaque nœud (xi , yi) de chaque élément du domaine ,

$$r_i = \sqrt{(x_i - x_T)^2 + (y_i - y_T)^2} \qquad \textbf{(3-8)}$$

Un vecteur **distanceVect** sera utilisé pour stocker les distances $r_i = x_i - R$ . Si cette distance est positive, cela signifie que le nœud se trouve en dehors du cercle, tandis qu'inversement, s'il est négatif, le nœud se situe à l'intérieur du cercle, ce qui suggère que le cercle coupe l'élément à un certain endroit

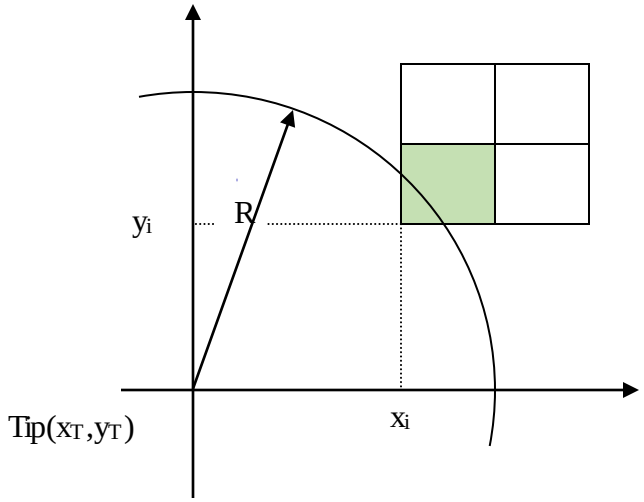



**Figure 3-14** Eléments intersectés par le cercle d'enrichissement

**d) ComputeSIFs(TimeStep* stepN)** et **computeK_eq(TimeStep* stepN) :** ces méthodes encapsulent le calcul des SIFs pour le mode I et II ainsi que le facteur d'intensité équivalent. Les facteurs d'intensité de contrainte KI et KII pour un pas de temps donné sont formulés comme suit :

$$K_i = A \,.\, I_i \quad , \quad K_{ii} = A \,.\, I_{ii} \qquad \textbf{(3-9)}$$

Ou :

*"A"* représente un coefficient matériau dépendant des propriétés mécaniques et des conditions de déformation (plane ou contrainte plane).

L'évaluation de ce coefficient est confiée à la méthode **computeSIFMultiplier()** , qui repose sur le type du domaine à un seul matériau homogène déjà enregistré dans **giveMatArray()** vecteur d'objets **matériel*** (notre scénario d'étude) , et qui fait référence à la méthode

**e) computeSIFMultiplierForOneMat().**

ceci fournit les valeurs du coefficient "*A*" en fonction de l'état de déformation spécifié:

$$\begin{cases} A = E/2 & \rightarrow\rightarrow Contra\,\text{int}\,e\ plane\ (plan\ stress) \\ A = E/2(1-\upsilon^2) & \rightarrow\rightarrow Déformation\ plane\ (plan\ strain) \end{cases} \quad \textbf{(3-10)}$$

Ou « E » module de Young et « n » coefficient de Poisson.

Alors que « $I_i$ », « $I_{ii}$ » sont les résultats de l'intégrale d'interaction pour chaque mode, calculées par la méthode **computeInteractionIntegral()**.cette méthode réalise numériquement l'intégrale d'interaction également appelée *M-integral*, qui est un outil efficace pour déterminer séparément les facteurs d'intensité de contrainte en mode I et II , en particulier dans le contexte de la méthode des éléments finis étendus **XFEM** .

```
std::valarray<double>  I = this->computeInteractionIntegral(stepN)
std::valarray<double>  I(0.,2) ;        // interaction integral I(1,2) of modes I and II
 Mu::Circle  *myBall  = this->makeBall();  // circle centered at tip, r = r_d
  td::list<Element*>  intDomain  = this->buildIntegrationDomain2();
```

**f) computeInteractionIntegral() :** Cette méthode agit comme un orchestre entre plusieurs méthodes de différentes classes du code, son rôle est de retourner un vecteur à deux valeurs visualisant l'intégrale d'interaction pour les modes I et II dans le cadre de la mécanique linéaire de rupture ($I=[I_1,I_2]$), lié à un pas de temps déterminé (stepN).c-à-d l'évaluation théorique une combinaison ( $I_1+I_2+W.jq/jx_1$ )pour chaque mode auxiliaire(.)$^{(aux)}$, issue de la forme faible du *J*-integral et interaction integral

Sachant que le concept de l'intégrale d'interaction joue un rôle fondamental dans la mécanique de la rupture. C'est une d'une extension du *J*-integral de Rice, qui permet de séparer les facteurs d'intensité de contrainte(SIFs) en mode I et en mode II pour une fissure situé dans un domaine quelconque.

Donc, l'idée centrale adopté consiste à superposer un champ auxiliaire (souvent une solution analytique reconnue, comme le champ de Westergaard) au champ réel (déterminé par la méthode des éléments finis) et d'utiliser la forme de *J*-integral pour ce champ superposé.

De ce fait, cette méthode commence par l'identification du domaine d'intégration par le biais de la méthode **buildIntegrationDomain2()** qui fait sélectionner les élément chevauchant avec le périmètre d'un cercle centré sur la pointe de la fissure, ayant un rayon définis au préalable ($r = r_d$ ) .cette dernière utilise deux méthodes de la classe **Element** : **conflicts**() et **intersects**() mentionnés précédemment.

```
std::valarray<double> I(0.,2) ;        // interaction integral I(1,2) of modes I and II
   Mu::Circle *myBall = this->makeBall(); // circle centered at tip, r = r_d
   std::list<Element*> intDomain = this->buildIntegrationDomain2();
//matrice de transformation
   FloatMatrix *r = new FloatMatrix(2,2);
   r->at(1,1) = cos(alpha) ; r->at(1,2) = sin(alpha) ;
         r->at(2,1) = -sin(alpha) ; r->at(2,2) = cos(alpha) ;
   rt = r -> GiveTransposition() ;
// initialisation des champs auxiliaires
   AuxiliaryField<Material,NullMaterial,PlaneStrain> *auxFieldHomo
      = new AuxiliaryField<Material,NullMaterial,PlaneStrain>();
```

Ensuite, la méthode **computeInteractionIntegral()** procède à l 'évaluation de champ réel qui exige l'identification de la matrice constitutive « D » associée aux conditions de déformation préétablies (plane ou contrainte plane) , grâce à la fonction **giveConstitutiveMatrix**() . de plus le vecteur de déplacement « U » pour chaque élément (*i) du domaine d'intégration qui est obtenue par la fonction **ComputeVectorOf**(), ainsi que la matrice des dérivées des fonctions de forme « N » par rapport au coordonnées globales (x,y).(ε=B.U pour les petites déformations) via la fonction **evaldNdx()** de la classe **FEInterpolation().**

L'étape suivante consiste à la détermination du gradient du déplacement réel (dU1dx),

$$\nabla u = \begin{bmatrix} \partial u_1/\partial x & \partial u_1/\partial y \\ \partial u_2/\partial x & \partial u_2/\partial y \end{bmatrix} \qquad \textbf{(3-11)}$$

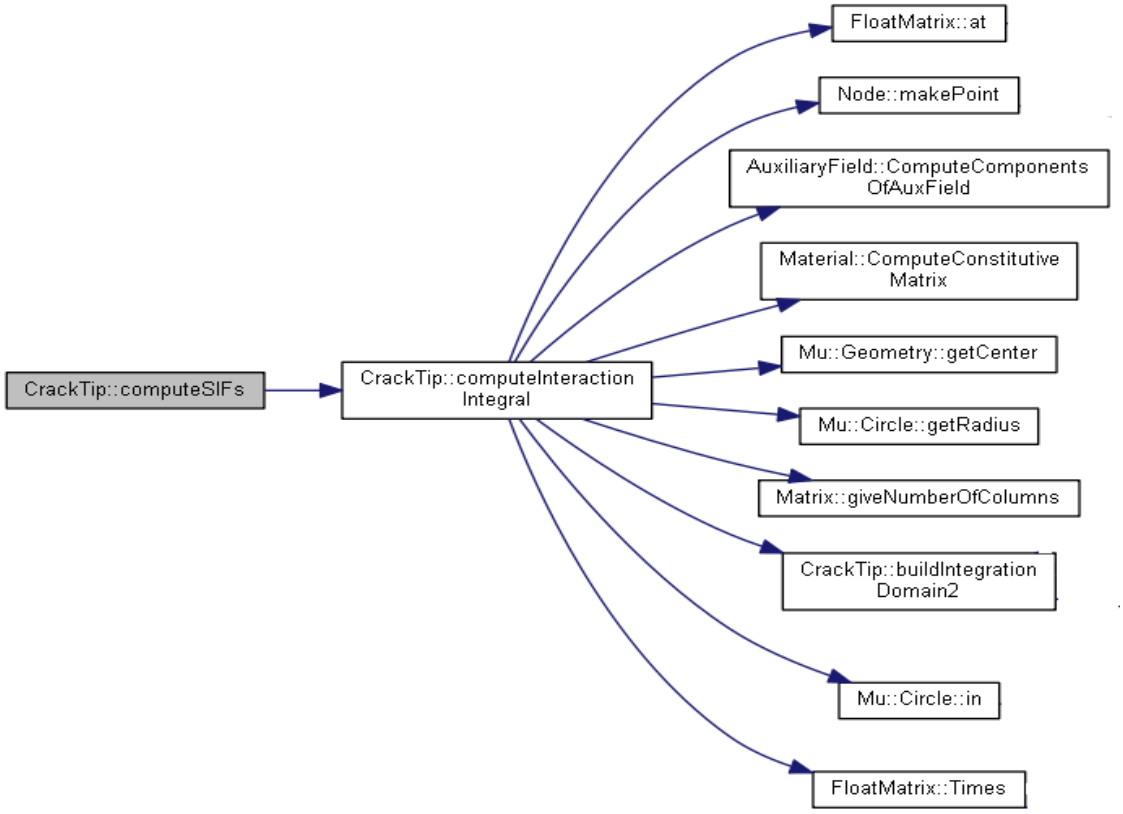



**Figure 3-15** Graphique d'appel de la méthode computeSIFs

Le vecteur de déformation (epsilon) et le vecteur de contrainte (sigma), ainsi que le gradient de la fonction poids « $q$[m] », sachant que tous les éléments du domaine considéré, sont enrichis par cette fonction scalaire qui est une fonction de pondération limitant l'intégrale à la zone d'intérêt. Cette dernière adopte la valeur 1 si le nœud est situé à l'intérieur du cercle sinon elle reste à zéro. Il est nécessaire aussi de calculer les dérivées de la fonction poids « $q$[m] » :

$$\frac{\partial q}{\partial x} = \sum_{m=1}^{n_{nodes}} \frac{\partial N_m}{\partial x} \cdot q_m \quad \text{....et.....} \quad \frac{\partial q}{\partial y} = \sum_{m=1}^{n_{nodes}} \frac{\partial N_m}{\partial y} \cdot q_m \qquad \textbf{(3-12)}$$

Ou « $N_m$ » fonction de forme. Ensuite ces dérivées sont transformé du système globale (x,y) au système locale (x1, x2) aligné à la fissure par le biais de la matrice de transformation $\mathrm{R}(\theta)$ (voir paragraphe 1.4.2.1)

$$\begin{bmatrix} \partial q/\partial x_1 \\ \partial q/\partial x_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \partial q/\partial x \\ \partial q/\partial y \end{bmatrix} \qquad \textbf{( 3-13)}$$

En troisième phase, on opte pour la transformation des quantités réelles vers le système locale de la fissure. étant donné que la matrice de rotation « R » n'est pas symétrique, en se basant la formule : (T_local= R* T_global*$R^T$). les expressions conventionnelles de transformations des contraintes planes prennent la forme suivante

$$\sigma' = R.\sigma.R^T \qquad \textbf{(3-14)}$$

Donc les contraintes locales **LocalSigma**[0, 1, 2] sont données comme suit:

$$\begin{cases} \sigma_{x1x1} = \frac{1}{2}\left(\sigma_{xx}+\sigma_{yy}\right)+\frac{1}{2}\left(\sigma_{xx}-\sigma_{yy}\right)\cos 2\theta+\sigma_{xy}\sin 2\theta \\ \sigma_{y1y1} = \frac{1}{2}\left(\sigma_{xx}+\sigma_{yy}\right)-\frac{1}{2}\left(\sigma_{xx}-\sigma_{yy}\right)\cos 2\theta+\sigma_{xy}\sin 2\theta \\ \sigma_{x1y1} = -\frac{1}{2}\left(\sigma_{xx}-\sigma_{yy}\right)\sin 2\theta+\sigma_{xy}\cos 2\theta \end{cases} \qquad \textbf{(3-15)}$$

L'utilisation des termes trigonométriques de double angle $\cos 2\theta$ et $\sin 2\theta$ proviennent du fait que la transformation agit de manière quadratique sur l'angle (effectue deux rotations : sur les deux indices de tenseur, ce qui est mathématiquement équivalent mais plus performant sur le plan numérique)

Dans la quatrième phase, on définit les champs auxiliaires $\left(\sigma^*,\varepsilon^*,\nabla u^*\right)$ issue des solutions asymptotiques au voisinage d'une pointe (singularité $r^{-1/2}$) qui représentent la solution analytique de référence pour une fissure (mode I et II) dans un milieu infinie. Cela se fait à l'aide de la méthode **ComputeComponentsOfAuxField**() de la classe **AuxiliaryField**. Elle fournit les gradients des déplacements auxiliaires (**AuxGradDisp**), les déformations auxiliaires (**AuxEps**), ainsi que les contraintes auxiliaires (**AuxStress**).

La phase finale consiste à Calculer les termes de l'intégrale d'interaction, dont la densité d'énergie de déformation d'interaction est donnée par la formule suivante:

$$W^{(1,2)} = \sigma_{ij}^{(1)}\varepsilon_{ij}^{(2)} = \sigma_{xx}^{(1)}\varepsilon_{xx}^{(2)} + 2.\sigma_{xy}^{(1)}\varepsilon_{xy}^{(2)} + \sigma_{yy}^{(1)}\varepsilon_{yy}^{(2)} \qquad \textbf{(3-16)}$$

(C'est le travail que ferait le champ de contrainte réel sur la déformation du champ auxiliaire ou vice-versa).l'état (1) = numérique local (**LocalSigma) ,** l'état (2) = auxiliaire (**AuxEps) ,**

Donc la formule de l'intégrale d'interaction pour chaque mode uv[*I,II*]

$$I_{\text{mod}} = \int_A \left[ \sigma_{ij}^{(1)} \frac{\partial u_i^{(2)}}{\partial x_j} + \sigma_{ij}^{(2)} \frac{\partial u_i^{(1)}}{\partial x_j} - W^{(1,2)} \delta_{1j} \right] \frac{\partial q}{\partial x_j} . dA \qquad \textbf{(3-17)}$$

Avec $\frac{\partial q}{\partial x_j}$ gradient de fonction poids, et $\delta_{1j}$ symbole de **Kronecker**

On somme ensuite, la forme intégrante de l'intégrale pour un matériau élastique est donnée par :

$$I_{\text{mod}} += -\left(I1 + I2 - W \,.\, dqdx1\right) . dV \qquad \textbf{(3-18)}$$

Ou :

**I1** : contribution du champ réel * gradient auxiliaire * dérivées de *q*

**I2** : contribution du champ auxiliaire * gradient réel * dérivées de *q*

Cette expression est évaluée de manière discrète, point de Gauss par point de Gauss, puis elle est accumulée dans $I_{\text{mod}}$ pour chaque mode. Finalement nous obtenons *I*(I) et *I*(II) , qui sont par le suite pour le calcul de KI et KII via la méthode **computeSIFs**().

la méthode **computeK_eq()** est utilisée pour calculer le facteur d'intensité équivalent qui est exprimé par :

$$K_{eq} = \sqrt{K_i^{\,2} + K_{ii}^{\,2}} \qquad \textbf{(3-19)}$$

**g) UpdateMyGeometry**() :

C'est une méthode conçue pour l'actualisation et la mise à jour de l'extrémité du fond de fissure en trois phase suivantes :

- L'évaluation de l'incrément de propagation : l'incrément de la fissure $\Delta a$ est évalué par la méthode **computeIncrementLength**() de la classe **crackgrowthincrementlaw** , ainsi que la direction de croissance $(\theta)$ est évalué aussi par la méthode **computeGrowthAngle**() de la classe **crackgrowthdirectionlaw**
- Mise à jour des coordonnées : translation du Tip du système local $(r, \theta)$ au système globale(x,y) .

$$\begin{cases} \Delta X = \Delta a \,.\cos\left(\theta + \alpha\right) \\ \Delta Y = \Delta a \,.\sin\left(\theta + \alpha\right) \end{cases} \qquad \textbf{(3-20)}$$

Avec (α) est l'orientation actuelle de la fissure.

Donc les nouvelles coordonnées du Tip qui permet de tracer le chemin de fissure dans le maillage sont :

$$\begin{cases} X_{tip}^{new} = X_{tip}^{old} + \Delta X \\ Y_{tip}^{new} = Y_{tip}^{old} + \Delta X \end{cases} \qquad \textbf{(3-21)}$$

• Mise à jour de la géométrie : révision des coordonnées et des segments de fissure, suivie d'une vérification de la cohérence avec l'élément qui contient le Tip via la méthode **interactsWith**() de la classe **geometryentity.**

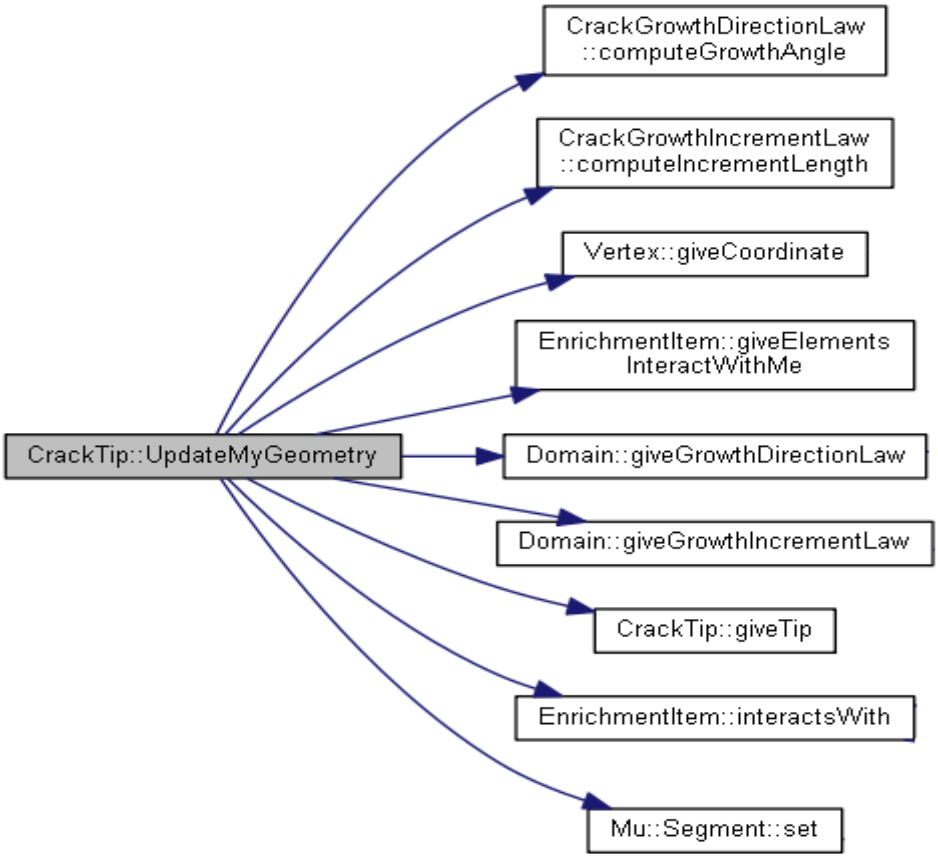



**Figure 3-16** Graphique d'appel de la méthode UpdateMyGeometry

### h) UpdateEnrichment() :

C'est une méthode prédominante dans la simulation de la propagation de la fissure, qui reconstruit l'état d'enrichissement des éléments et nœuds après le déplacement du tip , et identifié ceux dont la rigidité doit être recalculée.

Pour chaque nouveau pas de propagation de la fissure, cette méthode commence à actualiser la géométrie (tip, segment de la croissance) grâce à la méthode **UpdateMyGeometry()** décrit ci-dessus. cela se produit après sa finalisation de la réinitialisation de tous les éléments du domaine utilisant la méthode **reinitializeStateOfElement()** de la classe **Element**. Cette réinitialisation comprenne la suppression des anciens enrichissements (éléments, nœuds) via la méthode **eraseListOfEnrichmentItems()** , et la détermination de la liste des nouveaux éléments touchées par l'avancé via les deux méthode **DefineUpdatedElements()** et **intersects(tipSegment).**

```
std::list<Element*>  elems = this->DefineUpdatedElements();
 std::list<Element*>  newSplitElems ;
 for(std::list<Element*>  :: iterator i = elems.begin()  ; i != elems.end()  ; i++)
 {        if((*i)->intersects(tipSegment))
           newSplitElems.push_back(*i)  ;
            }
```

Enfin, la méthode **updateEnrichment()** met en ouvre de nouveaux enrichissements, et commence un procédure numérique similaire pour résoudre le problème. Elle distingue clairement le rôle du CrackTip et CrackInterior, en s'appuyant sur les différentes méthodes et fonctions mentionnées précédemment telles que : isEnrichedWith(), interactedElements(), setListOfInteractedElements(newTipElem), myAssociatedCrackInterior(), …..

```
std::cout  << " UPDATE ENRICHMENT ... " << endl ;
           newTipElem->isEnrichedWith(this);
           this->interactedElements->clear()  ;
           this->setListOfInteractedElements(newTipElem);
           newTipElem->setEnrichmentForMyNodes(this)  ;
```

```
for(std::list<Element*> :: iterator i = newSplitElems.begin() ; i != newSplitElems.end() ; i++)
{ (*i)->isEnrichedWith(myAssociatedCrackInterior) ;
myAssociatedCrackInterior->setListOfInteractedElements(*i);
(*i)->setEnrichmentForMyNodes(myAssociatedCrackInterior) ;
}
```

## 4.7. Classe enrichmentdetector:

L'objectif de cette classe est de permettre une sélection flexible des nœuds à enrichir dans le contexte de la méthode **XFEM**, ou la détection de l'enrichissement implique l'identification des nœuds du maillage qui bénéficient de fonctions supplémentaires pour illustrer la discontinuité/singularité est essentielle afin d'éviter la nécessite d'un remaillage. donc–elle définisse la stratégie de localisation de l'enrichissement

Pour atteindre cet objectif, cette classe employant ses trois méthodes effectue ce choix géométrique. La méthode **ElementAroundPointDetect :: setEnrichedNodes()** joue un rôle clé en enrichissant les nœuds de l'élément contenant le Tip, après avoir vérifié que l'élément sélectionné est effectivement de type CrackTip unique.

La seconde méthode **BallAroundPointDetect :: setEnrichedNodes()** enrichit tous les nœuds qui se trouve dans un périmètre spécifié du cercle de rayon donné centré sur le fond de fissure et crée par la classe (**Mu::Circle**) , ces nœuds sont identifiés et soumis au test des méthode **giveElementsInteractWithMe()** et **conflicts(ball)** mentionnées précédemment.

Tant que la dernière méthode **SplitElementDetect :: setEnrichedNodes()** enrichit tous les nœuds des éléments traversés par l'intérieur de la fissure après avoir vérifié que les éléments sélectionnées sont bien compatibles avec le type **CrackInterior**.

Compte tenue de la structure logicielle de cette classe, la conception illustre une utilisation judicieuse du polymorphisme. on y observe que la méthode **setEnrichedNodes()** est implémentée de différentes manière selon une logique spécifique : autour d'un point, dans un rayon fixé, ou dans les éléments traversés par la fissure. cela confère au code une flexibilité lors de l'ajout d'une nouvelle détection.

Pour illustrer l'encapsulation, prenons l'exemple de la méthode **BallAroundPointDetect**. Cette dernière crée un cercle, identifiée les éléments qui le croisent, vérifié que chaque nœuds

est à l'intérieur, et applique ensuite l'enrichissement, tous cela se fait sans que les autres classes qui sont en interaction avec celle-ci aient besoin de comprendre comment cette décision est prise, elles reçoivent simplement l'ordre d'être enrichies .

```
class ElementAroundPointDetect : public PointDetector
{ public:
  void setEnrichedNodes(EnrichmentItem *enrItem);
};

class BallAroundPointDetect : public PointDetector
{ public:
  void setEnrichedNodes(EnrichmentItem *enrItem);
};

class SplitElementDetect : public PolyLineDetector
{  public:
  void setEnrichedNodes(EnrichmentItem *enrItem);
        };
```

## 4.8.Enrichmentfunction:

C'est une classe cruciale dans l'architecture du code **OpenXFEM++**, qui rend plus facile la modélisation des fonctions d'enrichissements. Ces derniers sont essentielles pour capturer des singularités au voisinage du fond de la fissure ainsi que les discontinuités le long d'une fissure.

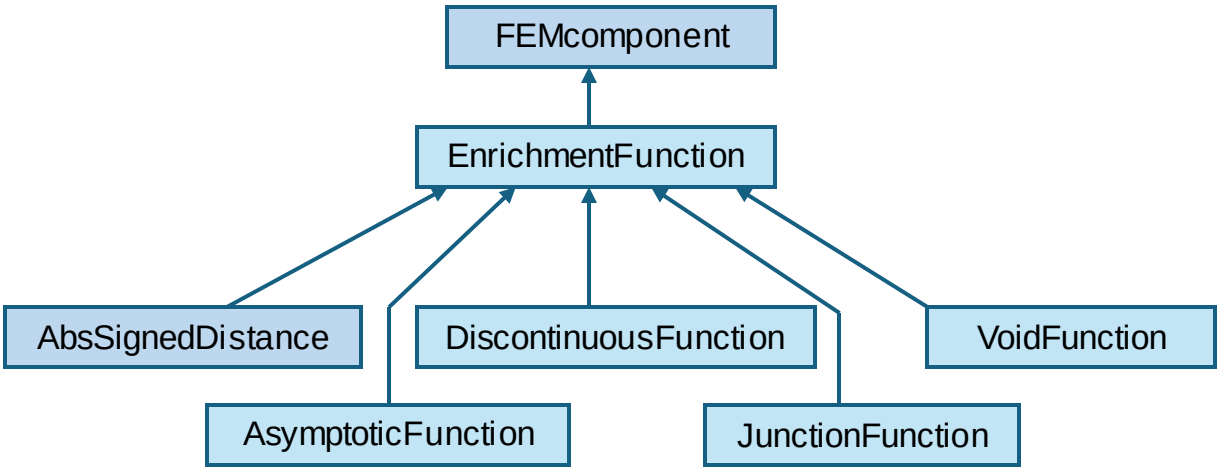



**Figure 3-17** Diagramme d'héritage de la classe EnrichmentFunction

Cette classe comporte trois methods principales :

- la méthode **ofType()** lit un attribut « class » depuis un fichier d'entrée et génère des instances typées de fonctions d'enrichissement correspondant au numéro de l'objet actuel, mais avec un type explicite tel que DiscontinuousFunction , AsymptoticFunction, …etc.

-La méthode **setMyEnrichmentItem()** à pour mission d'associer la fonction sélectionné à un ou plusieurs items d'enrichissement telles que CrackTip , CrackInterior, ….etc.

-Tant que la méthode **findActiveEnrichmentItem**() établit l'enrichissement active pour le calcul en cours,

La classe **enrichmentfunction** hérite plusieurs sous-classes qui définissent le type d'enrichissement basé sur la théorie de la mécanique de la rupture linéaire élastique (LEFM), telles que :

### 4.8.1. Discontinuousfunction

C'est une classe qui implémente une fonction discontinue H(x) (type Heaviside) employée pour décrire un saut de déplacement le long de la fissure sans nécessité de remaillage du domaine. Elle représente explicitement la position géométrique d'un point de Gauss par rapport à la fissure sous forme de valeur numérique (±1ou 0) via la méthode principale **EvaluateYourSelfAt(GaussPoint* gp)**

$$H(x)=\begin{cases}+1 & \textit{au dessus de la fissure}\\ -1 & \textit{au dessous de la fissure}\\ 0 & \textit{sur la fissure}\end{cases} \qquad \textbf{(3-22)}$$

Cette dernière utilise la méthode **giveMyElement**() de la classe **GauussPoint** pour déterminer l'élément auquel appartient le point d'intégration, puis elle transforme ses coordonnées du système locale au système globale dans le domaine en appelant la fonction **giveFEInterpolation**() de la classe **Element**. Par la suite , elle évalue la position du point de Gauss via la méthode **givePositionComparedTo**() de la classe **geometryentity**

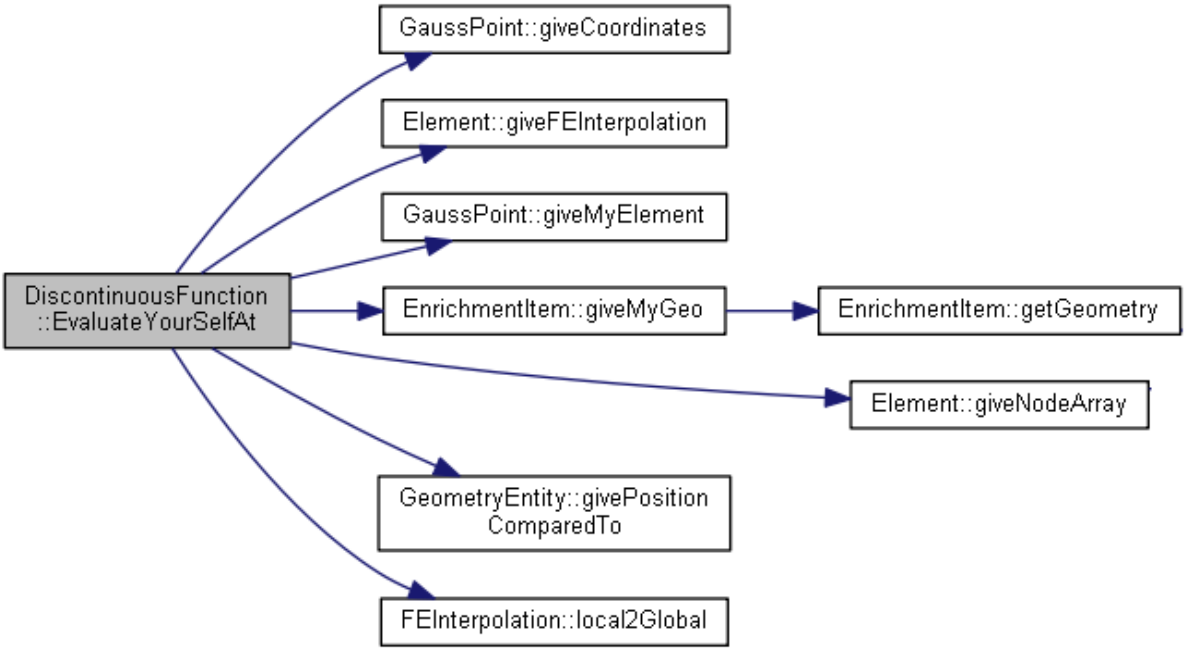



**Figure 3-18** Graphique d'appel de la méthode EvaluateYourSelfAt

### 4.8.2. HomogeneousElastCrackAsymp :

(Homogenious Elastic Crack Asymptotic) est une classe conçue pour implémentation des fonctions asymptotiques de Williams(1957) dans le contexte de la méthode **XFEM**. Elle vise à interpréter le comportement singulier du champ de déplacement et de contraintes au voisinage du fond de fissure dans un matériau homogène, et élastique linéaire.

Cette classe évalue les fonctions asymptotiques en un point d'intégration (point de Gauss) et aux nœuds en exploitant les deux similaires méthodes **EvaluateYourSelfAt(GaussPoint* gp)** et **EvaluateYourSelfAt(Element* e ,Node* n)** . successivement Suite à l'appel de la méthode **computePolarCoordOf**(). Cette dernière est employée pour réaliser le transformation global-polaire par rapport au tip de fissure, ce qui donne les formules de ces fonctions comme indiqué ci-dessous :

$$\phi_1(r,\theta)=\sqrt{r}\sin\left(\frac{\theta}{2}\right) \quad , \quad \phi_2(r,\theta)=\sqrt{r}\cos\left(\frac{\theta}{2}\right)$$

$$\phi_3(r,\theta)=\sqrt{r}\sin\left(\frac{\theta}{2}\right)\cos(\theta) \quad , \quad \phi_4(r,\theta)=\sqrt{r}\cos\left(\frac{\theta}{2}\right)\cos(\theta) \qquad \textbf{(3-23)}$$

Tandis que les dérivées de ces fonctions (**$VF_i$**) en coordonnées globales sont évaluées en utilisant la méthode **EvaluateYourGradAt(GaussPoint* gp)** après quelles sont évaluées dans le système locale , comme exemple la première fonctions qui est donnée comme suit :

$$\frac{\partial \phi_1}{\partial r} = -\frac{1}{2\sqrt{r}}\sin\left(\frac{\theta}{2}\right) \quad , \quad \frac{\partial \phi_1}{\partial \theta} = \frac{1}{2\sqrt{r}}\cos\left(\frac{\theta}{2}\right) \qquad \textbf{(3-24)}$$

puis, on effectue une rotation vers le système globale en utilisant l'angle (α) (angle d'inclinaison de la fissure par rapport à l'axe global x) utilisant la matrice de rotation « R » grâce à la fonction **local2Global()** de la classe **FEInterpolation**():

$$\begin{bmatrix} \partial\phi_i/\partial x \\ \partial\phi_i/\partial y \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \partial\phi/\partial r \\ \partial\phi/\partial\theta \end{bmatrix} \qquad \textbf{(3-25)}$$

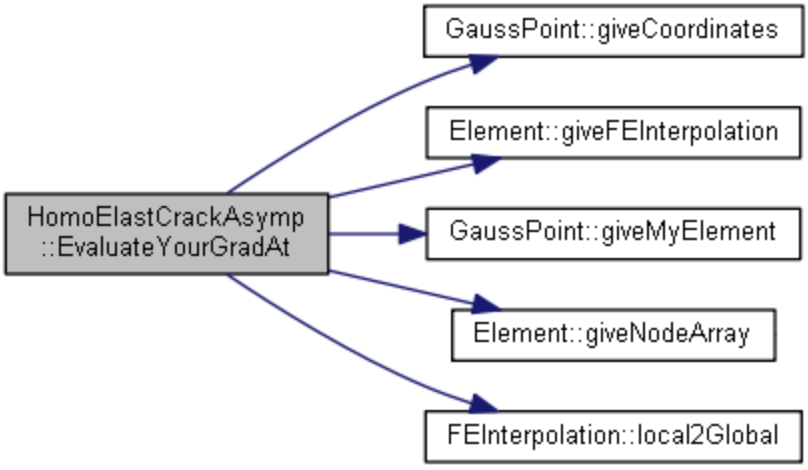



**Figure 3-19** Graphique d'appel de la méthode EvaluateYourGradAt

En ce qui concerne la programmation, cette classe met en œuvre de façon explicite les techniques de la programmation **orientée objet**. Cela se traduit par :

-l'héritage démontré par l'utilisation du mot-clé « this » pour accéder à des membres tels que (**this->activeEnrItem**), qui une méthode issue de la classe parent EnrichmentFunction et n'est pas définie dans la classe concernés, donc héritée.

-Le polymorphisme est manifeste de manière classique, ou la méthode **EvaluateYourSelfA()** est définit à deux reprise avec le même intitulé mais avec des paramètres distincts (**GaussPoint* gp**) et (**Element* e ,Node* n**).

-L'encapsulation : elle se manifeste lorsque cette classe fait appel à des méthodes pour interagir avec d'autres objets au lieu d'accéder directement à leurs données internes, habituellement en

utilisant les méthodes « **getter** » , par exemple (**gp->giveMyElement()**) qui récupère les données via une méthode **giveMyElement**() de la classe GaussPoint.

-L'abstraction :on l'appréhende aisément on observant la méthode **computePolarCoordOf()** de la classe **CrackTip** qui est invoquée sur l'objet **activeEnrItem** pour calculer les coordonnées polaires sans avoir à se préoccuper du processus de calcul , il suffit d'obtenir les résultats.

## 4.9.IntegrationRule

Cette classe gère les règles des intégrations et assure la discrétisation numérique des intégrales requise pour le calcul par la méthode des éléments finis étendus **XFEM**. Ces intégrales sont traitées en utilisant la quadrature de Gauss, qui repose sur l'ensemble des points d'intégration et du poids associés.

Elle a pour objectif de fournir les poids d'intégration ($w_i$) qu'elle stocke dans le vecteur **giveWeightArray(),**, ainsi que les coordonnées des points de Gauss ($\zeta,\eta$) qu'elle consigne par le biais de l'objet ( **Mu::Point** ), tout en gérant leur création en fonction de la géométrie de l'élément (triangle, carré, …etc) grâce à la méthode **setUpIntegrationPoints()**. Cette approche maintient l'encapsulation des données qui ne peuvent être modifiées qu'à partir de cette classe, tout en favorisant la modularité lors de la manipulation de ces points.

Cette classe est une classe parentale de deux sous-classes suivantes :

### 4.9.1. SplitGaussLegendreQuadrature

Cette sous classe est conçue pour mettre en œuvre la technique des éléments partitionnés, une stratégie robuste et systématique. Cette technique est adoptée pour éviter le souci d'inexactitude et la défaillance de convergence lors de la modélisation des discontinuités en utilisant les éléments entières affectés par la fissure. Sachant que l'intégration du produit des gradients de fonctions de formes numériques et la détermination de leurs matrices de rigidité posent un défi majeur.

La méthode centrale de cette classe est la méthode **SetUpPointsOn2dDomain()** qui a le rôle d'adapter la quadrature de Gauss-legendre classique à des domaines découpées en sous triangles utilisant la méthode (**elem->PartitionMySelf()**) de la classe parent **Element**, ensuite elle applique une quadrature de Gauss indépendante de chacun d'eux.

Pour un triangle standard, la quadrature de Gauss est donnée par :

$$T_{ref} = \{(\xi,\eta) | \quad \xi,\eta \geq 0, \quad \xi+\eta \leq 0\} \qquad \textbf{(3-26)}$$

Les points de Gauss sont bien connus selon **Zienkiewicz & Taylor**(2005) pour un triangle n=3 : points barycentriques ( 2/3, 1/6, 1/6 ), et le poids ($w_j$).

Chaque sous triangle issue de la partition est définie par ses sommets

$(x_1, y_1)$ , $(x_2, y_2)$ , $(x_3, y_3)$

On utilise les fonctions de forme linéaire du triangle P1 :

$$N_1(\xi,\eta) = 1-\xi-\eta \quad , \quad N_2(\xi,\eta) = \xi \quad , \quad N_3(\xi,\eta) = \eta \qquad \textbf{(3-27)}$$

La transformation vers le sous triangle physique est

$$\begin{cases} x = N_1(\xi,\eta).x_1 + N_2(\xi,\eta).x_2 + N_3(\xi,\eta).x_3 \\ y = N_1(\xi,\eta).y_1 + N_2(\xi,\eta).y_2 + N_3(\xi,\eta).y_3 \end{cases} \qquad \textbf{(3-28)}$$

Cette étape est implémenté dans

```
FloatArray *temp ;
                temp = FEInterpol->evalN((*coordArray)[i]);
                for (size_t j = 0 ; j < 3 ; j++)
                {       double x = coord->at(j+1,1) ;
                        double y = coord->at(j+1,2) ;
                        newIntPoint->x  += (*temp)[j]*x;
                        newIntPoint->y  += (*temp)[j]*y;
                       }
```

Le poids de gauss dans le triangle physique est $w_{phys} = J.w_{ref}$ , ou J est le déterminant du Jacobian de transformation, vaut dans un triangle $J = 2.Aire(T_{phys})$

```
double newWeight  = 2.0 * weight->at(i)  * area;
```

Ou : **weight->at(i)** → poids dans le triangle de référence

**Area** : l'aire de sous triangle physique

**Facteur (2.0)** : correction liés à l'aire du triangle de référence (1/2).

Après cette étape on opte pour une vérification et filtrage des sous triangles dont l'air nulles ou trop petits (aire > $10^{e-4}$ dans notre code)

```
if( ((*tri)[j]->isTriangle) && ((*tri)[j]->area() > 1e-4 ) )
```

## Exemple numérique :

Pour éclairer cette technique on adopte un élément initial carré [0,1] x [0,1], coupé par une fissure en deux sous-triangles :

Triangle1 : (0,0),(1,0),(0.5,0.5)
Triangle2 : (0,0),(0.5,0.5),(0,1)

On choisit la quadrature de Gauss à trois points (n=3) dans chaque triangle.

Pour un triangle de référence, la quadrature de Gauss est donnée par :

$$T_{ref} = \left\{(\xi,\eta)\right| \quad \xi,\eta \geq 0, \quad \xi+\eta \leq 0\}$$

Les points de Gauss sont bien connus : dont les coordonnées barycentriques sont :

(2/3, 1/6, 1/6), (1/6, 2/3, 1/6), (1/6, 1/6, 2/3), et le poids ($w_{j=}$1/6) pour chaque point (j=1,2,3)

-Transformation vers Triangle 1 :

Les sommets : $(x_1,y_1)=(0,0)$ , $(x_2,y_2)=(1,0)$ , $(x_3,y_3)=(1,0.5)$

Formule d'interpolation

$$\begin{cases} x = N_1(\xi,\eta).x_1 + N_2(\xi,\eta).x_2 + N_3(\xi,\eta).x_3 \\ y = N_1(\xi,\eta).y_1 + N_2(\xi,\eta).y_2 + N_3(\xi,\eta).y_3 \end{cases}$$

On fait le calcul pour les valeurs réelles, on trouve :

Point 1 : ($N_1$, $N_2$, $N_3$)= (2/3, 1/6, 1/6),

$$\begin{cases} x = \frac{2}{3}.0 + \frac{1}{6}.1 + \frac{1}{6}.1 = 0.5 \\ y = \frac{2}{3}.0 + \frac{1}{6}.0 + \frac{1}{6}.(0.5) = 0.0833 \end{cases}$$

Point 2 : ($N_1$, $N_2$, $N_3$)= (1/6, 2/3, 1/6)

$$\begin{cases} x = \frac{1}{6}.0 + \frac{2}{3}.1 + \frac{1}{6}.1 = 0.8333 \\ y = \frac{1}{6}.0 + \frac{2}{3}.0 + \frac{1}{6}.(0.5) = 0.0833 \end{cases}$$

Point 3 : ($N_1$, $N_2$, $N_3$)= (1/6, 1/6, 2/3),

$$\begin{cases} x = \frac{1}{6}.0 + \frac{1}{6}.1 + \frac{2}{3}.1 = 0.8333 \\ y = \frac{1}{6}.0 + \frac{1}{6}.0 + \frac{2}{3}.(0.5) = 0.3333 \end{cases}$$

L'Aire de triangle 1

$$A_1 = \frac{1}{2}\left|(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)\right| = \frac{1}{2}\left|(1-0)(0.5-0) - (1-0)(0-0)\right| = 0.25$$

Le Jacobian de transformation

$$J = 2\,.\,Aire\left(T_{phys}\right) = 2\,.\,0.25 = 0.5$$

Le poids transformé est $w_{phys} = J\,.\,w_{ref}$

$$w_i = 2\,.\,A_1\,.\,\frac{1}{6} = 2\,.\,0.25\,.\,\frac{1}{6} = 0.0833$$

⇨ Donc pour le triangle 1 on obtient :

⇨ Point (0.5, 0.0833), (0.8333, 0.0833) , (0.8333 0.3333)

⇨ Poids : 0.0833 pour chacun.

La même chose pour l'autre triangle 2 :

-Transformation vers Triangle 2 :

Les sommets : $(x_1, y_1) = (0,0)$ , $(x_2, y_2) = (1,0.5)$ , $(x_3, y_3) = (0,0.5)$

On fait le calcul pour les valeurs réelles, on trouve :

Point 1 : ($N_1$, $N_2$, $N_3$)= (2/3, 1/6, 1/6),

$$\begin{cases} x = \frac{2}{3}.0 + \frac{1}{6}.1 + \frac{1}{6}.0 = 0.1666 \\ y = \frac{2}{3}.0 + \frac{1}{6}.(0.5) + \frac{1}{6}.(0.5) = 0.1666 \end{cases}$$

Point 2 : ($N_1$, $N_2$, $N_3$)= (1/6, 2/3, 1/6)

$$\begin{cases} x = \frac{1}{6}.0 + \frac{2}{3}.1 + \frac{1}{6}.0 = 0.6666 \\ y = \frac{1}{6}.0 + \frac{2}{3}.(0.5) + \frac{1}{6}.(0.5) = 0.4166 \end{cases}$$

Point 3 : ($N_1$, $N_2$, $N_3$)= (1/6, 1/6, 2/3),

$$\begin{cases} x = \frac{1}{6}.0 + \frac{1}{6}.1 + \frac{2}{3}.0 = 0.1666 \\ y = \frac{1}{6}.0 + \frac{1}{6}.(0.5) + \frac{2}{3}.(0.5) = 0.4166 \end{cases}$$

L'Aire de triangle 1

$$A_1 = \frac{1}{2}\left|(0.5-0)(1-0)-(0-0)(0.5-0)\right| = 0.25$$

$$A_2 = \frac{1}{2}\left|(x_2-x_1)(y_3-y_1)-(x_3-x_1)(y_2-y_1)\right| = \frac{1}{2}\left|(1-0)(0.5-0)-(0-0)(0.5-0)\right| = 0.25$$

Le poids transformé est

$$w_i = 2 . A_1 . \frac{1}{6} = 2 . 0.25 . \frac{1}{6} = 0.0833$$

⇨ Donc pour le triangle 1 on obtient :

⇨ Point (0.1666, 0.1666), (0.6666, 0.4166) , (0.1666, 0.4166)

⇨ Poids : 0.0833 pour chacun.

⇨ Remarque : l'aire de chaque sous triangle égale à la somme du poids des trois points de Gauss (0.0833*3=0.25)

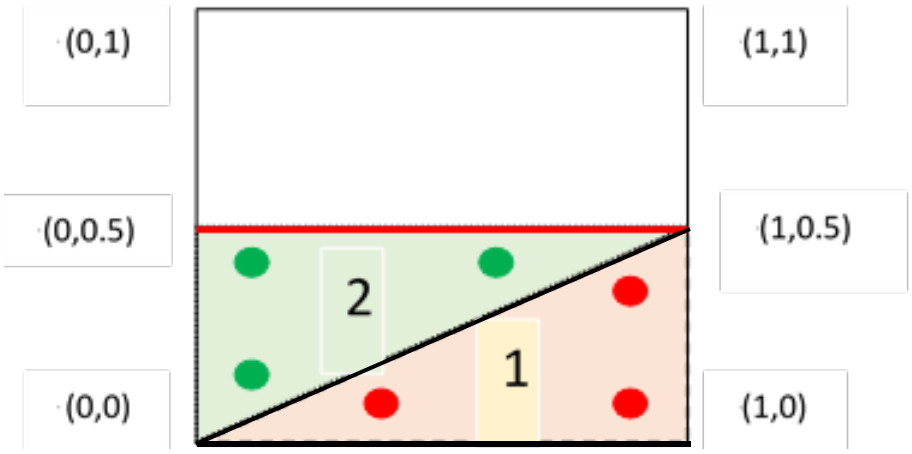



**Figure 3-20** Technique de partition élémentaire

### 4.9.2. StandardGaussLegendreQuadrature

Cette classe implémente des schémas d'intégration numérique basées sur les quadratures Gauss-Legendre, adaptées aux différents domaines géométriques : triangles, carrés,…etc. Donc, elle est responsable sur l'étape d'évaluation des intégrales sur les domaines élémentaires telles que les surfaces et les volumes, dans le but de constituer les matrices de rigidité et les vecteurs de charges.

Cette classe repose sur plusieurs méthodes telles que la méthode **SetUpPointsOnTriangle**() qui implémente les points de Gauss et leurs poids pour une intégration numérique dans un élément quadrilatère triangulaire ou les schémas classiques reconnues pour les nombres des points sont 1,3,7,13 respectivement.

Et aussi la méthode **SetUpPointsOnSquare**() , qui implémente les points de Gauss et leurs poids pour une intégration numérique dans un élément quadrilatère bilinéaire à quatre nœuds, dont les schémas classiques reconnues pour les nombres des points sont 1,7,9,16, 25, 36 ,49 et 56 respectivement.

Le nombre des points de Gauss est préalablement défini dans les sous classes **Tri3_U** , **Quad4_U** de la classe **PlaneProblem**, …etc., qui sont responsables des de la gestion des intégrations et partitions nécessaires pour le traitement **XFEM**, notamment les éléments infectées par les fissures.

En termes de mise en œuvre orientée objet, cette classe illustre une implémentation claire et robuste de la quadrature de Gauss dans le contexte de la méthode **XFEM**. On peut distinguer facilement l'encapsulation des données ainsi que le polymorphisme.

l'encapsulation des données se illustrée par l'inaccessibilité directement pour l'utilisateur des méthodes privées telles que **integrationPointVector** et **weightArray** configurées via la méthode **SetUpIntegrationPoints**() de la classe parente **IntegrationRule.**

La même méthode **SetUpIntegrationPoints** peut générer des quadratures adaptées à des domaines différents comme exemple quadrature 2D pour les éléments triangulaires et rectangulaires (Triangle, Square), illustrant ainsi le concept de polymorphisme

Pour un domaine maillé par des éléments quadrilatères bilinéaires à quatre nœuds, la quadrature de Gauss pour **XFEM** est la suivante (voir fig ci-dessous) :

- Élément de pointe est partitionné en cinq sous-triangles, le nombre des points de Gauss est 13 GP utilisées pour chaque sous-triangle.

-Elément divisé par la fissure (le long de la fissure) est partitionné en quatre sous-triangles, le nombre des points de Gauss est 13 GP utilisées pour chaque sous-triangle.

-Elément touché par la zone d'enrichissement fixe de rayon « $r_d$ » , le nombre des points de Gauss est 16 GP utilisées pour chaque rectangle.

-Elément non affecté par la fissure (loin de la fissure avec comportement standard) , le nombre des points de Gauss est 04 GP pour l'élément complet.

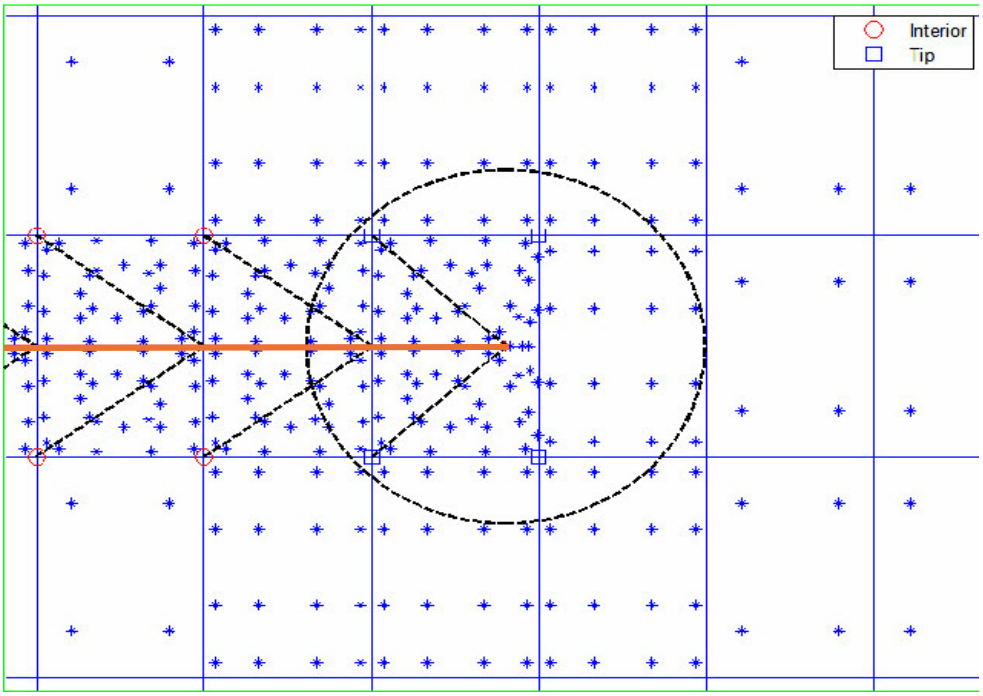



**Figure 3-21** Technique de partition des points de Gauss

## 4.10. Classe Load

La classe **Load** est une classe–qui prend en considération les charges appliquées (charge distribuée, charge nodale, conditions aux limites, etc.).

La classe **Load** stocke les valeurs des charges dans le tableau **'componentArray'**. Alors que les composants d'une charge à un pas de temps donné sont le produit de **'componentArray'** par la valeur de la fonction **'loadTimeFunction'** à ce pas de temps.

Cette classe est reliée avec quatre classes dérivées, comme la classe dérivée **BodyLoad** qui agit sur tout le volume des éléments finis (par exemple, le poids mort), ainsi que la classe dérivée **BoundaryLoad** : qui tiens en compte les conditions limites cinématique (déplacement, vitesse, température, etc.), d'autre classe dérivée **InitialCondition** qui tiens en compte des conditions initiales.

Finalement la classe dérivée **NodalLoad** prend en considération une charge concentrée (force, moment,...) qui agit directement sur un nœud (pas sur un élément).

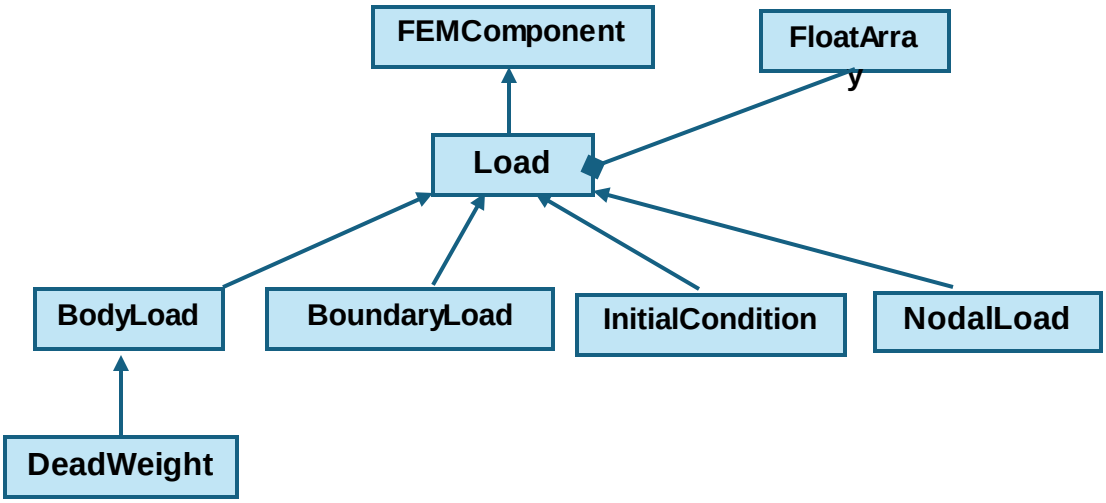



**Figure 3-22** Diagramme d'héritage de la classe Load

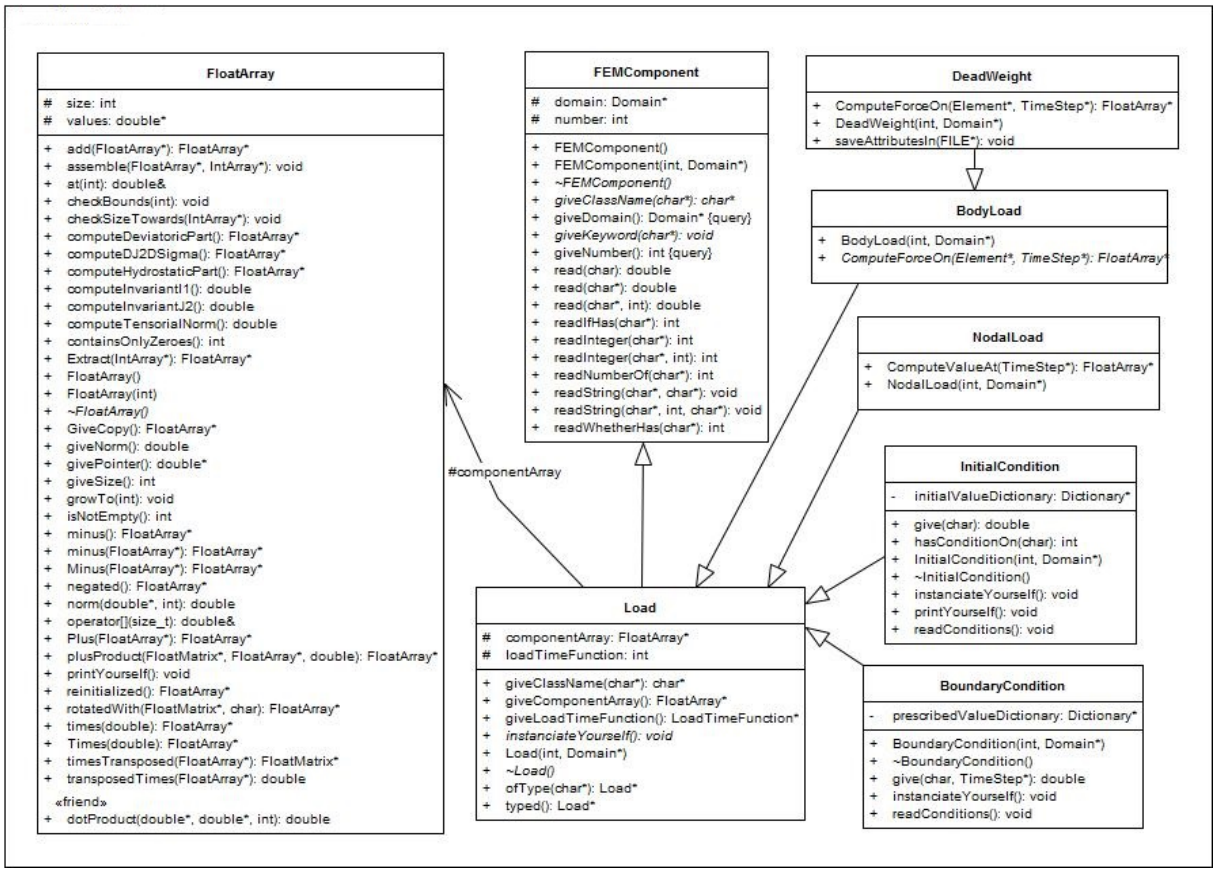



**Figure 3-23** Diagramme de collaboration et d'héritage pour la classe Load

Crack

Crack-Tip

Crack Interior

Non-enriched elements : (One Gauss point (GP))

Partially tip enriched elements : Element used to compute the interaction Integral ($r_d = 2{,}5h_{local}$ )

(13 Gauss points (GP) for each triangle)

Split and tip elements : (13 Gauss points (GPs) for each sub-triangle)

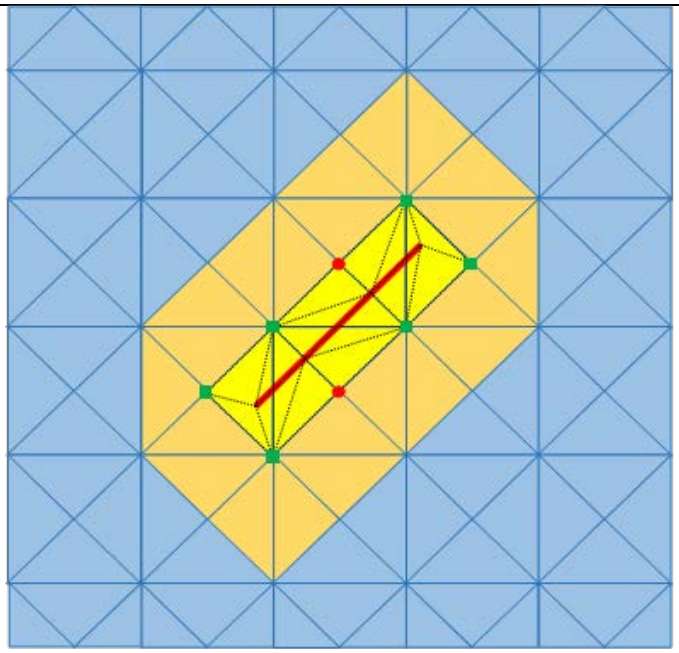


**Figure 3-24** Points de Gauss élémentaires pour le calcul numérique

## 5. Conclusion

Ce chapitre a présenté l'implémentation de la méthode XFEM au sein de la plateforme orientée objet OpenXFEM++, en mettant en évidence l'architecture logicielle, les principales classes développées ainsi que leurs interactions pour le traitement des discontinuités. Les stratégies d'enrichissement, d'intégration numérique et de calcul des facteurs d'intensité de contrainte ont démontré la modularité, l'extensibilité et la robustesse de l'approche proposée. Ce développement constitue la base informatique nécessaire pour la validation numérique et aux applications exposés dans le chapitre suivant.

# CHAPTRE 4 : VALIDATION ET DISCUSSION DES RESULTATS NUMÉRIQUES

# 1 Introduction

Dans ce chapitre en considère les mêmes conditions aux limites de déformation plane pour tous les exemples étudiés, où une variété de résultats de facteurs d'intensité de contrainte en mode mixte obtenus en utilisant programmation orientée objet en **C++** sont présentés et comparés à ceux fournis par des références théoriques et le logiciel commercial FEM Abaqus.

# 2 Fissure centrale inclinée sous chargement en traction

Cet exemple calcule le facteur d'intensité de contrainte en mode mixte (**MMSIF**) pour une plaque carrée finie isotrope présentée à la Figure 4-1. Les bords supérieur et inférieur sont soumis à une contrainte uniforme de traction et de compression « $\boldsymbol{\sigma}$ **=1 »**. La plaque possède une largeur **W=10** et une fissure centrale inclinée de longueur « **2a=1 »**. Les propriétés élastiques du matériau sont définies par le coefficient de Poisson « $\boldsymbol{\nu}$ » et le module de Young « $\boldsymbol{E}$ **»** [84, 86].

L'épaisseur de la plaque est supposé négligeable devant les autres dimensions. Une analyse bidimensionnelle (2D) est donc conduite en utilisant des éléments triangulaires structurés afin d'approximer la géométrie de la plaque.

Les facteurs d'intensité de contrainte en mode I et II, notés KI et KII, sont calculées au niveau de la pointe de la fissure pour un angle d'inclinaison $\boldsymbol{\beta}$ variant de 5° à 85° par un incrément de 5°, puis comparés aux solutions théoriques sous chargement en traction uniaxiale selon la formule proposée par Sih et al.[13] :

$$K_I = \sigma_{app}\sqrt{\pi a}.\cos^2\beta$$
$$K_{II} = \sigma_{app}\sqrt{\pi a}.\cos\beta.\sin\beta \qquad \textbf{(4-1)}$$

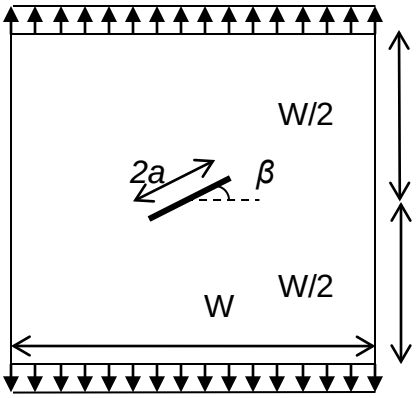



**Figure 4-1** Géométrie d'une fissure centrale inclinée dans une plaque infinie soumise à une traction uniaxiale

afin de simplifier l'analyse, le facteur d'intensité de contrainte est normalisé :

$$K_{normI,II} = K_{I,II} \Big/ \sigma_{app}\sqrt{\pi a} \qquad \textbf{(4-2)}$$

Pour le calcul du MMSIF, la méthode **XFEM** utilise une zone d'enrichissement fixe afin de déterminer le rayon optimal associé à l'intégrale $\boldsymbol{J}$, définie par $\boldsymbol{r_d = 2{,}5\, h_{local}}$.

Le tableau 4-1 met en évidence un très bon accord avec la solution exacte, où l'erreur relative est généralement inférieure à 5 %, ce qui montre que la précision obtenue est satisfaisante.

| Angle β(°) | K | (C++) | Ref | C++/ Ref (%) | Abaqus | C++/ Abaqus (%) |
|---|---|---|---|---|---|---|
| 15 | K I | 0.9197 | 0.9330 | 1.42 | 0.9178 | 0.21 |
| | K II | 0.2413 | 0.2500 | 3.48 | 0.2361 | 2.17 |
| 30 | K I | 0.7308 | 0.7500 | 2.56 | 0.7460 | 2.03 |
| | K II | 0.4210 | 0.4330 | 2.79 | 0.4050 | 3.79 |
| 45 | K I | 0.4830 | 0.5000 | 3.40 | 0.5119 | 5.64 |
| | K II | 0.4762 | 0.5000 | 4.76 | 0.4679 | 1.74 |
| 60 | K I | 0.2418 | 0.2500 | 3.28 | 0.2510 | 3.66 |
| | K II | 0.4239 | 0.4330 | 2.10 | 0.4054 | 4.36 |
| 75 | K I | 0.0663 | 0.0670 | 1.03 | 0.0685 | 3.21 |
| | K II | 0.2428 | 0.2500 | 2.88 | 0.2340 | 3.62 |

**Tableau 4-1** Convergence des SIF normalisés pour une fissure centrale inclinée sous chargement en traction

Comme illustre la Figure 4-2, la convergence entre les résultats obtenus avec Abaqus et ceux issue de la programmation orientée objet en C++ confirme l'efficacité et la robustesse de l'implémentation **OpenXFEM++**.

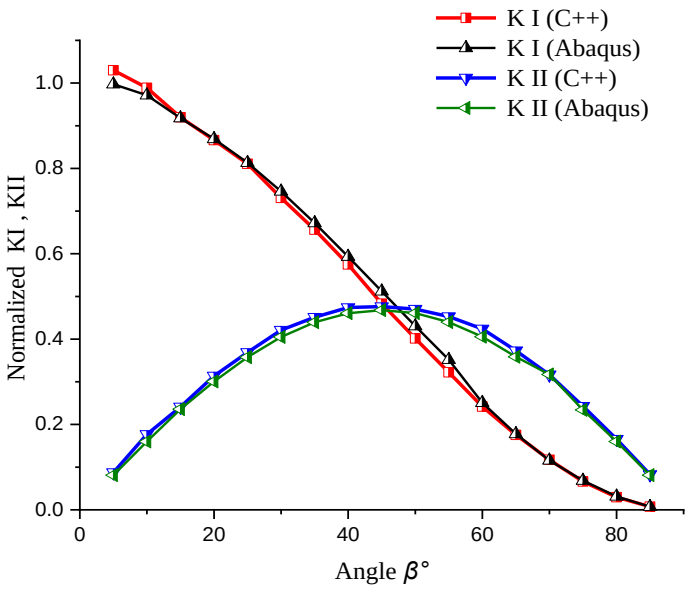



**Figure 4-2:** Valeurs normalisées des facteurs d'intensité de contrainte en modes I et II (calculées / Abaqus) en fonction de l'angle d'inclinaison $\beta$

Les valeurs normalisées des facteurs d'intensité de contrainte en modes I et II pour différentes discrétisations et dimensions géométriques avec un angle d'inclinaison fixe $\beta$=30°, sont présentées dans les tableaux 4-2 et 4-3.

| Num elements | $r_d/h_{local}$ | | | | | |
|---|---|---|---|---|---|---|
| | 2 | 2.5 | 3 | 3.5 | 4 | 4.5 |
| 1088 | 0.7095 | 0.7619 | 0.7678 | 0.7699 | 0.8216 | 0.8217 |
| 5928 | 0.7264 | 0.7258 | 0.7266 | 0.7252 | 0.7249 | 0.7214 |
| 10000 | 0.7309 | 0.7214 | 0.7178 | 0.7163 | 0.7170 | 0.7165 |
| 20000 | 0.7310 | 0.7295 | 0.7186 | 0.7200 | 0.7185 | 0.7178 |

**Tableau 4-2** KI normalisé pour diverses discrétisations et tailles de domaine

| Num elements | $r_d/h_{local}$ | | | | | |
|---|---|---|---|---|---|---|
| | 2 | 2.5 | 3 | 3.5 | 4 | 4.5 |
| 1088 | 0.4014 | 0.4544 | 0.4525 | 0.4483 | 0.4776 | 0.4781 |
| 5928 | 0.4222 | 0.4240 | 0.4258 | 0.4252 | 0.4242 | 0.4206 |
| 10000 | 0.4234 | 0.4232 | 0.4236 | 0.4247 | 0.4246 | 0.4238 |
| 20000 | 0.4342 | 0.4343 | 0.4284 | 0.4267 | 0.4255 | 0.4274 |

**Tableau 4-3** KII normalisé pour diverses discrétisations et tailles de domaine

Pour les deux modes **I** et **II**, il apparait que les valeurs obtenus des **MMSIF** deviennent indépendantes de la taille de la zone d'enrichissement lorsque $\mathbf{r_d > 2\, h_{local}}$. Ceci montre que le raffinement du maillage n'influence pas significativement les résultats.

La taille du domaine est caractérisée par un espacement local (pas) du maillage à la pointe de fissure $\boldsymbol{h_{local}}$, où :

$$\boldsymbol{r_d = r_k \,.\, h_{local}} \qquad \textbf{(4-3)}$$

Avec $\boldsymbol{r_k}$ un coefficient scalaire

$\boldsymbol{h_{local}}$ définie comme la racine carrée de la surface de l'élément à la pointe.

L'erreur relative demeure généralement inférieure à 5 %, ce qui indique que le niveau de précision est tout à fait acceptable. L'écart entre notre implémentation de recherche et le logiciel industriel Abaqus s'explique par plusieurs facteurs :

- Abaqus utilise une version industrialisée optimisée de **XFEM**, intégrant une fonction d'enrichissement très finement paramétrée.

- Il met en œuvre des algorithmes propriétaires permettant une intégration tronquée adaptée aux éléments **XFEM**, minimisant les erreurs dues aux discontinuités, tandis que le code de recherche adopte une approche moins optimisée.

- Les erreurs d'arrondi et la précision machine (simple/double précision) peuvent également influencer les résultats.

## 3 Effet de l'orientation de la fissure inclinée sur SIFs :

Dans cet exemple, le même modèle numérique est considéré : une plaque carré isotrope de largeur $\boldsymbol{W}$, comportant une fissure interne inclinée de longueur **2a**, avec un rapport géométrique fixé à $\mathbf{a/W=0.25}$, la plaque est soumise à des contraintes de traction uniformes « $\sigma$=1 », appliquées sur les deux côtés opposés.

La géométrie et les principales caractéristiques dimensionnelles sont illustrées dans la Figure 4-1.

Les propriétés mécaniques du matériau restent inchangées, à savoir un comportement linéaire élastique caractérisé par le module de Young ***E*** et le coefficient de Poisson ν. L'épaisseur étant supposée négligeable au regard des autres dimensions de la plaque.
Une modélisation bidimensionnelle est retenue.

Contrairement au premier cas d'étude, Les facteurs d'intensité de contrainte en mode **I** et **II**, notés **KI** et **KII**, sont calculées au niveau de la pointe de la fissure pour un angle d'inclinaison ***β*** variant de 0°à180° par un incrément de 5°

La mesure est effectuée successivement dans le sens des axes « *x* » puis « *y* », et dans le sens antihoraire. Et comme au premier cas d'étude, les valeurs obtenues sont comparées aux solutions théoriques sous chargement en traction uniaxiale.

Le tableau 4-4 présente les valeurs mesurées des facteurs d'intensité de contrainte en mode I et II pour différentes valeurs de l'angle d'inclinaison ***β,*** ainsi que l'erreur relative par rapport aux solutions théoriques :

| Angle β | Normalized KI calculated | Theoretical normalized KI | % error | Normalized KII calculated | Theoretical normalized KII | % error |
|---|---|---|---|---|---|---|
| 30° | 0,7308 | 0,7500 | 2,56 | 0,4210 | 0,4330 | 2,79 |
| 60° | 0,2418 | 0,2500 | 3,28 | 0,4239 | 0,4330 | 2,10 |
| 120° | 0.23928 | 0.2500 | 4.28 | -0.41586 | -0,4330 | 3.96 |
| 150° | 0.72854 | 0.7500 | 2.86 | -0.41996 | -0,4330 | 3.01 |

**Tableau 4-4 C**convergence des SIF normalisés pour une fissure centrale incliné

Les valeurs normalisés des facteurs d'intensité de contrainte en mode I et II, présentées dans tableau 4-4, montrent une convergence satisfaisante avec un taux d'erreur acceptable (environ 5%) par rapport aux solutions théoriques.

La Figure 4-3 présente l'évolution de la valeur normalisé des facteurs d'intensité de contrainte en mode I. on observe une décroissance marqué du $K_I$ jusqu'à son annulation dans l'intervalle des angles d'inclinaison de la fissure compris entre 0°et 90°. Cette évolution s'explique par transition physique de l'état de la fissure : les lèvres sont totalement ouvertes pour $\beta$=0°, puis ferment graduellement jusqu'à une fermeture complète pour β=90°.

Un comportement inverse apparait pour les angles compris entre 90° et 180°, ou l'on constate une augmentation progressive du $K_I$ jusqu'au maximum, traduisant l'inversion de phénomène précédent, à savoir le passage d'un état totalement fermé à un état d'ouverture complète.

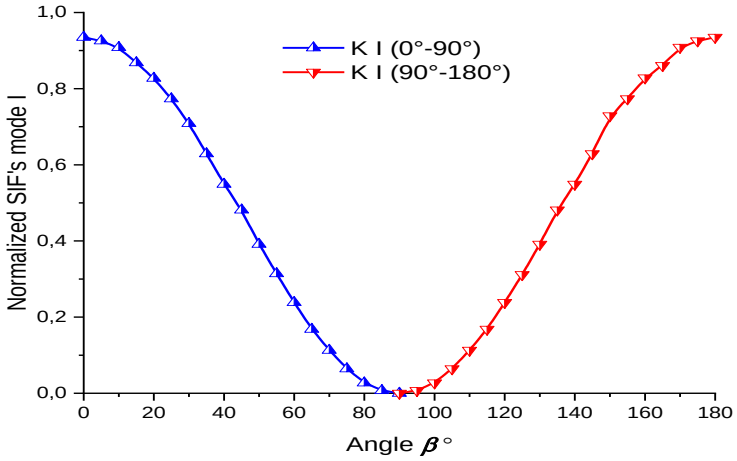



**Figure 4-3** Valeurs normalisées du SIF mode I en function de l'angle d'inclinaison " β " pour une plaque infinie comportant une fissure centrale inclinée sous traction uniaxiale..

Pour le facteur d'intensité de contrainte en mode II. $K_{II}$, le comportement présente des différences significatives (voir Figure 4-4). Pour un premier cas, une proportionnalité entre la valeur $K_{II}$ et l'angle $\beta$ est observée pour les angles compris entre 0° et 45°. Au-delà, cette relation est progressivement perdue entre 45° et 90°. Pour le deuxième cas, correspondant aux angles compris entre 90° et 180°, les valeurs normalisées de $K_{II}$ sont négatives, tout en conservant la même tendance générale d'évolution.

D'un point de vue mécanique, on peut interpréter ces résultats comme la présence d'un phénomène de glissement (slip) dans la zone positive de $K_{II}$, suivi d'un état de stabilité des lèvres de la fissure ($K_{II}$=0), un glissement de signe opposé dans la zone négative sur l'intervalle de 0° et 90°. Le même phénomène apparait pour l'intervalle de 90° et 180°, mais dans le sens inverse.

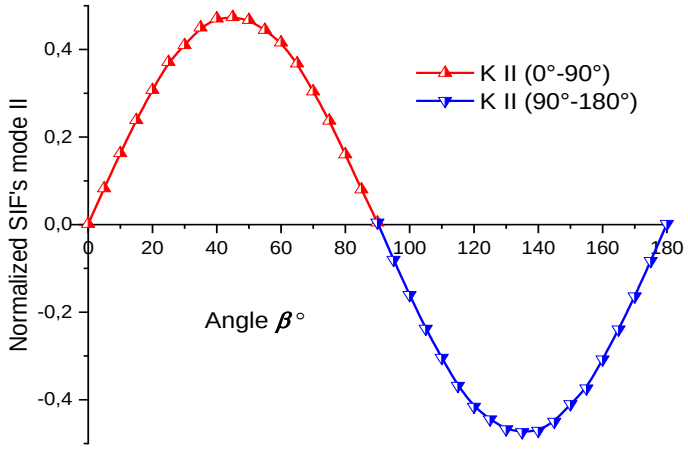



**Figure 4-4** Valeurs normalisées du SIF mode II en function de l'angle d'inclinaison " β " pour une plaque infinie comportant une fissure centrale inclinée sous traction uniaxiale..

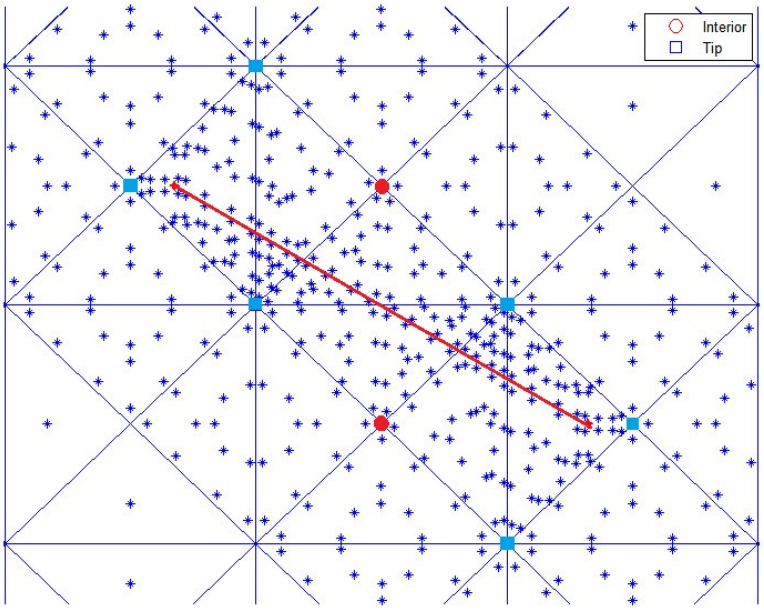



**Figure 4-5** Distribution des points de Gauss pour les éléments affectées par une fissure centrale inclinée dans une plaque infinie soumise à une traction uniaxiale..

.

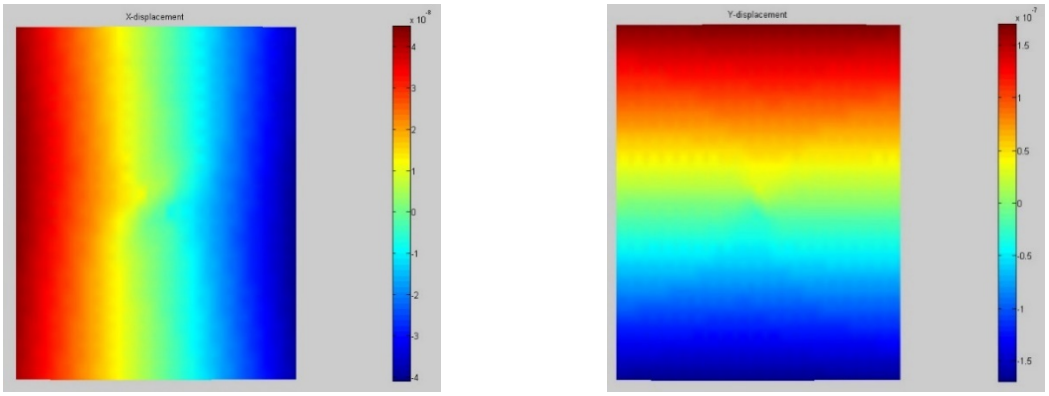



**Figure 4-6** Les déplacements selon XX et YY d'une plaque infinie carrée soumise à une traction uniaxiale munie d'une fissure centrale inclinée.

## 4 Deux fissures centrales inclinées similaires

Dans cette étude, on examine une plaque rectangulaire isotrope de dimensions ***L*=2*W***, comportant deux fissures internes inclinées de longueur **a/*W*=0.25.** La plaque est soumise à un chargement de traction uniforme « $\sigma$=1 », appliquées sur les deux bords supérieur et inférieur opposés.

La géométrie et les principales caractéristiques dimensionnelles sont illustrées dans la Figure 4-7. Les propriétés mécaniques du matériau sont présentées par un module de Young E et le coefficient de Poisson ν. L'épaisseur étant supposée négligeable par rapport aux autres dimensions de la plaque.

L'objectif de cette modélisation bidimensionnelle est de mettre en évidence l'efficacité et la robustesse de l'implémentation **OpenXFEM++** dans la modélisation de problèmes de fissuration complexe.

Le tableau 4-5 présente une comparaison des valeurs des facteurs d'intensité de contrainte mixte normalisées (MMSIF) en mode I et II, calculées aux extrémités « A » et « C », pour un angle d'inclinaison fixe $\beta=\alpha$=20° sous chargement de traction..

Une autre comparaison analogue est fournie dans le tableau 4-6, ou sont reportées les valeurs des (MMSIF) en mode I et II, calculées aux extrémités « B » et « D » pour une fissure centrale inclinée à $\beta=\alpha$=20° sous chargement de traction.

On compare ces valeurs normalisées des SIFs obtenues grâce au code **XFEM** orienté objet en **C++** avec celles publiées par Chen and Chen (1995) [87], obtenues via l'approche **Dual Boundary Element Method** (**DBEM**). Ces tableaux montrent aussi les variations relatives

entre les deux approches numériques, mettant en évidence une convergence remarquable et une concordance étroite.

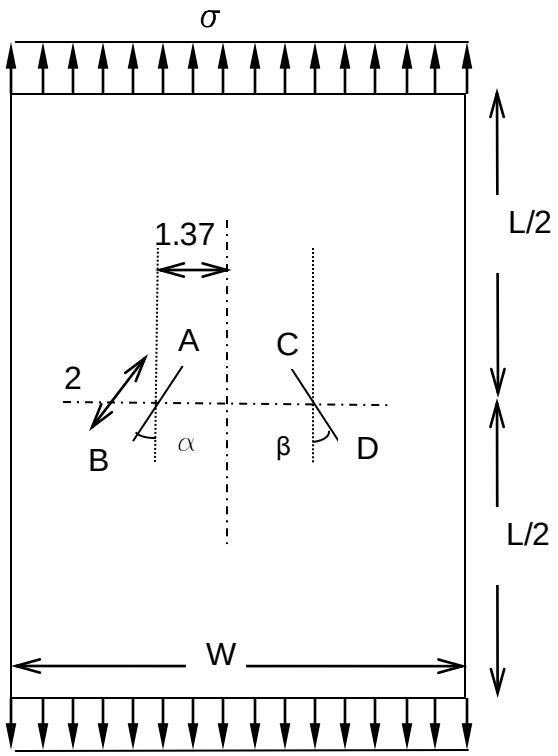


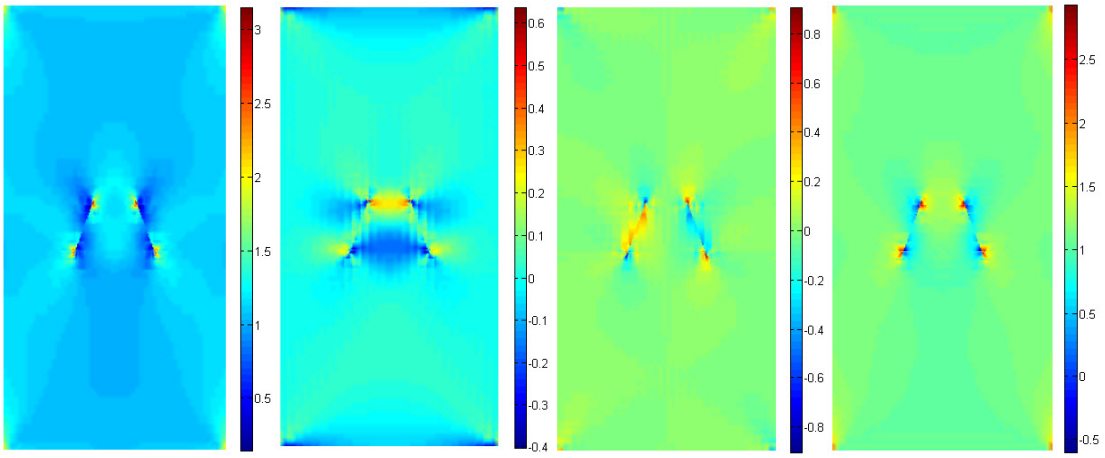

**Figure 4-7** Géométrie d'une double fissure inclinée dans une plaque rectangulaire soumise à un chargement de traction uniaxiale

| | Tip-A | | | Tip-C | | |
|---|---|---|---|---|---|---|
| K | (C++) | Ref | C++/ Ref (%) | (C++) | Ref | C++/ Ref (%) |
| K I | 0,14385 | 0,14497 | 0,77 | 0,14386 | 0,14483 | 0,67 |
| K II | 0,31649 | 0,32561 | 2,80 | 0,31650 | 0,32939 | 3,91 |

**Tableau 4-5** Convergence des facteurs d'intensité de contrainte mixte normalisées

| | Tip-B | | | Tip-D | | |
|---|---|---|---|---|---|---|
| K | (C++) | Ref | C++/ Ref (%) | (C++) | Ref | C++/ Ref (%) |
| K I | 0,10265 | 0,10479 | 2,04 | 0,10265 | 0,10354 | 0,86 |
| K II | 0,31935 | 0,33163 | 3,70 | 0,31936 | 0,33514 | 4,71 |

**Tableau 4- 6** Convergence des facteurs d'intensité de contrainte mixte normalisées




a) Von Mises stress b) $\sigma_{xx}$ c) $\sigma_{xy}$ d) $\sigma_{yy}$

**Figure 4-8** Champ de contrainte associés à une configuration à double fissures inclinées avec un angle d'inclinaison $\beta$=α=20°

## 5 Fissure centrale inclinée dans une plaque soumise à un chargement biaxial

Dans cet exemple, une étude menée sur une plaque carrée isotrope de longueur W, contenant une fissure centrale inclinée de longueur 2a=1. La plaque est soumise à un chargement biaxial de traction appliqué à distance sur les bords supérieur et inférieur.

La géométrie et les principales caractéristiques dimensionnelles sont illustrées dans la Figure 4-9. Les propriétés mécaniques du matériau sont définies par un module de Young E et le coefficient de Poisson ν.

L'épaisseur de la plaque est supposée négligeable au regard aux autres dimensions.

Deux charges normales sont appliquées : une charge de traction verticale unitaire $\sigma_y$=1 et une charge horizontale $\sigma_x$=$\lambda.\sigma_y$, où le paramètre $\lambda$ varient dans l'intervalle $-2, \leq, \lambda \leq 2$, avec un pas de 0.5, de façon d'un régime de compression à un régime de traction.

Les valeurs des facteurs d'intensité de contrainte en modes I et II sont évaluées pour des angles d'inclinaison de fissure $\beta$ allant de 5° à 85° avec un incrément de 5°.

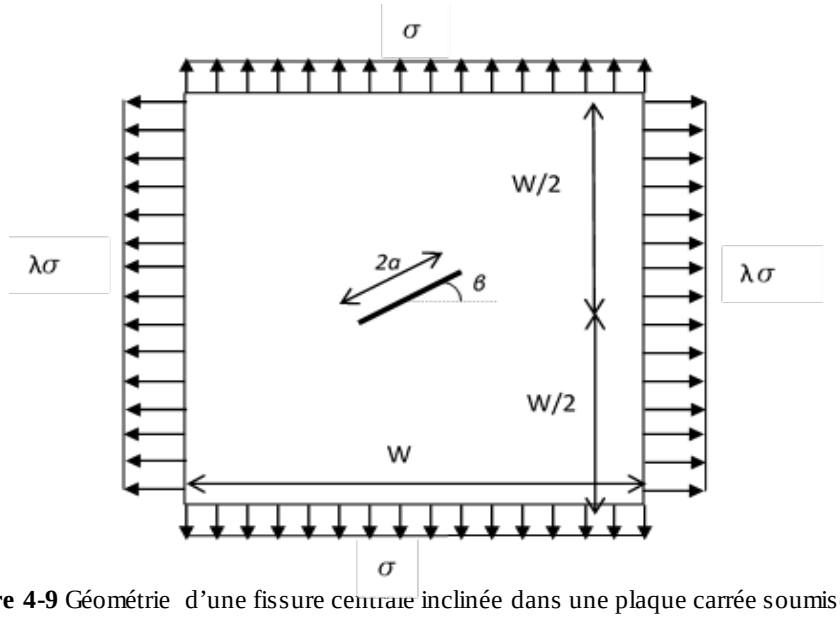



**Figure 4-9** Géométrie d'une fissure centrale inclinée dans une plaque carrée soumise à un chargement de traction biaxial

| | λ = -2 | | λ = 0 | | λ = 2 | |
|---|---|---|---|---|---|---|
| Angle | K I (C++) | K I (Abaqus) | K I (C++) | K I (Abaqus) | K I (C++) | K I (Abaqus) |
| 15 | 0,8723 | 0,8061 | 0,9197 | 0,9178 | 0,9699 | 1,0511 |
| 30 | 0,2761 | 0,2520 | 0,7308 | 0,7460 | 1,0678 | 1,2219 |
| 45 | - | - | 0,4830 | 0,5119 | 1,2230 | 1,5076 |
| 60 | - | - | 0,2418 | 0,2271 | 1,3549 | 1,2809 |
| 75 | - | - | 0,0663 | 0,0696 | 1,4838 | 1,4390 |

**Tableau 4-7** Convergence des SIFs normalisées en mode I pour une fissure centrale inclinée sous chargement biaxial

| | λ = -2 | | λ = 0 | | λ = 2 | |
|---|---|---|---|---|---|---|
| Angle | K II (C++) | K II (Abaqus) | K II (C++) | K II (Abaqus) | K II (C++) | K II (Abaqus) |
| 15 | 0,7968 | 0,7651 | 0,9197 | 0,9178 | -0,1796 | -0,1547 |
| 30 | 1,3437 | 1,3097 | 0,7308 | 0,7460 | -0,2904 | -0,2885 |
| 45 | 1,5431 | 1,5172 | 0,4830 | 0,5119 | -0,3279 | -0,3350 |
| 60 | 1,3239 | 1,3176 | 0,2418 | 0,2271 | -0,2652 | -0,2927 |
| 75 | 0,7652 | 0,7317 | 0,0663 | 0,0696 | -0,1585 | -0,2021 |

**Tableau 4-8** Convergence des SIFs normalisées en mode II pour une fissure centrale inclinée sous chargement biaxial

Comme indiqué dans les tableaux 4-**7** et 4-**8**, les valeurs des SIFs normalisés obtenues à l'aide du code **XFEM** orienté objet en **C++** présentent une excellente convergence et sont en très bon accord avec celles calculées avec le logiciel Abaqus.

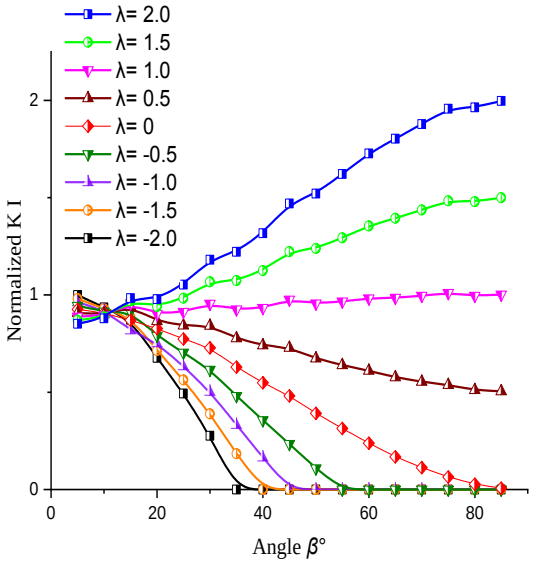



**Figure 4-10** Normalized SIFs mode I values vs horizontal loading controlled by factor λ and angle of inclination "β" for an infinite plate with an inclined center crack under vertical tensile loading

La Figure 4-10 représente l'évolution du facteur d'intensité de contrainte en mode I, normalisé, sous divers conditions de chargement biaxial, révélant trois régimes de comportement distincts :

a) Augmentation de KI : pour λ > 1, une augmentation nette de KI avec β, traduisant qui une ouverture accrue de la fissure sous l'effet d'une traction horizontale dominante.
b) Diminution modéré avec maintien de l'ouverture : pour **λ** = 0 et **λ** = 0,5, KI diminue mais demeure non nul pour toutes les valeurs de ***β***, indiquant une ouverture en mode mixte en cohérence avec les résultats de la section 1.
c) Transition vers la fermeture de la fissure : pour les cas dominés par la compression (λ = -0,5, -1, -1,5, -2), KI décroît en passant par zéro, traduisant une fermeture progressive de la fissure. l'angle critique de fermeture diminue lorsque la compression augmente et atteint environ 55°, 45°, 40° et 35° pour les valeurs de λ respectivement

Une observation notable est l'intersection de toutes les courbes autour de ***β***=10°, correspondant à un angle d'inclinaison neutre où l'effet de la charge horizontale sur l'ouverture de la fissure est

nul. Physiquement, cela correspond à un état ou l'interaction biaxiale est compensée. Pour $\beta$<10°, l'augmentation de $\lambda$ favorise la fermeture de la fissure, tandis $\beta$>10°, qu'au-delà de 10°, $\lambda$ favorise l'ouverture.

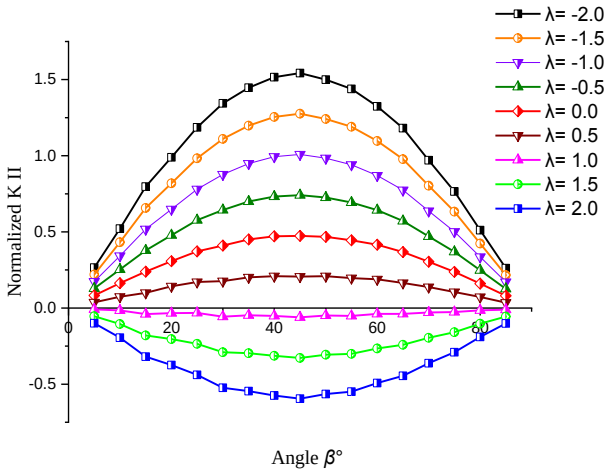



**Figure 4-11** La valeur des SIFs normalisés mode II en fonction du facteur de charge horizontale $\lambda$ et de l'angle d'inclinaison « $\beta$ » pour une plaque infinie avec une fissure centrale inclinée sous charge répartie.

La Figure 4-11 représente l'évolution du facteur $K_{II}$, normalisé, en fonction de $\lambda$ et de $\beta$. On y observe :

a) pour une compression horizontale $-2 < \lambda \leq -0{,}5$, $K_{II}$ croit avec $\beta$, atteint un maximum à 45°, puis décroit légèrement en raison des effets de compression.
b) pour $\lambda = 0$, la réponse est cohérente avec celle obtenue dans la section 1.
c) lorsqu'une traction horizontale domine $(0{,}5 < \lambda \leq 2)$, la réponse est similaire à celle dans la section 2, confirmant la cohérence du modèle (Zhu et al. 2006).

Indépendamment de la valeur de $\lambda$, $K_{II}$ atteint systématiquement un maximum aux, alentour de $\beta$=45°, puis évolue de façon monotone au-delà. Ces tendances confirment la forte sensibilité du mode mixte aux rapports de chargement biaxial et permettent d'identifier physiquement les changements d'orientation des lèvres de la fissure.

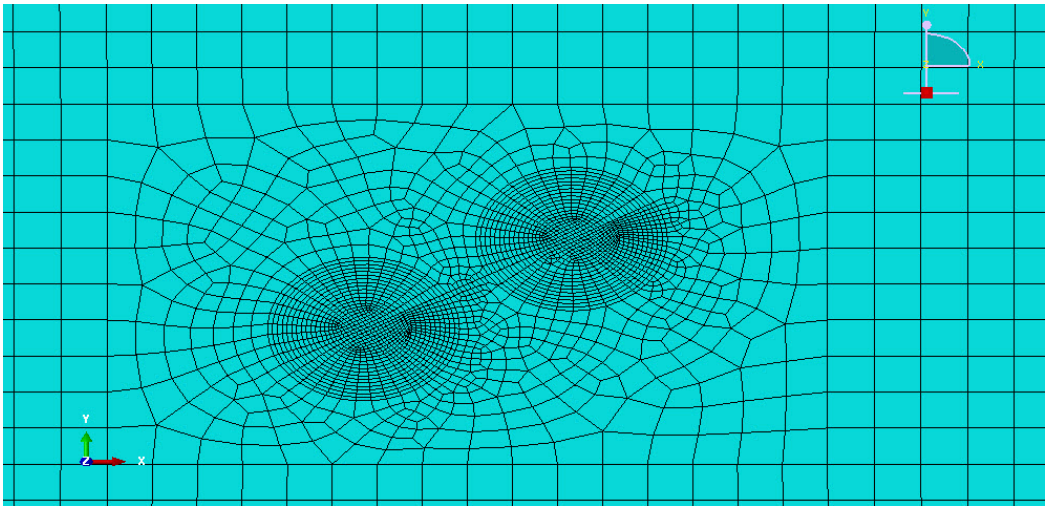


**Figure 4-12** discrétisation d'une fissure centrale incliné par Abaqus

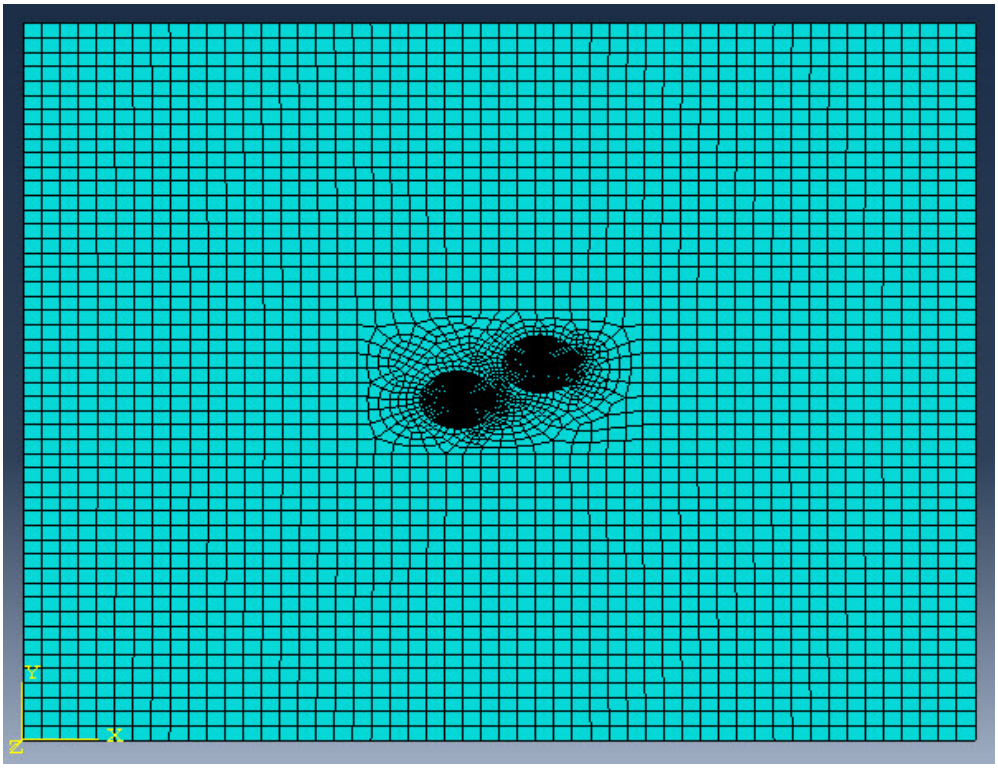


**Figure 4-13** discrétisation d'une fissure centrale inclinée dans une plaque carrée soumise à un chargement de traction biaxial par Abaqus

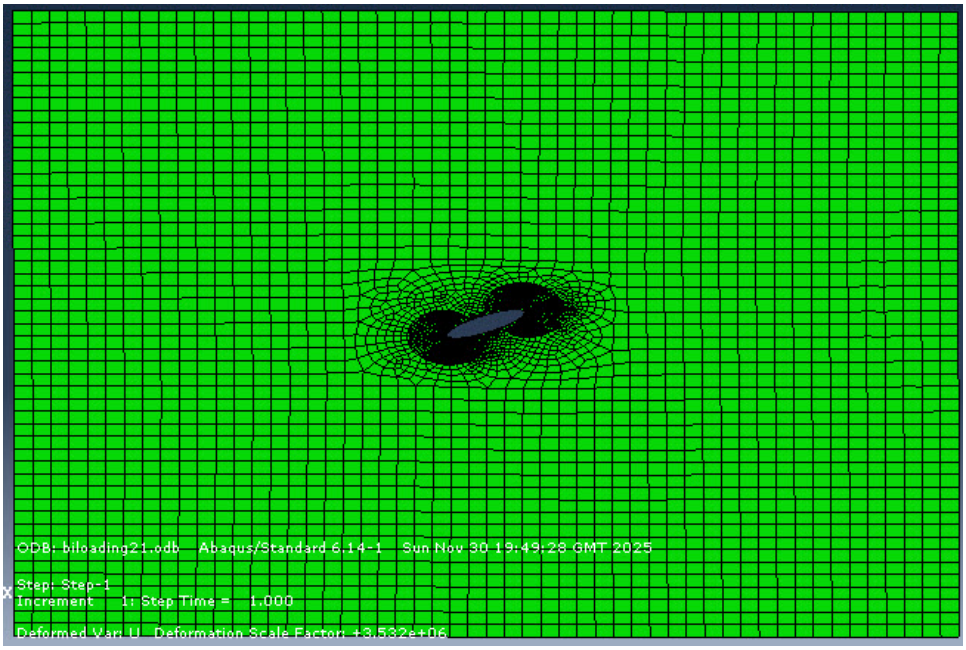



**Figure 4-14** Le déplacement U2 d'une plaque carrée soumise à un chargement de traction pure par Abaqus.

## 6 Conclusion

Ce chapitre a permet de valider la formulation XFEM implémentée dans la plateforme orientée objet OpenXFEM++ à travers plusieurs cas de références en mécanique de rupture. Les résultats obtenus présentent une excellente concordance avec les solutions analytiques, les données de la littérature et les simulations Abaqus. Cela confirme la précision, la solidité et la convergence indépendante du maillage de l'approche proposée. L'étude paramétrique a également révélé l'impact des conditions de chargement et des paramètres géométriques sur l'évolution des facteurs d'intensité de contrainte en mode mixte, démontrant de ce fait la pertinence de la plateforme utilisée pour l'analyse des structures fissurées.

# CHAPTRE 5 : MODELISATION NON-LINEAIRE PAR XFEM

## 1. Introduction

La modélisation élastoplastique dans le cadre de la méthode des éléments finis étendus (**XFEM**) représente une avancée méthodologique majeure pour la simulation numérique des structures fissurées en comportement ductile.

La mécanique de la rupture linéaire élastique (**LEFM**) expose ses limites dès que le phénomène d'écoulement plastique se développe à l'extrémité de la fissure, ce qui est typique pour les matériaux métalliques et de nombreux polymères.

Le couplage **XFEM**-plasticité [88] permet de surmonter cette limitation en combinant la capacité de **XFEM** à traiter la propagation de fissure sans remaillage avec la robustesse des modèles élastoplastiques pour saisir les processus irréversibles de déformation.

Le défi majeur est de développer des algorithmes numériques performants, notamment l'intégration des lois de comportement par le biais de l'algorithme retour radial et le calcul de matrice tangente consistante, qui assure la convergence optimale de Newton-Raphson dans un cadre d'enrichissement des fonctions de formes.

Cette approche pave la voie à une estimation plus réaliste de la durabilité en service des structures industrielles exposées à des charges sévères.

Il convient de préciser que, bien que le comportement du matériau soit modélisé par une loi élastoplastique non-linéaire de type Von Mises avec retour radial, les fonctions d'enrichissements utilisées au voisinage du fond de fissure de la formulation **XFEM** restent basées sur les solutions asymptotiques de la mécanique linéaire de la rupture champ classique de Williams/Westergaard [85.92] (exemple $B_1 = r^{\lambda} \cos\left(\theta/2\right)$ ).

En particulier, les enrichissements cinématiques adoptés correspondent à une singularité élastique en $\sqrt{r}$ , impliquant un exposant asymptotique $\lambda = 0.5$ .

Ce choix est justifié par l'hypothèse de plasticité confiné (small-**scall yielding**), selon laquelle la zone plastifiée demeure suffisamment petite devant les dimensions caractéristiques de la structure et de la fissure.

Dans ce cadre, la plasticité est prise en compte uniquement à travers la loi constitutive locale et la matrice tangentielle élasto-plastique cohérente, tandis que la cinématique enrichis au fond de fissure n'est pas modifiée.

L'integral de ***J*** conserve ainsi son rôle de paramètre énergétique de pilotage du champ proche de fissure, même en présence de non linéarités matérielles. Bien que dans un cadre théorique pleinement non linéaire, l'exposant asymptotique du champ de contrainte dépend de l'exposant

de l'écrouissage « $n$ » (paramètre matériau de la loi de Ramberg-Osgood) ou $\lambda = \frac{1}{1+n}$ (solution **H**utchinson-**R**ice-**R**osengren HRR) [88–90], cette dépendance n'est pas imposé cinématiquement .dans la présente formulation **XFEM**.

Les champs numériques obtenus résultent donc de l'interaction entre une singularité cinématique élastique et réponse matérielle élasto-plastique locale, ce qui constitue une approche standard largement adoptée dans la littérature.

## 2. Equations fondamentales de la mécanique des milieux continus :

### 2.1.Equations d'équilibre :

Considérons un solide fissuré occupant un domaine $\Omega$. Dont la frontière $\partial\Omega$ est partitionné en trois partis disjointes : $\Gamma_1$ ou les tractions sont imposées, $\Gamma_2$ ou les déplacements sont prescrits, $\Gamma_c$ qui présente les surfaces de la fissure .

L'équation différentielle aux dérivées partielles suivante gère l'équilibre statique d'un élément de volume :

$$\nabla.\sigma + \boldsymbol{f} = 0 \text{ dans } \Omega \qquad \textbf{(5-1)}$$

Ou : « $\sigma$ » représente le tenseur de contrainte de **cauchy** (symétrique $\sigma_{12} = \sigma_{21}$, etc..) et « $\boldsymbol{f}$ » le vecteur des forces volumiques ( exemple les forces d'inerties, gravité,…).

Cette équation (5-1) exprime la conservation de la quantité de mouvement dans le cas statique et découle directement de la deuxième loi de newton appliqué à un volume infinitésimale .

Physiquement le terme « $\nabla\sigma$ » représente la divergence du tenseur de contrainte, qui correspond à la résultante des forces de surface agissant sur l'élément. Tandis que « $\boldsymbol{f}$ » représente les forces de volume.

Dans un milieu fissuré, cette équation reste valable dans tout le domaine « $\Omega$ », mais la présence de la fissure introduit des conditions aux limites supplémentaires et potentiellement des singularités et des contraintes

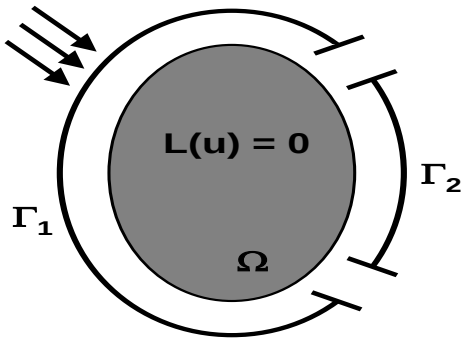



**Figure 5-1** L'état du problème

## 2.2.Conditions aux limites

Le problème aux limites complet s'exprime en trois types de conditions :

1- Conditions de **Neumann** (traction imposé) :

$$\sigma \,.\, \mathrm{n} = \bar{t} \quad \text{Dans } \Gamma_1 \qquad \textbf{(5-2)}$$

Cette condition spécifié les forces de surfaces « $\bar{t}$ » appliquées sur une partie de la frontière externe. Le vecteur « n » est la normale unitaire sortante à la surface. Cette condition est naturelle dans la formulation variationnelle du problème.

2- Conditions de **Dirichlet** (déplacement imposé) :

$$u = \bar{u} \quad \text{sur } \Gamma_2 \qquad \textbf{(5-3)}$$

Cette condition impose des déplacements spécifiques « $\bar{u}$ » sur une autre partie de la frontière externe. Les frontières $\Gamma_1$, $\Gamma_2$ doivent être complémentaires ($\bar{\Gamma}_1 \cup \bar{\Gamma}_2 = \partial\Omega / \bar{\Gamma}_c$ et $\bar{\Gamma}_1 \cap \bar{\Gamma}_2 = \varnothing$). Cette condition est essentielle pour éliminer les mouvements de corps rigide et assurer l'unicité de la solution [2].

3- Conditions sur la fissure (surface libre) :

$$\sigma \,.\, \mathrm{n} = 0 \quad \text{sur } \Gamma_c \qquad \textbf{(5-4)}$$

Cette condition stipule que les surfaces de la fissure « $\Gamma_c$ » sont libres de toute contrainte. C'est une condition de Neumann homogène. Dans les problèmes de fatigue ou de contact, cette condition a pour but de prendre en compte le contact entre les lèvres de la fissure. La modélisation précise de cette condition est cruciale pour une estimation correcte des facteurs d'intensité de contrainte [91].

## 2.3.Relations cinématiques

La relation entre le champ de déplacement « $u$ » et le champ de déformation « $\boldsymbol{\varepsilon}$ » est décrit par l'équation cinématique (5-2). Pour les petits déplacements et les petites déformations, on utilise le tenseur de déformation infinitésimale de Cauchy :

$$\varepsilon = \frac{1}{2}\left(\nabla u + (\nabla u)^{\mathrm{T}}\right) \qquad \textbf{(5-5)}$$

Ce tenseur, également symétrique ($\varepsilon_{ij} = \varepsilon_{ji}$), capture les changements de la longueur et d'angle dans le matériau. La partie symétrique du gradient de déplacement garantie l'objectivité matérielle dans le cadre des petites déformations.

## 2.4.Lois de comportement élastique linéaire

La relation constitutive reliée des déformations aux contraintes, pour un comportement élastique isotrope linéaire, la loi de Hooke généralisée s'applique :

$$\sigma = \mathrm{D}^{\mathrm{el}} : \varepsilon \qquad \textbf{(5-6)}$$

Où, « : » montre le produit qui a été contracté deux fois (indiquant qu'il a été doublement contracté). L'hypothèse de l'isotropie confère une configuration particulière au tenseur élastique isotrope du quatrième ordre « $\mathrm{D}^{\mathrm{el}}$ ». Il peut s'exprimer en fonction de seulement deux constantes indépendantes, les coefficients de Lamé « $\lambda$ » et « $\mu$ » :

$$\mathrm{D}^{\mathrm{el}}_{ijkl} = \lambda\delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) \qquad \textbf{(5-7)}$$

Où, « $\delta$ » est le symbole de Kronecker, le coefficient « $\mu$ » est le module de cisaillement, les coefficients de Lamé sont liées au module d'élasticité « $E$ » et au coefficient de Poisson « $\boldsymbol{\nu}$ » par les relations :

$$\lambda = \frac{E\nu}{\left[(1+\nu)(1-2\nu)\right]} \qquad \textbf{(5-8)}$$

$$\text{Et} \quad \mu = \frac{E}{\left[2(1+\nu)\right]} \qquad \textbf{(5-9)}$$

Cette relation constitutive est fondamentale pour décrire le comportement réversible du matériau en dehors de la zone plastifiée. Près de la pointe de la fissure, ou les déformations

peuvent devenir importantes et inélastiques, cette lois doit être remplacée par un modèle plus complexe, comme un modèle élasto-plastique [12, 92, 93].

## 3. Fondement de la théorie de la plasticité

### 3.1.Décomposition additives des déformations

La théorie de la plasticité en petites déformations repose sur l'hypothèse fondamentale de décomposition additive du tenseur de déformation totale. Cette hypothèse stipule que la déformation totale « $\varepsilon$ » peut être séparée en une partie élastique réversible « $\varepsilon^{el}$ » et une partie plastique irréversible « $\varepsilon^{p}$ ».

$$\varepsilon = \varepsilon^{el} + \varepsilon^{p} \qquad \textbf{(5-10)}$$

Sous forme incrémentale, cette décomposition s'écrit :

$$d\varepsilon = d\varepsilon^{el} + d\varepsilon^{p} \qquad \textbf{(5-11)}$$

La contribution élastique « $d\varepsilon^{el}$ » correspond à l'énergie stockée dans le matériau lors de la déformation, et est entièrement récupérable lors d'une décharge. En revanche, la partie plastique « $d\varepsilon^{p}$ » présente la déformation irréversible (permanente) due aux mécanismes irréversibles, tels que le glissement et mouvement de dislocation dans les matériaux.

Cette décomposition permet de décrire séparément le comportement réversible et irréversible dans la loi constitutive, et constitue le fondement de la formulation élastoplastique.

### 3.2.Lois de comportement incrémentale

Pendant l'écoulement plastique, les déformations plastiques dépendent du chemin de chargement. Par conséquence il n'existe pas de relation fonctionnelle unique entre contraintes et déformations. La loi constitutive doit donc exprimer sous forme différentielle ou incrémentale.

Dans un régime élasto-plastique. La loi de Hooke est appliquée uniquement à la partie élastique de l'incrément de déformation :

$$d\sigma = \mathrm{D}^{el} : d\varepsilon^{el} \qquad \textbf{(5-12)}$$

Ou « $\mathrm{D}^{el}$ » est le tenseur d'élasticité du quatriéme ordre. En utilisant la décomposition additive , on obtient :

$$\Rightarrow \qquad d\sigma = \mathrm{D}^{el} : (d\varepsilon - d\varepsilon^{p}) \qquad \textbf{(5-13)}$$

Cette formulation souligne que l'incrément de contrainte est uniquement due à l'incrément de déformation élastique .

La composition plastique de la déformation ne contribue pas à la l'augmentation de la contrainte, mais représente une dissipation d'énergie irréversible

## 3.3.Surface de charge et critères d'écoulement

La surface de charge délimite le domaine élastique dans l'éspace des contraintes. Elle est définie par une fonction scalaire qui dépend du tenseur des contraintes « $\sigma$ » et du variable « $\kappa$ » qui est une variable interne représentant l'écrouissage: $f(\sigma,\kappa)=0$,

- Le domaine élastique est définie par $f(\sigma,\kappa)<0$.
- Lorsque le point de contrainte atteint la surface $f(\sigma,\kappa)=0$, une déformation plastique supplémentaire peut se produire. L'évolution de cette surface avec l'écrouissage est cruciale pour décrire le comportement post-élastique des matériaux.

## 3.4.Loi d'écoulement plastique

La direction d'écoulement plastique qui exprime la direction du taux de la déformation plastique est définie par la loi d'écoulement. Pour un critére associé [93, 94]:

$$d\varepsilon^{p} = d\gamma \cdot \frac{\partial f}{\partial \sigma} \qquad \textbf{(5-14)}$$

Ou $d\gamma$ est un multiplicateur plastique et il est déterminé par l'application de la condition de consistance :

$$d\gamma = \frac{n(\sigma)\cdot \mathrm{D}\cdot d\varepsilon}{n(\sigma)\cdot \mathrm{D}\cdot n(\sigma)} \qquad \textbf{(5-15)}$$

Avec $n(\sigma)$ est le vecteur d'écoulement plastique, normale à la surface de charge dans l'espace des contraintes, donné comme suit

$$n(\sigma) = \frac{\partial f}{\partial \sigma} \qquad \textbf{(5-16)}$$

Cette normalité de l'écoulement plastique est une caractéristique fondamentale de la plasticité associée et est liée au principe du travail plastique maximal .

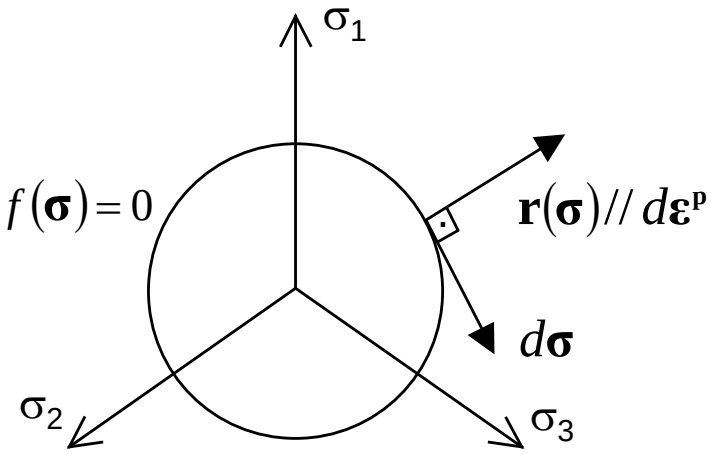



**Figure 5-2** surfaces d'écoulement dans le plan déviatorique

Comme résultats de la loi d'écoulement plastique ; « $d\varepsilon^p$ » et « $d\sigma$ » ne sont pas généralement colinéaires, contrairement au cas élastique

### 3.5. Condition de Khun-Tucker :

Le point de contrainte doit rester dans \ ou à l'intérieur de la surface d'écoulement, autrement dit, un chargement appliqué dans un état plastique conduit à un nouvel état plastique. Ainsi, les conditions de charge et de décharge en plasticité sont régies par les conditions de complémentarité **de Khun-Tucker**

$$f(\sigma,\kappa) \le 0 \qquad \mathrm{d}\gamma \ge 0 \qquad \mathrm{d}\gamma \cdot f(\sigma,\kappa) = 0 \qquad \textbf{(5-17)}$$

Ces conditions impliquent que :

- La déformation plastique n'apparait qu'au contact de la surface de charge.
- L'état de contrainte ne peut jamais dépasser la surface de charge (domaine admissible))
- L'écoulement plastique ne peut se produire que le point de contrainte se trouve exactement sur la surface de charge.

Ces conditions définissent un problème de complémentarité qui régit la transition entre comportement élastique et plastique.

### 3.6.Condition de consistance

Pendant l'écoulement plastique, le point de contrainte doit rester sur la surface de charge. Cette condition s'exprime mathématiquement par la condition de consistance :

$$f(\sigma,\kappa) = 0 \qquad \textbf{(5-18)}$$

Pendant la phase plastique, un nouvel incrément de chargement ne doit pas faire sortir le point de contrainte en dehors de la surface de charge. Donc, à l'incrément suivant :

$$f(\sigma + d\sigma, \kappa + d\kappa) = 0 \qquad \textbf{(5-19)}$$

En développant à l'ordre 1 :

$$df(\sigma, \kappa) = 0 \qquad \textbf{(5-20)}$$

Ce résultat s'appelle la condition de consistance.

Après la dérivation ce cette expression par rapport aux variables internes, on trouve la forme générale comme suit :

$$df = \frac{df}{d\sigma} d\sigma : \frac{df}{d\kappa} d\kappa = 0 \qquad \textbf{(5-21)}$$

Sachant que : le vecteur d'écoulement (la normale à la surface)est

$$n = \frac{df}{d\sigma} \qquad \textbf{(5-22)}$$

Et le paramètre d'écrouissage (positif) est

$$\mathrm{H} = -\frac{df}{d\kappa} \qquad \textbf{(5-23)}$$

Donc, en replace dans (5-20) :

$$n\, d\sigma - \mathrm{H}\, d\kappa = 0 \qquad \textbf{(5-24)}$$

Cette équation garantie que lors d'un incrément de déformation plastique, la contrainte évolue de telle sorte que la condition ( $f = 0$ ) reste satisfaite. Elle permet de déterminer le multiplicateur plastique ( $d\gamma$ ) et assurer la cohérence mathématique du modèle.

## 3.7.Représentation conceptuelle de la surface de charge

La surface de charge est visualisée comme une hypersurface dans l'espace des contraintes de six dimensions. Pour un état de contrainte donné « s », on distingue trois cas comme :

- Si $f < 0$ : le comportement est purement élastique.
- Si $f = 0$ : une déformation plastique peut se produire.( état limite)
- Si $f > 0$ inadmissible : correction plastique requise (Les conditions de **Khun-Tucker** assurent que le point de contrainte ne peut jamais se trouver à l'extérieur de cette surface)

Le vecteur d'écoulement plastique défini dans l'équation (5-22), est normal à la surface, indiquant la direction d'écoulement.

## 4. Critère de Von Mises

La théorie de plasticité de Von Mises constitue la base fondamentale de la modélisation du comportement des matériaux ductiles dans le domaine de la mécanique de rupture. Elle stipule que l'écoulement plastique s'initie lorsque l'énergie de distortion atteind une limite critique, indépendament de la pression hydrostatique.

En ce qui concerne les structures fissurées, cette théorie offre la possibilité d'évaluer la formulation et la croissance de la zone plastique à la pointe de la fissure, un phénoméne essentiel qui contrôle la résistance à la propagation de celle-ci.

### 4.1.Formulation du critère

Le critère de Von Mises correspond à une formulation standard de la plasticité des métaux en petites déformations : tant que la norme du tenseur déviatorique ne franchit pas un seuil (yield), les métaux demeurent élastiques

Sur le plan mathématique, Le critère de Von Mises suggére que l'écoulement plastique se produit lorsque le second invariant « $J_2$ » (théorie de l'énergie de distorsion) du tenseur de contrainte déviatorique « **s** » atteint une valeur critique « $k^2$ », (une propriété matérielle).

Pour un matériau isotrope et parfait, la fonction de charge est exprimée par :

$$f(\sigma) = \sqrt{J_2} - k = 0 \qquad \textbf{(5-25)}$$

La valeur « $k$ » présente la limite d'écoulement en cisaillement pure.

$$k = \frac{\sigma_y}{\sqrt{3}} \qquad \textbf{(5-26)}$$

Replaçant dans (26) on trouve

$$f(\sigma) = \sqrt{3J_2} - \sigma_y = 0 \qquad \textbf{(5-27)}$$

$$f(\sigma) = \sigma_{eq} - \sigma_y \qquad \textbf{(5-28)}$$

Ou « $s_{eq}$ » est la contrainte équivalente de Von Mises

$$\Rightarrow \qquad \sigma_{eq} = \sqrt{3J_2} \qquad \textbf{(5-29)}$$

Ainsi que « $J_2$ » le second invariant du tenseur de contrainte déviatorique est défini par :

$$\Rightarrow \qquad J_2 = \frac{1}{2} s : s \qquad \textbf{(5-30)}$$

Et « **s** » est le tenseur de contrainte déviatorique.

$$\Rightarrow \qquad s_{ij} = \sigma_{ij} - \frac{1}{3}.\mathrm{tr}(\sigma).\boldsymbol{I} \qquad \textbf{(5-31)}$$

$$\Rightarrow \qquad \text{ou } \mathrm{tr}(\sigma) = \sigma_{11} + \sigma_{22} + \sigma_{33} \qquad \textbf{(5-32)}$$

## 4.2. Vecteur d'écoulement plastique

Pour le critère Von Mises avec règle d'écoulement associé, le vecteur d'écoulement plastique est donné comme suit :

$$\Rightarrow \qquad n = \frac{\partial f}{\partial \sigma} = \frac{3}{2}\frac{s}{\sigma_{eq}} \qquad \textbf{(5-33)}$$

Ce vecteur est purement déviatorique, ce qui implique que la déformation plastique s'effectuée sans changement de volume (incompressibilité plastique). Cette propriété est cohérente avec les observations expérimentales sur les métaux qui se déforment plastiquement sans variation volumique significative.

Sachant que Le tenseur des contraintes totales est :

$$\Rightarrow \qquad \sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & 0 \\ \sigma_{21} & \sigma_{22} & 0 \\ 0 & 0 & \sigma_{33} \end{bmatrix} \qquad \textbf{(5-34)}$$

Pour un état de contrainte plane généralisé ($\sigma_{12} = \tau_{12}$)

$$\Rightarrow \; J_2 = \frac{1}{6}\left[(\sigma_{11} - \sigma_{22})^2 + (\sigma_{22} - \sigma_{33})^2 + (\sigma_{33} - \sigma_{11})^2\right] + \tau_{12}^2 \qquad \textbf{(5-35)}$$

$\Rightarrow$ La fonction de Von Mises (non normalisé) est

$$\Rightarrow \qquad f(\sigma) = \sqrt{3J_2} \;\rightarrow\; f(\sigma) = \sqrt{\frac{3}{2} s_{ij} \cdot s_{ij}} \qquad \textbf{(5-36)}$$

**La dérivée du critère par rapport à $\sigma$**

⇨ Le gradient du critère $n(\sigma)$ nécessaire pour le calcul de la direction d'écoulement plastique est donné par :

$$\Rightarrow \qquad n(\sigma)=\frac{\mathrm{d}\,f}{\mathrm{d}\sigma}=\frac{3}{2}\frac{\mathrm{s}_{ij}}{f} \qquad \textbf{(5-37)}$$

$$\Rightarrow \qquad \text{Donc : } n(\sigma)=\frac{\mathrm{d}\,f}{\mathrm{d}\sigma}=\frac{1}{2\sqrt{J_2}}\mathrm{s}_{ij} \qquad \textbf{(5-38)}$$

Expression composante par composante du tenseur des contraintes déviatorique est

$$\Rightarrow \quad \begin{cases} \mathrm{s}_{11}=\sigma_{11}-\dfrac{1}{3}.\mathrm{tr}\left(\sigma_{11}+\sigma_{22}+\sigma_{33}\right).\boldsymbol{I}=\dfrac{\left(2\sigma_{11}-\sigma_{22}-\sigma_{33}\right)}{3} \\ \mathrm{s}_{22}=\dfrac{\left(2\sigma_{22}-\sigma_{11}-\sigma_{33}\right)}{3} \\ \mathrm{s}_{33}=\dfrac{\left(2\sigma_{33}-\sigma_{11}-\sigma_{22}\right)}{3} \\ \sigma_{12}=\tau_{12} \end{cases} \qquad \textbf{( 5-39)}$$

Donc, une fois effectué le remplacement, les composants du gradient du critère sont présentées de cette manière :

$$\Rightarrow \quad \begin{cases} \dfrac{\mathrm{d}\,f}{\mathrm{d}\sigma_{11}}=\dfrac{1}{2\sqrt{J_2}}\mathrm{s}_{11}=\dfrac{\left(2\sigma_{11}-\sigma_{22}-\sigma_{33}\right)}{3\sqrt{J_2}} \\ \dfrac{\mathrm{d}\,f}{\mathrm{d}\sigma_{22}}=\dfrac{1}{2\sqrt{J_2}}\mathrm{s}_{22}=\dfrac{\left(2\sigma_{22}-\sigma_{11}-\sigma_{33}\right)}{3\sqrt{J_2}} \\ \dfrac{\mathrm{d}\,f}{\mathrm{d}\sigma_{33}}=\dfrac{1}{2\sqrt{J_2}}\mathrm{s}_{33}=\dfrac{\left(2\sigma_{33}-\sigma_{11}-\sigma_{22}\right)}{3\sqrt{J_2}} \\ \dfrac{\mathrm{d}\,f}{\mathrm{d}\tau_{12}}=\dfrac{3\tau_{12}}{\sqrt{J_2}} \end{cases} \qquad \textbf{(5-40)}$$

Pour les matériaux, ce critère est particulièrement approprié car il est indépendant de la pression hydrostatique, ce qui correspond au comportement observé expérimentalement ou la déformation plastique est principalement gouvernée par les conditions déviatorique.

## 4.3.Implémentation numérique

L'illustration théorique évoquée précédemment a été implémenté via une classe nommée « vonmisesmaterial » issue du code **OpenXFEM++**, cette dernière constitue une solution performante pour les matériaux faiblement écrouissants et les zones plastiques limitées. Dont le calcul de la fonction de charge est effectué par la méthode **computeYieldFunctionFor**() indiqué ci-dessous :

```
double VonMisesMaterial :: computeYieldFunctionFor (FloatArray* sigma)
{     double answer;
  double k = this→give('k');
  double j2 = sigma→computeInvariantJ2();
  answer = sqrt(j2) - k;
  return answer;       }
```

Pour que la méthode **computeYieldFunctionFor**() renvoie la valeur de la fonction de charge (sqrt(j2) – k), on fait appel à La méthode **computeInvariantJ2**() de la classe **flotarry** pour déterminer le second invariant du tenseur de contrainte déviatorique « $J_2$ », ainsi que la limite d'écoulement en cisaillement pure « k » est extraite du fichier d'entrée.

Ces composants sont implémentées dans la méthode **computeDFDSigma()**

```
FloatArray* VonMisesMaterial :: computeDFDSigma
(FloatArray* r)
{     FloatArray *answer = new FloatArray(r->giveSize());
  double a11 = (*r)[0];
  double a22 = (*r)[1];
  double a12 = (*r)[2];
  double a33 = (*r)[3];
  double j2 = r->computeInvariantJ2();
  if (j2 == 0) return answer;
  double factor = 1.0e0 / (6.0e0*sqrt(j2));
  (*answer)[0] = factor * ((2.0e0*a11) - a22 - a33);    //sxx
  (*answer)[1] = factor * ((2.0e0*a22) - a11 - a33);     //syy
  (*answer)[2] = factor * (6.0e0 * a12);                 //sxy
  (*answer)[3] = factor * ((2.0e0*a33) - a22 - a11);      //szz
  return answer;      }
```

## 4.4.Interprétation géométrique

Dans l'espace des contraintes principales ($\sigma_1, \sigma_2, \sigma_3$), la surface de Von Mises se représente sous la forme d'un cylindre circulaire dont l'axe correspond à la ligne hydrostatique ($\sigma_1=\sigma_2=\sigma_3$). Le rayon du cylindre est : $R=\sqrt{2}k$. Ou ($k$) est la limite d'écoulement en cisaillement pur, ce qui reflète (cette géométrie) l'indépendance du critère par rapport à la pression hydrostatique.

Dans le plan déviatorique (plan perpendiculaire à l'axe hydrostatique), la surface de charge apparait comme un cercle de rayon ($\mathrm{R}=\sqrt{\frac{2}{3}}\sigma_{\mathrm{y}}$), ou « $\sigma_{\mathrm{y}}$ » correspond à la limite d'écoulement en traction uniaxiale selon la relation $\mathrm{k}=\frac{\sigma_{\mathrm{y}}}{\sqrt{3}}$.

⇨ Cette dernière formulation est la plus courante en pratique industrielle, ou dans un essai traction uniaxiale, l'état de contrainte ($\sigma_1=\sigma_{\mathrm{y}}$, ($\sigma_2=\sigma_3=0$), et le tenseur déviatorique est : ($\mathrm{s}_1=\frac{2}{3}.\sigma_{\mathrm{y}}$, $\mathrm{s}_2=\mathrm{s}_3=-\frac{1}{3}.\sigma_{\mathrm{y}}$), ce qui donne l'expression du second variant comme suit :

⇨ $$J_2=\frac{1}{2}\left[\mathrm{s}_1^2+\mathrm{s}_2^2+\mathrm{s}_3^2\right]=\frac{1}{3}\sigma_{\mathrm{y}}^2 \qquad \textbf{( 5-41)}$$

Donc le critère de Von Mises donne :

$$\sqrt{J_2}=\frac{\sigma_{\mathrm{y}}}{\sqrt{3}}=k \qquad \textbf{(5-42)}$$

Ou sous une autre forme :

$$f\left(\sigma\right)=\sqrt{3J_2}-\sigma_{\mathrm{y}}^{\left(\varepsilon^{\mathrm{p}}\right)}=0 \qquad \textbf{(5-43)}$$

Ou :

$\sigma_{\mathrm{eq}}=\sqrt{3J_2}$ Est la contrainte équivalente de Von Mises

$\sigma_{\mathrm{y}}^{\left(\varepsilon^{\mathrm{p}}\right)}=\sigma_{\mathrm{y0}}+H^{\left(\varepsilon^{\mathrm{p}}\right)}$ Pour écrouissage linéaire

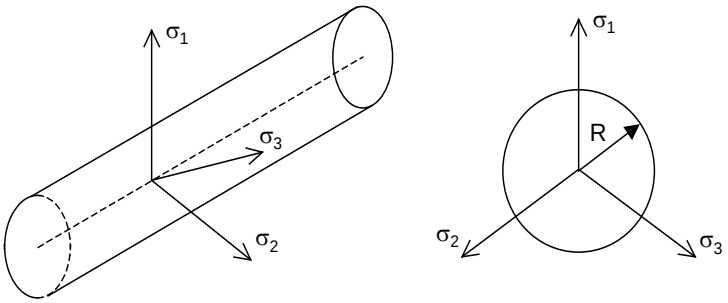



**Figure 5-3** La surface d'écoulement de Von Mises dans l'espace de contrainte et dans le plan déviatorique

## 5. Intégration numérique : algorithme de retour radial

### 5.1.prédiction élastique

L'algorithme de retour radial (return mapping algorithm) est la méthode couramment utilisée pour l'intégration des lois élastoplastiques. Il se divise en une étape de prédiction élastique suivie d'une étape de correction plastique.

L'étape de prédiction élastique (contrainte d'Essai) présuppose un comportement purement élastique lors de l'incrément (première itération), ou elle est utilisée comme un point initial stable pour l'algorithme Newton-Raphson, facilitant ainsi le contrôle d'un éventuel écoulement plastique [93].

Pour déterminer La contrainte d'essai $\sigma_{n+1}^{tr}$ associée à l'itération « i » et au pas « n+1 », l'algorithme général de la contrainte de retour peut être formulé comme suit :

$$\sigma_{n+1}^{tr} = \sigma_n + d\sigma^{tr} = \sigma_n + D^{el} : \Delta\varepsilon \qquad \textbf{(5-44)}$$

Ou : « $\sigma_n$ » contrainte de l'étape précédente, et « $\Delta\boldsymbol{\varepsilon}$ » : incrément de déformation total calculé à partir des déplacements nodaux d'Element,

### 5.2.Vérification du critère d'écoulement (plasticité) :

Le processus consiste à :

- Calculé la contrainte élastique d'essai ($\boldsymbol{\sigma}^{trial}$).

- Vérifier si elle est en dehors de la surface de charge comme suit :

Si $f\left(\sigma_{n+1}^{tr}, \overline{\varepsilon}_n^p\right) \leq 0$ , le comportement est élastique , donc : $\sigma_{n+1} = \sigma_{n+1}^{tr}$

Si $f\left(\sigma_{n+1}^{tr}, \overline{\varepsilon}_n^p\right) > 0$ , la contrainte d'essai est à l'extérieur de la surface de charge, donc un écoulement plastique se produit, nécessitant ainsi une correction plastique, pour projeter le point de contrainte sur la surface de charge.

Cette partie théorique est implémentée dans la méthode **ComputeStress**(), Ou la contrainte de l'étape précédente « $s_n$ » est appelée grâce à la méthode **givePreviousStressVector**() de la classe **gausspnt**, ainsi que incrément de déformation total « $\Delta\boldsymbol{\varepsilon}$ » : est calculé via la méthode **computeStrainIncrement**() issue de la classe **Element.**

```
// incrément de contrainte correspond to dxacc = delta x accumulé
//(depuis la dernière étape de convergence)
  deltaEpsilon = elem->computeStrainIncrement(gp,dElem);
  // delta sigma
  sigmaTrial = Del->Times(deltaEpsilon);
  // sigma(prev)+delta sigma
  sigmaTrial->add(gp->givePreviousStressVector());
  // calcul de f(sigmaTrial)
  double fSigTr = this->computeYieldFunctionFor(sigmaTrial);
```

## 5.3.Correction plastique principe du retour radial

La correction plastique nécessite l'imposition de la consistance globale afin de ramener le point sur la surface ( $\boldsymbol{f} = 0$ ) dans la direction normale donnée par ( $\frac{\partial \boldsymbol{f}}{\partial \sigma}$ ), donc :

$$\sigma_{n+1} = \sigma_{n+1}^{tr} - \Delta\gamma \, D^{el} : r \qquad \textbf{(5-45)}$$

Pour un écrouissage linéaire, le multiplicateur plastique $\Delta\gamma$ est déterminé en imposant la condition de consistance au pas te temps « n+1 ».

$$\Delta\gamma = \frac{f\left(\sigma_{n+1}^{trial}\right)}{\left(\frac{\partial f}{\partial \sigma}\right)^{T} D^{el} \left(\frac{\partial f}{\partial \sigma}\right)} \qquad \textbf{(5-46)}$$

Ce qui présente le retour radial dans la direction du gradient du critère qui est définie comme suit :

$$r = \frac{\partial f}{\partial \sigma} = \frac{s^{trial}}{2\sqrt{J_2^{trial}}} \qquad \textbf{(5-47)}$$

Cette formule est implémentée par la fonction (**dFDSigma**)

L'algorithme est dit radial, car pour les critères de types Von Mises, la correction s'effectue le long d'une direction radiale dans le plan déviatorique.

```
// check if the trial stress is outside the yield surface
  if (fSigTr  > 0) {
        // if true: plastic correction
        double deltaGamma;
        FloatArray *dFDSigma  = this->computeDFDSigma(sigmaTrial);
        deltaGamma  = fSigTr / (dFDSigma->transposedTimes(Del->Times(dFDSigma)));
        sigmaTrial->minus((Del->Times(dFDSigma))->times(deltaGamma));
        //tell the GaussPoint that it has reached the yield surface
        gp->isPlastic();
        gp->setDeltaGamma(deltaGamma);
        delete dFDSigma;
  }

  gp->letStressVectorBe(sigmaTrial);  //gp stores sigmaTrial (corrected), not a copy!
```

## 5.4.Cas spécifique de Von Mises

Pour le critère de Von Mises, l'algorithme de retour radial présent des simplifications importantes dues à la nature cylindrique de la surface de charge. la correction s'effectue exclusivement dans le plan déviatorique.

$$\boldsymbol{\sigma}_{\mathbf{n+1}} = \boldsymbol{p}_{n+1}^{tr}\, \mathrm{I} + \frac{\sigma_y\left(\bar{\varepsilon}_n^{pl}\right)}{\left\|\mathbf{s}_{\mathbf{n+1}}^{\mathbf{tr}}\right\|}\mathbf{s}_{\mathbf{n+1}}^{\mathbf{tr}} \qquad \textbf{(5-48)}$$

Où : $\boldsymbol{p}_{n+1}^{tr} = \frac{1}{3}\mathrm{tr}\left(\sigma_{n+1}^{tr}\right)$ est la pression hydrostatique prédite, et $\mathbf{s}_{\mathbf{n+1}}^{\mathbf{tr}}$ est le déviateur de contrainte prédit.

Cette expression montre que :

- La composante hydrostatique reste inchangée pendant la correction plastique.
- Le déviateur final est obtenu par un simple scaling du déviateur d'essai.
- La direction du déviateur ne change pas, seule son amplitude est réduite.

La solution analytique pour le multiplicateur devient :

$$\Delta\gamma = \frac{\left\|\mathbf{s}_{\mathbf{n+1}}^{\mathbf{tr}}\right\| - \sqrt{\frac{2}{3}}\,\sigma_y\left(\bar{\varepsilon}_n^{pl}\right)}{2\,\mu\left(1+\frac{\boldsymbol{H}'}{3\mu}\right)} \qquad \textbf{(5-49)}$$

Cette efficacité computationnelle explique pourquoi le critère de Von Mises est si largement utilisé dans les simulations numériques.

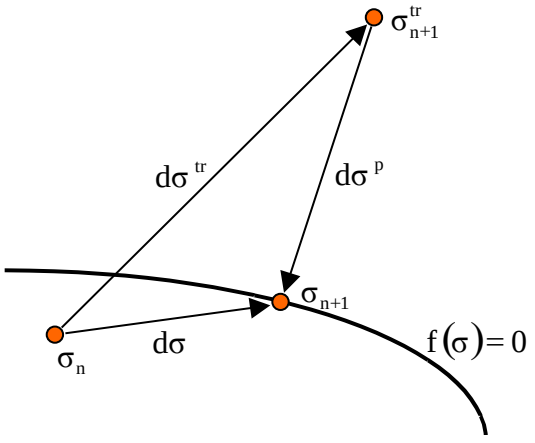



**Figure 1** Incrément de contrainte

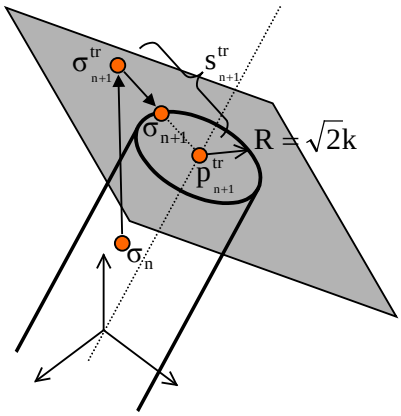



**Figure 5-5** La contrainte de retour dans le cas de Von Mises : vue dans l'espace de contrainte

## 6. Matrice constitutive élasto-plastique

### 6.1.Définition

Dans la mécanique non linéaire de la fracture simulée par la méthode des éléments finis étendus (**XFEM**), la matrice constitutive élasto-plastique « $D^{ep}$ », aussi connue sous le nom de la matrice de rigidité tangentielle élasto-plastique (ou matrice tangentielle cohérente), est fondamentale pour assurer la convergence quadratique des méthodes de résolution incrémentale, telles que la méthode de Newton-Raphson utilisée pour la résolution des équations non linéaires.

Dans l'implémentation numérique à l'aide des éléments finis, elle s'affiche sous la forme d'une matrice de taille ($n$ x $n$) (où « $n$ » change en fonction de la dimension du problème : 3x3 pour le cas plan de contrainte, 4x4 pour le cas plan de déformation, 6x6 pour le cas tridimensionnel 3D).

La matrice constitutive élasto-plastique « $D^{ep}$ » n'est constante que dans le cas de non-linéarité matérielle. Cette matrice procède à une évaluation à chaque incrément de charge et à chaque point d'intégration , c-à-d elle dépend de l'état de contrainte courante et de l'historique de déformation plastique

La matrice constitutive élasto-plastique « $D^{ep}$ » n'est pas seulement considérée comme le cœur physique du modèle étudié en incorporant toute la complexité du comportement du matériau (plasticité), elle illustre aussi la relation incrémentale et linéarisée entre l'incrément de

contrainte et l'incrément de déformations totales aux points de gauss considérées lors d'un chargement plastique

$$D^{ep} = \frac{d\sigma}{d\varepsilon} \qquad \textbf{(5-50)}$$

Avec « $d\sigma$ » est l'incrément du tenseur de contrainte de Cauchy, « dε » l'incrément du tenseur de déformation linéarisé.

Physiquement La matrice constitutive élasto-plastique « $D^{ep}$ » représente la pente de la courbe contrainte-déformation à l'état courant, capturant la rigidité instantanée du matériau sous chargement plastique.

Selon les conditions de **Kuhn-Tucker** on distingue deux étapes pour le calcul de la matrice constitutive (citée ci-dessus) :

1. Si la condition ( $f(\sigma) \leq 0$ ) est respectée, il n'y'a que seul le cas élastique qui se présente, ou $f(\sigma)$ la fonction d'écoulement. La matrice constitutive demeure purement élastique jusqu'à ce qu'une plasticité soit observée, donc : $D^{ep}=D^{el}$

Rappelant que pour un matériau isotrope, la matrice de Hooke pour état de contrainte plane (2D Voigt ($s_1$, $s_2$, $s_3$, $t_{12}$)) :

$$D^{el} = \frac{\mathrm{E}}{(1-\upsilon)^2}\begin{bmatrix} 1 & \upsilon & 0 & 0 \\ \upsilon & 1 & 0 & 0 \\ 0 & 0 & (1-\upsilon)/2 & 0 \\ 0 & 0 & 0 & (1-\upsilon) \end{bmatrix} \qquad \textbf{( 5-51)}$$

2. Si la condition de cons ( $f(\sigma) > 0$ ) est réalisée, elle impose que ( $df(\sigma) = 0$ ) pendant l'écoulement plastique. C-à-d un pris comme un point de départ.

La partie élastique de la déformation suit la loi de Hooke généralisée (loi élastique linéaire), ou la relation incrémentale constitutive est donnée comme :

$$d\sigma = D^{el} : \left(d\varepsilon - d\varepsilon^{p}\right) \qquad \textbf{(5-52)}$$

Ou « $D^{el}$ » représente la matrice constitutive élastique, symétrique et définie positive, ainsi que l'incrément de la déformation totale en plasticité classique, est considéré comme étant divisé en parties élastique et plastique. Ceci est applicable aux petites déformations (fondement de la théorie de la plasticité infinitésimale)

En introduisant la loi d'écoulement plastique :

$$\Rightarrow \qquad d\varepsilon^{\mathrm{p}} = d\gamma \frac{df}{d\sigma} = d\gamma \cdot m \qquad \textbf{(5-53)}$$

Ou : « $m$ » Est la direction d'écoulement plastique.

$$m = \frac{df}{d\sigma} = \frac{\mathrm{s}_{ij}}{2\sqrt{J_2}} \qquad \textbf{(5-54)}$$

Et « $d\gamma$ » le multiplicateur plastique (scalaire≥0),

Donc :

$$\Rightarrow \qquad d\sigma = D^{el} : \left(d\varepsilon - d\gamma * m\right) = D^{el} : d\varepsilon - d\gamma * \left(D^{el} : m\right) \qquad \textbf{( 5-55)}$$

Pour garder la contrainte sur la surface de charge pendant l'écoulement plastique, il est nécessaire de respecter la condition de consistance (pour simplifier, nous considérons d'abord une plasticité parfaite, sans variables internes d'écrouissage : H=0, dq=0):

$$\Rightarrow \qquad \mathrm{d}\boldsymbol{f} = \left(\frac{\partial \boldsymbol{f}}{\partial \sigma}\right) : \partial\sigma + \left(\frac{\partial \boldsymbol{f}}{\partial \mathrm{q}}\right) * \partial \mathrm{q} = 0 \qquad \textbf{(5-56)}$$

Donc :

$$\Rightarrow \qquad \frac{\partial \boldsymbol{f}}{\partial \sigma} : \partial\sigma = 0 \qquad \textbf{(5-57)}$$

$n = \dfrac{\partial f}{\partial \sigma}$ : est la normale à la surface de charge.

⇨ On remplace (5-55) dans (5-57), on trouve :

$$\Rightarrow \qquad \frac{df}{d\sigma} D^{el} \left(d\varepsilon - d\varepsilon^{p}\right) = 0 \qquad \textbf{( 5-58)}$$

$$\Rightarrow \qquad n\left(D^{el} : d\varepsilon - d\gamma * \left(D^{el} : m\right)\right) = 0 \qquad \textbf{( 5-59)}$$

$$\Rightarrow \qquad n : D^{el} : d\varepsilon - d\gamma * \left(n : D^{el} : m\right) = 0 \qquad \textbf{(5-60)}$$

$$\Rightarrow \qquad d\gamma^* = \frac{n : D^{el} : d\varepsilon}{n : D^{el} : m} \qquad \textbf{(5-61)}$$

Remplaçant (5-61) dans (5-55), on trouve :

$$d\sigma = D^{el} : \left(d\varepsilon - d\gamma \cdot m\right) = D^{el} : d\varepsilon - d\gamma^* \left(D^{el} : m\right) \qquad \textbf{(5-62)}$$

Et donc :

$$d\sigma = D^{el} : d\varepsilon - d\gamma^* \left(D^{el} : m\right) = \left[D^{el} - \frac{\left(D^{el} : m\right) \otimes \left(n : D^{el}\right)}{n : D^{el} : m}\right] d\varepsilon \qquad \textbf{(5-63)}$$

Ou $\otimes$ représente le produit tensoriel extérieur.

En substituant les expressions des incréments de contrainte et de déformation plastique on obtient l'expression de la matrice constitutive tangente élastoplastique suivante pour la plasticité parfaite (H=0):

$$\Rightarrow \qquad D^{ep} = D^{el} - \frac{\left(D^{el} : m\right) \otimes \left(n : D^{el}\right)}{n : D^{el} : m} \qquad \textbf{(5-64)}$$

**Remarque** : dans le cas des matériaux à écoulement associé ( tels que Von Mises, Drucker-Prager associé,…) : « $n$ » qui est la normale à la surface de charge (indiquant dans quelle direction la contrainte quitte ou entre dans la surface de charge) et la direction d'écoulement plastique « $m$ » (qui représente la tangente à la direction de déformation plastique , indiquant direction d'écoulement de la plasticité) sont identiques, et la matrice « $D^{el}$ » est symétrique. Contrairement aux matériaux à écoulement non-associé (par exemple Drucker-Prager non-associé, Mohr-Colomb,…) comme le béton, sol,…etc, ou la matrice « $D^{el}$ » serait non-symétrique.

Donc pour $n = m = r$ , on obtient l'expression de la matrice constitutive tangente élastoplastique :

$$\Rightarrow \qquad D^{ep} = D^{el} - \frac{\left(D^{el} : r\right) \otimes \left(r : D^{el}\right)}{r : D^{el} : r} \qquad \textbf{(5-65)}$$

La signification physique de cette dernière indique que le premier terme « $D^{el}$ » correspond à la raideur élastique (montée radiale dans l'espace des contraintes), le deuxième terme soustrait une composante suivant la direction plastique « $r$ », réduisant la raideur tangentielle. Tandis

que « $D^{ep}$ » assure la projection de la contrainte sur la surface de charge (principe de retour radial).

## 6.2.Cas avec écrouissage

Dans le cas d'une plasticité non parfaite, c-à-d un cas général avec module d'écrouissage (H), la condition de consistance se transforme en :

$$\Rightarrow \qquad n : \mathrm{d}\sigma - \mathrm{H} * \mathrm{d}\gamma = 0 \qquad \textbf{(5-66)}$$

Ou « H » représente le module d'écrouissage (tangent).

Lorsqu'on applique la même procédure, mais avec des conditions de consistance modifiées, on obtient :

$$\Rightarrow \qquad n : D^{el} : d\varepsilon - d\gamma * \left( n : D^{el} : m + H \right) = 0 \qquad \textbf{(5-67)}$$

⇨ D'où :

$$\Rightarrow \qquad d\gamma = \frac{n : D^{el} : d\varepsilon}{n : D^{el} : m + H} \qquad \textbf{(5-68)}$$

Donc pour une plasticité avec écrouissage isotrope, l'expression générale de la matrice constitutive tangente élastoplastique est

$$\Rightarrow \qquad D^{ep} = D^{el} - \frac{\left( D^{el} : \mathrm{m} \right) \otimes \left( n : D^{el} \right)}{n : D^{el} : m + H} \qquad \textbf{( 5-69)}$$

Dans le cas Von Mises $n = m = r$ , on obtient l'expression de la matrice constitutive tangente élastoplastique :

$$\Rightarrow \qquad D^{ep} = D^{el} - \frac{\left( D^{el} : r \right) \otimes \left( r : D^{el} \right)}{r : D^{el} : r + H} \qquad \textbf{(5-70)}$$

En terme Physique, la matrice constitutive tangente élastoplastique est décomposée en deux parties :

- La première partie « $D^{el}$ » illustre la rigidité élastique du matériau
- La seconde partie $\dfrac{\left( D^{el} : r \right) \otimes \left( r : D^{el} \right)}{r : D^{el} : r + H}$ exprime la réduction de rigidité causée par

l'écoulement plastique, également connu sous le nom d'adoucissement plastique du matériau.

on peut le décrire comme la rigidité effective qui diminue lors de l'écoulement plastique (ce qui est le cas des matériaux ductiles)

**Importance numérique :** l'utilisation de la matrice tangente cohérente (dérivée exactement à partir de l'algorithme d'intégration) plutôt d'une matrice tangente continue est cruciale pour préserver le taux de convergence quadratique de l'algorithme Newton-Raphson dans les simulations des éléments finis étendus. (question quelle est la différence matrice tangente cohérente et continue et c'est quoi le taux de convergence quadratique ??) il vaut mieux de ne pas trop parler

## 6.3. Implémentation dans XFEM

la méthode **ComputeConstitutiveMatrix**() de la classe **vonmisesmaterial** a été mise en œuvre pour réaliser le calcul de la matrice constitutive tangentielle **:**

```
FloatMatrix*  VonMisesMaterial :: ComputeConstitutiveMatrix
(GaussPoint*  gp, Element*  elem)
{  FloatMatrix  *Del, *Dep;
 Del = elem->giveConstitutiveMatrix()->GiveCopy();
 //pour la première itération, toujours renvoyé Del élastique
 if (elem->giveDomain()->giveNLSolver()->giveCurrentIteration()  == 1) {
        return Del;
 }
 //for Full Newton-Raphson, return Dep if plastic, Del else
 if (gp->givePlasticCode()  == 1) {
        //GaussPoint  is plastic
        FloatArray  *sigma,  *DFDSigma;
        sigma  = gp->giveStressVector();
        DFDSigma  = this->computeDFDSigma(sigma);
        int aSize  = sigma->giveSize();
 //test
        NLSolver*  nlSolver;
        nlSolver     = this->domain->giveNLSolver();
//if consistent,  compute DelStar, else Del if enough
        if (nlSolver->give('c')  == 1) {
                FloatArray  *kronDelta,  *a1;
```

```
                double deltaGamma,  j2, A1, A2, A3, e, nu, A4;
//deltaGamma  computation
                deltaGamma  =  gp->giveDeltaGamma();
//factors
                j2 = sigma->computeInvariantJ2();
                double  sqrtJ2 = sqrt(j2);
                A1 = deltaGamma*(-.25/(j2*sqrtJ2));
                A2 = deltaGamma*(0.5/sqrtJ2);
                e = elem -> giveMaterial()  -> give('E')  ;
                nu = elem -> giveMaterial()  -> give('n')  ;
                A3 = - nu / e;
                A4 = A3 - (A2 / 3.0e0);
//a1 computation  (dj2/dsigma)
                a1 = sigma->computeDJ2DSigma();
//kronDelta  definition
                kronDelta  = new FloatArray(aSize);
                kronDelta->at(1)  = 1;
                kronDelta->at(2)  = 1;
                kronDelta->at(3)  = 0;
                kronDelta->at(4)  = 1;
//T
                FloatMatrix*  T;
                T = new FloatMatrix(aSize,aSize);
                T->at(1,1) = 1;
                T->at(2,2) = 1;
                T->at(3,3) = 2;
                T->at(4,4) = 1;
//Ginv
                FloatMatrix*  Ginv;
                double  G;
                G = e / (2.0e0*(1.0e0+nu));
                Ginv  = T->Times(1.0e0/(2.0e0*G));
//Ninv
```

```
                FloatMatrix  *N, *Ninv;
                N = Ginv->Plus(T->Times(A2));
                Ninv = new FloatMatrix(aSize,aSize);
                for (int i = 1; i<=aSize; i++) {
                  Ninv->at(i,i) = 1 / N->at(i,i);
                }
//MathUtil
                MathUtil*  aMathUtil = new MathUtil();
//Kinv
                FloatMatrix*  Kinv;
                Kinv     =     aMathUtil->giveShermanMorrisonInverse(Ninv,
kronDelta,  A4);
//DelStar
        Del = aMathUtil->giveShermanMorrisonInverse(Kinv,  a1, A1);
//delete
                delete kronDelta;
                delete a1;
                delete T;
                delete Ginv;
                delete N;
                delete Ninv;
                delete aMathUtil;
        }
//Finally,  Dep
        FloatArray  *DelTimesDFDSigma;
        FloatMatrix  *aContribution;
        double aFactor;
        DelTimesDFDSigma  = Del->Times(DFDSigma);
        aFactor = 1. / DFDSigma->transposedTimes(DelTimesDFDSigma);
        aContribution                =                DelTimesDFDSigma-
>timesTransposed(DelTimesDFDSigma);
        Dep = Del->Minus(aContribution->times(aFactor));
//delete
```

```
            delete Del;
            delete DFDSigma;
//delete sigma;
            delete DelTimesDFDSigma;
            delete aContribution;
//return consistent (or not) tangent matrix Dep
            return Dep;
  } else {
//GaussPoint is Elastic
            return Del;
  }
}
```

## 7. Application numérique

Si on considère un exemple en 2D (état plan de contrainte), muni d'une fonction de charge $f$, et d'une contrainte d'essai (trial) , ainsi qu'un seuil d'écoulement $\sigma_y$, nous voulons faire un test élastique/plastique, puis une correction par retour radial si nécessaire , et enfin un calcul de la matrice tangente cohérente $D_{ep}$ selon la même la technique radial return

Ou nous considérons les caractéristiques suivantes :

- un état déviatorique de contrainte (initiale) (Voigt 2D [*xx*, *yy*, *xy*])
  $\sigma = \left[\sigma_{xx}, \sigma_{yy}, \sigma_{xy}\right]^T = \left[300, -100, 50\right]^T$ MPa,
- module d'élasticité $E$ = 210000 MPa,
- coefficient de poisson n=0.3.
- Limite d'écoulement $\sigma_y = 250 \quad \text{MPa}$
- Conditions : **état de contrainte plane.**
- La matrice constitutive élastique (plane stress)(Voigt 2D [*xx*, *yy*, *xy*]) est donnée comme suit :

$$D^{el} = \frac{\mathrm{E}}{(1-\upsilon)^2}\begin{bmatrix} 1 & \upsilon & 1 \\ \upsilon & 1 & 1 \\ 1 & 1 & (1-\upsilon)/2 \end{bmatrix} \approx \begin{bmatrix} 230\,769.23077 & 69\,230.76923 & 0 \\ 69\,230.76923 & 230\,769.23077 & 0 \\ 0 & 0 & 80\,769.23077 \end{bmatrix} \text{MPa}$$

- Module de cisaillement $G = \dfrac{\mathrm{E}}{2(1+\upsilon)} = 80\,769.23077 \quad \text{MPa}$

- Calculer la contrainte trial

Rappelons que :

a) pour Von Mises, la contrainte équivalente $\sigma_{\mathrm{eq}} = \sqrt{3J_2}$ ou $J_2$ est l'invariant du déviateur, ainsi que la fonction de charge scalaire $f(\sigma) = \sigma_{\mathrm{eq}} - \sigma_{\mathrm{y}}$,

b) Notre calcul consiste à faire un essai élastique (prédiction) suivi d'une correction plastique, ce qui donne lieu à deux scénarios distincts

- Si $f(\sigma) \le 0$ : le comportement est élastique ( $D=D^{el}$ )
- Si $f(\sigma) \ge 0$ : le comportement est plastique( $D=D^{pl}$ )

**Construire le prédicteur (Trial)**

$$\sigma^{\mathrm{trial}} = \left[\sigma_{\mathrm{xx}}, \sigma_{\mathrm{yy}}, \sigma_{\mathrm{xy}}\right]^{\mathrm{T}} = [300, -100, 50]^{\mathrm{T}}$$

Dans l'état plan de contrainte on prend $\sigma_{\mathrm{zz}} = 0$ donc

Calcul de la contrainte moyenne ( ~~le terme hydrostatique~~)

$$\sigma_{\mathrm{m}} = \frac{1}{3}\mathrm{tr}(\sigma).\boldsymbol{I} = \frac{1}{3}\left(\sigma_{\mathrm{xx}} + \sigma_{\mathrm{yy}} + \sigma_{\mathrm{xy}}\right) = \frac{1}{3}(300 - 100 + 50) = 66.66667 \text{ MPa}$$

⇨ Calcul du tenseur de contrainte déviatorique « s » :

$$\mathrm{s}_{\mathrm{ij}} = \sigma_{\mathrm{ij}} - \sigma_{\mathrm{m}}$$

Donc :

$$\begin{cases} s_{xx} = \sigma_{xx} - \sigma_m = (300 - 66.66667) = 233.33333 \\ s_{yy} = \sigma_{yy} - \sigma_m = (-100 - 66.66667) = -166.66667 \\ s_{zz} = \sigma_{zz} - \sigma_m = (0 - 66.66667) = -66.66667 \\ s_{xy} = 50 \end{cases}$$

**Calcul du second invariant du tenseur de contrainte déviatorique « $J_2$ » :**

$$\boldsymbol{J}_2 = \frac{1}{2} s_{ij} \,.\, s_{ij}$$

$$J_2^{trial} = \frac{1}{2}\left(s_{xx}^2 + s_{yy}^2 + s_{zz}^2\right) + s_{xy}^2$$

$$= \frac{1}{2}\left((233.33333)^2 + (-166.66667)^2 + (-66.66667)^2\right) + 50^2$$

$$= 45\,833.33333 \text{ MPa}$$

Donc la contrainte équivalente de Von Mises

$$\sigma_{eq}^{trial} = \sqrt{3 J_2^{trial}} = \sqrt{3 * 45\,833.33333} = 370.809 \text{ MPa}$$

⇨ Test de charge $f^{trial} = \sigma_{eq}^{trial} - \sigma_y = 370.809 - 250 = 120.809 \text{ MPa} \quad > 0$

⇨ L'état trial dépasse la surface de charge, donc nous somme dans l'état plastique

**Retour radial (Von Mises parfait)**

Calcul du multiplicateur $\Delta\gamma$ (coefficient scalaire sans dimension – celui qui multiplie la direction d'écoulement pour donner $d\varepsilon^{pl}$)

$$\Delta\gamma = \frac{\sigma_{eq}^{trial} - \sigma_y}{3G} = \frac{370.809 - 250}{3(80\,769.23077)} \simeq 4.986 \text{ x } 10^{-4}$$

**Mise à jour directe des déviateurs (projection radiale)**

$$s^{n+1} = \alpha \,.\, s^{trial} = \frac{\sigma_y}{\sigma_{eq}^{trial}} \,.\, s^{trial}$$

Ou α facteur scalaire $\alpha = \frac{\sigma_y}{\sigma_{eq}^{trial}} = \frac{250}{370.809} = 0.674199861$

Donc :

$$\begin{cases} s_{xx}^{(trial)} = \alpha \,.- s_{xx}^{trial} = 0.674199861 \,.\, 233.33333 \approx 157.008 \\ s_{yy}^{(trial)} = \alpha \,.- s_{yy}^{trial} = 0.674199861 \,.\, (-166.66667) \approx -112366644 \\ s_{zz}^{(trial)} = \alpha \,.- s_{zz}^{trial} = 0.674199861 \,.\, (-66.66667) \approx -44.946657 \\ s_{xy}^{(trial)} = \alpha \,.- s_{xy}^{trial} = 0.674199861 \,.\, 50 \approx 33.710 \end{cases}$$

Reconstruction des composantes de contraintes ( on rappelle que $\sigma_m$ reste inchangé pour cet exemple) :

$$\begin{cases} \sigma_{xx}^{(n+1)} = s_{xx}^{(n+1)} + \sigma_m = \approx 157.008 + 66.66667 = 223.6747 \\ \sigma_{yy}^{(n+1)} = s_{yy}^{(n+1)} + \sigma_m \approx -112.366644 + 66.66667 = -45.69997 \\ \tau_{xy}^{(n+1)} = s_{xy}^{(n+1)} \approx 33.710 \end{cases}$$

Vérification de consistance (contrainte équivalent après correction) :

$$\begin{aligned} J_2^{(n+1)} &= \frac{1}{2}\left(s_{xx}^2 + s_{yy}^2 + s_{zz}^2\right) + s_{xy}^2 \\ &= \frac{1}{2}\left((157.008)^2 + (-112.366644)^2 + (-44.946657)^2\right) + (33.710)^2 \\ &= 20\,785.3524616 \text{ MPa} \end{aligned}$$

Donc la contrainte équivalente de Von Mises

$$\sigma_{eq}^{(n+1)} = \sqrt{3J_2^{(n+1)}} = \sqrt{3 * 20\,7853524616} = 249.71194 \text{ MPa}$$

Cela confirme que la projection radiale a ramené le point sur la surface $f(\sigma) = 0$

Calcul de la direction plastique « *n* » (pour tangente) :

Nous utilisons la convention standard (tensorielle) pour Von Mises :

$$n = \frac{\partial f}{\partial \sigma} = \frac{3}{2}\frac{s^{(n+1)}}{\sigma_{eq}^{(n+1)}} = \frac{3}{2}\frac{s^{(n+1)}}{250} = 0.006 \,.\, s^{(n+1)}$$

Avec $\sigma_{eq}^{(n+1)} = \sigma_y \approx 250$ MPa

Donc : $$n = \begin{bmatrix} 0.006 \text{ x } 157.008 \\ 0.006 \text{ x } (-112366644) \\ 0.006 \text{ x } 33.710 \end{bmatrix} = \begin{bmatrix} 0.9420 \\ -0.6742 \\ 0.2023 \end{bmatrix}$$

Les composantes du vecteur « *n* » exprimées sous forme tensorielles ; en représentation Voigt, il est nécessaire de suivre la convention pour la composante de cisaillement si l'on réalise des produits matricielles dans la base de Voigt. Ici on maintient la même convention que celle fournie par la matrice « $D^{el}$ » donnée

**Calcul de la tangente cohérente « $D^{el}$ »**

Pour la plasticité parfaite et l'écoulement associé (*n*=*m*), la tangente cohérente est (la projection rang-1)

$$D^{ep} = D^{el} - \frac{\left(D^{el} : m\right) \otimes \left(n : D^{el}\right)}{n : D^{el} : m}$$

En substituants les valeurs de « $D^{el}$ » et de « *n* » , on trouve que :

D'abord on calcul ($v=D^{ep} * n$)

$$v=D^{ep} . n = \begin{bmatrix} 1.7070923078 . \text{e} + 05 \\ -9.0369230775 . \text{e} + 04 \\ 1.6339615385 . \text{e} + 04 \end{bmatrix}$$

On calcul ($num=n^{T} * v= n^{T} * v$)

$$num = \begin{bmatrix} 2.9141641472\text{e}+10 & -1.5426861872\text{e}+10 & 2.7893231735\text{e}+09 \\ -1.5426861872\text{e}+10 & 8.1665978709\text{e}+09 & -1.4765984735\text{e}+09 \\ 2.7893231735\text{e}+09 & -1.4765984735\text{e}+09 & 2.6698303092\text{e}+08 \end{bmatrix} \text{MPa}$$

En suite , on calcul ($den= n * D^{ep} * n$)

$$den = 2.2504053497 . \text{e} + 05$$

Donc $D^{ep} = D^{el} - \frac{num}{den}$

$$D^{ep} = \begin{bmatrix} 1.01274.e+05 & 1.37782.e+05 & -1.23948.e+04 \\ 1.37782.e+05 & 1.94480.e+05 & 6.56148.e+03 \\ -1.23948.e+04 & 6.56148.e+03 & 7.95829.e+04 \end{bmatrix} \text{MPa}$$

**Fissure Centrale inclinée sous chargement de traction**

Dans cet exemple, le même modèle numérique dans la section (4.1) est considéré : une plaque carré isotrope de largeur W, comportant une fissure interne inclinée de longueur 2a, avec un rapport géométrique fixé à a/W=0.25, la plaque est soumise à des contraintes de traction uniformes s=1, appliquées sur les deux côtés opposés.

La géométrie et les principales caractéristiques dimensionnelles sont illustrées dans la **Figure 5.6** .

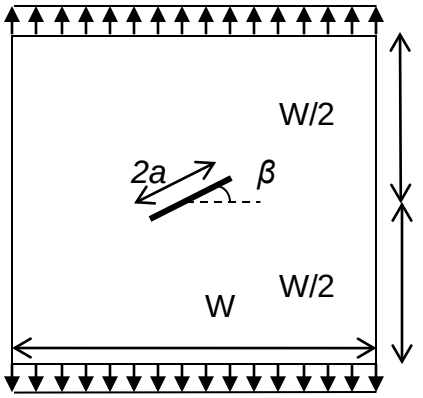



**Figure 5-6** Géométrie d'une fissure centrale inclinée dans une plaque soumise à une traction uniaxiale

Les propriétés mécaniques du matériau restent inchangées, à savoir un comportement linéaire élastique caractérisé par le module de Young E et le coefficient de poisson n. L'épaisseur étant supposée négligeable au regard des autres dimensions de la plaque.

Une modélisation bidimensionnelle est retenue.

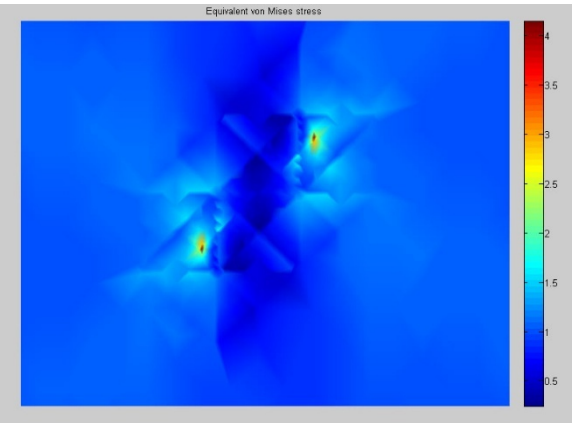



**Figure 5-7** Champ de contrainte Von Mises d'une plaque carrée soumise à une traction uniaxiale munie d'une fissure centrale inclinée

La figure 5.7. met en évidence la distribution de la contrainte équivalente de Von Mises, ou le champ de contrainte souligne des concentrations distinctes au voisinage du fond de fissure, caractéristiques du comportement singulier induit par la présence d'une discontinuité interne.
Ces zones de forte contrainte correspondent également à l'initiation et au développement de zones plastifiées localisées autour du fond de fissure. De plus, une décroissance rapide du niveau de contrainte en s'éloignant de ces zones indique une plasticité confinée, compatible avec l'hypothèse ***small-scale yielding***. .
L'apparence globale du champ obtenue confirme donc la capacité du modèle numérique à représenter simultanément les impacts de la concentration des contraintes et de non-linéarité matérielle.
Cette analyse constitue une étape préliminaire, les recherches concernant l'évolution de la zone plastique, l'impact des paramètres géométriques et l'utilisation de critères énergétiques de rupture étant actuellement en progression.

## 8. Conclusion

Ce chapitre a présenté l'extension de la plateforme orientée objet OpenXFEM++ au domaine non linéaire par l'intégration d'un modèle de Von Mises reposant sur un algorithme de retour radial et une matrice tangente consistante. Les résultats obtenus soulignent l'efficacité de la formulation proposée pour présenter précisément les concentrations des contraintes et les zones plastifiées près de la pointe de la fissure, tout en assurant la solidité et la convergence de la résolution numérique. Cette extension constitue une avancée importante de la plateforme développée et ouvre la voie à l'étude des problèmes plus complexes en mécanique de la rupture élastoplastique.

# Conclusion générale

# Conclusion générale

Cette thèse met en évidence l'enrichissement et la validation d'une approche numérique intégrée pour la modélisation des discontinuités en mécanique de la rupture quasi-statique. Fondée sur la méthode des éléments finis étendus (**XFEM**), elle répond au défi scientifique consistant à modéliser avec précision des fissures soumises à des sollicitations en mode mixte, tout en s'affranchissant des contraintes liées au remaillage conforme. Elle s'appuie la bibliothèque **OpenXFEM++**, conçue selon les principes de la programmation orientée objet en **C++**.

L'un des principaux apports de cette recherche réside dans l'enrichissement et l'évolution d'une architecture logicielle modulaire, extensible et orientée objet permettant l'intégration des enrichissements XFEM, des fonctions Level Set et ce nouvelles lois de comportement. Cette architecture amélioré al maintenabilité, la réutilisabilité et l'évolutivité de la plateforme OpenXFEM++, tout en facilitant l'intégration de nouvelles modèles numériques.

Sur le plan méthodologique, l'étude porte sur l'analyse systématique d'une plaque élastique comportant une fissure centrale inclinée, soumise à des chargements uniaxiaux et biaxiaux, appliqués séparément ou en combinaison. Les développements réalisés ont permis de mettre en œuvre une formulation XFEM robuste capable de représenter avec précision les champs singuliers au voisinage de la pointe de fissure sans recourir au remaillage, tout en garantissant une excellente stabilité numérique.

Les résultats numériques, systématiquement comparés aux solutions analytiques de référence et aux simulations Abaqus, démontrent la robustesse, la précision et la convergence indépendante du maillage. Ils mettent également en évidence une excellente concordance avec les résultats apportés dans la littérature scientifique, confirmant ainsi la fiabilité de la formulation proposée.

L'étude paramétrique met en évidence une relation entre la longueur et l'orientation de la fissure, les conditions de chargement er l'évolution des facteurs d'intensité des contraintes en mode I et II. Elle confirme que le mode de chargement gouverne directement le mécanisme de rupture : l'ouverture de la fissure mode I apparait sous traction et s'annule en compression, traduisant un phénomène de fermeture, tandis que le mode II demeure présent pour l'ensemble des configurations étudiées, son signe étant directement lié au sens de glissement des lévres de

fissure. Ces résultats contribuent à une meilleure compréhension du comportement mécanique des structures fissurées soumissent à des sollicitations complexes.

Les principales contributions scientifiques de cette thèse peuvent être résumées comme suit :

- Enrichissement et évolution de la plateforme OpenXFEM++ selon une architecture orientée objet modulaire et extensible ;
- Implémentation de la formulation XFEM pour l'analyse des fissures en mode I, II et mixte ;
- Validation numérique de la formulation à travers plusieurs cas de références et comparaison avec les solutions analytiques, les résultats de littérature et les simulations Abaqus ;
- Réalisation d'une étude paramétrique mettant en évidence l'influence de la configuration de chargement et des paramètres géométriques sur les facteurs d'intensité de contrainte en mode mixte ;
- Extension de la plateforme au domaine élastoplastique par l'intégration d'un modèle de plasticité de Von Mises reposant sur un algorithme de retour radial et une matrice tangente consistante.

Sur le plan de l'implémentation, cette thèse démontre clairement que l'adoption d'une approche orientée objet constitue un levier majeur pour la structuration et l'évolution des méthodes numériques avancées.

L'architecture d'**OpenXFEM++**, fondée sur les principes de l'approche OOP à savoir la modularité, l'héritage, le polymorphisme, l'encapsulation, la non-anticipation et la généricité offre une extensibilité et une maintenabilité accrues, favorisant l'intégration de nouveaux modèles physiques.

Les perspectives ouvertes par cette recherche peuvent être inclinées selon trois axes complémentaires :

- Perspectives scientifiques : extension aux comportements fortement non linéaires, prise en compte de la propagation automatique de la fissure et intégration de modèles d'endommagement plus avancés ;
- Perspectives numériques : développement d'une version tridimensionnelle d'OpenXFEM++, amélioration des stratégies d'intégration numérique et optimisation des performances de calcul ;

- Perspectives applicatives : validation expérimentale de la formulation proposée, application des structures industrielles complexes et développement de couplages multiphysiques notamment thermomécanique et hydromécanique.

Ainsi cette recherche apporte une contribution significative au développement des méthodes numériques avancées en mécanique de la rupture. Au-delà de la validation de la formulation XFEM proposée, elle fournit une plateforme de calcul évolutive susceptible de service de base à de futurs développements scientifiques et à des applications industrielle dans le domaine de la modélisation des structures fissurées.

# Références bibliographiques

# Références bibliographiques :


1. Griffith, A.A.: VI. The phenomena of rupture and flow in solids. Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character. 221, 163‑198 (1921). https://doi.org/10.1098/rsta.1921.0006
2. Irwin, G.R.: Analysis of Stresses and Strains Near the End of a Crack Traversing a Plate. Journal of Applied Mechanics. 24, 361‑364 (1957). https://doi.org/10.1115/1.4011547
3. Westergaard, H.M.: Bearing Pressures and Cracks: Bearing Pressures Through a Slightly Waved Surface or Through a Nearly Flat Part of a Cylinder, and Related Problems of Cracks. Journal of Applied Mechanics. 6, A49‑A53 (1939). https://doi.org/10.1115/1.4008919
4. Williams, M.L.: On the Stress Distribution at the Base of a Stationary Crack. Journal of Applied Mechanics. 24, 109‑114 (1957). https://doi.org/10.1115/1.4011454
5. Gdoutos, E.: Fracture mechanics criteria and applications. (1990)
6. TADA, H., PARIS, P.C., IRWING, G.R.: The Stress Analysis of Cracks Handbook.. (1975)
7. BUI, H.D.: Mécanique de la rupture fragile. (1978)
8. Cherepanov, G.P., Halmanov, H.: On the theory of fatigue crack growth. Engineering Fracture Mechanics. 4, 219‑230 (1972). https://doi.org/10.1016/0013-7944(72)90038-0
9. Cherepanov, G.: The propagation of cracks in a continuous medium. (1967)
10. Rice, J.R.: A Path Independent Integral and the Approximate Analysis of Strain Concentration by Notches and Cracks. Journal of Applied Mechanics. 35, 379‑386 (1968). https://doi.org/10.1115/1.3601206
11. Rice, J.R.: Elastic Fracture Mechanics Concepts for Interfacial Cracks. Journal of Applied Mechanics. 55, 98‑103 (1988). https://doi.org/10.1115/1.3173668
12. Anderson, T.L.: Fracture Mechanics CRC Press. (1995)
13. Sih, G.C., Paris, P.C., Erdogan, F.: Crack-Tip, Stress-Intensity Factors for Plane Extension and Plate Bending Problems. Journal of Applied Mechanics. 29, 306‑312 (1962). https://doi.org/10.1115/1.3640546
14. Shih, C.F., Asaro, R.J.: Elastic-Plastic Analysis of Cracks on Bimaterial Interfaces: Part I—Small Scale Yielding. Journal of Applied Mechanics. 55, 299‑316 (1988). https://doi.org/10.1115/1.3173676
15. Stern, M., Becker, E.B., Dunham, R.S.: A contour integral computation of mixed-mode stress intensity factors. Int J Fract. 12, 359‑368 (1976). https://doi.org/10.1007/BF00032831
16. Fracture mechanics. 19: San Antonio, TX, 30 June - 2 July 1986. ASTM, Philadelphia, Pa (1988)
17. Lorentzon, M., Eriksson, K.: A path independent integral for the crack extension force of the circular arc crack. Engineering Fracture Mechanics. 66, 423‑439 (2000). https://doi.org/10.1016/S0013-7944(00)00030-8
18. Sih, G.: Handbook of stress-intensity factors. (1973)
19. Murakami, Y., Keer, L.M.: Stress Intensity Factors Handbook, Vol. 3. Journal of Applied Mechanics. 60, 1063‑1063 (1993). https://doi.org/10.1115/1.2900983
20. Clough, R.W.: The finite element in plane stress analysis, (1960)
21. Argyris, J.H., Kelsey, S.: Energy Theorems and Structural Analysis: A Generalised Discourse with Applications on Energy Principles of Structural Analysis Including the

Effects of Temperature and Non-Linear Stress-Strain Relations. Springer US, Boston, MA (1960)
22. Zienkiewicz, O.C., Cheung, Y.K.: The Finite Element Method in Structural and Continuum Mechanics. (1967)
23. uritzinn, T., Blanchet, N.: Analyse du critère KIa pour l'arrêt de fissure. (2004)
24. Symposium on Crack Arrest Methodology and Applications: Crack arrest methodology and applications. American Society for Testing and Materials, Philadelphia, Pa (1980)
25. Kobayashi, A.S., Seo, K., Jou, J.Y., Urabe, Y.: A dynamic analysis of modified compact-tension specimens using homalite-100 and polycarbonate plates: Dynamic photoelasticity, dynamic finite-element analysis and streaking photography are used to study the dynamic fracture and crack-arrest-responses of a modified compact-tension specimen machined from Homalite-100 and polycarbonate sheets. Experimental Mechanics. 20, 73–79 (1980). https://doi.org/10.1007/BF02321231
26. Bouyne E, Propagation et arrêt de fissure de clivage dans l'acier 2 1/4 Cr-1 Mo, Thèse, Paris : Ecole des Mines de Paris, 1999.
27. Vaezzadeh, A., Ahmadizadeh, M., Dolatshahi, K.M.: Three-dimensional nonlinear dynamic analysis of slack cable structures using node Relaxation method. Structures. 29, 586–599 (2021). https://doi.org/10.1016/j.istruc.2020.11.026
28. Fang, H.-Y., Wang, T., Hu, J.-F., Yang, J.-G.: Node dynamic relaxation method: principle and application. Front. Mater. Sci. 5, 179–195 (2011). https://doi.org/10.1007/s11706-011-0127-3
29. Komori, K.: Simulation of shearing by node separation method. Computers & Structures. 79, 197–207 (2001). https://doi.org/10.1016/S0045-7949(00)00118-8
30. Dugdale, D.S.: Yielding of steel sheets containing slits. Journal of the Mechanics and Physics of Solids. 8, 100–104 (1960). https://doi.org/10.1016/0022-5096(60)90013-2
31. Barenblatt, G.I.: The Mathematical Theory of Equilibrium Cracks in Brittle Fracture. In: Advances in Applied Mechanics. p. 55–129. Elsevier (1962)
32. Elices, M., Guinea, G.V., Gómez, J., Planas, J.: The cohesive zone model: advantages, limitations and challenges. Engineering Fracture Mechanics. 69, 137–163 (2002). https://doi.org/10.1016/S0013-7944(01)00083-2
33. Kuna, M.: Finite Elements in Fracture Mechanics: Theory - Numerics - Applications. Springer Netherlands, Dordrecht (2013)
34. Kang, Y., Zheng, Y., Li, S., Zhang, J., Tang, J., Yang, C., Luo, Y.: GPU-accelerated approach for 2D fracture analysis of structures combining finite particle method and cohesive zone model. Engineering Fracture Mechanics. 306, 110198 (2024). https://doi.org/10.1016/j.engfracmech.2024.110198
35. Cui, H.: Viscoelastic cohesive zone modeling for mode II fracture between propellant and insulation. International Journal of Adhesion and Adhesives. 114, 103118 (2022). https://doi.org/10.1016/j.ijadhadh.2022.103118
36. Nairn, J.A., Aimene, Y.E.: A re-evaluation of mixed-mode cohesive zone modeling based on strength concepts instead of traction laws. Engineering Fracture Mechanics. 248, 107704 (2021). https://doi.org/10.1016/j.engfracmech.2021.107704
37. Linke, M., Lammering, R.: On the calibration of the cohesive strength for cohesive zone models in finite element analyses. Theoretical and Applied Fracture Mechanics. 124, 103733 (2023). https://doi.org/10.1016/j.tafmec.2022.103733
38. Parks, D.M.: A stiffness derivative finite element technique for determination of crack tip stress intensity factors. Int J Fract. 10, 487–502 (1974). https://doi.org/10.1007/BF00155252

39. Hellen, T.K.: On the method of virtual crack extensions. Numerical Meth Engineering. 9, 187‑207 (1975). https://doi.org/10.1002/nme.1620090114
40. Delorenzi, H.G.: Energy release rate calculations by the finite element method. Engineering Fracture Mechanics. 21, 129‑143 (1985). https://doi.org/10.1016/0013-7944(85)90060-8
41. deLorenzi, H.G.: On the energy release rate and the J-integral for 3-D crack configurations. Int J Fract. 19, 183‑193 (1982). https://doi.org/10.1007/BF00017129
42. Bruyneel, M., Delsemme, J.P., Jetteur, P., Germai: A study of complex delaminations in laminated composite structures with SAMCEF, (2008)
43. Ramirez Tamayo, D., Montoya, A., Millwater, H.: A virtual crack extension method for thermoelastic fracture using a complex-variable finite element method. Engineering Fracture Mechanics. 192, 328‑342 (2018). https://doi.org/10.1016/j.engfracmech.2017.12.013
44. Montoya, A., Ramirez-Tamayo, D., Millwater, H., Kirby, M.: A complex-variable virtual crack extension finite element method for elastic-plastic fracture mechanics. Engineering Fracture Mechanics. 202, 242‑258 (2018). https://doi.org/10.1016/j.engfracmech.2018.09.023
45. Millwater, H., Wagner, D., Baines, A., Montoya, A.: A virtual crack extension method to compute energy release rates using a complex variable finite element method. Engineering Fracture Mechanics. 162, 95‑111 (2016). https://doi.org/10.1016/j.engfracmech.2016.04.002
46. Rybicki, E.F., Kanninen, M.F.: A finite element calculation of stress intensity factors by a modified crack closure integral. Engineering Fracture Mechanics. 9, 931‑938 (1977). https://doi.org/10.1016/0013-7944(77)90013-3
47. Shivakumar, K., Tan, P.W., Newman Jr, J.C.: A virtual crack-closure technique for calculating stress intensity factors for cracked three dimensional bodies., (1988)
48. Liu, Y.-P., Li, T.-J.: Application of the virtual crack closure technique (VCCT) using tetrahedral finite elements to calculate the stress intensity factor. Engineering Fracture Mechanics. 253, 107853 (2021). https://doi.org/10.1016/j.engfracmech.2021.107853
49. Farkash, E., Banks-Sills, L.: The Multi-Virtual Crack Closure Technique for three-dimensional interface crack problems. Theoretical and Applied Fracture Mechanics. 128, 104117 (2023). https://doi.org/10.1016/j.tafmec.2023.104117
50. Zhou, L., Wang, J., Wang, Y., Li, X., Chai, Y.: The enriched finite element method-virtual crack closure technique for cracked structures. Thin-Walled Structures. 187, 110756 (2023). https://doi.org/10.1016/j.tws.2023.110756
51. Swenson, D.V., Ingraffea, A.R.: Modeling mixed-mode dynamic crack propagation using finite elements: Theory and applications. Computational Mechanics. 3, 381‑397 (1988). https://doi.org/10.1007/BF00301139
52. Taupin, E., Breitling, J., Wu, W., Altan, T.: Material fracture and burr formation in blanking results of FEM simulations and comparison with experiments. Journal of Materials Processing Technology. 59, 68‑78 (1996). https://doi.org/10.1016/0924-0136(96)02288-1
53. An Essay on the Application of mathematical Analysis to the theories of Electricity and Magnetism. Journal für die reine und angewandte Mathematik (Crelles Journal). 1854, 161‑221 (1854). https://doi.org/10.1515/crll.1854.47.161
54. Albuquerque, E.L., Sollero, P., Aliabadi, M.H.: Dual boundary element method for anisotropic dynamic fracture mechanics. Numerical Meth Engineering. 59, 1187‑1205 (2004). https://doi.org/10.1002/nme.912

55. Seelig, Th., Gross, D., Pothmann, K.: Numerical Simulation of a Mixed-mode Dynamic Fracture Experiment. International Journal of Fracture. 99, 325‑338 (1999). https://doi.org/10.1023/A:1018769521499
56. Leitao, V.M.A., Aliabadi, M.H.: Boundary element methods for the analysis of crack growth in non-linear fracture. IJMPT. 15, 104 (2000). https://doi.org/10.1504/IJMPT.2000.001239
57. Bonnet, M.: Mathematical aspects of boundary element methods2024) .)
58. Chongmin, S., John P., W.: Finite-element modelling of unbounded media. (c1996)
59. Fracture Mechanics, 4(2) :219–230 Rice, James R. « Mathematical analysis in the mechanics of fracture. » Fracture: an advanced treatise 2 (1968): 191-311
60. Yang, Z.: Fully automatic modelling of mixed-mode crack propagation using scaled boundary finite element method. Engineering Fracture Mechanics. 73, 1711‑1731 (2006). https://doi.org/10.1016/j.engfracmech.2006.02.004
61. Yang, Z.J., Deeks, A.J., Hao, H.: Transient dynamic fracture analysis using scaled boundary finite element method: a frequency-domain approach. Engineering Fracture Mechanics. 74, 669‑687 (2007). https://doi.org/10.1016/j.engfracmech.2006.06.018
62. Belytschko, T., Krongauz, Y., Organ, D., Fleming, M., Krysl, P.: Meshless methods: An overview and recent developments. Computer Methods in Applied Mechanics and Engineering. 139, 3‑47 (1996). https://doi.org/10.1016/S0045-7825(96)01078-X
63. Krysl, P., Belytschko, T.: The Element Free Galerkin method for dynamic propagation of arbitrary 3-D cracks. Int. J. Numer. Meth. Engng. 44, 767‑800 (1999). https://doi.org/10.1002/(SICI)1097-0207(19990228)44:6<767::AID-NME524>3.0.CO;2-G
64. Ventura, G., Xu, J.X., Belytschko, T.: A vector level set method and new discontinuity approximations for crack growth by EFG. Numerical Meth Engineering. 54, 923‑944 (2002). https://doi.org/10.1002/nme.471
65. Belytschko, T., Organ, D., Gerlach, C.: Element-free galerkin methods for dynamic fracture in concrete. Computer Methods in Applied Mechanics and Engineering. 187, 385‑399 (2000). https://doi.org/10.1016/S0045-7825(00)80002-X
66. Lu, Y.Y., Belytschko, T., Tabbara, M.: Element-free Galerkin method for wave propagation and dynamic fracture. Computer Methods in Applied Mechanics and Engineering. 126, 131‑153 (1995). https://doi.org/10.1016/0045-7825(95)00804-A
67. Belytschko, T., Tabbara, M.: DYNAMIC FRACTURE USING ELEMENT-FREE GALERKIN METHODS. Int. J. Numer. Meth. Engng. 39, 923‑938 (1996). https://doi.org/10.1002/(SICI)1097-0207(19960330)39:6<923::AID-NME887>3.0.CO;2-W
68. Sukumar, N., Moran, B., Black, T., Belytschko, T.: An element-free Galerkin method for three-dimensional fracture mechanics. Computational Mechanics. 20, 170‑175 (1997). https://doi.org/10.1007/s004660050235
69. Belytschko, T., Organ, D., Krongauz, Y.: A coupled finite element-element-free Galerkin method. Computational Mechanics. 17, 186‑195 (1995). https://doi.org/10.1007/BF00364080
70. Rao, B.N., Rahman, S.: An enriched meshless method for non-linear fracture mechanics. Numerical Meth Engineering. 59, 197‑223 (2004). https://doi.org/10.1002/nme.868
71. Duflot, M., Nguyen-Dang, H.: A meshless method with enriched weight functions for fatigue crack growth. Numerical Meth Engineering. 59, 1945‑1961 (2004). https://doi.org/10.1002/nme.948

72. Melenk, J.M., Babuška, I.: The partition of unity finite element method: Basic theory and applications. Computer Methods in Applied Mechanics and Engineering. 139, 289‑314 (1996). https://doi.org/10.1016/S0045-7825(96)01087-0
73. Babuška, I., Melenk, J.M.: THE PARTITION OF UNITY METHOD. Int. J. Numer. Meth. Engng. 40, 727‑758 (1997). https://doi.org/10.1002/(SICI)1097-0207(19970228)40:4<727::AID-NME86>3.0.CO;2-N
74. Chessa, J., Wang, H., Belytschko, T.: On the construction of blending elements for local partition of unity enriched finite elements. Numerical Meth Engineering. 57, 1015‑1038 (2003). https://doi.org/10.1002/nme.777
75. Farhat, C., Harari, I., Franca, L.P.: The discontinuous enrichment method. Computer Methods in Applied Mechanics and Engineering. 190, 6455‑6479 (2001). https://doi.org/10.1016/S0045-7825(01)00232-8
76. Strouboulis, T., Copps, K., Babuška, I.: The generalized finite element method. Computer Methods in Applied Mechanics and Engineering. 190, 4081‑4193 (2001). https://doi.org/10.1016/S0045-7825(01)00188-8
77. Moës, N., Dolbow, J., Belytschko, T.: A finite element method for crack growth without remeshing. Int. J. Numer. Meth. Engng. 46, 131‑150 (1999). https://doi.org/10.1002/(SICI)1097-0207(19990910)46:1<131::AID-NME726>3.0.CO;2-J
78. Belytschko, T., Black, T.: Elastic crack growth in finite elements with minimal remeshing. Int. J. Numer. Meth. Engng. 45, 601‑620 (1999). https://doi.org/10.1002/(SICI)1097-0207(19990620)45:5<601::AID-NME598>3.0.CO;2-S
79. Bordas, S.P.A., Menk, A., Natarajan, S.: Partition of unity methods. Wiley, Hoboken, NJ Chichester, West Sussex (2024)
80. Fleming, M., Chu, Y.A., Moran, B., Belytschko, T.: ENRICHED ELEMENT-FREE GALERKIN METHODS FOR CRACK TIP FIELDS. Int. J. Numer. Meth. Engng. 40, 1483‑1504 (1997). https://doi.org/10.1002/(SICI)1097-0207(19970430)40:8<1483::AID-NME123>3.0.CO;2-6
81. Laborde, P., Pommier, J., Renard, Y., Salaün, M.: High-order extended finite element method for cracked domains. Int. J. Numer. Meth. Engng. 64, 354‑381 (2005). https://doi.org/10.1002/nme.1370
82. Gordeliy, E., Peirce, A.: Enrichment strategies and convergence properties of the XFEM for hydraulic fracture problems. Computer Methods in Applied Mechanics and Engineering. 283, 474‑502 (2015). https://doi.org/10.1016/j.cma.2014.09.004
83. Stroustrup, Bjarne.: The C++ programming language. (2013)
84. Bordas, S., Nguyen, P.V., Dunant, C., Guidoum, A., Nguyen-Dang, H.: An extended finite element library. International Journal for Numerical Methods in Engineering. 71, 703‑732 (2007)
85. Dubois-Pe`lerin, Y., Zimmermann, T.: Object-oriented finite element programming: III. An efficient implementation in C++. Computer Methods in Applied Mechanics and Engineering. 108, 165‑183 (1993). https://doi.org/10.1016/0045-7825(93)90159-U
86. PHU, N.V.: An object oriented approach to the extended finite element method with applications to fracture mechanics, (2005)
87. Chen, W.H., Chen, T.C.: An efficient dual boundary element technique for a two-dimensional fracture problem with multiple cracks. Numerical Meth Engineering. 38, 1739‑1756 (1995). https://doi.org/10.1002/nme.1620381009

88. Belytschko, T., Liu, W.K., Moran, B., Elkhodary, K.I.: Nonlinear finite elements for continua and structures. Wiley, Chichester, West Sussex, United Kingdom (2014)
89. Hutchinson, J.W.: Singular behaviour at the end of a tensile crack in a hardening material. Journal of the Mechanics and Physics of Solids. 16, 13‑31 (1968). https://doi.org/10.1016/0022-5096(68)90014-8
90. Rice, J.R., Rosengren, G.F.: Plane strain deformation near a crack tip in a power-law hardening material. Journal of the Mechanics and Physics of Solids. 16, 1‑12 (1968). https://doi.org/10.1016/0022-5096(68)90013-6
91. Belytschko, T.: Nonlinear Finite Elements for Continua and Structures. John Wiley & Sons, Incorporated, Newark (2014)
92. Lemaitre, J., Chaboche, J. L., Benallal, A., & Desmorat, R. (2020). Mécanique des matériaux solides-3e éd. Dunod..
93. Motivation. One-Dimensional Plasticity and Viscoplasticity. In: Computational Inelasticity. p. 1‑70. Springer-Verlag, New York (1998)
94. Commend, S., Zimmermann, T.: Object-oriented nonlinear finite element programming: a primer. Advances in Engineering Software. 32, 611‑628 (2001). https://doi.org/10.1016/S0965-9978(01)00011-4