Two-dimensional Fracture Modeling with the Generalized/Extended Finite Element Method: An Object-Oriented Programming Approach

Mohammad Malekanabc1, Leandro L. Silvaa, Felicio B. Barrosa, Roque L.S. Pitangueiraa, Samuel S. Pennaas

aGraduate Program in Structural Engineering (PROPEEs), School of Engineering, Federal University of Minas Gerais (UFMG), Belo Horizonte-MG, Brazil

Abstract

This work presents an object-oriented implementation of the G/XFEM to model the crack nucleation and propagation in structures made of either linear or nonlinear materials. A discontinuous function along with the asymptotic crack-tip displacement fields are used to represent the crack without explicitly meshing its surfaces. Different approach are explained in detail that are used to capture the crack nucleation within the model and also determine the crack propagation path for various problems. Stress intensity factor and singularity of the localization tensor (which provides the classical strain localization condition) can be used to determine the crack propagation direction for linear elastic materials and nonlinear material models, respectively. For nonlinear material model, the cohesive forces acting on the crack plane are simulated in the enrichment process by incorporating a discrete constitutive model. Several algorithms and strategies have been implemented, within an object-oriented framework in Java, called INSANE. This implementation will be presented in detail by solving different two-dimensional problems, for both linear and nonlinear material models, in order to show the robustness and accuracy of the proposed method. The numerical results are compared with the reference solutions from the analytical, numerical or experimental results, where applicable.

Keywords: Generalized/Extended Finite Element Method, Fracture Mechanics, Object-Oriented Programming, Linear and Nonlinear Materials

1. Introduction

Fracture analysis using standard finite element method (FEM) is quite limited. In order to model crack propagation, remeshing is always needed to match the new geometry of the crack, except for some techniques such as smeared cracking approaches [22] that do not require remeshing. Generalized or extended FEM (G/XFEM) has been proposed to facilitate the modeling of arbitrary crack geometry and its evolution. It also eliminates need for remeshing and conformity to element boundaries. In G/XFEM [16, 80], as in the FEM, the approximation is built over a mesh of elements using interpolation functions. Special functions multiply the original FEM functions and smooth as well as non-smooth solutions can be modeled independently of the mesh. In addition, there are other methods that can handle the crack propagation with their special techniques, such as: Mesh-free method [6, 13], cracking particle method [5, 75] or efficient remeshing techniques [4, 48].

Application of object-oriented programming for FEM has been receiving great attention over the last two decades [56], for example: a FE analysis to solve structural problems using OOP approach within: Object NAP code [39], FEMOBJ [93], and OOFEM code [70]; a finite element differential equations analysis library [11]; an object-oriented environment to solve multidisciplinary problems (combination of thermal, fluid dynamics, and structural different fields) [27]; implementation of a unified library of nonlinear solution schemes in FE programming scheme [54]. Beside this, the OOP has been successfully used to represent also different numerical methods, such as the boundary element method [51] and meshfree methods [12]. Also, a bunch of G/XFEM codes used object-oriented concept as their implementation strategy: an extension of a FEM code by adding the G/XFEM enrichment strategy [81, 82]; demonstration of an open source architecture for G/XFEM so-called openxfem++ [19]; an extension the OOFEM code to include the G/XFEM method [24]; implementing a G/XFEM code from scratch [35]; automated meshing for integrated experiments project proposed by Dunant [34]; and a G/XFEM implementation in Python by Neto et al [68].

Email address: mmalekan1986@gmail.com, malekan@dees.ufmg.br (Mohammad Malekan)

1Corresponding author
Initial implementation of the G/XFEM for crack propagation problems was introduced by Belytschko and Black [16] and Moës et al [67]. After that, this method has been extensively used for the simulation of crack propagation problems, such as: three-dimensional crack propagation was proposed by Duarte et al [33]; a quasi-static crack growth was proposed by Sukumar and Prévost [83] using G/XFEM; three-dimensional modeling of initiation, branching, growth of crack in non-linear solids including statics and dynamics problems was presented by Bordas et al [20]; Rabczuk et al [76] proposed some crack tracking techniques for three-dimensional problems using partition of unity and meshfree technique; and an enhanced G/XFEM method for modeling of dynamic crack branching presented by Xu et al [90]. Dolbow et al [30] modeled the fracture in Reissner-Mindlin plate with XFEM and computed the mixed-mode stress intensity factor using the domain forms of interaction integral. Subsequently, different researchers have used G/XFEM formulations based on Reissner-Mindlin plate theory to develop fracture modeling in shell and plate structures, see for example [14, 52, 65, 92].

Although, there are many academic code implementing the G/XFEM approach for either static, quasi-static or dynamic analyses, as discussed earlier, but few of them address detail implementation aspects, such as openxfem++ [19] (for stationary and crack propagation problems), OOFEM code [24] (for stationary crack problems), and G/XFEM in [3] (for stationary crack problems). However, none of them considered the crack nucleation into the problems. The work presented in [19] was almost an extension library which was designed to add XFEM capabilities to an existing FEM code, FEMOBJ [93]. The challenging issue is the integration of new implementation with an existing code. Chamrová and Patzák [24] presented an object-oriented approach implementation of the G/XFEM by extending the OOFEM, an available FEM code. The object-oriented structure of the this code was described in detail, including the role of individual classes and their mutual relations. In addition, an available FEM programming environment was expanded to enclose the standard version of the G/XFEM in [3], to analyze the static problems only for linear materials. This environment, so-called INSANE (INteractive Structural ANalysis Environment)³ is an open source software available at http://www.insane.dees.ufmg.br.

The aim of this work is to fill this lack in the literature by presenting a new computational framework for crack nucleation and propagation that covers both linear and nonlinear material models. The whole implementation is based on object-oriented programming with capabilities of the G/XFEM that facilitates the expansion of the current implementations for other numerical approaches, such as meshless method. The advantages of the current OOP design as well as the whole implementations are discussed in detail with the use of various OOP features, such as abstract classes and single/multiple inheriances. A discontinuous function along with the asymptotic crack-tip displacement fields are used to represent the crack without explicitly meshing its surfaces. Different approach are explained in detail that are used to capture the crack nucleation within the model and also determine the crack propagation path for various problems. Stress intensity factor and singularity of the localization tensor (which provides the classical strain localization condition) can be used to determine the crack propagation direction for linear elastic materials and nonlinear material models, respectively. For nonlinear material model, the cohesive forces acting on the crack plane are simulated in the enrichment process by incorporating a discrete constitutive model. This constitutive model is defined as the traction-displacement relation in the crack path and is based on the concept of cohesive crack. An outline of the present paper is as follows. A general formulation of the classical G/XFEM is presented in section 2. Section 4 presents a crack nucleation criterion and related formulations for fracture modeling of nonlinear medium. Section 5 provides explanation along with corresponding formulation for crack propagation process. The object-oriented implementation environment, INSANE, and its explanation for the present work is discussed in section 6. In section 7, the fracture modeling approach is applied to different problems which emphasizes the main ideas of the current implementations, and concluding remarks are brought in the final section.

2. Generalized/Extended FEM

The G/XFEM was developed for modeling structural problems with discontinuities [16, 32, 33, 64]. Furthermore, it can be considered an instance of the Partition of Unity Method, PUM [10], in the sense that it employs a set of Partition of Unity, PU, functions to guarantee interelement continuity. Such strategy creates conforming approximations which are improved by a nodal enrichment scheme. The enrichment scheme is obtained by multiplying a PU function of $C^0$ type with compact support $\omega_j$ by the function $L_{ji}(x)$, named as a local approximation (also called enrichment function). The resulting shape function $\phi_{ji}(x)$ inherits characteristics of both functions, i.e., the compact support and continuity of the PU and the approximate character of the local function.

³The source code is available at the Git repository at https://git.insane.dees.ufmg.br/insane/insane.git.
As a consequence, the generalized global approximation, denoted by \( \tilde{u}(x) \), can be described as a linear combination of the shape functions associated with each node:

\[
\tilde{u}(x) = \sum_{j=1}^{N} N_j(x) \left\{ u_j + \sum_{i=2}^{q} L_{ji}(x) b_{ji} \right\}
\]  

(1)

where \( u_j \) is a nodal parameter associated with standard FE shape function \( N_j(x) \), \( b_{ji} \) is nodal parameter associated with G/XFEM shape functions \( N_j(x) \cdot L_{ji}(x) \). The enrichment function can be either continuous or discontinuous function, depending on the problem type. An example of the enrichment function, \( L_{ji} \), by considering the singularities can be defined as [32]:

\[
xL_{j\alpha}^s(x)_{\alpha=1} = \frac{1}{2G} r^{\lambda_1} \left\{ [\kappa - Q_1(\lambda_1 + 1)] \cos \lambda_1 \theta - \lambda_1 \cos(\lambda_1 - 2)\theta \right\}
\]  

(2)

\[
yL_{j\alpha}^s(x)_{\alpha=1} = \frac{1}{2G} r^{\lambda_1} \left\{ [\kappa + Q_1(\lambda_1 + 1)] \sin \lambda_1 \theta + \lambda_1 \sin(\lambda_1 - 2)\theta \right\}
\]  

(3)

where \( r \) and \( \theta \) are the polar coordinates centered on the crack tip, \( \lambda_1 = 0.5, Q_1 = 1/3 \) are coefficients of the first term of the solution in the neighborhood of the crack-tip, considering only the mode-I displacement field. Also, \( \kappa = 3 - 4\nu \) and \( \kappa = \frac{3 - \nu}{1 + \nu} \) for plane strain and plane stress analysis, respectively, and \( G = \frac{E}{2(1 + \nu)} \). The superscripts \( x \) and \( y \) are referred to \( x \)- and \( y \)-directions, respectively. Another example of the enrichment function in which called the near-tip enrichment, is defined as [16]:

\[
[F_i(r, \theta)] = \left[ \sqrt{r} \sin \frac{\theta}{2}, \sqrt{r} \cos \frac{\theta}{2}, \sqrt{r} \sin \frac{\theta}{2} \sin \theta, \sqrt{r} \cos \frac{\theta}{2} \sin \theta \right]
\]  

(4)

where \( i \) is the number of crack-tip functions \( F_i(r, \theta) \) and \( (r, \theta) \) denotes the local polar coordinate defined at the crack-tip.

3. Reissner-Mindlin Plate Formulation

A structural element which is thin and flat is called plate. The thin means that the plate transverse dimension, or thickness, is small compared to the length and width dimensions. The Reissner-Mindlin plate theory are applied to very thin \( L/t > 100 \), moderately thin, \( 20 < L/t < 100 \), and plates, where \( t \) and \( L \) represent the plate thickness and a representative length or width dimension. Reissner-Mindlin plate theory assumes that the normals to the plate do not remain orthogonal to the mid-plane after deformation, thus allowing for transverse shear deformation effects. This allows to use \( C^0 \) continuous elements.

The assumptions of the Reissner-Mindlin plate theory are the following: (1) the points belonging to the middle plane \( (z = 0) \), \( u = v = 0 \), which means the points on the middle plane, only move vertically; (2) the points along a normal to the middle plane have the same vertical displacement (i.e., the thickness does not change during deformation); (3) the normal stress \( \sigma_z \) is negligible (plane stress assumption); and (4) straight line normal to the undeformed middle plane remains straight but not necessarily orthogonal to the middle plane after deformation. More importantly, it is assumed that the middle plane is placed exactly at the same distance from the upper and lower faces.

Following [30], equilibrium equations for a Reissner-Mindlin plate under bending moment and shear force are:

\[
\nabla \cdot M - Q = 0
\]  

(5a)

\[
\nabla \cdot Q = 0
\]  

(5b)

where \( M \) and \( Q \) are vectors representing the moment and shear force components applied over the plate. The constitutive relationships can be obtained by energetic equivalence between the plate and three-dimensional model. Assuming the plate is made of an isotropic homogeneous material, the constitutive relations are given by:
\[
\begin{bmatrix}
M_{xx} \\
M_{yy} \\
M_{xy}
\end{bmatrix} = \frac{Et^3}{12(1-\nu^2)}
\begin{bmatrix}
1 & \nu & 0 \\
\nu & 1 & 0 \\
0 & 0 & 1-\nu
\end{bmatrix}
\begin{bmatrix}
\epsilon_{b}^{xx} \\
\epsilon_{b}^{yy} \\
\epsilon_{b}^{xy}
\end{bmatrix}
\] 
(6a)

\[
\begin{bmatrix}
Q_{xz} \\
Q_{yz}
\end{bmatrix} = \mu kt
\begin{bmatrix}
\epsilon_{b}^{xz} \\
\epsilon_{b}^{yz}
\end{bmatrix}
\] 
(6b)

where \(\epsilon_{b}\) and \(\epsilon_{s}\) are bending and shear deformations, respectively, \(E\) is the Young’s modulus, \(\nu\) is Poisson’s ratio, \(\mu\) is the shear modulus, and \(t\) represents the plate thickness. The correction factor \(k\) accounts for the parabolic variation of the transverse shear stresses through the thickness of the plate, and is taken to be \(k = 5/6\).

4. Crack Nucleation Procedure

In the fracture modeling procedure, the discontinuity initiation can be identified by the appearance of a discrete failure from an initial degradation which can be considered as the crack nucleation stage. The formation of a crack is a transient process that can be divided into three stages: diffuse failure, weak discontinuity and strong discontinuity [45]. At the diffuse failure stage, the strain and displacement fields are continuous, but there is a concentration of strains in the zone where the material begins to soften. At the weak discontinuity stage, the strain field becomes discontinuous, but the displacement field remaining continuous, across the limits of a narrow band (strain localization band). This stage precedes the strong discontinuity stage, in which the region of strain localization collapses and forms a discrete crack. Therefore, the phenomenon of strain localization can be investigated to characterize the crack nucleation threshold stage [50]. With this purpose, the second-order localization tensor can be defined as follows:

\[
Q = n \cdot D \cdot n
\] 
(7)

where \(n\) is a unit vector normal to the discontinuity surface and \(D\) is the tangent stiffness tensor. According to [46], the localization tensor is singular at the early formation of a weak discontinuity. This brings us to the classical localization condition:

\[
\det Q = 0
\] 
(8)

From the mathematical point of view, singularity of the localization tensor indicates the so-called loss of ellipticity of the governing differential equation. The localization tensor defined in Eq. (7) depends on the tangent stiffness tensor \(D\) and on the unit vector normal to the discontinuity surface, \(n\). With certain exceptions, the tangent stiffness can be known if it is considered dependent only on the current state, which is strain state. The vector \(n\), however, is not given in advance. Therefore, localization analysis consists in searching for a unit vector \(n\) for which the localization tensor becomes singular. If such a vector does not exist, the strain field must remain continuous. Singularity of the localization tensor for a certain vector \(n\) indicates that a strain jump can develop across a surface with normal \(n\). In the fracture modeling of nonlinear medium presented here, the singularity of the localization tensor is investigated to indicate crack nucleation as well as crack propagation.

The strategy (in 2D analysis) adopted to investigate a unit vector \(n\) for which the localization tensor is singular does not directly determine the vector \(n\), but the system defined by this vector and the direction of the crack. Therefore, the singularity of the localization tensor is investigated only to indicate the crack nucleation/propagation and not for the determination of the unit vector \(n\). To determine the crack orientation, this direction is defined as the normal direction to the maximum principal strain. Due to the step size of the nonlinear analysis and the mesh refinement, it is possible that the balanced system results in the introduction of a crack in a line of elements simultaneously, rather than in only one element. Therefore, the element where the nucleation phenomenon was identified can actually be part of a line of nucleated elements, not the only one where the crack should be introduced at this moment. The strategy for introducing a crack starts from the determination of the element in this line of elements that should contain the initial segment of the crack. Once this element is defined, it is assumed that the initial segment of the crack passes in its centroid and has the direction defined as the normal direction to the maximum non-local principal strain calculated at this point. The other segments are disposed from the crack tip to propagate with the direction defined as the direction normal to the maximum non-local principal strain calculated at this crack-tip.
5. Representation of Fixed and Moving Discontinuities

This section presents the procedure of discontinuity modeling in order to analyze a static crack propagation problem. The term crack growth or crack propagation here is referred to quasi-static crack growth, in which inertia effects are neglected. In this approach, the problem is assumed in equilibrium at all time steps.

5.1. Crack Representation Procedure

The traditional approach to analyze a problem with discontinuity is to generate the mesh to conform to the line, or surface, of discontinuity $\Gamma_c$ in Fig. 1. However, in the G/XFEM the discontinuity along $\Gamma_c$ is modeled using special enrichment functions that can describe the discontinuity as well as the stress field behavior close to the crack tip. In this case, the appropriate enrichment function must be selected and applied to those nodes that are around/close to the discontinuity surfaces. The signed distance function along with the so-called Heaviside function are used here to represent the discontinuity in a model. For linear elastic fracture mechanics, the crack-tip singularity can be captured with either singular enrichment (Eqs. (2) and (3)) or near-tip enrichment function (Eq. (4)). Following subsections are presented in detail the procedure of modeling a discontinuity within a problem using the G/XFEM approach and also the corresponding formulation that is used for this research.

5.1.1. Signed Distance Function

The level-set method is a numerical tool for the tracking of the moving interfaces [78]. The signed distance function is one particularity of the level-set method which is used to represent the moving interface.

Consider a domain $\Omega$ divided into two non-overlapping domains $\Omega_A$ and $\Omega_B$, sharing an interface, or surface of discontinuity, denoted by $\Gamma_c$, as shown in Fig. 1. The signed distance function is defined for the representation of the interface position as:

$$
\phi(x) = \| x - x^* \| \text{ sign} (n_{\Gamma_c} \cdot [x - x^*])
$$

where $x^*$ is the closest point projection of $x$ onto the discontinuity $\Gamma_c$, and $n_{\Gamma_c}$ is the vector normal to the interface at point $x^*$. In this definition, $\| \|$ denotes the Euclidean norm, where $\| x - x^* \|$ specifies the distance of point $x$ to the discontinuity $\Gamma_c$ (Fig. 1).
5.1.2. The Heaviside Function

A strong discontinuity is defined as a jump in the displacement field. The discontinuity in the displacement occurs where the displacement of one side of the crack is completely different from the displacement field of the other side. In such cases, the kinematics of the strong discontinuity can be defined based on the Heaviside function [17]. This function is one of the most popular functions used to model the crack discontinuity in the Gi/XFEM formulation and is defined as:

\[ H(x) = \begin{cases} 
1 & \text{if } \phi(x) > 0 \\
0 & \text{if } \phi(x) < 0 
\end{cases} \]  

(10)

in which \( \phi(x) \) is the signed distance function, defined in Eq. (9), and the discontinuity can be represented using this function as, according to Fig. 1:

\[ \phi(x) = \begin{cases} 
> 0 & \text{if } x \in \Omega_A \\
< 0 & \text{if } x \in \Omega_B 
\end{cases} \]  

(11)

The approximation field, Eq. (1), can now be written as, by including the Heaviside enrichment function \( H(x) \):

\[ \tilde{u}(x) = \sum_j N_j(x)u_j + \sum_i N_i(x)H(x)b_i \]  

(12)

5.1.3. Node Selection for Enrichment Strategy

Heaviside enrichment function enriches the nodes that belongs to those element that are completely cut by the discontinuity. According to [31], a direct use of this approach could provide an ill-conditioned stiffness matrix. Consider a crack/discontinuity cutting through some elements, as shown in Fig. 2. Since the crack doesn’t cross through element \( E \) from Fig. 2(a), nodes \( i \) and \( j \) are enriched by Heaviside function whereas nodes \( k \) and \( l \) are not enriched. In other case, if crack crosses the element \( E \) completely, then all nodes must be enriched by the Heaviside function. However, the classical and enriched shape functions at these nodes will only differ in the very thin band of width \( \varepsilon \) (Fig. 2(b)), leading to ill-conditioned system of equations. This happens because the resulting basis functions are almost identical.

\[ \frac{A^+}{A^+ + A^-} \quad \text{or} \quad \frac{A^-}{A^+ + A^-} \]

(13)

In this particular situation, nodes \( k \) and \( l \) must not be enriched by the Heaviside function. To overcome this situation, a criterion is defined by Dolbow [31] in which, for a certain node \( j \) (see Fig. 2(c)), if the values of \( A^+/(A^+ + A^-) \) or \( A^-/(A^+ + A^-) \) are smaller than the allowable tolerance value of \( 10^{-4} \), the node must not be enriched. The \( A^+ \) and \( A^- \) are the portions of the area of the influence domain of a node above and below the crack, respectively.
5.1.4. Construction of a Discontinuous Approximation

In a more general case such as that shown in Fig. 3, the crack tip will not coincide with an element edge, and in this instance the discontinuity cannot be adequately described using only a function such as \( H(x) \). The jump enrichment of the circled nodes in this case only provides for the modeling of the discontinuity until point \( P \). To seamlessly model the entire discontinuity along the crack, the squared nodes are enriched with the asymptotic crack tip functions with the technique developed in [16]. The approximation for the case of an arbitrary crack, as shown in Fig. 3, takes the form:

\[
\hat{u}(x) = \sum_{j \in I} N_j(x) u_j + \sum_{i \in I} N_i(x) H(x) b_i + \sum_{k \in K_1} N_k(x) \left( \sum_{l=1}^n C_{kl}^1 L_1^l \right) + \sum_{k \in K_2} N_k(x) \left( \sum_{l=1}^n C_{kl}^2 L_2^l \right)
\]

in which \( J \) is the set of all nodes, \( I \) is the set of nodes enriched with Heaviside function, \( K_1 \) and \( K_2 \) are the sets of nodes to be enriched for the first and second crack tip, respectively. The function \( L_1^l(x) \) and \( L_2^l(x) \) are the crack tip enrichments that can be either the singular enrichment of Eqs. (2) and (3) or near tip enrichment function (Eq. (4)), and \( n \) is number of enrichment functions used.

![Figure 3: Local axes for the polar coordinates at the crack tips for an arbitrary crack shape and types of nodes in a general case.](image)

5.1.5. Traction on the Crack Surface

To simulate the cohesive tractions acting on the crack surfaces, a discrete constitutive model is incorporated into the enrichment process, which consists of the traction-displacement relation in the crack path. In this case, only Heaviside enrichment is used, i.e., the crack-tip enrichment is omitted, and the crack has to cross the entire element. According to [88], the displacement fields (Eq. (12)) in the matrix form is rewritten and the field of deformations is obtained by calculating its gradient, as follows:

\[
u(x) = N(x)a + N(x)Hb
\]

Therefore, the strain field in elements can be expressed as:

\[
\varepsilon = Ba + HBb + (\delta_{\Gamma_n} n) Nb
\]

where \( N \) is the matrix of functions from conventional FEM, \( a \) is the vector of conventional nodal degrees of freedom, \( b \) is the vector of additional nodal degrees of freedom due to enrichment, \( H \) is the Heaviside function, \( B \) is the deformation approximation matrix, \( \delta_{\Gamma_n} \) is the Dirac delta function centered on the crack, and \( n \) is the matrix of the terms of the vector normal to the crack.

Furthermore, consider the equation of the virtual works without body forces:

\[
\int_\Omega \nabla^s v : \sigma d\Omega = \int_{\Gamma_N} v \cdot \bar{t} d\Gamma
\]
where \( v \) is admissible displacement variation, \( \sigma \) is the stress field and \( \bar{t} \) is external traction force, applied on the Neumann boundary \( \Gamma_N \), as it was shown in Fig. 1. Inserting the Equations (14) and (15) into the virtual work, Eq. (16), and using the variations of degrees of freedom \((a', b')\) and the integration property of the Dirac delta, results in:

\[
\alpha^T \int_{\Omega} B^T \sigma \, d\Omega + b'^T \int_{\Omega} H B^T \sigma \, d\Omega + b'^T \int_{\Gamma} N t \, d\Gamma = a'^T \int_{\Gamma_N} N t \, d\Gamma + b'^T \int_\Gamma H N \bar{t} \, d\Gamma
\]

in which, \( \Gamma_c \) is the crack surface and \( t = (\sigma \cdot n)_{\Gamma_c} \) is the cohesive traction applied on the crack. Using the constitutive relations in the continuum and discontinuous domain, one can obtain:

\[
\bar{\sigma} = D \bar{\varepsilon} = D(B \bar{a} + H \bar{b})
\]

where \( D \) relates the instantaneous stress and strain rates. Similarly, the traction rate at the discontinuity can be expressed in terms of the enhanced nodal velocities as:

\[
\bar{t} = T(\bar{u}) = TN \bar{b}
\]

where \( T \) relates the instantaneous traction and crack displacement rates. Substitution of the stress and traction rates into the discretized form of the virtual work Eq. (17) leads to:

\[
\begin{bmatrix}
\int_{\Omega} B^T DB \, d\Omega & \int_{\Omega} H B^T DB \, d\Omega \\
\int_{\Omega} H B^T DB \, d\Omega & \int_{\Omega} H^2 B^T DB \, d\Omega + \int_{\Gamma} N T N \, d\Gamma
\end{bmatrix}
\begin{bmatrix}
d\alpha \\
db
\end{bmatrix}
= \begin{bmatrix}
f^a_{\text{ext}} \\
f^b_{\text{ext}}
\end{bmatrix} - \begin{bmatrix}
f^a_{\text{int}} \\
f^b_{\text{int}}
\end{bmatrix}
\]

where \( K \) is the stiffness matrix, \( da \) and \( db \) are incremental displacements, and \( f^\text{int} \) and \( f^\text{ext} \) are the internally and externally applied forces, respectively.

### 5.2. Criteria for Mixed-Mode Crack Propagation

In crack propagation problems, there are two main requirements at each time step: crack propagation status and propagation direction. The crack propagation criteria may be a function of the stress intensity factors (SIFs), the strain energy density, and so on. The direction of the crack can be determined based on the singular term solutions of stress at the crack tip can be used to determine the crack propagation angle, where the shear stress becomes zero. Assuming the mixed-mode loading conditions, the asymptotic crack-tip circumferential stress can be defined in polar coordinate system as [66, 49]:

\[
\sigma_r = \frac{1}{\sqrt{2\pi r}} \cos \theta \left\{ K_{I} \left[ 1 + \sin^2 \frac{\theta}{2} \right] + \frac{3}{2} K_{II} \sin \theta - 2 K_{I} \tan \frac{\theta}{2} \right\}
\]

\[
\sigma_\theta = \frac{1}{\sqrt{2\pi r}} \cos \frac{\theta}{2} \left\{ K_{I} \cos^2 \frac{\theta}{2} - \frac{3}{2} K_{II} \sin \theta \right\}
\]

\[
\tau_{r\theta} = \frac{1}{\sqrt{2\pi r}} \cos \frac{\theta}{2} \left\{ K_{I} \sin \theta + K_{II} \left[ 3 \cos \theta - 1 \right] \right\}
\]

where \( K_{I} \) and \( K_{II} \) are stress intensity factors of mode-I and mode-II fracture, respectively, and \( r \) and \( \theta \) are polar coordinate of a point with respect to the crack tip point, as previously shown in Fig. 3. The crack is represented in this work as a set of straight line segments that are connected to each other. It is necessary to compute the critical crack propagation angle, \( \theta_c \), and increment length, \( \Delta \alpha \), for the new propagation step, see Fig. 4. The critical angle can be determined by setting the shear stress \( \tau_{r\theta} \) to zero which leads to:
\[
\frac{\partial \tau_{r\theta}}{\partial \theta} = \frac{1}{\sqrt{2\pi r}} \cos \frac{\theta}{2} \left\{ K_I \sin \theta + K_{II} [3 \cos \theta - 1] \right\} = 0
\] (22)

Solving Eq. (22) gives the \( \theta_c \) as follows:

\[
\theta_c = 2 \arctan \left( \frac{K_I}{K_{II}} \pm \sqrt{\left( \frac{K_I}{K_{II}} \right)^2 + 8} \right)
\] (23)

Figure 4: Representation of the crack extension and its new orientation, \( \theta_c \).

The result that gives the sign as opposite to sign of \( K_{II} \) is the correct one. Using Eq. (22) in mode-I loading (\( K_{II} = 0 \)), the crack propagation angle is zero. In mode-II loading, by solving the equation \( K_{II} [3 \cos \theta - 1] = 0 \), the crack propagation angle is \( \pm 70.5^\circ \). So the maximum range of the crack propagation angle under linear elastic fracture mechanics (LEFM) approach is limited to an angle range of \([-70.5^\circ \) to \( 70.5^\circ \]). If \( K_{II} > 0 \), the crack growth direction \( \theta_c < 0 \), and if \( K_{II} < 0 \), the crack growth direction \( \theta_c > 0 \). An efficient expression of the critical angle of crack propagation can also be given as:

\[
\theta_c = 2 \arctan \left( \frac{-2K_{II}/K_I}{1 + \sqrt{1 + 8(K_{II}/K_I)^2}} \right)
\] (24)

6. Object-oriented Implementations

This section illustrates the implementation aspects regarding to theoretical procedures and models explained in the previous sections. The whole implementations are done in an open source computational platform, so-called INSANE (INteractive Structural ANalysis Environment) environment [38], which is implemented in Java, a powerful OOP language. Similar to all FEM codes, it is composed of three parts: pre-processor, processor (numerical core) and post-processor. The numerical core is composed by the interfaces Assembler and Persistence and the abstract classes Solution and Model, aiming to analyze different problems, from linear to nonlinear problems using different numerical approaches (FEM [38], BEM [71], classical, stable, and two-scale G/XFEM [3, 57, 58, 59, 60], Meshless method [40, 42], and Hp-cloud method [73]). Figure 5 shows the unified modeling language (UML) diagram of the INSANE numerical core.

Different parts of the INSANE numerical core are written and linked to each other to solve the following generic representation of an initial or boundary value problem:

\[
C \ X = D
\] (25)

where \( X \) is the solution vector; \( C \) is a matrix with the properties of the problem and \( D \) is a vector that represents the system excitation. Following subsections present in detail the new implementations and also modifications of the existing classes corresponding to the crack growth process.

6.1. Persistence Interface

Interface Persistence interacts with the the input and output data. The persistence of data is performed by structured data files written in eXtensible Markup Language (XML) format. Appendix A presents the main part of the Persistence
interface regarding to current implementations and also different parts of the XML input file for crack propagation procedure with the G/XFEM.

6.2. Model Abstract Class

The Model abstract class contains the data of the discrete model and provides information for the Assembler to assemble the final matrix system (25). For current work, several classes are implemented or modified under this abstract class aiming to provide required information for the process. These classes are explained in the following subsections.

6.2.1. AnalysisModel Abstract Class

Analysis model is defined by different classes derived from the AnalysisModel abstract class, see Fig. 6. There are various models already implemented within INSANE code: space frame, plane stress/strain, plate and solid. Therefore, when the Assembler starts to assemble the system of equations (25), it asks the instance of AnalysisModel class for characteristics about the model, such as how the deformation matrix should be calculated. Beside this, the GFemAnalysisModel interface is created to have additional information for G/XFEM analysis. As can be seen in Fig. 6, all three classes of GFemPlaneStress, GFemPlaneStrain and GFemReissnerMindlinPlate (which is based on Reissner-Mindlin formulation presented in section 3) are derived from the AnalysisModel abstract class and simultaneously implements the methods from the GFemAnalysisModel interface.
6.2.2. EnrichmentType Package

The *EnrichmentType* package provides required information for the enrichment strategy of the fracture analysis approach. Figure 7 shows different classes from this package. As it can be seen from this figure, there are various enrichment types that can be used either for classical elasticity problems or linear elastic fracture mechanic problems (LEFM).

Furthermore, other classes such as the *StableCrackEnrichment* are available to facilitate the use of the stable G/XFEM approach [57, 59], providing a better convergence rate as well as an improved conditioning system of equations. Figure 8 presents the *discontinuousEnrichment* class that contains the necessary information for the Heaviside function calculation, see section 5.1.2. This function will be used along with *NearTipEnrichment* (Eq. (4)) and/or *CrackEnrichment* (Eqs. (2) and (3)) (see Fig. 7) to facilitate the crack propagation approach.

6.2.3. Integration Scheme

The numerical integration is implemented in INSANE code through the *IntegrationPoint* class, in which it is designed an integration point of an element and its geometrical and physical properties. This class represents a degeneration of the geometry, in which this degeneration can be represented the Gauss points or sub-cells, depending on the problem type. In the present implementation, the *IntegrationPoint* is associated to the entity Gauss quadrature point. The Gauss quadrature is not produce accurate results if the enrichment function is singular or discontinuous. However, higher order Gauss quadrature is used for this work which showed to provide very satisfactory results. Further information on the numerical integration implemented in INSANE can be found in [3].

6.3. Assembler Interface

The task of the *Assembler* interface is to mount the linear equation system from Eq. (25), returning the stiffness matrix and its partitions related to free/restrained degrees of freedom. In order to better understand the relationship between the classes described before, the process of assembling the stiffness matrix of the problem will be described in this section, as shown in Fig. 9, as the main part of the *Assembler* interface. An instance of the *Solution* class presented as “Actor”, asks for one of its attributes, an *Assembler*, i.e., *GFemAssembler*, to perform the assembly of the stiffness matrix $C$ (from Eq. (25)). The instance of the *Assembler* class, in turn, performs a loop in the list of elements that is stored in the class *GFemModel*, an attribute of the *Assembler*.

Different FEM formulations originally provided within the INSANE environment are implemented by different classes derived from *SolidMech* abstract class (inherited from the *ProblemDriver* abstract class), each one with a specific way to
calculate the different parts of the Eq. (25). Some of these classes are: FEM parametric (class \textit{Parametric}), G/XFEM parametric (class \textit{GFemParametric}), and frame (class \textit{Frame}), as it can be seen in Fig. 10. The \textit{GFemParametric} class is also designed to carry out the necessary tasks related to the G/XFEM analysis, along with its inheritance the \textit{GFemPhysicallyNonLinear} class to facilitate the nonlinear analysis. Depending on the strategy used to solve the nonlinear analysis via Newton-Raphson method, the stiffness matrix can be elastic (\textit{getC()} method), secant (\textit{getTotalC()} method) and tangent (\textit{getIncrementalC()} method). Therefore, in the sequence of stiffness matrix mounting, \textit{Element} calls one of its attributes, an instance of the \textit{GFemParametric} class which is responsible for constructing the element’s contribution to the stiffness matrix. The \textit{GFemParametric} queries \textit{Element} to obtain certain information that will be used in the construction of the stiffness matrix of the element. The first required information is the type of the analysis model that is provided here by an instance of the \textit{GFemAnalysisModel} class, either \textit{GFemPlaneStress/GFemPlaneStrain} or \textit{GFemReissnerMindlinPlate} types.

The \textit{Frame}, and \textit{Parametric/GFemParametric} classes are derived from the \textit{SolidMech} class since they use different approaches to mount the stiffness matrix. \textit{Parametric/GFemParametric} use a numerical integration approach, \textit{Frame} uses analytical formulation to mount the stiffness matrix, with respect to their corresponding formulations provided by the \textit{Element} class (the types of shape functions, for example). The \textit{Element/GFemElement} class has an instance of some of the classes derived from \textit{AnalysisModel/GFemAnalysisModel} class that provides the mathematical formulation of the analysis.
model (see section 6.2.1). In fact, plane and plate cases can use the same approach to mount the stiffness matrix and only their degrees of freedom and procedure to calculate the required derivatives matrices are different. Since, this work is dealing only with plane stress and Reissner-Mindlin problems, the formulation of the Frame wasn’t presented here.

Afterwards, the instance of class Element queries another attribute, an instance of Degeneration class (see Fig. 9). This instance has stored the section and material properties and coordinates of the integration points. This list of integration points is used for a loop that runs through each integration point of the current element in order to calculate the portion of the stiffness matrix of the element at each integration point. At each step of this loop, the derivatives of the shape function and its enriched part must be evaluated. This is done by an instance of the EnrichedShape class, which is also an Element attribute. The derivatives of the shape function depends on PU from underlying FE mesh and enrichment types from G/XFEM approach, either continuous or discontinuous functions. The EnrichedShape instance manages this dependency between these two parts.

For each node, the EnrichedShape instance is evaluated at the corresponding integration point. The PU is given by an instance of Shape class which is a member of the EnrichedShape. Enrichment functions are obtained from a list of instances of EnrichedType class, which is the attribute of the Node. This list is stored in GFemModel and is accessed through a list of associated instances for the corresponding node. The original and derived shape functions, computed by the EnrichedShape, after a loop through the nodes of the element, are sent to the GFemParametric instance which is responsible for building the array of the element stiffness. GFemParametric sends this information to the instance of GFemAnalysisModel class, which provides the matrix of derivatives, integration factors, and the Jacobian for a specific analysis model. Finally, the instance of Degeneration class is queried to provide the numerical integration weights for a particular point of the integration. Thus, the GFemParametric instance can return the stiffness matrix of the element, which uses this portion to form the stiffness matrix $C$, from Eq. (25), of the problem.

6.4. Solution Abstract Class

Solution abstract class starts the solution process and has the necessary resources for solving the matrix system of the fracture analysis approach. As it can be seen in Fig. 11, it shows different classes derived from class Solution to handle either static (SteadyState, GlobalLocal, ThermoStructural, StaticEquilibriumPath), dynamic (DynamicEquilibriumPath), or modal analysis (ModalVibration). The StaticEquilibriumPath class is responsible for the nonlinear analysis and it is used here to handle the quasi-static linear elastic fracture analysis based on the G/XFEM method.

![Figure 11: Solution abstract class](image)

A nonlinear FEM solution strategy was presented in Yang and Shieh [91], which is sufficiently generic that can be applied to many different control methods, by a simply redefinition of a constraint equation [71]. Figure 12 presents the nonlinear solution procedure used for current implementations. The implementation of this flowchart is shown in Code 1, lines 13-36. It starts with the assembling of the force vector (lines 13-15 from Code 1), then starting a do-while loop over the time steps (lines 17-36 from Code 1, where details for crack propagation modeling are presented in section 6.5). Afterwards, a loop over iterations must be followed (line 26 from Code 1). The iterative solver here is the well-known standard Newton-Raphson method. After that, the convergence at the end of each iteration will be checked (lines 28-29 from Code 1). If the solution is converged, then it must followed by the next time step (lines 31-34 from Code 1), by updating the corresponding part of the code and writing the output file for the current step.

The implementation of the control method process is highlighted in Fig. 12. The method execute() from class StaticEquilibriumPath (line 26 from Code 1) triggers the iterative process for each step, through running the Code 2 from StandardNewtonRaphson class. At the beginning, the displacement vector is provided by the Assembler (line 3 from...
Figure 12: Flowchart of the Nonlinear solution procedure used in this work.

Code 2). Then, the reference and residual displacement vectors are initialized (lines 6 and 7 from Code 2). Afterwards, the loading is being updated with the corresponding factor (lines 8-10 from Code 2), and solver adapter is initialized with the stiffness matrix (lines 13-14 from Code 2). Lines 11 and 12 from Code 2 are initializing the internal and residual force vectors, respectively. After that, the iterative loop will be started by setting the corresponding stiffness matrix to different equilibrium types (lines 23-31 from Code 2) that is defined by the user, see Appendix A.

```java
public void execute() throws Exception {
    int cont = 0;
    step.addObserver(this);
    step.setIterativeStrategy(iterativeStrategy[cont]);
    step.getAssembler().setXp();
    this.setCurrentStep(0);
    step.setLabel(this.getCurrentStep());
    step.update();
    step.getAssembler().update();
    this.setCurrentStep(1);
    step.getAssembler().addLoading(loadCombination);
    this.step.setProportionalLoad(this.getAssembler().setProportionalLoad());
    this.step.setConstantLoad(this.getAssembler().setConstantLoad());
    do {
        if (step.getAssembler().getCrackPropagation()) {
            this.step.setProportionalLoad(this.getAssembler().setProportionalLoad());
            this.step.setConstantLoad(this.getAssembler().setConstantLoad());
        }
        iterativeStrategy[cont].setStep(step);
        step.setLabel(this.getCurrentStep());
        step.executeUpdate();
        step.getAssembler().setLabel();
        if (!step.getConvergence()) {
            break;
        }
        step.getAssembler().getCrackPropagation();
        step.getAssembler().setIterationEnd();
        if (!step.getConvergence()) {
            break;
        }
        loadCombination = step.getAssembler().getModel().getLoadCombinationsList().get(0);
        step.getAssembler().addLoading(loadCombination);
        step.getAssembler().update();
        this.setCurrentStep(this.getCurrentStep() + 1);
    } while (this.getCurrentStep() <= this.getNumMaxSteps());
}
```

Code 1: Main code block of the StaticEquilibriumPath class
Method `getXPandXQ()` allows to evaluate the incremental values of the state variable for residual and reference parts (line 32 from Code 2). The `IterativeStrategy` is then responsible to return the predicted value of the load factor increment at the first iteration (lines 35-40 from Code 2), or its correction for the other iterations (lines 41-43 from Code 2). Various types of control method can be used here (Direct displacement control, Load control, Arc-length control, Cylindrical arc-length control, etc.), since the iterative procedure is written in such a way that only procedure for obtaining the load factor from corrector or predictor would be different. The updating of both the load factor and the nodal parameters vector is performed by the `assignStepState(...)` method (line 44 from Code 2). Subsequently, the vector of residual forces is calculated using the internal forces vector (lines 45-46 from Code 2). Finally, the convergence is checked and the load factor and displacement in the iteration, i.e. the `initialX` vector, will be updated (lines 51-52 from Code 2). The convergence at the end of each iteration will be checked through comparing the tolerance value with the error of the problem, with the aim of the method `setConvergence()`. One of the main feature of the current solution procedure is that it is highly modular and it doesn’t have any direct connection with a specific numerical method or constitutive model. This is because the only relationship with the other parts of the code is through some methods from the `Assembler` interface, such as `getCuu()`, `getIncrementalCuu()`, and `getFp()`.

6.5. Crack Propagation Modeling

Figure 13 shows the crack propagation modeling flowchart from the current implementation, in which the numbers specify the step numbers for crack propagation process at each time step. In addition, Code 1 showed the main block (`execute()` method) of the `StaticEquilibriumPath` class that is related to steps shown in Fig. 13. For nonlinear material models, the solution starts for each time step with a `do-while` loop (step 1 from Fig. 13 and line 17 from Code 1). First, it checks whether the solution is converged (step 2 from Fig. 13 and line 28 from Code 1), if so, the variables and instances will be updated (steps 3 and 4 from Fig. 13 and line 33 from Code 1).
Figure 13: Flowchart of the current fracture modeling approach.
The crack will be propagated if there is a pre-existing crack in the model, by evaluating the propagation status and its direction, then extending the crack line along this direction (steps 8 to 12 from Fig. 13). For nonlinear materials, singularity of the localization tensor will be checked, the crack can grow if the strain localization phenomenon occurs at any Gauss point. This step can be repeated for all available cracks in the model (steps 7 to 12 from Fig. 13). If the problem study is for crack nucleation procedure (i.e., there is no crack in the model to propagate), the model evaluates this situation (step 13 from Fig. 13). In this step, the model can experience the possibility of having multiple cracks. This must be defined by the user, see code 6 from Appendix A. Nucleation of more than one crack in the analysis can be allowed (step 14 from Fig. 13). As a consequence, the existing cracks propagation is first verified and then, the cracks nucleation process start. The opposite could result in a crack nucleating into an element, where the propagation of an existing crack should actually occur. After the crack nucleation process, the analysis will follow a similar process to the crack propagation described earlier (steps 21 to 25 from Fig. 13). If the crack is not able to propagate at the current time step, the loop goes for the next time step to apply a bigger loading magnitude (steps 5, 14, 15, and 22 from Fig. 13).

As discussed earlier, the \textit{LEFMcrackGrowthByGFem} and \textit{DiscontinuityByGFem} classes are the main core for the quasi-static crack propagation approach based on the G/XFEM methodology for linear and nonlinear material models, respectively, as their UML diagram is shown in Fig. 15. If the crack propagation status is \textit{true}, then the \textit{update()} method from \textit{GFemModel} class will start to analyze the problem and follow the procedure from SIF calculation (linear elastic material), by using the procedure described in Appendix B, or localization tensor (nonlinear material) and adding Heaviside and singular/near-tip enrichment functions for the corresponding nodes, explained in section 6.2.2. The nodes selection procedure here is based on the definition from section 5.1.3. The \textit{update()} method from the \textit{GFemModel} class is presented in Code 3. If the solution procedure is quasi-static, i.e., there must be some crack propagation, then following lines of code must be taken into account in the process. As an example, line 4 checks whether the problem type is nonlinear or not. If so, it will call \textit{evaluateCracksPropagation()} method, which is shown in the same block. This method will call \textit{update()} method from \textit{DiscontinuityByGFem} class which is responsible for the crack propagation process. Also, if the problem can interact with multiple cracks, beside the predefined crack or there is no predefined crack, it will call the \textit{evaluateCracksNucleation()} method to start the crack nucleation process. In the case of existence of a crack in the nonlinear material model, due to either crack nucleation or propagation of a pre-existing crack, a cohesion force will be applied over the crack lines, according to section 5.1.5.

At the first step of the crack propagation procedure, the \textit{PersistenceAsXml} class is used to fill the data, constructing an instance of the \textit{LEFMcrackGrowthByGFem} or \textit{DiscontinuityByGFem} classes and then the \textit{buildNotch()} method is called in order to create the initial crack from the user-inserted data and Heaviside function. In the case of crack nucleation modeling for nonlinear materials, the method \textit{evaluateCrackNucleation()} is responsible for call \textit{buildCrack()} method from the \textit{DiscontinuityByGFem} class to evaluate the model for possible crack nucleation.

```java
public void update() throws Exception {
    super.update();
    if (withCrackPropagation) {
        if (this.getDiscontinuityType().equals(DiscontinuityByGfem.class.getSimpleName())) {
            this.evaluateCracksPropagation();
        } else if (onlyOneCrack) {
            this.evaluateCracksNucleation();
        } else if (!onlyOneCrack) {
            this.evaluateCracksNucleation();
            this.evaluateCracksPropagation();
        } else if (this.getDiscontinuityType().equals(LEFMcrackGrowthByGfem.class.getSimpleName())) {
            this.evaluateCracksPropagationLEFM();
        } else if (true) {
            this.evaluateCracksPropagation();
        }
    } else if (true) {
        this.evaluateCracksNucleation();
    }
}
protected void evaluateCracksPropagation() {
    for (i = 0; i < discontinuities.size(); i++) {
        discontinuities.get(i).update();
    }
}
protected void evaluateCracksNucleation() {
    listIterator = elements.list().listIterator();
    while (listIterator.hasNext()) {
        Element elm = listIterator.next();
        if (!elm.isSplit()) {
            if (!elm.checksStrainLocalization()) {
                this.discontinuities.add(new DiscontinuityByGfem("D" + discontinuities.size() + 1), this, elm);
            }
        } else if (onlyOneCrack) {
            break;
        }
    }
}
```

Code 3: Code block of the \textit{update()} method from \textit{GFemModel} class

The crack propagation procedure for linear elastic materials is similar to nonlinear materials, but with some differences. Considering the Fig. 13 again, the process starts at step 1 and neglecting step 2, since there is no convergence study for
### Figure 14: Structure of the GFemModel class

<table>
<thead>
<tr>
<th>GFemModel</th>
</tr>
</thead>
<tbody>
<tr>
<td>- discontinuitiesLEFcrack: ArrayList&lt;LEFcrackGrowthByGFEM&gt;</td>
</tr>
<tr>
<td>- discontinuities: ArrayList&lt;DiscontinuityByGFEM&gt;</td>
</tr>
<tr>
<td>- withCrackPropagation: boolean</td>
</tr>
<tr>
<td>- onlyOneCrack: boolean</td>
</tr>
<tr>
<td>+ update(): void</td>
</tr>
<tr>
<td>- evaluateCrackPropagation(EMF): void</td>
</tr>
<tr>
<td>- evaluateCrackPropagation(): void</td>
</tr>
<tr>
<td>- evaluateCrackNucleation(): void</td>
</tr>
<tr>
<td>- getDiscontinuities(): ArrayList&lt;DiscontinuityByGFEM&gt;</td>
</tr>
<tr>
<td>+ getDiscontinuitiesLEFcrack(): Array&lt;DiscontinuityByGFEM&gt;</td>
</tr>
<tr>
<td>+ getDiscontinuitiesType(): String</td>
</tr>
<tr>
<td>+ isWithCrackPropagation: boolean</td>
</tr>
<tr>
<td>+ isOnlyOneCrack: boolean</td>
</tr>
<tr>
<td>+ setDiscontinuities(): void</td>
</tr>
<tr>
<td>+ setDiscontinuitiesLEFcrack(): void</td>
</tr>
<tr>
<td>+ setDiscontinuitiesType(): void</td>
</tr>
<tr>
<td>+ setWithCrackPropagation(withCrackPropagation: boolean): void</td>
</tr>
<tr>
<td>+ setOnlyOneCrack(withOnlyOneCrack: boolean): void</td>
</tr>
</tbody>
</table>

### Figure 15: UML diagram of the LEFcrackGrowthByGFem and DiscontinuityByGFem classes together

```
LEFcrackGrowthByGFem/DiscontinuityByGFem

- label: String
- firstPoint: Point3d
- lastPoint: Point3d
- firstElement: Element
- lastElement: Element

DiscreteConstitutiveModel: CrackingConstitutiveModel // For DiscontinuityByGFem class only

- enrichmentDiscontinuity: DiscontinuousEnrichment
- enrichmentCrack: CrackEnrichmentModel
- enrichmentNodeBySingularFunc: Array<List<Element>>
- model: Model
  - enrichmentIntegral: computeInteractionIntegral // For LEFcrackGrowthByGFem class only
  - fractureMode: int
  - integralRadiusMultiplier: double
  - crackIncrement: double

+ getLabel(): String
+ setLabel(label: String): void
+ getFirstPoint(): Point3d
+ setFirstPoint(firstPoint: Point3d): void
+ getLastPoint(): Point3d
+ setLastPoint(lastPoint: Point3d): void
+ getEnrichment(): DiscontinuousEnrichment
+ setEnrichment(enrichment: DiscontinuousEnrichment): void
+ getCrackEnrichment(): CrackEnrichmentModel
+ setCrackEnrichment(crackEnrichment: CrackEnrichmentModel): void
+ getRadiusMultiplier(): double
+ setRadiusMultiplier(radiusMultiplier: double): void
+ getCrackIncrement(): double
+ setCrackIncrement(crackIncrement: double): void
```

---

**GFemModel**

- DiscontinuousEnrichment

**NearTipEnrichment**

**CrackEnrichment**

**computeInteractionIntegral**

**Integration**

**ComptationalGeometry**
linear materials. Then, it follows the step 3 to 12 (only for crack propagation), with two exceptions which the loop will be directly out to the step 1, and step 11 will be also excluded from the analysis procedure, since there is no traction defined for linear materials. The stress intensity factors are calculated at step 9 in order to evaluate whether the crack tip(s) can grow or not. In the case of crack propagation state, obtaining the crack propagation orientation using Eq. (23). Having the crack orientation and also the crack extension increment, the new crack tip(s) can be easily calculated at step 10. So, the model can be updated with new crack tip(s) and solution process can be followed until reaching the maximum time step defined by the user.

7. Numerical Applications

This section presents four problems in \( \mathbb{R}^2 \). Section 7.1 presents two linear-elastic problems and section 7.2 presents two nonlinear material model problems. The number of integration points that are used for these problem were selected quite big enough to accurately capture the crack propagation direction within the element boundary, even for elements containing singular enrichment functions. For simplicity, the same number of integration points is used for the all elements with and without singular/Heaviside enrichment functions.

As mentioned earlier, the implemented strategy here is applicable to the analysis of linear and nonlinear problems. In nonlinear problems, the distributed degradation history variables are calculated and stored at the integration points. For this reason, and in order to continue evaluating the possible distributed degradation around the crack, the integration process (or the position/distribution of the integration points) is not modified when the crack cuts the element, i.e., the conventional Gauss integration is used, except for the introduction of integration points along crack segment to integrate the parts of stiffness and of internal forces vector related to cohesion (see section 5.1.5), specifically for nonlinear material models where a cohesion force is applied over the crack lines.

7.1. Linear Material Model Problem

This section presents two linear-elastic problems in \( \mathbb{R}^2 \). Section 7.1.1 presents a plate, under state of plane stress, with an inclined crack under tension loading and a Reissner-Mindlin plate with a crack is analyzed in section 7.1.2. The geometry and boundary conditions are very simple and the goal of choosing them is to demonstrate the capabilities of the G/XFEM implementation for quasi-static crack propagation process. All problems have the following parameters (in consistent units): modulus of elasticity \( E = 1.0 \), Poisson ratio \( \nu = 0.3 \), and the tension/shear stress \( \sigma = 1.0 \). The number of integration points for all two problems is considered equal to \( 8 \times 8 \). The domain size of the interaction integral used to calculate the SIF, according to Appendix B, is considered here by a circle with radius \( r \) defined by \( r = r_m h_{elem} \), in which the element characteristic length, \( h_{elem} \), is the square root of the crack tip element area and \( r_m \) is a scalar multiplier [67]. To have an accurate SIF results, one have to select a proper multiplier \( r_m \). This scalar multiplier can be chosen by performing numerical experiments with different values to have an independent J-integral path. The crack increment length, \( \Delta \alpha \), should be chosen in such a way to have a reasonable and stable crack propagation procedure. According to [43], an appropriate value must be chosen according to the type of crack propagation, i.e., straight or curved crack, and mesh size to have a reliable crack propagation path. Small values could help to obtain a better accuracy, however, if \( \Delta \alpha \) is too small with respect to the element size, multiple changes in the direction of the crack path may occur. The scalar multiplier for following two problems is considered equal to 2.0, but the crack increment length has different values for each problem. The crack propagation criteria and the propagation direction are defined based on the stress field around the crack tip, as it was explained in section 5.2. In addition, the Heaviside (Eq. (10)) and singular enrichment (Eqs. (2) and (3)) functions are used to model the crack lines and capture the crack-tip singularities within the element boundaries.

7.1.1. Inclined Crack Under Tension

This section presents the results for a problem with an inclined crack, as shown in Fig. 16. The objective of this problem is to illustrate the mixed-mode crack propagation using G/XFEM method. The problem is analyzed under plane stress state. The geometrical parameters of this problem are: \( W = 3.0 \), \( 2a = 0.35 \), and \( \beta = 48.5^\circ \).

The element size is equal to 0.25 for this problem with total of 144 elements, a uniform mesh of \( 12 \times 12 \) elements. The crack increment length considered here is equal to 0.19. Displacement distributions in \( y \) direction along with crack propagation path are shown in Fig. 17. There are some small fluctuations in the crack propagation path, but this figure clearly shows the mixed-mode propagation path, in accordance with [8].
Figure 16: Geometry and loading of the problem with an inclined crack.

Figure 17: Contour of displacement in $y$ direction for inclined crack problem, at different stage of the crack propagation process.
7.1.2. A Reissner-Mindlin Plate with Bending Moment

An infinite plate subjected to a far-field moment, \( M \), is shown in Fig. 18 to have a purely mode-I loading. The aim of this example is to illustrate the crack propagation using G/XFEM method for Reissner-Mindlin plate problems. These are the plate parameters: the crack length is taken to be \( 2a = 1.2 \), the plate width, \( W \), is taken to be equal to 6.0, and thickness \( t = 1.0 \). Thanks to the symmetry about the \( y \) axis, only one-half of the plate is modeled with the finite elements.

![Figure 18: Schematic of geometry and loading for Reissner-Mindlin plate under bending.](image)

Numerical SIFs from current G/XFEM are compared with the results from ref. [30] for the Reissner-Mindlin plate. This comparison for various thickness over semi-crack length, \( t/a \), is shown in Fig. 19. Maximum error of the \( K_I \) for various values of \( t/b \) are less than 8%.

![Figure 19: Numerical SIFs from current G/XFEM and from ref. [30] for single-edge cracked plate, with the values of: \( E = 200 \text{ GPa}, \nu = 0.3 \), \( W = 10 \), and \( a = 0.5 \).](image)

The element size used is equal to 0.25 and the model consists of 144 elements, a uniform mesh of \( 12 \times 12 \) elements. The crack increment length considered here is equal to 0.125. Figure 20 shows the rotation distributions in \( y \) direction along with crack propagation path. As it can be seen from this figure, the crack propagation path is along with mode-I propagation, since the loading is pure mode-I.
7.2. Nonlinear Material Model Problem

This section presents two nonlinear problems in two-dimensional domain to show both nucleation and propagation of the crack for different problems. Section 7.2.1 presents simulation of a L-shaped panel experimentally analyzed by Winkler et al [89] and section 7.2.2 presents a four point shear test simulation, experimentally analyzed by Arrea and Ingraffea [7]. The localization tensor, which provides the classical strain localization condition, is used to determine the crack propagation direction for nonlinear material models presented in this section. In the case of nonlinear material models, only Heaviside enrichment function (Eq. (10)) is used to model the crack lines within the element boundaries.

Several constitutive models based on elastic degradation are available in the INSANE computational platform, such as: Mazars Isotropic Damage Model [62]; Mazars and Lemaitre Isotropic Damage Model [63]; Simo and Ju Isotropic Damage Model [79]; Ju Isotropic Damage Model [47]; Lemaitre and Chaboche Isotropic Damage Model [53]; de Vree Isotropic Damage Model [29]; Smeared Cracking Models, with Fixed or Rotating Direction; Smeared Damage Model by de Borst and Gutierrez [28]; and Volumetric Damage Model [72].

7.2.1. L-shaped Panel

This example simulates a L-shaped panel shown in Fig. 21, experimentally analyzed by Winkler et al [89]. This example is mainly designed for modeling the crack nucleation and also propagation.

The constitutive model of volumetric damage by Penna [72] was adopted in the modeling to evaluate the initial degradation in distributed manner and indicate the introduction of discrete failure. The material parameters are: modulus of elasticity $E_0 = 25850.0 \text{ N/mm}^2$ and Poisson ratio $\nu = 0.18$. To follow the damage evolution, polynomial laws defined in
were adopted for tension and compression. The evolution law for tension damage is described by the following parameters: equivalent strength limit \( f_e = 1.43 \, \text{N/mm}^2 \), value of the equivalent strain from which the damage process starts \( \kappa_0 = 0.000215 \) and equivalent modulus of elasticity \( \tilde{E} = 13463.0 \, \text{N/mm}^2 \), whereas, for compression: \( f_e = 16.0 \, \text{N/mm}^2 \), \( \kappa_0 = 0.0022 \) and \( \tilde{E} = 13463.0 \, \text{N/mm}^2 \). The panel was modeled with 636 triangular elements of three nodes \( T3 \) in state of plane stress.

The direct displacement control method was used in the solution of the model, increasing the maximum vertical displacement by 0.01 \( \text{mm} \), with tolerance for convergence in forces of \( 1 \times 10^{-3} \) and reference load \( q = 28.0 \, \text{N/mm} \). Figures 22 and 23 show, respectively, the deformed with scale factor equal to 100 and the evolution pattern of tension damage along the domain to the steps 9 (19.6 \( \text{N/mm} \)), 10 (21.3 \( \text{N/mm} \)), 25 (14.3 \( \text{N/mm} \)) and 200 (3.4 \( \text{N/mm} \)). As it can be seen from this figure, there is no crack at step 9, while the crack starts to nucleate and propagate at larger loading magnitude at step 10.

![Figures 22 and 23: L-shaped panel: Deformed shape](image)

Figures 22 and 23 also highlight the jump in the displacements field caused by the presence of the crack, as well as the evolution of the crack along the region with the greater damage. The observed damage pattern is similar to that obtained by Winkler et al [89], as shown in Fig. 24.
Figure 23: L-shaped panel: Tension damage evolution

Figure 24: Damage pattern observed by [89]
The equilibrium path of the maximum vertical displacement is shown in the Fig. 25, compared to the experimental and numerical results of Winkler et al [89].

![Equilibrium path of the maximum vertical displacement](image)

The behavior obtained is similar to the one observed in the experiments, however, a more fragile response is observed in the displacements range of 0.15 mm to 0.4 mm, which may be justified by an estimate of cohesion below the toughening mechanisms. Regarding the numerical results of [89], the equilibrium path is similar to that of numerical model 2, with respect to obtaining the limit load and the post-critical regime until the displacement equal to 0.25 mm. From this displacement, the behavior obtained becomes more ductile than numerical model 2, similar to the numerical model 1.

7.2.2. Four Point Shear Beam

This example simulates a four point shear test shown in Fig. 26, experimentally analyzed by Arrea and Ingraffea [7]. The constitutive model of fixed smeared crack with Carreira and Chu law [23] for compression and Boone et al. law [18] for tension was adopted in the modeling to evaluate the initial degradation. The material parameters are: modulus of elasticity \(E_0 = 24800 \text{ N/mm}^2\), Poisson ratio \(\nu = 0.18\), compressive strength \(f_c = 34.0 \text{ N/mm}^2\), tensile strength \(f_t = 3.4 \text{ N/mm}^2\), strain relative to the elastic limit in compression \(\varepsilon_c = 0.002\), fracture energy \(G_f = 0.120 \text{ N/mm}\), characteristic length \(h = 40 \text{ mm}\) and shear retention factor \(\beta_r = 0.02\). The beam was modeled with 432 triangular elements of three nodes T3 in plane stress.

![Geometry and loading of the four point shear test](image)

The preexisting crack was modeled as a notch, by means of the enriching the corresponding nodes near the crack line in order to simulate a jump in the displacements field. The direct displacement control method was used in the solution of
the model, increasing the degree of freedom added to one of the nodes of the edge containing the “mouth” of the notch by 0.00065 mm. This degree of freedom refers to a portion of the relative tangential sliding of this “mouth”, i.e., a portion of the Crack Mouth Sliding Displacement (CMSD) was controlled (Fig. 26), with tolerance for convergence in forces of $1 \times 10^{-3}$ and reference load $P = 130.0 \text{kN}$.

Figures 27 and 28 show the deformed with scale factor equal to 500 and the shear stress $\tau_{xy}$ along the domain, respectively, for time steps of 36 ($P = 143.5 \text{kN}$) and 200 ($P = 42.0 \text{kN}$). The capability of the numerical model to predict the crack path is highlighted in Fig. 27, as well as the simulation of the jump in the displacements field. The equilibrium path of the CMSD is shown in the Fig. 29, compared to the experimental results of Arrea and Ingraffea [7] and numerical results of Fang et al [37], which in turn analyzed the beam with a constitutive model of cohesive crack that adopts bilinear and exponential laws in a model of extended finite elements.
It is observed that the model was able to represent the experimental results, describing adequately the evolution of the CMSD. The behavior obtained also resembles the numerical results of Fang et al [37].

8. Concluding Remarks

This work presented an object-oriented implementation of two-dimensional crack nucleation and propagation for linear and nonlinear material models into an in-house code called INSANE. The capabilities of the generalized/extended finite element method were used to simulate the discontinuity propagation within the models. Different enrichment types, from Heaviside to singular enrichment functions were used to model the discontinuities and also capture the crack-tip singularities to obtain a better and more accurate crack propagation path. The stress intensity factor and singularity of the localization tensor are used to obtain the propagation path for linear and nonlinear materials, respectively. The OOP aspects were discussed in detail by providing different UML diagrams from abstract classes to interfaces levels as well as different classes interactions aiming to show the whole implementation with regards to these work. Also, various blocks of codes are brought in order to show a part of implementations in a practical way. These blocks of codes were accompanied with some parts of the XML input file to demonstrate how the user input file interacts and defines the solution procedure, from solution type to nodes and elements, and also the discontinuity definition.

The validation of these implementations were presented by different numerical examples for solid mechanics, from plane stress problems to a Reissner-Mindlin problem for both linear and nonlinear material models, aiming to cover all aspects and features of the current implementations. Crack nucleation (for nonlinear materials) and propagation (for both linear and nonlinear material models) were modeled. The numerical results presented here were compared with the reference solutions from the analytical, numerical or experimental results, where applicable. These results clearly show the capability of the current G/XFEM implementations to overcome almost all kinds of two-dimensional crack propagation problems.

Currently, the effects of singular and jump enrichment functions in the accuracy and conditioning of the G/XFEM method are important issues that have been tackled in some recent works. One alternative is the stable strategy of the G/XFEM [9, 41]. This stable G/XFEM strategy is already implemented in the INSANE platform and some results have been shown in [57]. Other techniques such as the shifting type enrichment function proposed by Agathos et al [1] or the use of global functions to weight the enrichment functions presented in [2], could also be pursued, require adding new types of enrichment strategies to the current G/XFEM method in INSANE. Error estimators and refining adaptive procedures are
also interesting topics to be discussed under the OOP approach. Several works have been proposed to deal with the error estimator issue under the G/XFEM approach applied to fracture mechanics problems, such as [15, 44, 55, 74]. In addition, extending current implementations to solve 3D problems is another potential topic, in which various available researches can be used, such as [76, 84, 85, 86]. The strain smoothing technique proposed by [25] can be an efficient approach to simplify the cumbersome numerical integration of the Heaviside and singular enriched functions. The main idea behind this technique is using the strain smoothing to transform the domain integration into boundary integration. This technique has been extended to the PU methods, using three methods: smoothed extended finite element method [21], edged-based smoothed extended finite element method [26] and node-based smoothed extended finite element method [87]. Therefore, the strain smoothing using the aforementioned references can be another promising development topic for the INSANE code. These aforementioned research themes are among the future works of our research group, aiming to empower the INSANE computational platform to solve wider range of the solid mechanics problems. Finally, a data set along with input files of this work can be found at https://figshare.com/s/6458f016f6dc521ff3c2 [61], in order to reproduce the results presented here and also use for other stuff.

Acknowledgments

This research is funded by the Brazilian research agencies: CNPq (National Council for Scientific and Technological Developments - Grants No. 308785/2014-2 and 308932/2016-1), FAPEMIG (Research Support Foundation of the State of Minas Gerais - Grant No. PPM-00669-15), and CAPES (Coordination for the Improvement of Higher Education Personnel).

Appendix A. XML Input File

This section presents a detail explanation on corresponding methods from PersistenceAsXml class to perform the current G/XFEM implementations. It also presents different parts of the XML input file for both linear and non-linear analysis. Figure A.30 shows the UML diagram of PersistenceAsXml class in which presents those methods that are modified or created in order to facilitate the crack growth procedure.

The method fillDiscontinuityByGFEMListFromFile and fillIsWithCrackPropagationFromFile are added to the PersistenceAsXml class aiming to provide initial information for crack propagation procedure, such as fracture modes and radius of the integral domain for SIF calculation, see Appendix A. In addition, fillIsWithCrackPropagationFromFile looks into the input file if the crack is stationary or can be propagated during the analysis (see Code 4). It also checks whether the model can interact with single or multiple cracks. This code looks into the corresponding part of the XML file and informs the discontinuity type, whether it can be static or quasi-static crack. Also, in the case of crack propagation procedure, it informs if the problem is dealing with single or multiple cracks. This is particularly useful when the problem study is about both crack initiation and propagation phases.
// Read "OnlyOneCrack ..."
OMElement xmlElement = isWithCrackPropagationElement.getFirstChildWithName(new QName(ns, "OnlyOneCrack"));
if (!(xmlElement == null)) {
    onlyOneCrack = xmlElement.getText().trim();
    ((GFemModel) model).setOnlyOneCrack(Boolean.parseBoolean(onlyOneCrack));
}

// Read "crackPropagationType ...
xmlElement = isWithCrackPropagationElement.getFirstChildWithName(new QName(ns, "crackPropagationType"));
if (!(xmlElement == null)) {
    crackPropagationType = xmlElement.getText().trim();
    ((GFemModel) model).setDiscontinuityType(crackPropagationType);
}

Code 4: Code block of the fillIsWithCrackPropagationFromFile method from PersistenceAsXml class

Code 5 shows the input data for Solution type definition. It needs to define total number of steps, required parameters for iterative approach such as: maximum number of iterations; tolerance; convergence type (Force:1, Displacement:2, Both:3); and equilibrium type (1, 2, 3), and iterative strategy (such as displacement control and cylindrical arc-length control).

Code 5: XML input file for Solution definition

Code 6 has required tags to define the discontinuity type, crack line, and other parameters for the static and quasi-static analysis based on G/XFEM method. This code defines the crack geometry by setting two points, representing a straight line. Lines 5-6 and 11-13 are parameters related to the singular (crack) or near-tip enrichment functions. Lines 15-18 fill some parameters specifically for LEFM crack propagation, such as: fractureModes to define mode(s) of fracture (1, 2, 3, 4, 5, and 6 for mode-I, mode-II, mode-III, mode-I and II, mode-I and III, and mode-II and III, respectively), InteractIntegRadiusMultiplier to define the interaction integral multiplier, i.e. \( r_m \), SingleOrDoubleCrackTip to define whether both crack-tips are inside of the model boundaries or not, crackIncrement which defines the \( \delta a \), from section 5.2. The tag IsWithCrackPropagation specifies if the problem is a static (‘false’) analysis or quasi-static (‘true’). Finally, the OnlyOneCrack forces the problem to be analyzed with a single or multiple cracks. This belongs only for nonlinear material analysis procedure. The crack nucleation works by putting only lines 21-24 in the input file, and activation the crack propagation process by setting the IsWithCrackPropagation status to ‘true’.

Code 6: XML input file for Discontinuity definition
Code 6: XML input file for Discontinuity definition

The main part to specify the linear or nonlinear materials is in the element definition, as shown in the Code 7. Incidence defines the nodal incidence of the element and AnalysisModel specifies the problem type, from plane stress (GFemPlaneStress) and plane strain (GFemPlaneStrain) to Reissner-Mindlin plate (GFemReissnerMindlinPlate), for G/XFEM analysis. Gaussian quadrature order is defined with IntegrationOrder tag. ConstitutiveModel defines if the material mode is linear (LinearElasticConstModel) or nonlinear (MLFOCM_SCM_FD) and ElmDegenerations sets the element section properties.

Code 7: XML input file for Element definition

Appendix B. Stress Intensity Factor Calculation

The interaction energy integral method is adapted here to calculate the stress intensity factor, and hence the crack propagation orientation. A summary on plane stress problem will be discussed here, while the reader can find detailed formulation for plane stress and Reissner-Mindlin plate problems in [67, 30]. The J-integral contour for a plane stress problem is defined as [77]:

\[
J = \oint_{\Gamma} \left[ W \delta_{ij} - \sigma_{ij} \frac{\partial u_{i}}{\partial x_{1}} \right] n_{j} d\Gamma \quad (B.1)
\]

where \(W\) is the strain energy density defined as \(W = \frac{1}{2} \sigma_{ij} \epsilon_{ij}\), \(\sigma_{ij}\) and \(\epsilon_{ij}\) are the stress and strain tensor, respectively, \(u_{i}\) is the displacement field, \(n_{j}\) is the unit outward normal vector to the contour integral \(\Gamma\) that contains the crack, and \(\delta\) is the Kronecker delta.

Following [16, 69], assume two states of a cracked body: State (1) \((\sigma_{ij}^{(1)}, \epsilon_{ij}^{(1)}, u_{i}^{(1)})\) represents the current state and State (2) \((\sigma_{ij}^{(2)}, \epsilon_{ij}^{(2)}, u_{i}^{(2)})\) is an auxiliary state which will be chosen as the asymptotic fields for modes I and II. The J-integral for the sum of these two states is:

\[
J^{(1+2)} = J^{(1)} + J^{(2)} + I^{(1,2)} \quad (B.2)
\]

where \(J^{(1)}\) and \(J^{(2)}\) are obtained from Eq. (B.1) and the states (1) and (2), respectively. \(I^{(1,2)}\) is called the interaction integral for states (1) and (2) and defined as:

\[
I^{(1,2)} = \oint_{\Gamma} \left[ W^{(1,2)} \delta_{ij} - \sigma_{ij}^{(1)} \frac{\partial u_{i}^{(2)}}{\partial x_{1}} - \sigma_{ij}^{(2)} \frac{\partial u_{i}^{(1)}}{\partial x_{1}} \right] n_{j} d\Gamma \quad (B.3)
\]

in which \(W^{(1,2)}\) is the interaction strain energy.

For general mixed mode problems, the following relationship between the value of the J-integral and the stress intensity factors can be generally defined as \(J = \frac{1}{E'} (K_{I}^{2} + K_{II}^{2})\), in which \(E' = E\) for plane stress and \(E' = E/(1-\nu^{2})\) for plane strain problems. This J-integral definition can be rewritten for the combined states (1) and (2), using \(K_{I} = K_{I}^{(1)} + K_{I}^{(2)}\) and \(K_{II} = K_{II}^{(1)} + K_{II}^{(2)}\), and hence the interaction energy integral can be extracted as:
\[ I^{(1,2)} = \frac{2}{E'} \left( K_I^{(1)} K_I^{(2)} + K_{II}^{(2)} K_{II}^{(1)} \right) \]  

(B.4)

\( K_I^{(1)} \) and \( K_{II}^{(1)} \) can be obtained by choosing an appropriate auxiliary field for state (2). As an example, for an auxiliary field of mode I (with \( K_I^{(2)} = 1 \) and \( K_{II}^{(2)} = 0 \)), the interaction integral \( I^{(1,2)} \) is expressed as \( I_{\text{mode I}}^{(1)} \):

\[ K_I^{(1)} = \frac{E'}{2} I_{\text{mode I}}^{(1)} \]

(B.5)

Aiming to simplify the numerical integration of Eq. (B.3), it can be converted into an area integral by multiplying the integrand with a smooth bounded weighting function \( q \), described below. Then for each contour \( \Gamma \), assuming the crack faces are stress free and straight in the interior of the region \( A \) and bounded by the outer prescribed contour \( C_0 \), the interaction integral may be written as:

\[ I^{(1,2)} = \oint_C \left[ W^{(1,2)} \delta_{1j} - \sigma_{ij}^{(1)} \frac{\partial u_i^{(2)}}{\partial x_1} - \sigma_{ij}^{(2)} \frac{\partial u_i^{(1)}}{\partial x_1} \right] q m_j \, d\Gamma \]

(B.6)

with \( q = 1 \) on \( \Gamma \) and \( q = 0 \) on \( C_0 \). In this equation, \( C = \Gamma + C_+ + C_- + C_0 \) and \( m_j \) is the outward unit normal vector, equal to \( n_j \) on \( C_0 \cup C_+ \cup C_- \) to the contour \( C \), see Fig. B.31. The divergence theorem is used, which gives the following equation for the interaction integral in domain form:

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

(B.7)

Figure B.31: J-integral domain definition. Domain \( A \) is enclosed by \( \Gamma, C_+, C_- \), and \( C_0 \). Unit normal \( m_j = n_j \) on \( C_+ \), \( C_- \), and \( C_0 \) and \( m_j = -n_j \) on \( \Gamma \).

Figure B.32 shows the structure of the StressIntensityFactors class which is written based on the formulation presented in this Appendix. This class is mainly used to calculate the stress intensity factors, either for plane stress/strain or Reissner-Mindlin problems, specifically for linear elastic materials. It returns the SIFs for different modes, so the crack propagation status and its direction angles can be calculated for the crack analysis process.

References


Figure B.32: Structure of the StressIntensityFactors class

- model: Model
  - thetaCrack: double
  - crackTipPoint: Point3D
  - intRadius: double
  - fractureModes: Int

+ getInteractionIntegral(): Array<LineDouble>
+ getInteractionIntegralPlates(): Array<LineDouble>
+ getAux chị(element: Element, gaussPoint: double[][], fracMode: int[]): Vector
+ getAux Stress(element: Element, gaussPoint: double[], fracMode: int[]): Vector
+ getAuxGradDisp(element: Element, gaussPoint: double[], fracMode: int[]): Matrix
+ getAuxGradDispPlates(element: Element, GaussPoint: double[], fracMode: int[]): Vector
+ getAuxBendingPlates(element: Element, GaussPoint: double[], fracMode: int[]): Vector
+ getAuxDxwPlates(element: Element, GaussPoint: double[], fracMode: int[]): Vector
+ getAuxEpsPlates(element: Element, GaussPoint: double[], fracMode: int[]): Vector
+ getCrackTipElement(): Element
+ getMyArea(element: Element): double


