The Extended Finite Element Method and Its Implementation in 2D in the Aster Code

FRANCK HAZIZA

KTH Computer Science
and Communication

Master of Science Thesis
Stockholm, Sweden 2006

The Extended Finite Element Method and Its Implementation in 2D in the Aster Code

F R A N C K H A Z I Z A

Master's Thesis in Numerical Analysis (20 credits)
at the Scientific Computing International Master Program
Royal Institute of Technology year 2006
Supervisor at CSC was Michael Hanke
Examiner was Axel Ruhe

TRITA-CSC-E 2006:043
ISRN-KTH/CSC/E--06/043--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden

URL: www.csc.kth.se

Abstract

The extended finite element method is a new approch based on finite element method. Where finite element method need to change the mesh at every step of a crack propagation, extended finite element method does not because the mesh does not need to follow the crack anymore, that's why this method is very well suited for fracture problems. This master's thesis is the development of the 2D part of this method in the Aster code, the finite element code for Electricité de France. This method uses a definition of the crack by level sets to add new degrees of freedom and new functions to the nodes around a crack. It also solves the problem of contact and friction between the lips of the crack.

Contents

Chapter 1

Introduction

The Finite Element Method (FEM) has been widely used when dealing with problems of Linear Fracture Mechanics. A crack are usually due to microscopic defects in the material. Depending on the loadings on the material, a crack can become larger and propagate through the material. If a crack or numerous crack become too important, it might lead the structure might simply break or loose some properties such as impermeability. To know what will happen with the material, the crack and his propagation has to be studied. The classical way to propagate a crack is to create a new mesh at each step of the propagation. Besides, it is also costly to perform a parametric study on the location and shapes of cracks. The re-meshing step can be easily automatically generated in 2D (see [4]), but in 3D automatic meshing programs often generate a large number of badly-shaped elements which are not reliable and imply ill-conditioned stiffness matrices. As a consequence, human work to supervise the remeshing process increases dramatically as the geometry of the 3D crack gets more complex (as helix-shaped cracks in rotor shafts). In addition to these practical difficulties, the projection of quantities such as stresses from a mesh to the next-step mesh raises fundamental theoretical problems (verification of the conservation of energy, quantity of movement and mass). Alternative methods, such as Meshless methods have been proposed to avoid a mesh which must follow the crack geometry (see [3]). A recent method named eXtended Finite Element Method (X-FEM) allows one to consider a crack in a unique and simple mesh within the classical framework of the FEM. The crack, represented explicitly, is independent of the mesh, so the mesh does not need to follow the geometry of the crack faces and re-meshing is avoided. X-FEM uses an enrichment of the classical shape functions. To represent displacement discontinuities through the interface, a generalized Heaviside function is introduced. Moreover, adding singular asymptotic fields at the crack tip gives accurate results in linear elastic fracture mechanics. In addition, the level set method is a convenient way to describe a crack in 3D and efficient for the propagation phase (see [5]). However, to take into account the possible closure of the crack, penetration of one side into the other one must be prevented. Indeed, surveys of real industrial cases when fatigue occurs for example have shown that

contact between the crack faces should not be neglected. The aim of this project was to complete the XFEM already partially developed in 3D in order to be able to compare 2D and 3D cases and for further development of 2D problems. The method is implemented in the Aster code, the finite element code of Electricité de France.

This document is divided into 4 parts. The first one is this introduction. The second one presents the extended finite element method and more precisely the one included in the Aster code. It shows how the level sets are used to represent the fracture, how the finite element functions are enriched and how the final problem can be solved with contact and friction at the crack. The third part presents the results obtained and the validation cases which have been done to test particular aspects of the method. The last part presents an overview of Aster, the process for working on such a code and the different pieces of code needed to work with Aster.

Chapter 2

Aster's extended finite element method

The extended finite element method is very useful to deal with crack growth without remeshing. In order to do so, level sets have to be introduced to represent the crack, some degrees of freedom have to be added, the matrices to solve the new problem have to be changed and this has to be applied to fracture with contact and friction.

2.1 Level sets

2.1.1 Level sets definition

A crack can be represented by 2 functions, a tangential level set (F_t) and a normal level set (F_n). F_n defines on which side of the fracture the point is located, and F_t defines if the projection of the point on the fracture plane is inside or outside the fracture. That will define geometrically a crack (see Figure 2.1). These functions must have the following properties at a point X of the mesh:

$F_n(X) < 0$ on one side of the crack.

$F_n(X) > 0$ on the other side of the crack.

$F_n(X) = 0$ on the crack plane.

$F_t(X) < 0$ over or under the crack.

$F_t(X) > 0$ outside the crack.

$F_t(X) = 0$ on the crack tip.

The crack is therefore defined by $F_n(X) = 0$ and $F_t(X) \leq 0$ and the crack tip is defined by $F_n(X) = 0$ and $F_t(X) = 0$. Level sets are often taken as signed distance function, what allows to use the gradient of these level sets later on. F_n usually represents the distance to the fracture plane and F_t the distance to the crack tip projected on the fracture plane. It's not a required condition to use level sets but we use it in Aster as soon as there is a crack tip in the mesh.

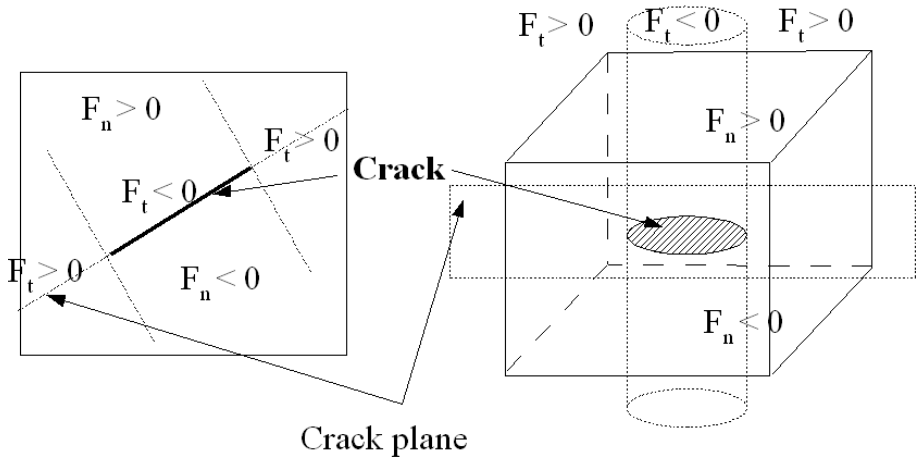

Figure 2.1. Level sets graphics in 2D and 3D

2.1.2 Level sets calculation

Those level sets have to be calculated on every node of the mesh. There are two ways to do it in the Aster code. Either a function is given for each level set, or the crack is given by a group of elements to localize the crack and the crack tip. The first case is quite easy because the functions can be evaluated at every node of the mesh knowing the coordinates (x, y) of the nodes. For example, consider a squared sheet going from $x = 0$ to $x = 2$ and from $y = 0$ to $y = 2$, and a horizontal crack from the middle of the sheet $(x, y) = (1, 1)$ to a border of the sheet $(x, y) = (0, 1)$. The level sets for this crack are given by $F_n(x, y) = y - 1$ and $F_t(x, y) = x - 1$. With those formulas, over the crack for example, i.e. for all node with coordinate $y > 1$, $F_n > 1 - 1$ so $F_n > 0$. All the level set properties for this crack can be found from those formulas. The second case requires a little bit more work. The nodes coordinates and the coordinates of the elements of the crack are known, therefore geometric calculations can be made. For F_n, the distance to the crack for each node needs to be calculated. First all the elements are sorted to obtain a list of contiguous elements. Then for each node M, and for all the elements the distance to the element is calculated by taking the distance from M to its projection P on the element (or to one vertex of the segment element if P is outside the element). Only the smallest distance MP is then kept. But a signed distance is needed, so the first element will define an arbitrary orientation, and all the following elements will take the same orientation. That's why the elements were sorted at the beginning. For F_t, for each node M and each crack tip element I, the point R_I is determined, R_I being the projection of M along the normal to the crack element connected to I. The shortest distance IR_I is then kept. Once again a signed distance is needed, but vertices are known to be on the crack only or on the crack tip so a simple scalar product determines if R_I is on the crack tip side or on the other side of the crack

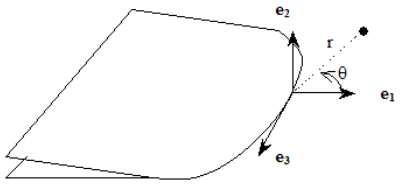

Figure 2.2. Crack tip local basis

element and therefore give us the sign of F_t. After that some of those level sets have to be adjusted to prevent ill conditioned stiffness matrices, because if the crack is too close to the border of an element, the ratio of the surfaces between both side of the crack will be too large. Therefore all F_n values that are under 1% of the value at a node on the same edge will be set to 0, the crack is "moved" to the edge of the element. Those things done, a good approximation of the level sets on every point of the structure and a fairly exact value on the nodes of the mesh are obtained.

2.1.3 Crack tip local basis

For a further use, a crack tip local basis is also defined with the level sets (see Figure 2.2). This basis is defined as follows.

$$\begin{aligned} e_1 &= \sum_{nodes\ i} \phi_i \nabla F_n(Xi) \\ e_2 &= \sum_{nodes\ i} \phi_i \nabla F_t(Xi) \\ (and \quad e_3 &= e_1 \wedge e_2 \quad in\ 3D) \end{aligned} \tag{2.1}$$

With $\wedge$ being the operator for cross product. This basis will be used to define polar coordinates for a crack tip enrichment.

2.2 Fracture problem with XFEM

2.2.1 General problem

The general problem equations of a fractured structure are needed here. The situation is sketched in Figure 2.3. We consider a domain Ω with a crack Γ_c. $\partial\Omega$ is the border of Ω with exterior normal $\mathbf{n}_ext$. Crack lips are called Γ_1 and Γ_2 with outward normals $\mathbf{n}_1$ and $\mathbf{n}_2$. The stress and displacement fields are called σ and $\mathbf{u}$ respectively. A quasi-static load is put on the structure with a density of volume force $\mathbf{f}$ and a density of surface forces $\mathbf{t}$ on Γ_t. The structure is fixed on Γ_u. Equilibrium

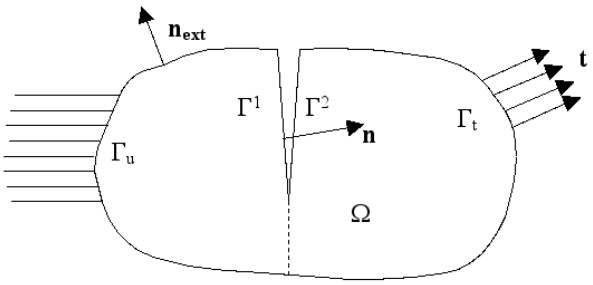

Figure 2.3. General problem notations

equations can be written:

$$\begin{aligned} \nabla \cdot \sigma &= \mathbf{f} \quad in \quad \Omega \\ \sigma \cdot \mathbf{n_{ext}} &= \mathbf{t} \quad on \quad \Gamma_t \\ \mathbf{u} &= \mathbf{0} \quad on \quad \Gamma_u \end{aligned} \tag{2.2}$$

To keep the problem simple, only small displacements and deformations are considered here, therefore can be written :

$$\varepsilon = \nabla_s(\mathbf{u}) = \frac{1}{2}(\nabla(\mathbf{u}) +^T \nabla(\mathbf{u})) \tag{2.3}$$

where ∇_s is the symmetric part of the gradient, ^{T} the operator for transposition and ε the deformation tensor. We consider a linear elastic material, therefore :

$$\sigma = \mathbf{C} : \varepsilon \tag{2.4}$$

where $\mathbf{C}$ is the Hooke tensor which depend on the type of material and : the operator for tensor product. The following weak form is finally obtained :

$$\int_\Omega \sigma(\mathbf{u}) : \varepsilon(\mathbf{v}) d\Omega = \int_\Omega \mathbf{f} \cdot \mathbf{v} d\Omega + \int_{\Gamma_t} \mathbf{t} \cdot \mathbf{v} d\Gamma \quad \forall \mathbf{v} \tag{2.5}$$

2.2.2 XFEM enrichment

The idea of X-FEM is to enrich the basis functions in order to represent the fracture behavior. New degrees of freedom are added for the crack jump, and new degrees of freedom are added for the crack tip opening. Classic FEM functions are usually written like that:

$$\mathbf{u}(\mathbf{x}) = \sum_{i \in N_n(\mathbf{x})} \mathbf{a_i} \phi_i(\mathbf{x}) \tag{2.6}$$

Where ϕ_i represent finite element nodal basis functions. They will be changed to the following formulas :

$$\mathbf{u}(\mathbf{x}) = \sum_{i \in N_n(\mathbf{x})} \mathbf{a_i} \phi_i(\mathbf{x}) + \sum_{i \in N_n(\mathbf{x}) \bigcap K} \mathbf{b_i} H(F_n(\mathbf{x})) \phi_i(\mathbf{x})$$

$$+ \sum_{i \in N_n(\mathbf{x}) \bigcap L} \sum_{\alpha=1}^{4} c_i^\alpha \phi_i(\mathbf{x}) F^\alpha(F_n(\mathbf{x}), F_t(\mathbf{x})) \tag{2.7}$$

The same functions ϕ_i are kept but two new terms are added, a Heaviside function $H(F_n(x))$ to represent the jump and four singular functions $F^\alpha(F_n(x), F_t(x))$ to represent the behavior at the crack tip. $N_n(x)$ is the group of nodes with a support that contains x and K and L are two domains where we apply those enrichments. The basis functions ϕ_i that have been used in this project are the standard bilinear basis functions. For the quadrilaterals they are for example (when centered in the middle of the element):

$$\begin{aligned} \phi_1(\xi,\eta) &= (1-\xi)(1-\eta)/4 \\ \phi_2(\xi,\eta) &= (1+\xi)(1-\eta)/4 \\ \phi_3(\xi,\eta) &= (1+\xi)(1+\eta)/4 \\ \phi_4(\xi,\eta) &= (1-\xi)(1+\eta)/4 \end{aligned} \tag{2.8}$$

Other basis function could of course have been used with this method. Those enrichment are the core of the XFEM and will be detailed thereafter. The first enrichment is an heaviside enrichment. H is defined as follows:

$$\begin{aligned} H &: \mathbb{R} \rightarrow \{-1, 1\} \\ H(x) &= -1 \quad if \quad x < 0 \\ H(x) &= 1 \quad if \quad x \geq 0 \end{aligned} \tag{2.9}$$

Combined with F_n, we have $H(F_n)$ positive on one side of the crack and $H(F_n)$ negative on the other side. Therefore it will help to represent the displacement jump at the crack. K is the group of nodes with a support completely cut by the crack. The b_i are enriched degrees of freedom. The second enrichment is needed to represent the singularity at the crack tip. Four functions F^α are used. Those functions have been chosen so that one can build the expressions of the asymptotic expansion of the displacement field in mechanics of the elastic linear fracture. Those expression were determined for a plane crack with infinite boundaries (see [6]). The c_i^α are enriched degrees of freedom. The basis chosen is therefore :

$$F^\alpha \in \{\sqrt{r}sin\frac{\theta}{2}, \sqrt{r}cos\frac{\theta}{2}, \sqrt{r}sin\frac{\theta}{2}sin\theta, \sqrt{r}cos\frac{\theta}{2}sin(\theta)\} \tag{2.10}$$

If the level sets have be chosen to represent a signed distance, the polar coordinates of the node can be easily expressed from the values of the level sets (see Figure 2.4):

$$r = \sqrt{F_n + F_t} \tag{2.11}$$

$$\theta = arctan\frac{F_n}{F_t} \tag{2.12}$$

Those enrichment should be chosen for every node of the mesh. To find which kind

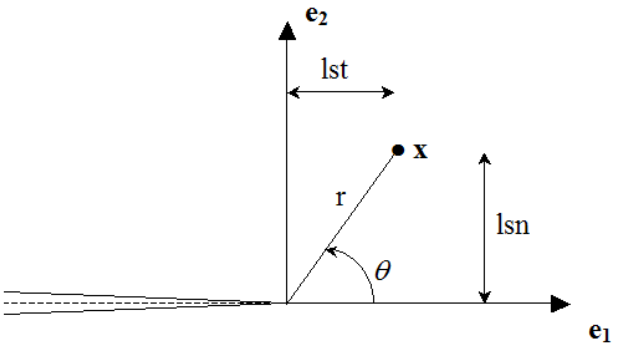

Figure 2.4. Polar coordinates in the local basis

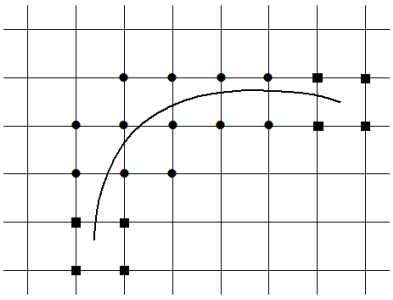

Figure 2.5. Nodes enrichment around a crack (circled nodes : heaviside enrichment, squared nodes : crack tip enrichment)

of enrichment is needed, tests on the level sets signs are performed. But to avoid having a huge number of elements types, the number of new element types in the code was restricted to 3:

- Heaviside elements with only b_i as new degrees of freedom, with at least one node enriched with heaviside.
- Crack tip elements with c_i^α with at least one node enriched with singular functions.
- Combined elements with both degrees of freedom added and with both types of nodes.

Other elements types with different types of enriched nodes are not created, therefore some elements have too many degrees of freedom, those degrees will be removed later on.

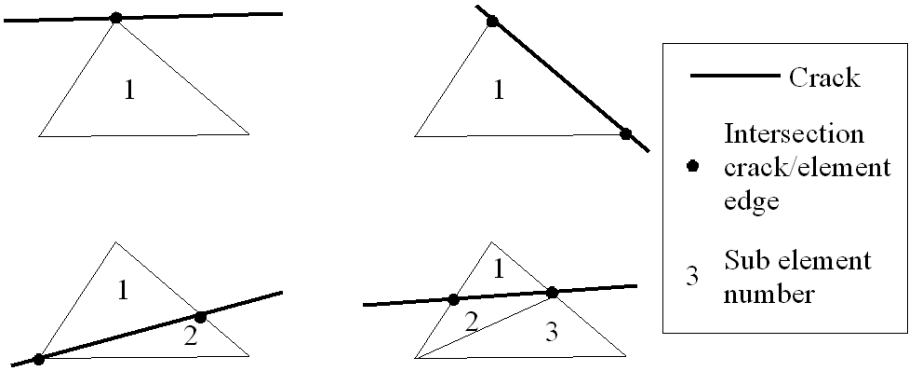

Figure 2.6. Possible cases of cutting for triangular elements

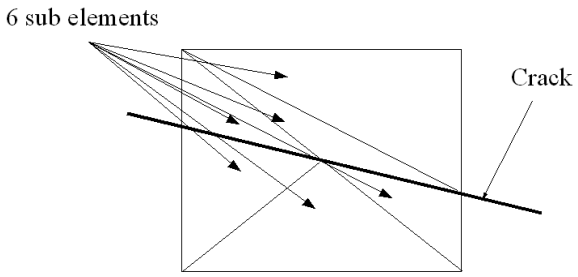

Figure 2.7. Cutting of a quadrilateral element

2.2.3 Elements cutting

With the XFEM enrichment, there are some discontinuous terms to integrate to solve the problem, due to the Heaviside term and to one of the singular function which are both discontinuous when the crack is crossed. The integration use Gaussian quadrature, so we need to integrate only continuous functions. The elements will therefore be cut into smaller elements on both side of the crack. To stay close to the realization in 3D, it was chosen to first cut all the quads in triangles and to deal with both types of meshes with triangles then. One, two or three sub elements can be obtained depending on how the crack is cutting the triangle (see Figure 2.6). For a quad, a maximum of six sub-elements can be obtained (see Figure 2.7). All the sub elements are triangles.

2.2.4 Integration

The displacement field in every element can be written with XFEM:

$$\mathbf{u} = \begin{pmatrix} \phi_1 & H\phi_1 & F^1\phi_1 & F^2\phi_1 & F^3\phi_1 & F^4\phi_1 & & \\ & & & \dots & & & & \\ & \phi_n & H\phi_n & F^1\phi_n & F^2\phi_n & F^3\phi_n & F^4\phi_n \end{pmatrix} \begin{pmatrix} a_1 \\ b_1 \\ c_1^1 \\ c_1^2 \\ c_1^3 \\ c_1^4 \\ \vdots \\ a_n \\ b_n \\ c_n^1 \\ c_n^2 \\ c_n^3 \\ c_n^4 \end{pmatrix} \tag{2.13}$$

n being the number of nodes of the element or

$$\mathbf{u} = \mathbf{N}\mathbf{g} \tag{2.14}$$

Where $\mathbf{N}$ is the enriched base of basis functions and $\mathbf{g}$ if the vector of nodal degree of freedom. All the types of elements don't need all the degrees of freedom, therefore some of them will be set to 0 later on. Deformations can be written :

$$\varepsilon = [B]\mathbf{g} \tag{2.15}$$

where $[B]$ is equal to $\nabla_s(N)$. In a domain Ω the stiffness matrix is :

$$[K] = \int_\Omega [B]^t[D][B]d\Omega \tag{2.16}$$

And after cutting this domain in sub elements, we can write with only continuous functions:

$$[K] = \sum_{sub-elements} \int_{\Omega_{se}} [B]^t[D][B]d\Omega_{se} \tag{2.17}$$

With the help of the element cutting a Gaussian quadrature is suitable. The Gaussian quadrature chosen use three Gauss points for integration, because the maximum degree of the integrated terms is four, and three gaussian points should therefore be sufficient for the triangular sub elements.

2.3 Contact and friction with XFEM

2.3.1 Contact laws

Be P a point of Γ_c. We call P^1 and P^2 the corresponding points on Γ^1 and Γ^2 (See figure 2.8). The condition of non interpenetration between P^1 and P^2 can be

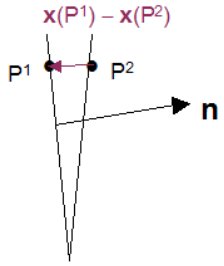

Figure 2.8. Jump definition

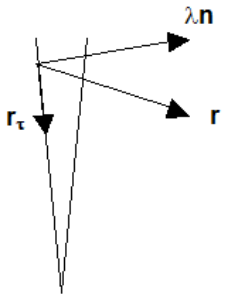

Figure 2.9. Definition of contact effort density

written in direction of $\mathbf{n}$, the normal to Γ^1 :

$$d_n = (\mathbf{x}(P^1) - \mathbf{x}(P^2)) \cdot \mathbf{n} \leq 0 \tag{2.18}$$

We decompose the contact effort density $\mathbf{r}$ in a normal part λ, the normal contact pressure and a tangential part $\mathbf{r}_t$ (see Figure 2.9).

$$\mathbf{r} = \lambda \mathbf{n} + \mathbf{r}_t \tag{2.19}$$

With those notations, contact laws are written :

$$d_n \leq 0, \quad \lambda \leq 0, \quad \lambda d_n = 0 \tag{2.20}$$

In order to get rid of the inequality these equations are rewritten :

$$\lambda - \chi(g_n) g_n = 0 \tag{2.21}$$

Where χ is a step function defined by :

$$\chi(x) = \begin{cases} 1 & \text{if} \quad x < 0 \\ 0 & \text{if} \quad x \geq 0 \end{cases} \tag{2.22}$$

and g_n is the term of augmented contact :

$$g_n = \lambda - \rho_n d_n, \tag{2.23}$$

where ρ_n is a strictly positive parameter arbitrarily chosen.

2.3.2 Friction laws

For friction phenomenon, we use Coulomb's law :

$$\begin{array}{rcl} \|\mathbf{r}_\tau\| & \leq & \mu|\lambda| \\ \text{if } \|\mathbf{r}_\tau\| & < & \mu|\lambda| \text{ then } \mathbf{v}_\tau = 0 \\ \text{if } \|\mathbf{r}_\tau\| & = & \mu|\lambda| \text{ then } \exists \alpha \geq 0 \text{ such as } \mathbf{v}_\tau = -\alpha \mathbf{r}_\tau, \end{array} \tag{2.24}$$

where μ is the Coulomb's friction coefficient and $\mathbf{v}_\tau$ is relative tangential speed of the contacting bodies. Those laws can be rewritten as for contact :

$$\begin{array}{rcl} \mathbf{r}_\tau & = & \mu\lambda\mathbf{\Lambda} \\ \mathbf{\Lambda} - P_{B(0,1)}(\mathbf{g}_\tau) & = & 0 \\ \mathbf{g}_\tau & = & \mathbf{\Lambda} + \rho_\tau \mathbf{v}_\tau \end{array} \tag{2.25}$$

$\mathbf{\Lambda}$ is the half friction multiplier, $\mathbf{g}_\tau$ is the half augmented friction multiplier, $P_{B(0,1)}$ is the projection on the unity sphere and ρ_τ is a strictly positive parameter. A complementary condition to those friction laws is the following exclusion law :

$$d_n\mathbf{\Lambda} = 0 \text{ or, equivalently, } (1 - \chi g_n)\Lambda = 0 \tag{2.26}$$

2.4 Mixed weak form

The following mixed weak form is finally obtained. Find (**u**,**r**), such as :

$$\int_\Omega \sigma(\mathbf{u}) : \varepsilon\mathbf{v} d\Omega = \int_\Omega \mathbf{f}\cdot\mathbf{v} d\Omega + \int_{\Gamma_t} \mathbf{t}\cdot\mathbf{v} d\Gamma + \int_{\Gamma_c} \mathbf{r}\cdot[[\mathbf{v}]] d\Gamma_c \quad \forall v \tag{2.27}$$

Where $[[\mathbf{v}]]$ is the jump of $\mathbf{v}$ and is equal to $\mathbf{v}(P^1) - \mathbf{v}(P^2)$. Including contact and friction, the weak form becomes : Find $(\mathbf{u}, \lambda, \Lambda)$ such as for all $(\mathbf{v}, \lambda^*, \Lambda^*)$:

$$\begin{array}{c} \displaystyle\int_\Omega \sigma(\mathbf{u}) : \varepsilon(\mathbf{v}) d\Omega - \int_\Omega \mathbf{f}\cdot\mathbf{v} d\Omega - \int_{\Gamma_t} \mathbf{t}\cdot\mathbf{v} d\Gamma \\ \displaystyle - \int_{\Gamma_c} \chi(g_n) g_n \mathbf{n}.[[\mathbf{v}]] d\Gamma_c - \int_{\Gamma_c} \chi(g_n)\mu\lambda P_{B(0,1)}(\mathbf{g}_\tau).[[\mathbf{v}]] d\Gamma_c = 0 \\ \displaystyle \int_{\Gamma_c} \frac{-1}{\rho_n}(\lambda - \chi(g_n) g_n)\lambda^* d\Gamma_c = 0 \\ \displaystyle \int_{\Gamma_c} \frac{-\mu\chi(g_n)\lambda}{\rho_t}(\lambda - P_{B(0,1)}(\mathbf{g}_\tau))\lambda^* d\Gamma_c + \int_{\Gamma_c} (1 - \chi(g_n))\lambda\lambda^* d\Gamma_c = 0 \end{array} \tag{2.28}$$

2.5 Discretization of finite elements

To calculate λ and $\mathbf{\Lambda}$ the values at the intersection points between edges and the fracture are used. These intersections define a polygon inside an element (contact segment for 2D case). The contact and friction multiplier are calculated on these elements. The following equation are obtained with the segment basis functions :

$$\lambda(\mathbf{x}) = \sum_{i=1}^{2} \lambda_i \psi_i(\mathbf{x}) \tag{2.29}$$

The same approximation is used for $\mathbf{\Lambda}$. The segment basis functions used are linear functions (centered on the middle of the element) :

$$\begin{aligned} \psi_1 &= (1-\xi)/2 \\ \psi_2 &= (1-\eta)/2 \end{aligned} \tag{2.30}$$

We can notice that those intersection points aren't in the initial mesh. In order to use them within the classic element code of Aster, we associate those points to the middle points of the edges of the elements even if they are not geometrically at the middle position. If the crack is located inside an element, only middle points will have activated contact degree of freedom. If it falls on a side of an element, contact points are nodes of the mesh and contact is activated at the vertex nodes of the elements.

2.6 Resolution strategy

The strategy of resolution is the same as the one used with classic finite element method in Aster (see [7]). There is a nonlinear term in every equation, for example, $\chi(g_n)\mathbf{u}$ in the equilibrium equation. Therefore, we will suppress these nonlinearities by putting some parameters in some parts of the computation. We first initialize a friction threshold to some arbitrary value λ_s and we begin the friction loop, at every step of this loop this term will be calculated and put as a constant in the inner part. Then we do the same thing for the contact status χ and we begin a contact loop ($\chi(g_n)\mathbf{u}$ becomes $\chi\mathbf{u}$ therefore the nonlinearity disappear). In the innermost loop we solve the last nonlinear term by a Newton method.

2.7 Linearized elementary terms

2.7.1 Matrix form the linear problem

The linearized system obtained is:

$$\begin{pmatrix} K_{meca} + A_u + B_u & A^T & B^T \\ A & C & 0 \\ B & 0 & F_r \end{pmatrix} \begin{pmatrix} \delta\mathbf{u} \\ \delta\lambda \\ \delta\mathbf{\Lambda} \end{pmatrix} = \begin{pmatrix} L^1_{meca} + L^1_{cont} + L^1_{frot} \\ L^2_{cont} \\ L^3_{frot} \end{pmatrix}, \tag{2.31}$$

which can be written :

$$[K]\delta\mathbf{u} = \mathbf{F}. \tag{2.32}$$

- The unknowns are an increment of the terms since the last Newton iteration,
- K_{meca} is the mechanical stiffness matrix defined by (2.17),
- A_u is the augmented stiffness matrix due to contact,
- B_u is the augmented stiffness matrix due to friction,
- A is the matrix linking the displacement and contact terms,
- B is the matrix linking displacement and friction terms,
- C is the matrix used to determine contact pressure for non-contact case,
- F_r is the matrix used to determine friction terms for non-frictional case,
- L^1_{meca} is the right-hand side representing the intern forces and the loading increments,
- L^1_{cont} and L^2_{cont} are the right-hand sides due to contact,
- L^1_{frot} and L^3_{frot} are the right-hand sides due to friction

2.7.2 Contact elementary matrices

From the previous equations and discretizations, the contact matrices can be calculated.

$$[A]_{ij} = \int_\Gamma \chi\psi_i\lambda_i\phi_j(2b_j + 2\sqrt{r}c^1_j)\mathbf{n}d\Gamma, \tag{2.33}$$

$$[A_u]_{ij} = \int_\Gamma \chi\rho_n\phi_i(2b_i + 2\sqrt{r}c^1_i)\mathbf{n}\phi_j(2b_j + 2\sqrt{r}c^1_j)\mathbf{n}d\Gamma, \tag{2.34}$$

$$[C]_{ij} = -\int_\Gamma \frac{1}{\rho_n}(1-\chi)\psi_i\lambda_i\psi_j\lambda_j d\Gamma. \tag{2.35}$$

Note that all the continuous terms ($\mathbf{a}_j$,$\mathbf{c}^2_j$,$\mathbf{c}^3_j$,$\mathbf{c}^4_j$) disappear in the contact part. The $[C]$ matrix is the same than without XFEM. Terms in $\sqrt{r}$ come from derivation of crack tip enrichment functions F^α.

2.7.3 Right-hand sides for contact forces

Right-hand sides for contact forces use the previous Newton iteration, the variables of the previous iteration are indicated by the superscript $k-1$.

$$[L^1_{cont}] = -\int_\Gamma \chi(\lambda^{k-1} - \rho_n d_n^{k-1})((2b_i + 2\sqrt{r}c^1_i)\phi_i\mathbf{n}d\Gamma \tag{2.36}$$

$$[L^2_{cont}] = -\int_\Gamma (\frac{1-\chi}{\rho_n}\lambda^{k-1} + \chi d_n^{k-1})\psi_i\lambda_i d\Gamma \tag{2.37}$$

2.7.4 Friction matrices

The following expressions can be obtained for the friction matrices :

$$[B_u]_{ij} = -\int_\Gamma \chi\mu\rho_\tau\phi_i(2b_i + 2\sqrt{r}c_i^1)\phi_j(2b_j + 2\sqrt{r}c_j^1)[P]^T[K_n][P]d\Gamma, \tag{2.38}$$

where the matrix P represent the projection on the contact plane with a normal $\mathbf{n}$. This matrix can be written as:

$$[P] = \begin{pmatrix} 1-n_x^2 & -n_x n_y \\ -n_x n_y & 1-n_y^2 \end{pmatrix}, \tag{2.39}$$

$$[B]_{ij} = -\int_\Gamma \chi\mu\rho_\tau\psi_i\Lambda_i\tau_i[K_n][P]\phi_j(2b_j + 2\sqrt{(}r)c_j^1)d\Gamma, \tag{2.40}$$

$$[F_r]_{ij} = -\int_\Gamma \frac{\chi\mu\lambda_s}{\rho_\tau}\psi_i\mathbf{\Lambda}_i\tau_i[Id - K_n][P]\psi_j\Lambda_i\tau_i d\Gamma + \int_\Gamma(1-\chi)\psi_i\psi_j\tau_i\tau_j d\Gamma. \tag{2.41}$$

2.7.5 Right-hand sides for friction forces

Right-hand sides for friction forces use the previous Newton iteration, the variables of the previous iteration are indicated by the superscript $k-1$.

$$L^1_{frott} = -\int_\Gamma \chi\mu\lambda_s\phi_i(2b_i + 2\sqrt{r}c_i^1)[P]^T P_{B(0,1)}(g_\tau^k - 1)d\Gamma \tag{2.42}$$

$$L^3_{frott} = \int_\Gamma \frac{\chi\mu\lambda_s}{\rho_\tau}\psi_i\mathbf{\Lambda}_i\tau_i(\mathbf{\Lambda}^{k-1} - P_{B(0,1)}(g_\tau^k - 1)d\Gamma \tag{2.43}$$

Where k-1 is the previous Newton iteration.

Chapter 3

Results and test cases

Different test have been performed. The aim of those test was to validate the method and to complete the database of test cases which is used to verify the integrity of the Aster code. Some of them are therefore performed on very simple meshes.

3.1 Level sets validation

The level sets shapes are tested on different simple cases: horizontal fracture with a crack given with functions, or with segment elements, and also a curved crack. The methodology is shown below on the curved crack. Figure 3.1 shows the mesh given and the level sets obtained. The mesh consists of 20*100 square elements. The crack is the curved shape close to the middle of the mesh. Both values and shapes of level sets seem correct on those figures as well as on all the cases obtained. Figure 3.2 shows the node status given to the different nodes. Those node status help us to attribute element types later in the program. The different types of nodes on the figure are shown. There are normal nodes (status = 0, dark blue), the heaviside nodes, for elements completely cut by the crack (status = 1, light blue), the nodes around the crack tip (status = 2, yellow) and mixed nodes (status = 3, red). Knowing that the crack coincide with a node of the mesh on the upper side and finishes inside an element on the lower side, we get the results we wanted. The nodes with status = 1 are circled nodes and those with status = 2 are squared nodes in Figure 2.5 on p.8. The fracture is at the same place as in Figure 3.1. Finally investigations on a bigger mesh have been done, using a mesh previously created for other Aster tests. It consists of an inclined crack with an angle to the horizontal plane of 37 degrees. This mesh is composed of 14888 nodes and 6674 elements (see Figure 3.3). On this mesh the crack tip local basis is tested (see Figure 3.4). This basis corresponds to level set gradients and therefore the first vector can be written $\vec{e_t} = \pm(cos(\beta)\vec{e_x} + sin(\beta)\vec{e_y})$ depending on which side of the crack the point is located. 12 points are tested with both methods of crack definition (functions or with segment elements). The maximum error range that we find is of 10^{-13} which is conclusive.

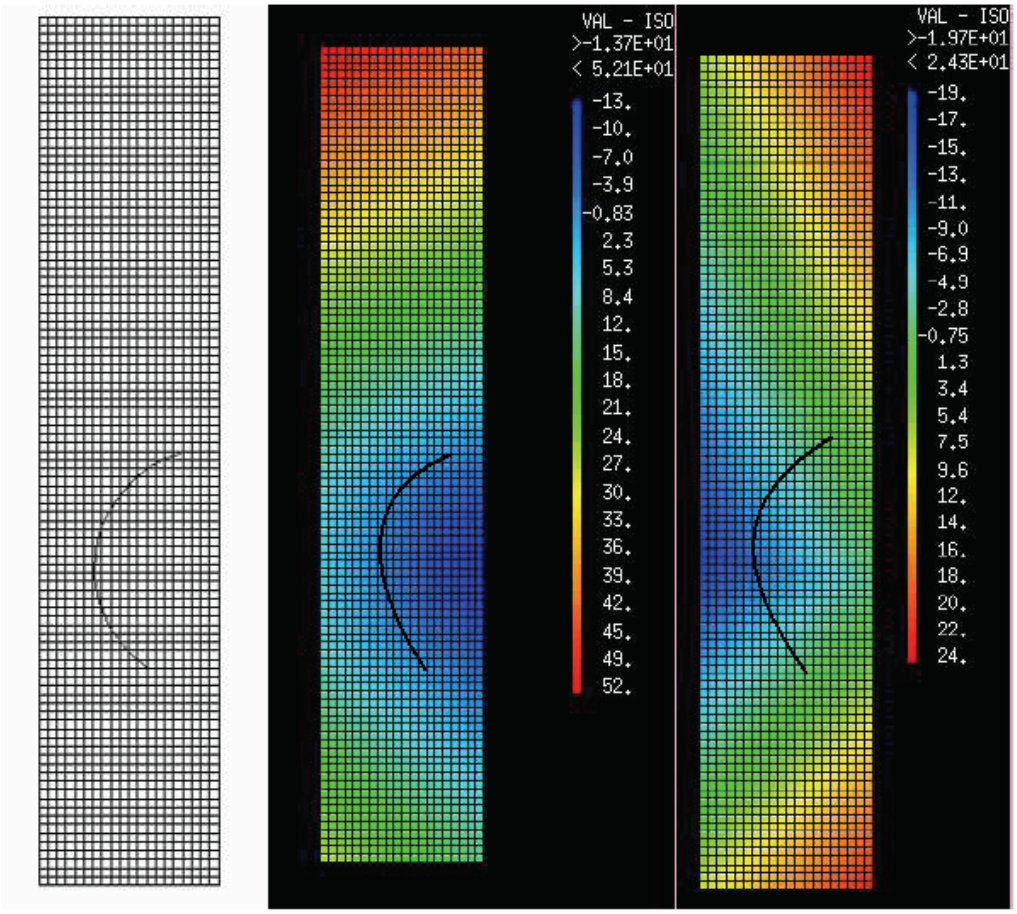

Figure 3.1. Curved crack in a mesh of quadrilaterals and corresponding level sets (normal then tangential)

3.2 Resolution validation in traction (no contact or friction)

3.2.1 5 and 10 elements meshes

It was first decided to test the method on a simple mesh consisting of 5 elements and cut in the middle (see figure 3.5 (a) and (b)). Doing so, it was possible to put displacements on normal elements, without imposing loadings directly on XFEM elements. The model is a simple sheet in traction with quadrilateral elements first (see figure 3.5 (c)), then with 10 triangular elements (see figure 3.5 (d)). The crack is defined by the following functions :

$$F_n(x,y) = y - \frac{LY}{2} \quad F_t(x,y) = -x - 1 \tag{3.1}$$

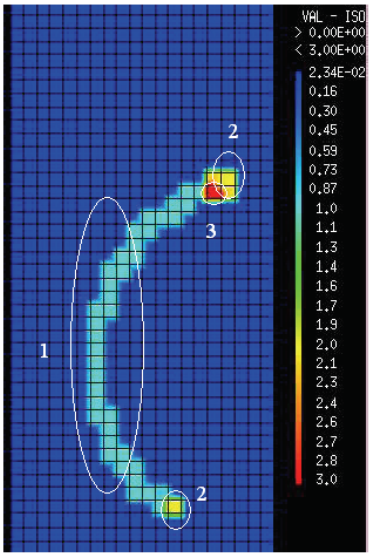

Figure 3.2. Node status

The crack tip is therefore located outside the sheet which permits us to have a completely cut sheet and work only with heaviside terms. On these mesh with quadrilateral elements, there is a cut element in the middle, two other XFEM elements around it and two normal elements at the top and at the bottom. Those elements have different degrees of freedom. Normal elements have the usual degrees of freedom that are called A_x and A_y. XFEM elements have additionally enriched degrees of freedom, to keep the same notation as in the second chapter they are called a_x, a_y and b_x and b_y for the Heaviside degrees of freedom. We bind the bottom level (no displacement possible) and we put a fixed displacement of $U_y = 10^{-6}m$ on the top of the mesh. Theoretical results give no displacements on the X axis, therefore null degrees of freedom, and the following degrees of freedom for the Y axis:

- 1st level of nodes (bottom): $A_y = 0$
- 2nd level of nodes : $A_y = a_y - b_y = 0$ (the displacement for the node is of course equal for both side, therefore for both kind of elements with this node which is between a normal and an XFEM element). $a_y = b_y = U_y/2$
- 2nd level of nodes : $a_y - b_y = 0$ and $a_y = b_y = U_y/2$
- 3nd level of nodes : $a_y - b_y = 0$ and $a_y = b_y = U_y/2$
- 4nd level of nodes : $a_y + b_y = U_y$ and $a_y = b_y = U_y/2$
- 5nd level of nodes : $A_y = a_y + b_y = U_y$ and $a_y = b_y = U_y/2$

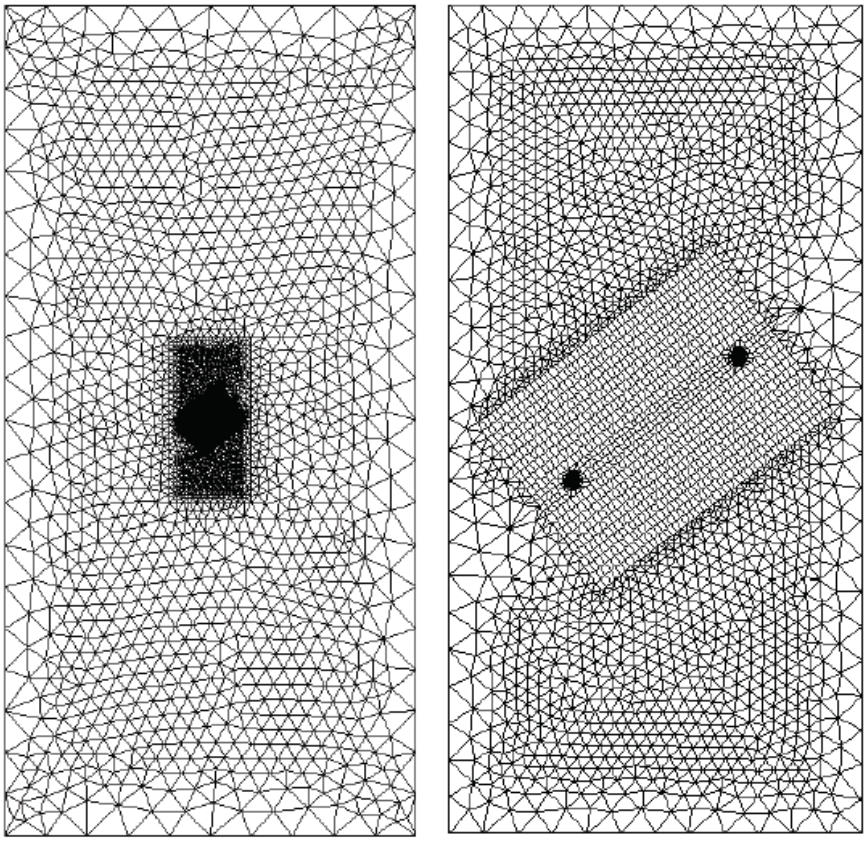

Figure 3.3. Global and inner mesh

- 5nd level of nodes : $A_y = U_y$

Results calculated with Aster are identical to these theoretical values. The maximum error range is in the order of 10^{-12}. We find the expected opening of the crack if we test points over and under the crack. The triangular element mesh gives the same kind of results.

3.2.2 1 element mesh

This case uses a single quadrilateral element. This element is cut by a crack in its middle. This is a XFEM element and therefore loadings have to be imposed directly on XFEM degrees of freedom because normal degrees of freedom no longer exists. Imposed displacements are :

- on the lower nodes, no displacement, $a_y - b_y = 0$ and $a_x - b_x = 0$
- on the upper nodes, displacement of $U_y = 10^{-6}$ on the Y axis, $a_y - b_y = U_y$ and $a_x - b_x = 0$

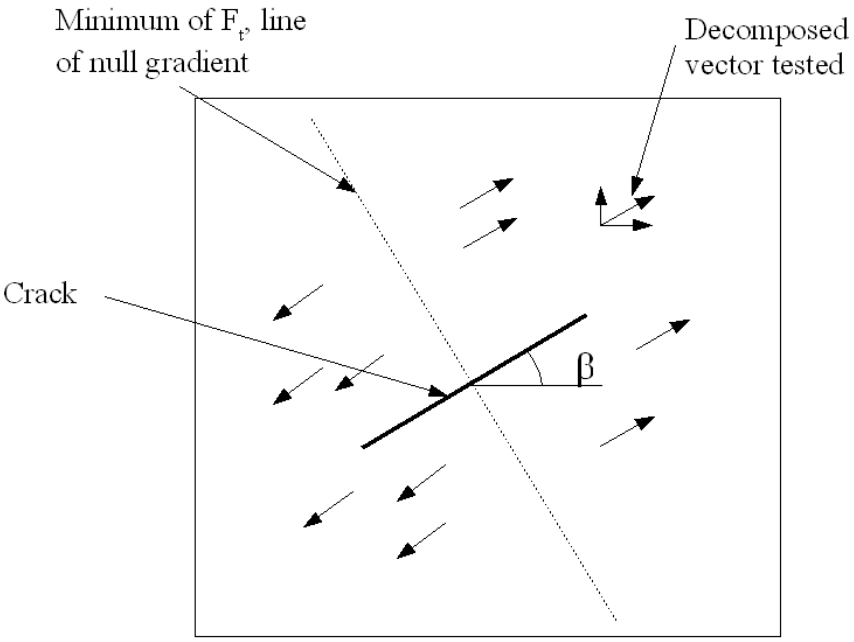

Figure 3.4. Global and inner mesh

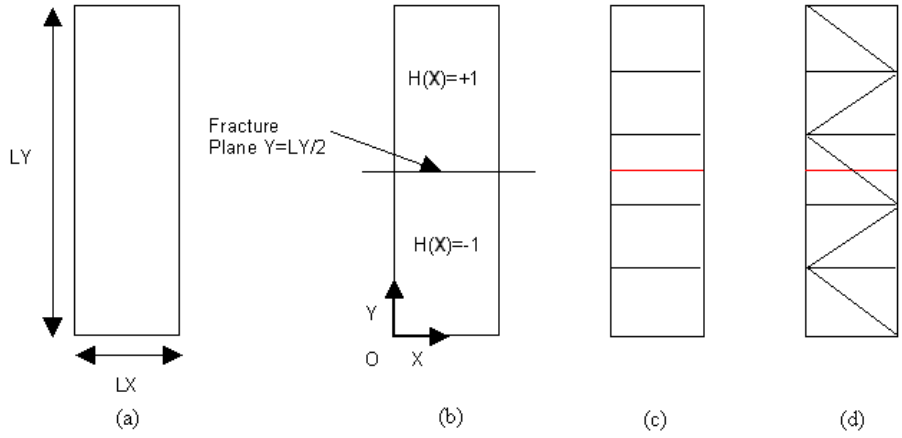

Figure 3.5. Sheet geometry and meshes

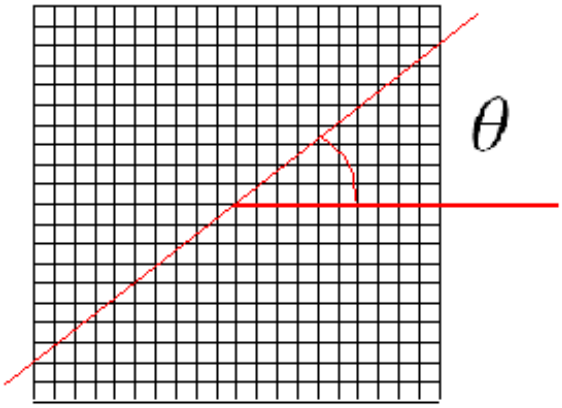

Figure 3.6. Mesh and crack for compression cases

The analytic solution is an opening of the block with $a_y = b_y = U_y/2$ and $a_y = b_y = 0$ for the 4 nodes. These are the results found by Aster, with a very good precision once again, the error is at the maximum in the order of 10^{-13}.

3.3 Resolution validation in compression

This time we use a bigger mesh. We use a 20*20 mesh of quadrilaterals of size $1m^2$ and a fracture with an angle θ with the horizontal plane (see Figure 3.6). This fracture completely cuts the crack. The lower part is blocked and an imposed displacement of $-10^{-6}m$ is imposed in these tests. The Young's modulus of the material is put to $10^8 Pa$.

3.3.1 Contact validation

For the first compression case, the angle θ is equal to 0. Therefore the crack is horizontal and there is no friction involved. That allows to test the contact part first. From (2.28) the expression of the contact multiplier with a linear compression can be found :

$$\lambda = \sigma_{yy} = E\varepsilon = E\frac{U_y}{LY} \tag{3.2}$$

So with the given values $\lambda = -5Pa$ .This value is tested on all the elements along the crack and once again we obtain very good results. The maximum error in the order of 10^{-12}.

3.3.2 Friction validation

The second compression case is with an angle θ equal to -30 degrees. The friction terms can therefore be tested. To stay in small displacement and to avoid sliding

and any need of any new pairing of the nodes, the friction threshold is taken equal to 1000. λ is defined by :

$$\lambda = \mathbf{n} \cdot \sigma \cdot \mathbf{n} = n_y \sigma_{yy} n_y = \sigma_{yy} cos(\theta)^2 = E \frac{U_y}{LY} cos(\theta)^2, \tag{3.3}$$

where $\mathbf{n}$ is the normal to the interface. The half friction multiplier $\mathbf{\Lambda}$ is defined by:

$$\mathbf{r}_\tau = \lambda \mu \mathbf{\Lambda}, \tag{3.4}$$

with the tangential effort density that can be written :

$$\mathbf{r}_\tau = (\tau \cdot \sigma \cdot \mathbf{n})\tau, \tag{3.5}$$

and τ is the tangent to the interface. So :

$$\mathbf{\Lambda} \cdot \tau = \frac{\tau \cdot \sigma \cdot \mathbf{n}}{\mu \mathbf{n} \cdot \sigma \cdot \mathbf{n}} = \frac{\tau_y}{\mu n_y} \tag{3.6}$$

With the given values it holds $\lambda = -3,75Pa$ and $\mathbf{\Lambda} \cdot \tau = \frac{1}{1000\sqrt{3}}$. That's the results obtained with a maximum error in the order of of 10^{-12}.

Chapter 4

A short overview of the Aster code

4.1 Aster, a free finite element software for engineering

The Aster code is a code of calculation developed by EDF since 1989 and based on the finite element method. It collects all the work done by the R&D department of EDF in structure mechanics and the feedback from other users. EDF mainly works on power plants, electricity transportation and distribution. This tool has been developed to answer the needs of engineers for development and maintenance purposes. Therefore ASTER is used by engineers of EDF in many different areas of applications, e.g. mechanics, acoustics, metallurgy, fluid-structure interaction, fatigue, fracture, and so on. It is used to model linear and nonlinear behaviors, statically and dynamically. In order to facilitate the use of Aster by other companies working with EDF, EDF decided 3 years ago to give a free access to Aster to other users. This also helps its development by creating a wider community of users. Aster is therefore now under a gnu general public license and can be downloaded at the website of the Aster code [2].

Aster in numbers means :

- 1 million of lines of code in 3 different language, mostly Fortran, but also C and python,
- about 1500 non-regression and qualification tests,
- 10 000 pages of documentations,
- 200 users at EDF,
- a team of 20 developers,
- an industrial version every 2 years,
- a development version updated every week.

To ensure a non-regression of the code, every modification is tested with previous cases studied on different platforms. Every modification also follows a complicated

process in order not to collide with other modifications developed at the same time. To add a piece of code, the developer first needs to run a verification software, to verify that the code conforms to the Aster rules of code writing (no lower case, no common variables, no unreached piece of code...). Then the restitution program checks that the developer worked with the latest version and that there is no conflict with other developers. The changes need to be discussed and accepted during a development meeting and after that they have to pass all the test cases. An history of modifications is also kept. This procedure allows EDF to ensure the quality quality of the software.

4.2 Development in Aster and work done on the 2D method

The different parts of an Aster program are shown here, with the help of some of the files created for the XFEM implementation in 2D. Those parts can be sorted into Fortran subroutines, catalogs of elements, mesh files and command files. They will be described here. The files are a little bit shorter than they are in practice to keep things readable. The compilation process or any hardware aspect are not included here, because that part was managed by the software Astk, an interface for Aster, but the code was tested on different platforms with the complete Aster program. All the detailed parts of the code can be found at Aster's website [2].

4.2.1 Mesh files

First a mesh has to be created, to model the problem to be solved or as a support for development routines. Gibi, a mesher originally developed for Cast3m [1], was usually used for this part during this master's thesis. For example, to create a mesh of 20*20 quadrilateral elements, the mesh file looks like the following:

- dimension and element choice

```
opti dime 2  elem qua4 ;
```

- Mesh size and number of elements

```
LX = 20; LY = 20;
NX = 20; NY = 20;
```

- Mesh construction : first the points are defined, then the lines and the surface

```
p1 = 0. 0.;
p2 = LX 0.;
p3 = LX LY;
p4 = 0. LY;
lig1 = droit p1 p2 NX;
lig2 = droit p2 p3 NY;
lig3 = droit p3 p4 NX;
lig4 = droit p4 p1 NY;
surf=DALL lig1 lig2 lig3 lig4 PLAN;
```

- Graphic output generation for visual verification

```
trac surf;
```

- Mesh saving and end of file

```
MAILLE = surf;
opti sauv format './mesh.mgib' ;
sauv format maille ;
fin ;
```

4.2.2 Command files

When the mesh file is finished, the problem can be treated by Aster. A command file contains all the command that Aster will execute. This file is written in Python and can also be divided into different parts.

- Name and version information

```
# AJOUT
# TITRE BLOC AVEC INTERFACE EN CONTACT FROTTANT AVEC X-FEM
DEBUT(CODE=_F(NOM='SSNV182D', NIV_PUB_WEB='INTERNET',),);
```

- Mesh reading and modification, the mesh is first read here, then nodes are added in the middle of the edges to allow friction and contact multiplier on these middle nodes.

```
PRE_GIBI();
MAILLAG1=LIRE_MAILLAGE(INFO=1,);
MAILLAG2=CREA_MAILLAGE(MAILLAGE=MAILLAG1,
                       LINE_QUAD=_F(GROUP_MA='SURF',),);
```

- Model declaration.Except for the upper and lower lines where we want to impose displacements, the mesh has an XFEM model with plane stress (C_PLAN_X) .

```
MODELEIN=AFFE_MODELE(MAILLAGE=MAILLAG2,
                     AFFE=(_F(GROUP_MA=('SURF',),
                              PHENOMENE='MECANIQUE',
                              MODELISATION='C_PLAN_X',),
                           _F(GROUP_MA=('LIG1','LIG3',),
                              PHENOMENE='MECANIQUE',
                              MODELISATION='C_PLAN',),),);
```

- Level set and fracture definition. Here the definition of the crack by functions is used (LN and LT for F_n and F_t), and the contact and friction mode and their parameters are specified (μ is COEF_REGU_CONT for example).

```
THETA=-30./180.*pi
LN=FORMULE(NOM_PARA=('X','Y'),VALE='(Y-10)*cos(THETA)-sin(THETA)*(X-10)');
LT=FORMULE(NOM_PARA=('X','Y'),VALE='-X*cos(THETA)-Y*sin(THETA)-100');
FISS=DEFI_FISS_XFEM(MODELE=MODELEIN,
                    DEFI_FISS=_F(FONC_LT=LT,
                                 FONC_LN=LN,),
                    GROUP_MA_ENRI='SURF',
                    CONTACT=_F(INTEGRATION='GAUSS',
                               COEF_REGU_CONT=100.,
                               ITER_CONT_MAXI=4,
                               CONTACT_INIT='OUI',
                               FROTTEMENT='COULOMB',
                               COULOMB=1000.0,
                               ITER_FROT_MAXI=6,
                               COEF_REGU_FROT=1000.,
                               SEUIL_INIT=-5,
                               ),);
```

- The following command will affect the different types of elements and remove irrelevant degrees of freedom.

```
MODELEK=MODI_MODELE_XFEM(MODELE_IN=MODELEIN, FISSURE=FISS,);
```

- Material description : this part specifies the physical properties of the material like the Young's modulus.

```
E=100.0E6
nu=0.
ACIER=DEFI_MATERIAU(ELAS=_F(E=E,NU=nu,RHO=7800.0,),);
CHAMPMAT=AFFE_MATERIAU(MAILLAGE=MAILLAG2,
                       MODELE=MODELEK,
                       AFFE=_F(GROUP_MA=('SURF','LIG1','LIG3',),
                               MATER=ACIER,
                               TEMP_REF=0.0,),);
```

- Loadings : this part impose the different displacement of pressures and the second command links XFEM degrees of freedom and normal degrees of freedom.

```
CH1=AFFE_CHAR_MECA(MODELE=MODELEK,
                   DDL_IMPO=(_F(GROUP_MA='LIG1',
                                DX=0.0,
                                DY=0.0,),
                             _F(GROUP_MA='LIG3',
                                DX=0.0,
                                DY=-1.E-6,),),);
CHXFEM=AFFE_CHAR_MECA(MODELE=MODELEK,LIAISON_XFEM='OUI',);
```

- The next part calls the non linear solver and specifies the time step (here only 1 step), uses the previously defined loadings, models, materials and defines some resolution parameters (number of iteration for the solver, method).

```
L_INST=DEFI_LIST_REEL(DEBUT=0.0,INTERVALLE=_F(JUSQU_A=1.0,NOMBRE=1,),);
UTOT1=STAT_NON_LINE(MODELE=MODELEK,
                    CHAM_MATER=CHAMPMAT,
                    EXCIT=(_F(CHARGE=CHXFEM,),
                           _F(CHARGE=CH1,),),
                    COMP_ELAS=_F(RELATION='ELAS',
                                 GROUP_MA='SURF',),
                    INCREMENT=_F(LIST_INST=L_INST,
                                 INST_FIN=1.0,),
                    CONVERGENCE=(_F(ARRET='OUI',
                                    RESI_GLOB_RELA=1E-9)),
                    SOLVEUR=_F(METHODE='MUMPS',
                               PCENT_PIVOT=250,),
                    NEWTON=_F(REAC_ITER=10,),);
```

- Finally a table is created, which contains the contact and friction variables for all the nodes along the crack. Then this table is tested and the values are compared to known analytic values. If any test fails, Astk will point it out to the user.

```
LAG=POST_RELEVE_T(ACTION=_F(INTITULE='DEPLE',
                            NOEUD=('N261','NS222','NS223','NS224',
                            'NS259','NS260','NS261','NS298','NS299',
                            'NS335','NS336','NS337','NS372','NS373',
                            'NS374','NS411','NS448','NS449','NS485',
                            'NS486','NS487','NS522','NS523',
                            'NS524','NS561','NS562','NS598',
                            'NS599','NS600','NS636','NS638',),
                            RESULTAT=UTOT1,
                            NOM_CHAM='DEPL',
                            NUME_ORDRE=1,
                            NOM_CMP=('LAGS_C','LAGS_F1',),
                            OPERATION='EXTRACTION',),);
REF=-5.*3./4.
REF2=.001/(3.**.5)
# TESTS
TEST_TABLE(TABLE=LAG,
           NOM_PARA='LAGS_C',
           TYPE_TEST='MAX',
           VALE=REF,
           CRITERE='RELATIF',
           PRECISION=1.E-10,
           REFERENCE='ANALYTIQUE',);
TEST_TABLE(TABLE=LAG,
           NOM_PARA='LAGS_C',
           TYPE_TEST='MIN',
           VALE=REF,
           CRITERE='RELATIF',
           PRECISION=1.E-10,
           REFERENCE='ANALYTIQUE',);
TEST_TABLE(TABLE=LAG,
           NOM_PARA='LAGS_F1',
           TYPE_TEST='MAX',
           VALE=REF2,
```

```
           CRITERE='ABSOLU',
           PRECISION=1.E-12,
           REFERENCE='ANALYTIQUE',);
TEST_TABLE(TABLE=LAG,
           NOM_PARA='LAGS_F1',
           TYPE_TEST='MIN',
           VALE=REF2,
           CRITERE='ABSOLU',
           PRECISION=1.E-12,
           REFERENCE='ANALYTIQUE',);
FIN();
```

4.2.3 Fortran code

Now that a mesh and a command file is done, Aster can be run on this simple case. With this kind of support, the code can be developed and the Fortran routines modified. That's the part where the most of the work had been done. The code was greatly inspired from the 3D code already done. Sometimes, there were only small modifications necessary (split the program in a 2D part and a 3D part, test the dimension to change some variables) and sometimes more work was needed. For example for integration, informations have to be kept about the intersection points, the number of the edges concerned, the number of sub-elements and their topology. The size of the objects containing this information is quite different in 2D and in 3D. For example, we have at the most 3 intersection points in 2D for a quadrilateral and we can have until 11 intersections points in 3D for a cube. Therefore all those objects have therefore been resized. The Fortran routine for calculating contact and friction is presented hereafter:

- Routine name and parameters, this option has very few parameters, because it is an elementary routine. The needed variables are obtained later in the program.

```
      SUBROUTINE TE0533(OPTION,NOMTE)
      IMPLICIT   NONE
      CHARACTER*16 OPTION,NOMTE
```

- Version and copyright information and small description of the routine utility :

```
C            CONFIGURATION MANAGEMENT OF EDF VERSION
C MODIF ELEMENTS  DATE 22/02/2006   AUTEUR MASSIN P.MASSIN
C ======================================================================
C COPYRIGHT (C) 1991 - 2005  EDF R&D                  WWW.CODE-ASTER.ORG
...
C         CALCUL DES MATRICES DE CONTACT FROTTEMENT POUR X-FEM
C                        (METHODE CONTINUE)
```

- Standard common memory allocation variables. Aster has it's own memory allocation management with the procedure JEVEUX. For example to get a integer stored at a particular address, one will just use ZI(address).

```
C --------- DEBUT DECLARATIONS NORMALISEES  JEVEUX --------------------
      INTEGER ZI
      COMMON /IVARJE/ZI(1)
      ...
      COMMON /KVARJE/ZK8(1),ZK16(1),ZK24(1),ZK32(1),ZK80(1)
C --------- FIN  DECLARATIONS  NORMALISEES  JEVEUX --------------------
```

- Declaration of Fortran variables used in the routine.

```
      INTEGER      I,J,K,L,IJ,IFA,IPGF,INO,ISSPG,NI,NJ,NLI,NLJ,PLI,PLJ
      INTEGER      JINDCO,JDONCO,JLSN,IPOIDS,IVF,IDFDE,JGANO,IGEOM
      ...
      REAL*8       PTKNP(3,3),TAUKNP(2,3),TAIKTA(2,2),IK(3,3),NBARY(3)
      REAL*8       LSN,LST,R,RR
```

- Initialization of the different variables and objects needed for computation, for example dimension, gauss points families, address for the level sets or topology of contact elements.

```
C     INITIALISATIONS
      CALL ELREF1(ELREF)
      CALL ELREF4(' ','RIGI',NDIM,NNO,NNOS,NPG,IPOIDS,IVF,IDFDE,JGANO)
      ...
      CALL JEVECH('PLSN','L',JLSN)
      CALL JEVECH('PLST','L',JLST)
      CALL JEVECH('PPINTER','L',JPTINT)
      ...
```

- Here comes the main program, large pieces were removed from that part. There were different kinds of modification testing the dimension, here can be seen an IF loop testing the dimension to choose the gauss points family for integration (FPG). Then the contribution of 1 gauss point of 1 contact element to the matrix A_u is calculated, this matrix has now a size which will depend of the dimension as almost every other object in the program.

```
      IF (NDIM .EQ. 3) THEN
       IF (INTEG.EQ.1) FPG='XCON'
       IF (INTEG.EQ.4) FPG='FPG4'
       IF (INTEG.EQ.6) FPG='FPG6'
       IF (INTEG.EQ.7) FPG='FPG7'
      ELSE
      FPG='MASS'
      ENDIF
      ...
      DO 100 IFA=1,NFACE
        DO 110 IPGF=1,NPGF
          IF (OPTION.EQ.'RIGI_CONT') THEN
            IF (INDCO(ISSPG).EQ.0) THEN
            ...
            ELSE IF (INDCO(ISSPG).EQ.1) THEN
C             I.2. CALCUL DE A_U
              DO 140 I = 1,NNO
                DO 141 J = 1,NNO
```

```
                DO 142 K = 1,DDLH
                  DO 143 L = 1,DDLH
                    MMAT(DDLS*(I-1)+NDIM+K,DDLS*(J-1)+NDIM+L) =
    &               MMAT(DDLS*(I-1)+NDIM+K,DDLS*(J-1)+NDIM+L)+
    &               4.D0*RHON*FFP(I)*FFP(J)*ND(K)*ND(L)*JAC*MULT
143               CONTINUE
142             CONTINUE
141           CONTINUE
140         CONTINUE
          ...
110     CONTINUE
100   CONTINUE
```

- Data saving and end of routine. The calculated matrix is stored in an Aster object.

```
      DO 200 J = 1,NDDL
        DO 210 I = 1,J
          IJ = (J-1)*J/2 + I
          ZR(IMATT+IJ-1) = MMAT(I,J)
210     CONTINUE
200   CONTINUE
      END
```

4.2.4 Catalogs of elements

The previous routine was an elementary subroutine. Aster will first call global routines, but then some part of the calculations will have to be done on each element. To go from global subroutine to elementary subroutine, Aster calls a routine that looks for the type of element, and for each type of element Aster will choose a certain elementary subroutine and associated parameters. This fork is done with the help of catalogs. For example, 4 catalogs have been developed for XFEM in 2D. They use new routines, or routines from XFEM in 3D with other parameters and define some properties of the elements. The different parts of a catalog are described here (pieces of the catalog have been removed). The following catalog is used for all mixed elements (both enrichments on quadrilaterals and triangles).

- Version and copyright information, as for other Aster's files.

```
%& MODIF TYPELEM  DATE 09/01/2006   AUTEUR GENIAUT S.GENIAUT
%            CONFIGURATION MANAGEMENT OF EDF VERSION
% ======================================================================
% COPYRIGHT (C) 1991 - 2005  EDF R&D                  WWW.CODE-ASTER.ORG
% THIS PROGRAM IS FREE SOFTWARE; YOU CAN REDISTRIBUTE IT AND/OR MODIFY ...
```

- First the catalog name and elements description are given (name, type, sub elements or edges, gauss points families and group of nodes).

```
GENER_MECPL2_XHT
TYPE_GENE__
...
```

```
ENTETE__ ELEMENT__ MECPQU8_XHT      MAILLE__ QUAD8
   ELREFE__  QU4       GAUSS__  RIGI=FPG4  MASS=FPG4  XFEM=XFEM72
   ELREFE__  TR3       GAUSS__  RIGI=FPG1  MASS=FPG3  XFEM=XFEM36
   ELREFE__  SE2       GAUSS__  RIGI=FPG2  MASS=FPG3
   ENS_NOEUD__  EN2     =     5   6   7   8
   ENS_NOEUD__  EN1     =     1   2   3   4
```

- After that the variables used are shown, here for example the displacement and its degrees of freedom for both kind of nodes (vertex nodes or middle node), a variable for intersection topology and the geometric coordinates of the nodes.

```
MODE_LOCAL__
    DDL_MECA = DEPL_R   ELNO__ DIFF__
                                EN1   (DCX    DCY    H1X    H1Y    E1X    E1Y
                                       E2X    E2Y    E3X    E3Y    E4X    E4Y
                                       LAGS_C LAGS_F1 )
                                EN2   (LAGS_C LAGS_F1 )
    E8NEUTI  = NEUT_I   ELEM__         (X1     X2     X3     X4     X5     X6
                                       X7     X8      )
    NGEOMER  = GEOM_R   ELNO__ IDEN__  (X      Y       )
```

- Here the elementary subroutines are called, the previous subroutine might be called for 2 different options RIGI_CONT and RIGI_FROT, one calculates the contact stiffness matrix and the other one calculates the friction stiffness matrix. The catalog gives the option name, the name of the subroutine called (te0xxx.f, here te0533.f) and the structures used by the subroutine.

```
OPTION__
 RIGI_CONT   533 IN__  NGEOMER  PGEOMER  DDL_MECA PDEPL_M  DDL_MECA PDEPL_P
                       E1NEUTI  PINDCOI  CONTX_R  PDONCO   N1NEUT_R PLST
                       N1NEUT_R PLSN     E4NEUTR  PPINTER  E8NEUTR  PAINTER
                       E2NEUTI  PCFACE   E2NEUTI  PLONCHA  E8NEUTR  PBASECO
                       E1NEUTR  PSEUIL
                  OUT__ MMATUUR  PMATUUR
 RIGI_FROT   533 IN__  NGEOMER  PGEOMER  DDL_MECA PDEPL_M  DDL_MECA PDEPL_P
                       E1NEUTI  PINDCOI  CONTX_R  PDONCO   N1NEUT_R PLST
                       N1NEUT_R PLSN     E4NEUTR  PPINTER  E8NEUTR  PAINTER
                       E2NEUTI  PCFACE   E2NEUTI  PLONCHA  E8NEUTR  PBASECO
                       E1NEUTR  PSEUIL
                  OUT__ MMATUUR  PMATUUR
```

Chapter 5

Conclusion

Aster's extended finite element method in 2D has been implemented and verified in different cases. All the results obtained seem to show that there is no more problem with level sets, XFEM elements, contact and friction resolution. But there are still a lot of improvement to do on this method to be able to use it in a wider context as just a complement to the Aster classic code.

Different possible improvement have already been started or scheduled. Some of those improvement are:

- Calculation of stress intensity factors from the displacement obtained at the crack tip in order to calculate how will behave the crack.
- Propagation of the level set with propagation of the crack in order to model crack propagation.
- Implementation of multiples fractures within XFEM, with multiple level sets and enrichment functions for use in various application, for example thermic cracking.

Bibliography

[1] http://www-cast3m.cea.fr/.

[2] http://www.code-aster.com.

[3] T. Belytschko, Y. Krongauz, D. Organ, and M. Flemming. Meshless methods : an overview and recent developments. *Computer Methods in Applied Mechanics and Engineering*, 139:3–47, 1996.

[4] D. Colombo and M. Giglio. A methodology for automatic crack propagation modelling in planar and shell fe models. *Engineering Fracture Mechanics*, 73:490–504, 2006.

[5] A. Gravouil, N. Moës, and T. Belytschko. Non-planar 3d crack growth by the extended finite element and level sets - part ii : Level set update. *International Journal for Numerical Methods in Engineering*, 53:2569–2586, 2002.

[6] G. R. Irwin. Analysis of stresses and strains near the end of crack traversing a plate. *Journal of applied mechanics*, September 1957.

[7] P. Massin, H. Ben Dhia, and M. Zarroug. *Contact elements derived from a continuous hybrid formulation.* Aster reference documentation, number [R5.03.52].

TRITA-CSC-E 2006:043
ISRN-KTH/CSC/E--06/043--SE
ISSN-1653-5715

www.kth.se