Calculation of modified J-integral in presence of initial state

Summary:

This document aims at describing the theoretical formulations and implementation to compute the modified *J* -integral in *3D and 2D* via the macro_command POST_JMOD .

The theoretical background of *J* -integral is presented in Section 1 . It is first introduced by the continuous mathematical description of the standard *J* in *2D* [1]. This standard formulation without initial state is extended successively to the full *3D* formulation [2] and the modified formulations to account for the presence of an initial strain field [3] and the non-proportional loading conditions [4]. In Section 2 , the domain integral formulations are detailed in discretized form adapted for the finite element method. The overall strategy for implementation of the macro_command POST_JMOD in *code_aster* , and general information regarding the supported element types, loads, and required mechanics fields are presented in Section 3 . Specific details of the procedures used in the identification of key terms such as the virtual crack extension field, the cracked surface advancement and the gradient of the deformation and strain energy terms are presented in Section 2 . Finally, an overview list of the test cases used to validate POST_JMOD is presented in Section 3 .

The current development version of POST_JMOD is applicable for:

- elastic and plastic material behaviors;
- proportional/non-proportional and mixed mode loads;

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- initial strain (associated with an equilibrium residual stress);
- Standard FEM with a meshed crack

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Table of Contents

1Theoretical background....5
1.1 Standard J -integral formulation in 2D....5
1.2Extension of standard J -integral 3D....6
1.3Modified J -integral for initial strain under proportional loads....7
1.4Modified J -integral for initial strain under non-proportional loads....8
1.5Evaluation of initial strain....9
2 Discretized formulation implemented within POST_JMOD....12
3Algorithm of POST_JMOD J -integral in code_aster....14
1.1Approach to compute fracture mechanics parameters in presence of initial strain....14
1.2Steps in strategy implementation....14
1.3Restrictions on Element types and loads....17
1.4Required existing code_aster concepts....18
2Developments specific to POST_JMOD formulation....19
2.1Computation of virtual crack extension field q....19
2.2 Extension of contour domain identification from 2D to 3D....20
2.3Identifying node and element sets within the Integral domain....22
2.4Computation of the area increase of the cracked surface....24
2.5Computation of gradients of deformation terms in code_aster....26
2.6Computation of strain energy density term and its gradient....28
3Test cases....30
4References....31

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Table of Figures

Figure 1 : Definition of the local J-integral evaluation for 3D problems.....6
Figure 2 : Definition of area increase A c at node N i in discretization problem.....7
Figure 3 : Definition of the volume integration V around crack front.....7
Figure 4 : Definition of the volume integration domain (V 1 = V - V') around crack front.....9
Figure 5 - Example of the sequence of calculations that produce: the initial state, the CBA with initial state, modified J-integral calculation on the CBA. Example based on computations done within code_aster.....10
Figure 6 - Steps in computation of fracture mechanics.....14
Figure 7 : Definition of the pentahedral element for J calculation.....17
Figure 8 : A crack with crack front and its surface.....19
Figure 9 : The defined contours around crack tip in 2D problem.....20
Figure 10 : The local plane defined on each node of crack front in 3D.....21
Figure 11 : Definition of contour on the local plane (Pi) through the node (Ni) on crack front.....22
Figure 12 : The defined contour on each plane in discretized problem: (left) N* node is in plane and (right) N* node is out of plane.....22
Figure 13- Domain extraction step 1. A ll elements with at least one node connected to the crack front are extracted for each node along the crack front. The difference in element sets between neighbouring groups is then taken.....23
Figure 14 - Domain extraction step 2. The nodes within each element set is extracted for each crack front nodal position. The neighbouring node sets are compared to obtain the intersecting nodes.....23
Figure 15 - Domain extraction repeat step 1 for next contour.....24
Figure 16 - Domain extraction repeat step 2 for next contour.....24
Figure 17 - Crack front nodes and the crack surfaces.....25
Figure 18 : Local crack advances at node Ni in the plane of crack.....25

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

1 Theoretical background

1.1 Standard *J* -integral formulation in 2D

The *J* -integral [1] is widely used in fracture mechanics analyses to assess the structural integrity of industrial components. In its basic definition, the value of the *J* -integral is calculated along a small closed contour around the crack tip in a two-dimensional (*2D*) problem for an elastic (linear or non-linear) material subjected to a proportional monotonic stress, as shown in Error: Reference source not found .:

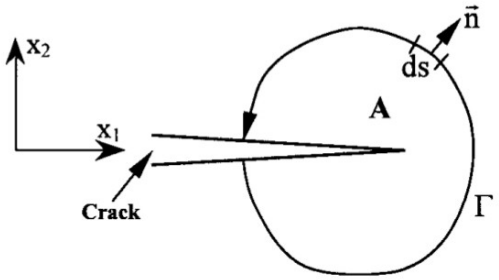


Figure : Contour integral path Γ and domain A around crack tip.

For a *2D* analysis, in the absence of the body forces and the crack face traction, under quasi-static conditions with a crack directed along the x_1 -axis (Error: Reference source not found), a path independent formulation of the standard *J* -integral definition is defined as ([1], [2],):

$$J = \lim_{\Gamma \to 0} \int_{\Gamma} (W \delta_{1i} - \sigma_{ij} \frac{\partial u_j}{\partial x_1}) n_i \mathrm{d}s \tag{1}$$

where δ_{ij} is the Kronecker coefficient, σ_{ij} and u_i are components of stress and displacement in the Cartesian coordinates, respectively. In Equation 1, Γ represents a curve surrounding the crack tip which begins at the lower face of the crack and ends at the upper one while n_i represents the unit vector normal to Γ and d *s* represents the infinitesimal path length along Γ.

The strain energy density *W* is defined at Gauss points as:

$$W = \int_0^{\varepsilon_{ij}} \sigma_{ij} \mathrm{d}\varepsilon_{ij} \tag{2}$$

Applying the divergence theorem, Equation 1 is transformed into a domain integral formulation as shown in [2]:

$$J = \int_A (\sigma_{ij} \frac{\partial u_j}{\partial x_1} - W \delta_{1i}) \frac{\partial q}{\partial x_i} \mathrm{d}A \tag{3}$$

where *q* is any smoothing function defined over the area *A* , which takes the values of unity at the crack-tip node, zero on the contour Γ, and varies from 0 to 1 within the ring of elements containing Γ. Physically, *q* represents the virtual crack extension. It should be noted that the *J* -integral approach in this

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

macro_command is equivalent to the *G-theta* method used in the operator CALC_G of *code_aster* but with a different numerical implementation. Also, *q*- field plays a similar role as the theta-field in CALC_G . Only the *u* and *q* fields are evaluated on nodes, and the remaining parameters are evaluated at the Gauss points.

1.2 Extension of standard *J* -integral 3D

Assuming planar crack surfaces, the standard *J* -integral formulation defined in Equation (1) in *2D* can be extended to a three-dimensional (*3D*) problem where a *2D* formulation is defined locally on each plane S_i perpendicular to the crack front line at each crack tip point for all the points along the crack front.

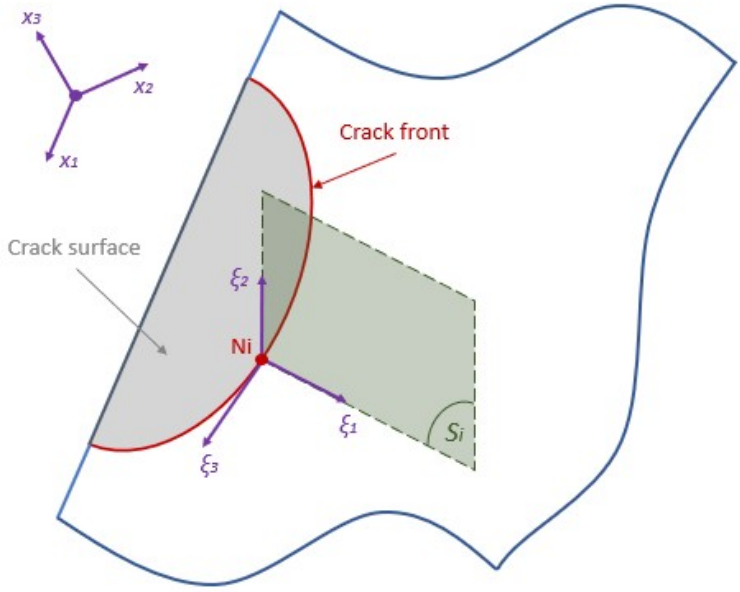


Figure 1 : Definition of the local J-integral evaluation for 3D problems.

For a *3D* problem, a crack front with a global coordinate system $x_1\ x_2\ x_3$ is first defined. Then, a local coordinate system $\xi_1\ \xi_2\ \xi_3$ is introduced at arbitrary crack tip point N_i of the crack front (Figure 1). The coordinate axes ξ_1 and ξ_2 lie in a plane S_i normal to the crack front through N_i , ξ_2 is perpendicular to the crack surface, and ξ_3 is tangent to the crack front. According to the basic definition of *J* -integral in *2D* , the local *J*- integral can be computed along a small contour around N_i in the $\xi_1\ \xi_2$ plane. For an arbitrary crack, the general *3D* expression of *J* -integral to perform computations in the global coordinate system $x_1\ x_2\ x_3$ can be expressed by Equation 4 at each local crack tip node N_i, as shown in [5]:

$$J = \frac{1}{A_c}\int_V (\sigma_{ij}\frac{\partial u_j}{\partial x_k} - W\delta_{ki})\frac{\partial q_k}{\partial x_i}\mathrm{d}V \tag{4}$$

In this equation A_c is the area increase in the cracked surface (Figure 2) generated by the virtual crack extension *q* and *V* represents the volume (Figure 3). Details of the term *Ac* are given in Section 2.4 .In the discretised finite element space, the area increase A_c at the local node N_i is determined by an only moving N_i to N_i' (Figure 2). It is noted that there is no specific requirement for the shape of the volume *V* around the crack tip when performing this domain integral. *W* is the strain energy density as defined in Equation 2 and the remaining terms are the same as in Equation 3.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

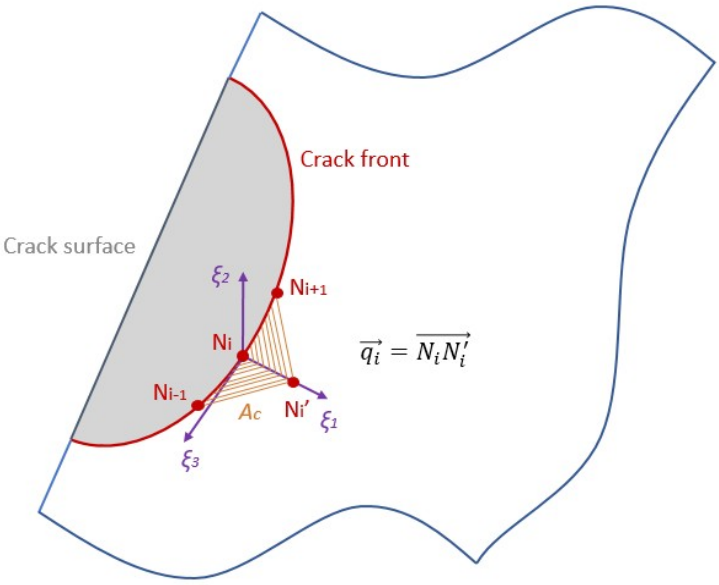


Figure 2 : Definition of area increase A_c at node N_i in discretization problem.

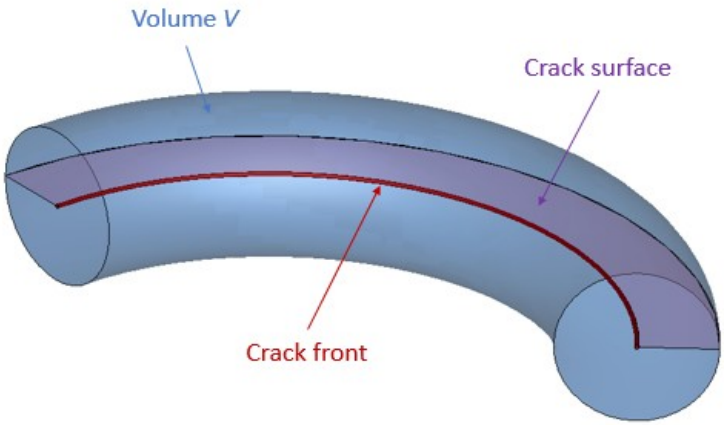


Figure 3 : Definition of the volume integration V around crack front.

1.3 Modified *J* -integral for initial strain under proportional loads

For cases of initial strains, we define the “initial state” as the moment in time prior to the application of external loading to a cracked body. When initial strains are present, the total strain during mechanical loading ε_{ij} is the sum of the mechanical strain ε^m_{ij} under load and the initial strain ε^0_{ij} at initial state, as shown in Equation 5:

$$\varepsilon_{ij} = \varepsilon^m_{ij} + \varepsilon^0_{ij} \qquad (5)$$

$$\varepsilon_{ij} = \varepsilon^m_{ij} + \varepsilon^0_{ij}\Big|_{initial\ state}$$

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

It is assumed that the initial strain ε^0_{ij} remains constant during subsequent deformation and the mechanical strain ε^m_{ij} is related to the applied stress through the material constitutive law. Note that ε^0_{ij} arises from a preliminary calculation of the initial state and more detail on the initial state is given in Section 1.1 . A 3-dimensional path independent J -integral (Equation 6) for this loading condition can be obtained under the conditions of proportional loading as shown in [2]:

$$J = \frac{1}{A_c}\int_V [(\sigma_{ij}\frac{\partial u_j}{\partial x_k} - W\delta_{ki})\frac{\partial q_k}{\partial x_i} + \sigma_{ij}\frac{\partial \varepsilon^0_{ij}}{\partial x_k} q_k]\mathrm{d}V \tag{6}$$

In this modified J -integral, W is defined as the mechanical strain energy density:

$$W = \int_0^{\varepsilon^m_{ij}} \sigma_{ij}\mathrm{d}\varepsilon^m_{ij} \tag{7}$$

Note that ε^m_{ij} is the sum of the elastic and inelastic strain obtained during loading of a cracked body, corrected for the inelastic strains at the initial state (Equation 11 in Section 1.5). The calculation procedure illustrated in Figure 6 of Section 1.5 shows that the initial state arises from a preliminary calculation step before the cracked body analysis (CBA) occurs. The initial strains at the initial state is effectively a constant field used in the post-processing of the results from the cracked body analysis.

1.4 Modified J -integral for initial strain under non-proportional loads

Lei [3] proposed a more general modified J -integral formulation which covers both cases with the presence of initial strains and non-proportional stress occurring during applying mechanical load (after initial state) while accounting for initial strains. Following the method of Wilson and Yu [6], Equation 4 is modified as:

$$J = \frac{1}{A_c}\int_V [(\sigma_{ij}\frac{\partial u_j}{\partial x_k} - W\delta_{ki})\frac{\partial q_k}{\partial x_i} + (\sigma_{ij}\frac{\partial \varepsilon_{ij}}{\partial x_k} - \frac{\partial W}{\partial x_k})q_k]\mathrm{d}V \tag{8}$$

with strain energy density W defined by Equation 7 and the total strain ε_{ij} by Equation 5.

Practically, the J values obtained on the first, second, or third contours are generally dropped because of the inaccurate stress, strain, and displacement rate at the vicinity of the crack tip (this may depend on the mesh refinement). As a general rule of thumb, it is best practice to have a sufficiently fine mesh around the crack tip for calculations. Moreover, the correction part in the right-hand side of Equation 8 can be also affected by the inaccurate near tip field data, resulting in an incorrect correction to the modified J . Note that the correction part of the modified J is evaluated using the volume integration. Therefore, any errors evaluated at the near-tip volume affect all the following calculations and force the path-independent modified J to approach an incorrect value; typically this occurs on the first or second domain. This source of error may lead to an underestimation of the modified J , and it is found that the error due to this may be significant at lower load levels (for small scale yielding conditions).

However, this error reduces with increasing load level and almost disappears when the load exceeds approximately half of the yield stress-based limit. The specific reason for the error reduction at high load level is not fully identified but it is conjectured that the higher absolute J value at a high load level reduces the relative error [4]. In order to minimize the effect of the inaccurate near-tip data on the correction term of the modified J , a very small volume surrounding the considered points on crack front is excluded from the integration volume and Equation 8 can then be modified following the method in [4]:

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

$$J = \frac{1}{A_c}\int_V (\sigma_{ij}\frac{\partial u_j}{\partial x_k} - W\delta_{ki})\frac{\partial q_k}{\partial x_i}\mathrm{d}V + \frac{1}{A_c}\int_{V_1}(\sigma_{ij}\frac{\partial \varepsilon_{ij}}{\partial x_k} - \frac{\partial W}{\partial x_k})q_k \mathrm{d}V_1 \tag{9}$$

Here the volume V_1 is limited by two volumes V and V'. The size of V' may be determined empirically by selecting the first three contours around crack front for a reasonably finely focused mesh (see Figure 4 below). It is noted that there is no specific requirement for the shape of the volumes V and V' around the crack tip when performing this domain integral.

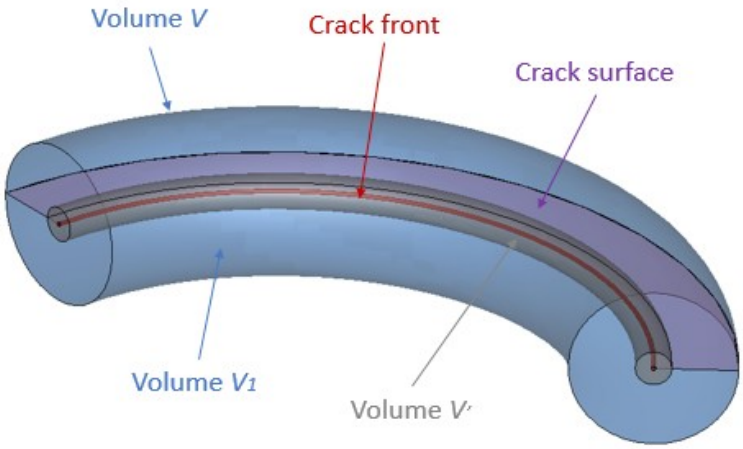


Figure 4 : Definition of the volume integration domain ($V_1 = V - V'$) around crack front.

Using Lei's modification and in presence of the initial strain, Equation 6 can be modified to:

$$J = \frac{1}{A_c}\int_V [(\sigma_{ij}\frac{\partial u_j}{\partial x_k} - W\delta_{ki})\frac{\partial q_k}{\partial x_i} + \sigma_{ij}\frac{\partial \varepsilon_{ij}^0}{\partial x_k}q_k]\mathrm{d}V + \frac{1}{A_c}\int_{V_1}(\sigma_{ij}\frac{\partial \varepsilon_{ij}}{\partial x_k} - \frac{\partial W}{\partial x_k})q_k \mathrm{d}V_1 \tag{10}$$

1.5 Evaluation of initial strain

Numerous studies in the literature show that the values of the standard J-integral in Equation 4 ceased to be path-independent under the presence of *residual stresses* . The path-dependence leads to the non-validity of J-integral as a useful constant fracture mechanics parameter. Lei *et al.* [2] proposed the modification of Rice's J-integral to correct the formulation to produce a path-independent J- integral to account for *initial strains* under proportional loads.

Note that, generally speaking, a residual stress field can arise from internal strains, which could form due to non-uniform plastic deformation, non-uniform heat expansion, or local volume changes owing to phase transformations, etc. If the internal strain field produces an additional mechanical strain field (without external loading) residual stresses occur in the body. It is these such residual stress problems that can be considered as initial strain problems [2]. For this reason, only the initial strain state is treated in the modified J-integral calculation illustrated in Figure 6.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Figure 6 shows an example process to evaluate the initial strains at the “initial state”. The first step may be a preliminary non-linear analysis of the problem to generate a residual stress field in the body, which features an associated initial strain field. A user would then, in the second step, perform a CBA under loading that includes the effects of the residual stresses in initial state. The modified J-integral computation would then occur in the third step, as a post-processing operation on the results output from the second step, using the initial strain field at the initial state evaluated in step 1.

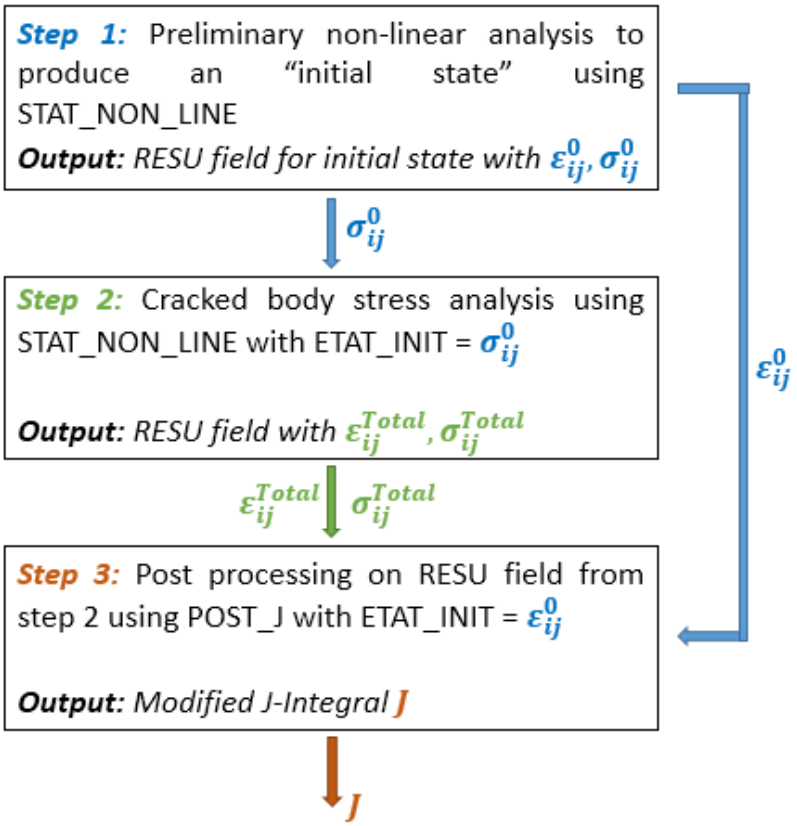


Figure 5 - Example of the sequence of calculations that produce: the initial state, the CBA with initial state, modified J-integral calculation on the CBA. Example based on computations done within code_aster.

From the user’s viewpoint, it is emphasized that only the initial strain state is required as an input, and not a distinction as to the type of strain. However, the user must be aware that they need to carefully assign the initial strain state depending on the specific engineering problems to be assessed. For example, thermal loading cases allow the initial strains to be evaluated directly, but more generally, the initial strains can be determined from the total and elastic strains at the initial state (Step 1).

$$\varepsilon_{ij}^0 = \left(\varepsilon_{ij}^{Total} - \varepsilon_{ij}^e \right)\Big|_{initial\ state} = \varepsilon_{ij}^p\Big|_{initial\ state} \qquad (11)$$

From the developer viewpoint, it is necessary to be aware that the CBA in step 2 generates total strains which include the effect of the initial strain field, as shown by Equation

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

$$\varepsilon_{ij}^{Total} = \varepsilon_{ij}^{e} + \varepsilon_{ij}^{p} + \varepsilon_{ij}^{0}$$

Hence, when evaluating ε^{m}_{ij} (Equation 7) required for modified J-Integral calculations in step 3, it may be necessary to separate the initial strain from the total mechanical strain arising from the CBA in step 2. For this reason, the concept “initial state” was introduced earlier. All plastic strains before the initial state must be treated as initial strains in the post-processing calculation of step 3. Elastic strains, at the initial state, however, are recoverable and therefore contribute to the mechanical strain during step 2. The mechanical strain (Equation 7) from the CBA in step 2 is then decomposed as:

$$\varepsilon_{ij}^{m} = \varepsilon_{ij}^{Total} - \varepsilon_{ij}^{0} = \left(\varepsilon_{ij}^{e} + \varepsilon_{ij}^{p} \right) - \varepsilon_{ij}^{plas} \Big|_{initial\ state} \quad (12)$$

where ε^{e}_{ij} and ε^{p}_{ij} are the total elastic and plastic strains, respectively, and $\varepsilon^{p}_{ij}|_{\text{initial state}}$ is the plastic strain at the initial state, which is detailed by Lei in [2].

Furthermore, normally only the total strain energy density W^{total} is available in finite element software, which is not equal to W in Equation 7 since the plastic strains will include plastic strain accumulated before the initial state. Therefore, W must be obtained from the adjusted W^{Total} as:

$$W = W^{\text{total}} - W^{p} \Big|_{\text{initial state}} \quad (13)$$

where $W^{p}|_{\text{initial state}}$ is the plastic work done during the initial state.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

2 Discretized formulation implemented within POST_JMOD

The domain integral formulations presented in Section 1 are conveniently implemented in discretized form. The general equations of modified J -integral without and with presence of initial strain Equation 9 and Equation 10 are re-written here by the finite element method.

The first, the global coordinates x_i may be written as:

$$x_i = \sum_{k=1}^{ne} N_k X_{ik} \qquad \text{with } i = 1,2,3 \qquad \ldots \tag{14}$$

where ne is the number of nodes per element, N_k are the shape functions and X_{ik} are the nodal coordinates.

The nodal fields describing the displacements u_i , the q_i fields and the initial strain ε^0 are written in a similar way:

$$u_i = \sum_{k=1}^{ne} N_k U_{ik} \tag{15}$$

$$q_i = \sum_{k=1}^{ne} N_k Q_{ik} \tag{16}$$

$$\varepsilon_{ij}^0 = \sum_{k=1}^{ne} N_k \varepsilon_{ijk}^0 \tag{17}$$

where U_{ik} are the nodal displacements, Q_{ik} are the nodal values of virtual crack extension, and ε_{ijk}^0 are the nodal values of initial strain.

It is noted that the strain ε and the energy W are evaluated at Gauss points. For calculating their gradients, these fields are first converted to global nodal fields (`XXXX_ELGA` to `XXXX_NOEU` in *code_aster*). This conversion introduces an approximation error due to the fact that a simple arithmetic mean (with no consideration of mesh size) of the connected nodes is used to evaluate the global nodal values. However, this approximation allows us to avoid the issue that locally the strain gradients are null on the element, when considering classical finite element methods.

The gradients of these fields are given by:

$$\frac{\partial u_i}{\partial x_j} = \sum_{k=1}^{ne} \sum_{l=1}^{3} \frac{\partial N_k}{\partial \eta_l} \frac{\partial \eta_l}{\partial x_m} U_{ik} \tag{18}$$

$$\frac{\partial q_i}{\partial x_j} = \sum_{k=1}^{ne} \sum_{l=1}^{3} \frac{\partial N_k}{\partial \eta_l} \frac{\partial \eta_l}{\partial x_m} Q_{ik} \tag{19}$$

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

$$\frac{\partial \varepsilon_{ij}^{0}}{\partial x_n} = \sum_{k=1}^{ne} \sum_{l=1}^{3} \frac{\partial N_k}{\partial \eta_l} \frac{\partial \eta_l}{\partial x_m} \varepsilon_{ijk}^{0} \tag{20}$$

$$\frac{\partial \varepsilon_{ij}}{\partial x_n} \approx \sum_{k=1}^{ng} \sum_{l=1}^{3} \frac{\partial N_k}{\partial \eta_l} \frac{\partial \eta_l}{\partial N_k} \varepsilon_{ijk}\big|_{NOEU} \tag{21}$$

$$\frac{\partial W}{\partial x_n} \approx \sum_{k=1}^{ng} \sum_{l=1}^{3} \frac{\partial N_k}{\partial \eta_l} \frac{\partial \eta_l}{\partial N_k} W_k\big|_{NOEU} \tag{22}$$

where η_l represents the local coordinates (η_1 , η_2 , η_3) and $\partial \eta_l / \partial x_m$ is the inverse Jacobian matrix of the transformation in Equation 14, and the terms with $|_{NOEU}$ Indicate that the global nodal values have been used in the computation, which were obtained from Gaussian values as described earlier.

The domain integral expression of modified *J* without initial stress in Equation Error: Reference source not found is:

$$J = \sum_{p=1}^{nV} \sum^{ng} \left\{ \left[\frac{1}{A_c} \left((\sigma_{ij} \frac{\partial u_j}{\partial x_k} - W \delta_{ki}) \frac{\partial q_k}{\partial x_i} \right) \right] \det\left(\frac{\partial x_l}{\partial \eta_l} \right) \right\}_p w_p +$$
$$\sum_{p=1}^{nV_1} \sum^{ng} \left\{ \left[\frac{1}{A_c} \left((\sigma_{ij} \frac{\partial \varepsilon_{ij}}{\partial x_k} - \frac{\partial W}{\partial x_k}) q_k \right) \right] \det\left(\frac{\partial x_l}{\partial \eta_l} \right) \right\}_p w_p \tag{23}$$

where nV represents the number of elements in the volume V , nV_1 represents the number of elements in the volume V_1 and *ng* is the number of gauss points per element. The quantities within $\{.\}_p$ are evaluated at all Gauss points in an element and w_p are the respective weights.

The domain integral expression of modified *J* with initial stress in Equation 10 is:

$$J = \sum_{p=1}^{nV} \sum^{ng} \left\{ \left[\frac{1}{A_c} \left((\sigma_{ij} \frac{\partial u_j}{\partial x_k} - W \delta_{ki}) \frac{\partial q_k}{\partial x_i} + \sigma_{ij} \frac{\partial \varepsilon_{ij}^{0}}{\partial x_k} q_k \right) \right] \det\left(\frac{\partial x_l}{\partial \eta_l} \right) \right\}_p w_p +$$
$$\sum_{p=1}^{nV_1} \sum^{ng} \left\{ \left[\frac{1}{A_c} \left((\sigma_{ij} \frac{\partial \varepsilon_{ij}}{\partial x_k} - \frac{\partial W}{\partial x_k}) q_k \right) \right] \det\left(\frac{\partial x_l}{\partial \eta_l} \right) \right\}_p w_p \tag{24}$$

In the development of POST_JMOD , the Equations 23-24 are calculated using POST_ELEM operator.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

3 Algorithm of POST_JMOD *J* -integral in code_aster

1.1 Approach to compute fracture mechanics parameters in presence of initial strain

There are three main steps in the computation of fracture mechanics parameters in the presence of initial state:

1. Create a field at the initial state

   There are two types of initial state: the first is a result field from a computation step (as described in section 1.5) and the second ay result from importing an external residual stress/strain field.. It is important to verify that the initial state is in auto-equilibrium. For example, from an initial stress field, one can deduce an equivalent initial strains field. By performing a dummy step with no applied loads, one can verify if the initial state is in auto-equilibrium. The user must be aware that the fields given to POST_JMOD are appropriate.

2. Perform a mechanics computation including initial state for a cracked structure

3. Post-processing using POST_JMOD

STEP 1 : Create an initial state Using STAT_NON_LINE or by CREA_CHAMP If using CREA_CHAMP , verify that the initial stress is in auto-equilibrium
STEP 2 : Mechanical calculation with initial stress and cracked structure SIG_0 ≠ 0 + CRACK → (STAT_NON_LINE) → SIG ≠ 0 , EPS ≠ 0
STEP 3 : J calculation in presence of initial state POST_JMOD=(… ETAT_INIT=_F(EPSI=EPS_0)

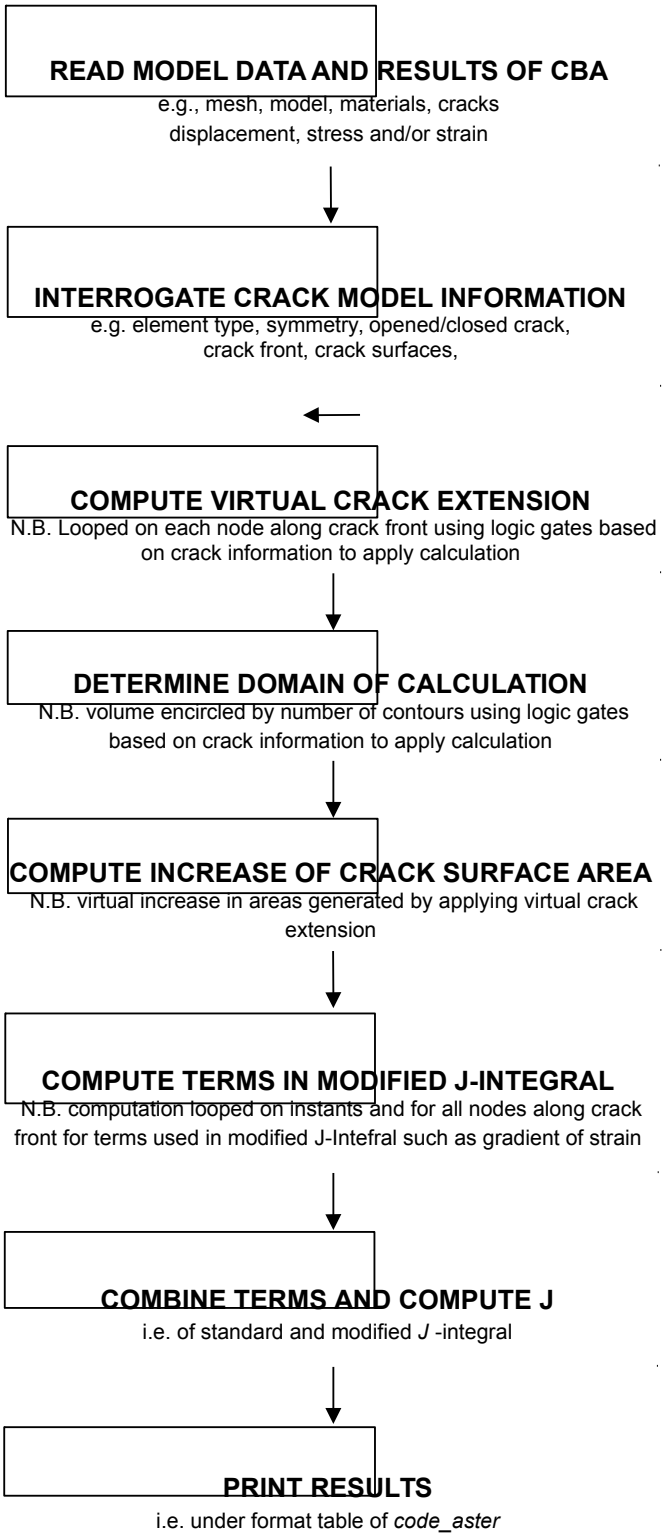
Figure 6 - Steps in computation of fracture mechanics

1.2 Steps in strategy implementation

POST_JMOD is developed as a python based macro command in order to execure step 3 of Figure 7. The programmed algorithm is contained within POST_JMOD_ops.py . The script contains all the necessary functions in order to perform the calculation described within Section 2 . However, some minor changes to the FOTRAN90 code were made in order to enable computation for large numbers of contours. This is a simple change to the size of the number of nodes defined by INFNORM_NOEU2 within defi_fond_fiss.

The general sequence of computation is described by the following diagram:

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)



Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- Firstly, using in built operators, the mesh geometrical data, model is extracted and the field results containing the global displacement field at node points and stress, strain, energy fields at gauss points. This represents the raw information used to setup the algorithm and computation. An example of the syntax used is given:

```
MAIL = self.getModel().getMesh()
MODE = self.getModel()
__RESU = RESULTAT
__RESU = CALC_CHAMP(reuse=__RESU,
                    RESULTAT=__RESU,
                    CONTRAINTE=('SIGM_ELGA'),
                    DEFORMATION=('EPSI_ELGA'),
                    ENERGIE=('ETOT_ELGA'),)
```

- Secondly, from the data structure of the crack defined by DEFI_FOND_FISS , information regarding the crack is extracted: element type, the presence of symmetry, opened/closed crack type, crack surfaces. This information is used for determining number of nodes on the crack front, set of nodes on the crack surfaces through nodes of crack front. The information is also used to create logic gates within the POST_JMOD_ops.py files in order to correctly apply the computation as data structures are different sizes and types.

```
elemType = FOND_FISS.getCrackTipCellsType()
ir, ib, symeType = aster.dismoi('SYME',FOND_FISS.getName(),'FOND_FISS', 'F')
NP, TPFISS, closedCrack = no_fond_fiss(self, FOND_FISS)
```

- Then, the virtual crack extension field is computed. This field is substitute as the *q* field in the *J* -integral computation. Further details of this step are given in Section 2.1
- Next, the volume domain *V* for the calculation is identified according to user specified number of contours. Once the integral volume is determined, all nodes are extracted from this volume and the *q* fields are assigned to these nodes. In this development, the number of contours must be greater than or equal to 2. Details of this step are given in Section 2.2 and 2.3 .
- Then, the virtual area increase of the cracked surface generated by application of virtual crack extension is calculated. To do this, the nodes on the crack surface are translated according to the virtual crack extension vectors to compute the area. Note that this is only performed to calculate the area increase and the mesh remains unchanged for the J-Integral computation. Further details of this step are given in Section 2.4 .
- Next, following the problem as linear/non-linear problem, the *J* -integral is calculated at each instant of calculation and at each node of crack front. The calculated fields such as virtual crack extension *q* , stress, strain, displacement, energy, and their gradients/divergence can be extracted from the results or can be computed using CREA_CHAMP , CREA_RESU , and CALC_CHAMP operators. The non-conventional terms are detailed in Section 2.5 and Section 2.6 .

  Here is a pseudo-code for the loops on instants and on nodes and the calculation/combination of terms:

```
Q_FIELD
for INST in LIST_INST:
      STRESS_FIELD
      DISPLACEMENT_FIELD
      GRAD_OF_DISPLACEMENT_FIELD
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
        STRAIN_FIELD
        GRAD_OF_STRAIN_FIELD

        if presence_of_initial_strain:
              INITIAL_STRAIN_FIELD
              GRAD_OF_INITIAL_STRAIN_FIELD

        STRAIN_ENERGY_FIELD
        GRAD_OF_STRAIN_ENERGY_FIELD
        for INODE in LIST_NODE:
              GRAD_OF_Q_FIELD
              COMPUTE EACH TERME OF EQUATIONS Error: Reference source not
found  AND Error: Reference source not found  IN SECTION 2
              J=COMBINATION OF TERMES
```

- At the end of `POST_JMOD` macro_command, the results of *J* are printed under format table of *code_aster* using `CREA_TABLE` and `CALC_TABLE` .

1.3 Restrictions on Element types and loads

The *J* -integral may be applied to a *3D* problem where it is defined locally around the crack tip points along the crack front in the planes perpendicular to the crack front (see Figure 1), as discussed in section 1. The procedure to calculate *J* has only been developed with hexahedral elements in mind. However, for complex 3D cracks, it is possible to use the prism elements for the first ring around the crack front. It must be noted that one of the quadrangle surfaces of these prism elements should be located on one of the crack surfaces (Figure 7). This restriction comes from the difficulty in determination of crack propagation direction and of the area increase of the cracked surface with irregular meshes. The user must take to ensure an appropriate mesh is utilised in the computation.

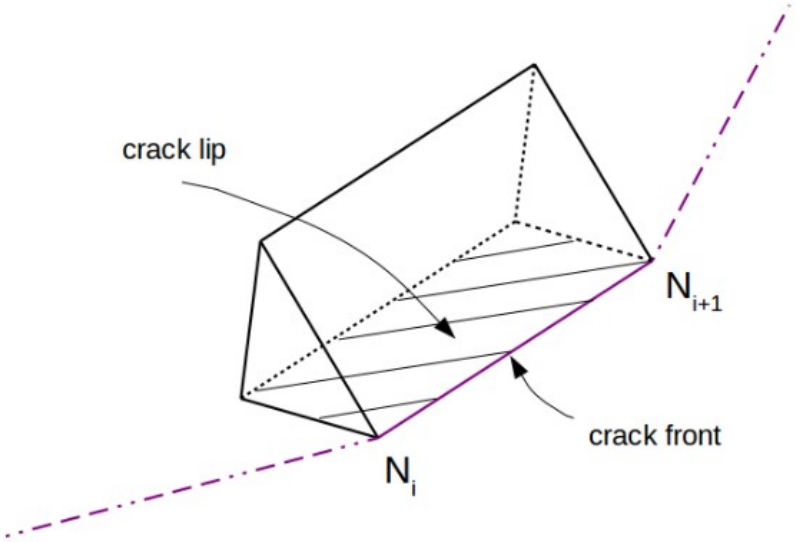


Figure 7 : Definition of the pentahedral element for J calculation.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

It should be emphasized that the same limit was observed in the *J* -integral development of Abaqus software [7]: " *The contour integral evaluation capability of ABAQUS assumes that the elements that lie within the domain used for the calculations are quadrilaterals in two-dimensional or shell models or bricks in continuum three-dimensional models. Triangles, tetrahedrals, or wedges should not be used in the mesh that is included in the contour integral regions* ".

In *code_aster* , it is possible to calculate the *J* -integral using the command POST_JMOD [U4.82.41]. The virtual crack extension field of the crack *q* is calculated automatically in this command from the information provided by the user. The current development is valid for the *3D* modeling and deals with the following conditions only:

- both elastic and plastic behaviors of materials;
- proportional, non-proportional and mixed mode loads;
- initial strains (representing residual stress fields)

It cannot deal with cases where there is pressure on the crack surfaces.

1.4 Required existing code_aster concepts

For the calculation of the energy release rate, it is necessary to characterize beforehand the meshed crack via the concept of the FOND_FISS type with the command DEFI_FOND_FISS [U4.82.01]. The current version of POST_JMOD has not been yet available for non-meshed (X-FEM) crack. As discussed later, hexahedral elements need to be used in the regions around the crack tip which are used to compute domain integral.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

2 Developments specific to POST_JMOD formulation

2.1 Computation of virtual crack extension field q

In this section, we show how to determine the virtual crack extension field *q* which is defined by Equation 3 at each node point of crack front when the crack front is defined by straight segments and the crack surface is defined by flat element faces. The vector ***q*** is defined here on all the nodes within volume integration limited by number of contours and not on a line, and *q=0* for all nodes on boundary. Then, t he integral is evaluated on this volume. In this development, the ***q*** value calculated by Equation 30 is used for assigning (this is not case that was done in *2D* by Lei).

We consider a crack fissure with crack front on which the node N^i and its neighbors N^{i-1} , N^{i+1} are presented and its surface (Figure 8). We will determine $\boldsymbol{q}_i$ at node N^i .

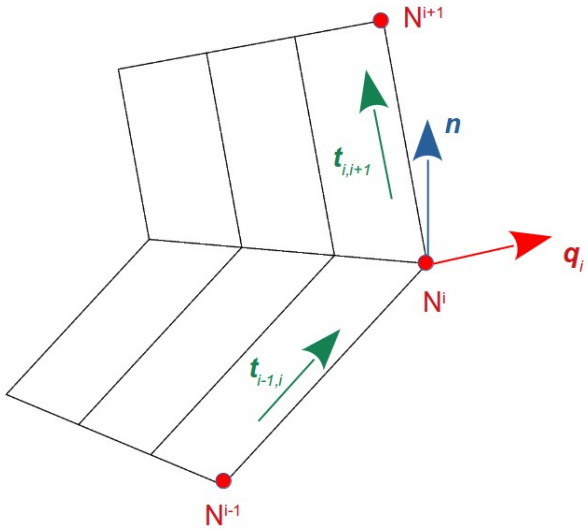


Figure 8 : A crack with crack front and its surface.

First, two vectors $\boldsymbol{t}_{i-1,i}$ and $\boldsymbol{t}_{i,i+1}$ along two neighboring segments of the crack front are determined:

$$t_{i-1,i} = \frac{u_i - u_{i-1}}{\|u_i - u_{i-1}\|} \tag{25}$$

$$t_{i,i+1} = \frac{u_{i+1} - u_i}{\|u_{i+1} - u_i\|} \tag{26}$$

where $\boldsymbol{u}_{i-1}$, $\boldsymbol{u}_i$, $\boldsymbol{u}_{i+1}$ are coordinates of N^{i-1} , N^i , N^{i+1} , respectively, and ||.|| denotes vector length.

The virtual crack extension $\boldsymbol{q}_i$ at the crack front node N^i is obtained by averaging the two following vectors:

$$q_{i-1,i} = t_{i-1,i} \times n \tag{27}$$

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

$$q_{i,i+1} = t_{i,i+1} \times n \tag{28}$$

where ***n*** is a normalized vector of crack surface at node N^i and can be determined as:

$$n = t_{i-1,i} \times t_{i,i+1} \tag{29}$$

Finally, the virtual crack extension *q* is calculated as:

$$q_i = \frac{1}{2}(q_{i-1,i} + q_{i,i+1}) \tag{30}$$

2.2 Extension of contour domain identification from 2D to 3D

In this section, the methods to identify the domain integral in *2D* and *3D* are discussed.

For a *2D* meshed crack, an example of two contours around crack tip N are shown in Figure 9 . As shown in the figure, the contours are made to coincide with the edges of the elements, the *q* fields are assigned on the nodes of these contours and the J-integral is evaluated on the 2D surface area enclosed by each contour.

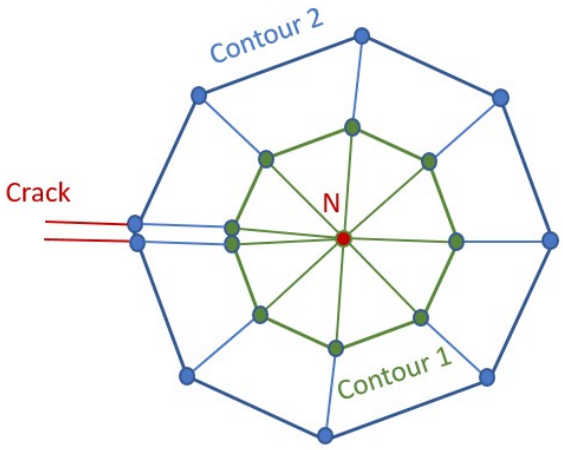


Figure 9 : The defined contours around crack tip in *2D* problem.

In *3D* problems, in order to illustrate the extension, we first assume that for each node along the crack front, there exists a plane perpendicular to the crack front through this node (Figure 10), e.g. plane (Pi) through node Ni. On each plane, the contour is identified in the same manner as the *2D* case for two successive planes as shown in Figure 11 . These nodes become the two surfaces of the torus along the $N_i N_{i+1}$ direction. The contours are made of the surfaces of the elements, the *q* fields are assigned on the nodes of these contours and the integral is evaluated on the volume *3D* .

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

As mentioned in Section 1.3 , in 3 *D* , hexahedral meshes are used in the contour integral regions and i t is possible to use the pentahedral elements for the first ring around the crack front. This allows the existence of a plane that passes each node of crack front.

When considering the discretised problem by the finite element method, we note that it is possible that the set of nodes defining the contours that for each nodal position along the crack front may not all lie on a single mathematical plane, as the assumed theory of *J* -integrals in *3D* discussed earlier. However, they are well defined on a single surface. Note that a surface here is defined as a group of element surfaces and they do not necessarily all lie on a plane. More specifically, the left hand image in Figure 12 shows the set of nodes through node Ni where all nodes lie on a plane perpendicular to the crack front. Likewise, the right hand image of Figure 12 shows the case where the nodes may lie out of plane.

N.B. In the algorithm, no specific shape of domain is required, but it is reiterated that the user must take care in the definition of the mesh to ensure the mesh regularity coincides with the theory.

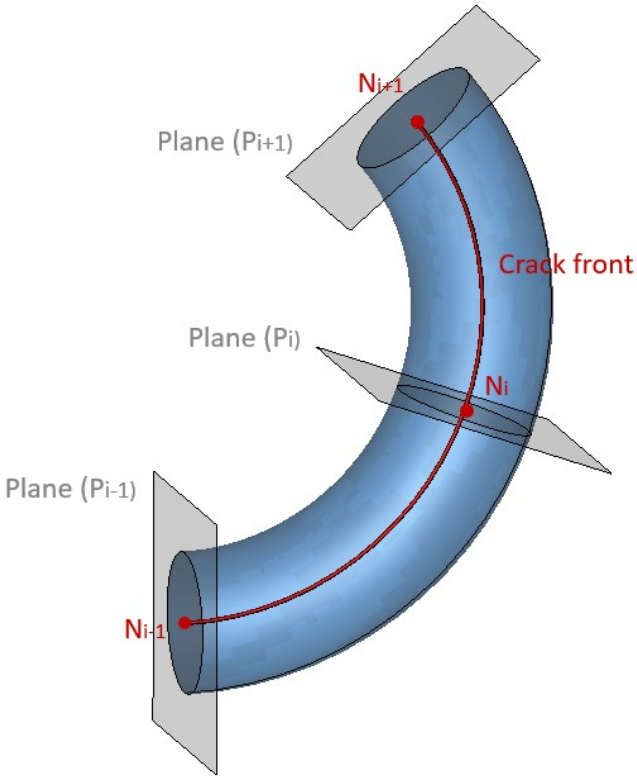


Figure 10 : The local plane defined on each node of crack front in 3D.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

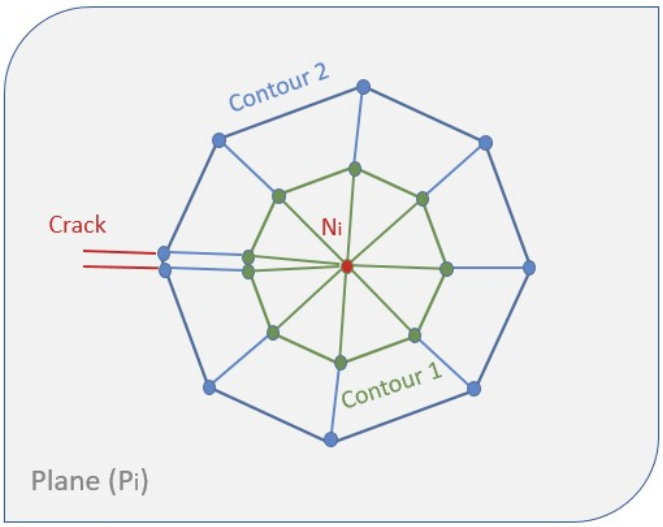


Figure 11 : Definition of contour on the local plane (Pi) through the node (Ni) on crack front.

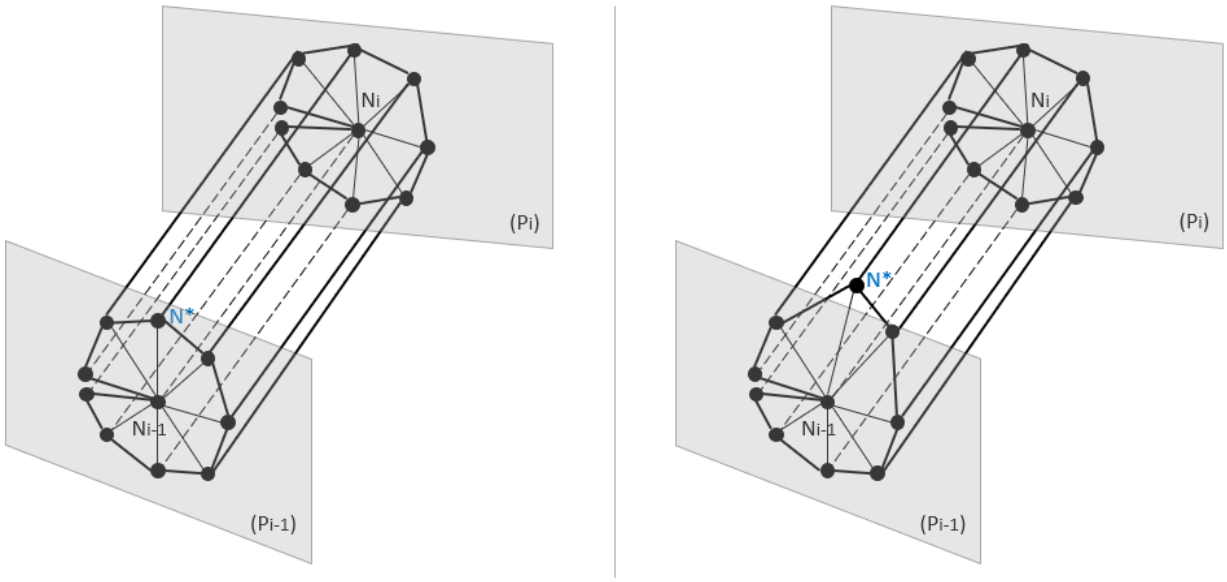


Figure 12 : The defined contour on each plane in discretized problem: (left) N* node is in plane and (right) N* node is out of plane.

2.3 Identifying node and element sets within the Integral domain

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

To determine the elements that compose each contour domain, POST_JMOD needs only to know the set of nodes along the crack front and the number of contours evaluated.

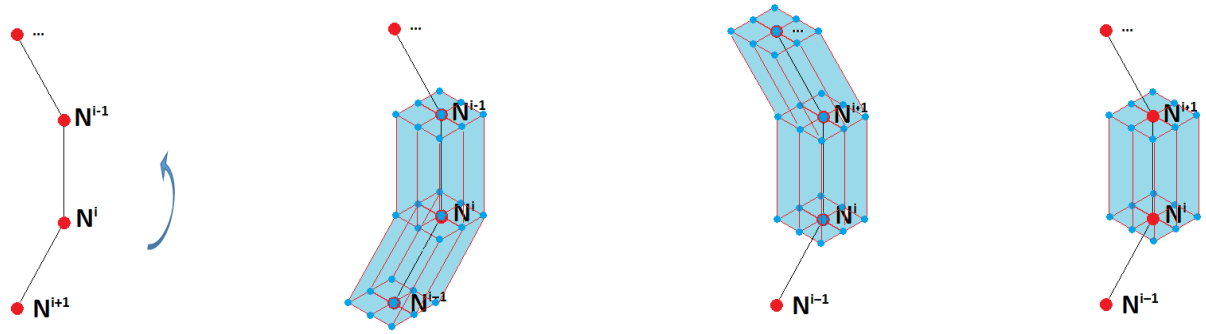


Figure 13- Domain extraction step 1. A ll elements with at least one node connected to the crack front are extracted for each node along the crack front. The difference in element sets between neighbouring groups is then taken.

Firstly, the nodes of the crack front are interrogated sequentially (Figure 13). This allows the creation of the sub sets of elements that have at least one node contained within each of those crack front nodes. By comparing the first element set along the crack front with its neighbouring element set, the elements contained within both sets can be extracted. These elements correspond to the elements lying on the plane emanating from each crack front nodal position.

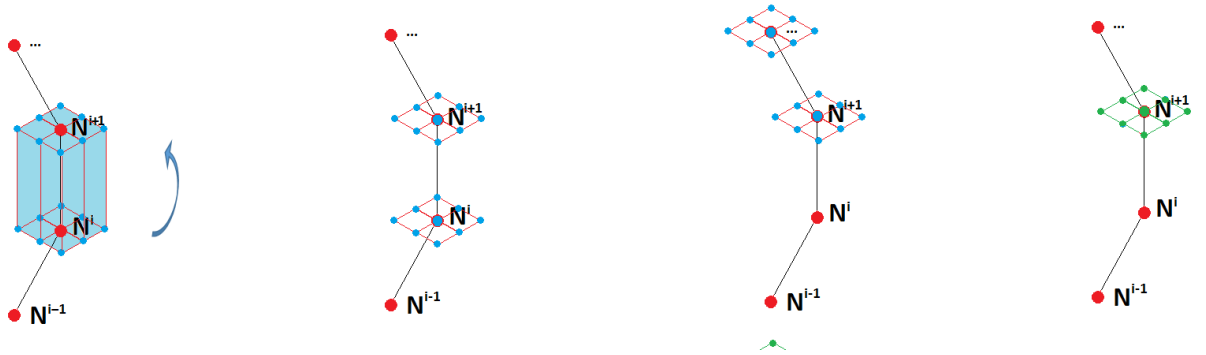


Figure 14 - Domain extraction step 2. The nodes within each element set is extracted for each crack front nodal position. The neighbouring node sets are compared to obtain the intersecting nodes.

Then, the set of elements at the crack front nodal positions are interrogated sequentially (Figure 14). This allows the extraction of the node sets corresponding to the element sets. The intersection between neighbouring nodal sets is used to extract the node set attached to the first contour of elements at each crack front nodal position. These nodes correspond to the nodes lying in the plane emanating from each crack front nodal position.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

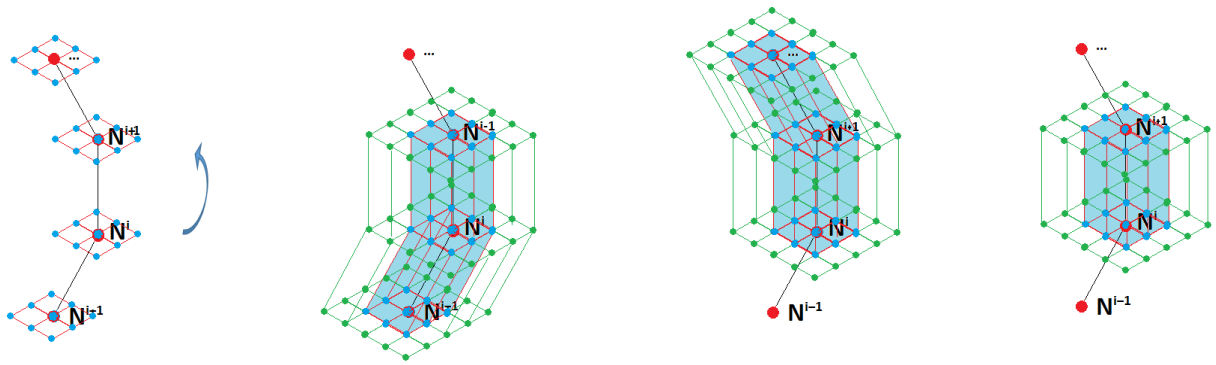


Figure 15 - Domain extraction repeat step 1 for next contour

The process is now repeated for the second contour, first allowing the extraction of elements from each crack front nodal position (Figure 15), and then allowing the extraction of node sets lying in the plane emanating from each crack front nodal position (Figure 16). The elements within the domain are utilised later in the POST_JMOD algorithm to perform the J computation; this is computed using with POST_ELEM command using in-built functionality to integrate over the volume (INTEGRAL). Note that only local values of J are computed along the crack front.

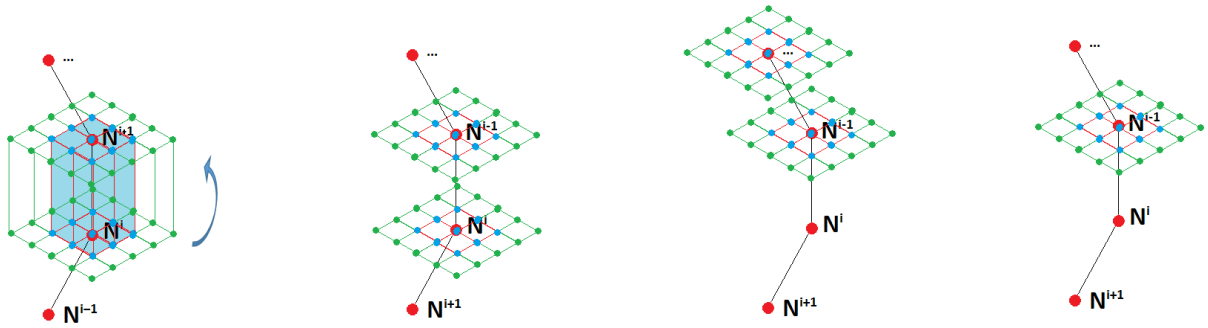


Figure 16 - Domain extraction repeat step 2 for next contour

The algorithm that does this makes use of in-built *code_aster* commands in order to define groups (or sets) of nodes and elements (DEFI_GROUP) and makes uses the specific functionality to check the difference in element sets (DIFFE) and to obtain intersections (INTERSEC). The algorithm also employs specific conditions to deal with the end most crack front nodal positions.

The domain elements and node sets are also used to assign the balues of the *q* -field. This allows the boundary nodes of each contour to be assigned a value of zero, and all other nodes within the contour to be assigned the virtual crack extension as a *q*- field.

2.4 Computation of the area increase of the cracked surface

In a *3D* analysis, *J* -integral takes the increase in cracked area A_c due to a virtual crack extension. This increase is due to the shift of that particular node and depends on a number of considered contours.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Delorenzi [5] showed an approximate formulation for a constant strain element like 8-nodes and 20-nodes 3 *D* isoparametric elements but for only first contour.

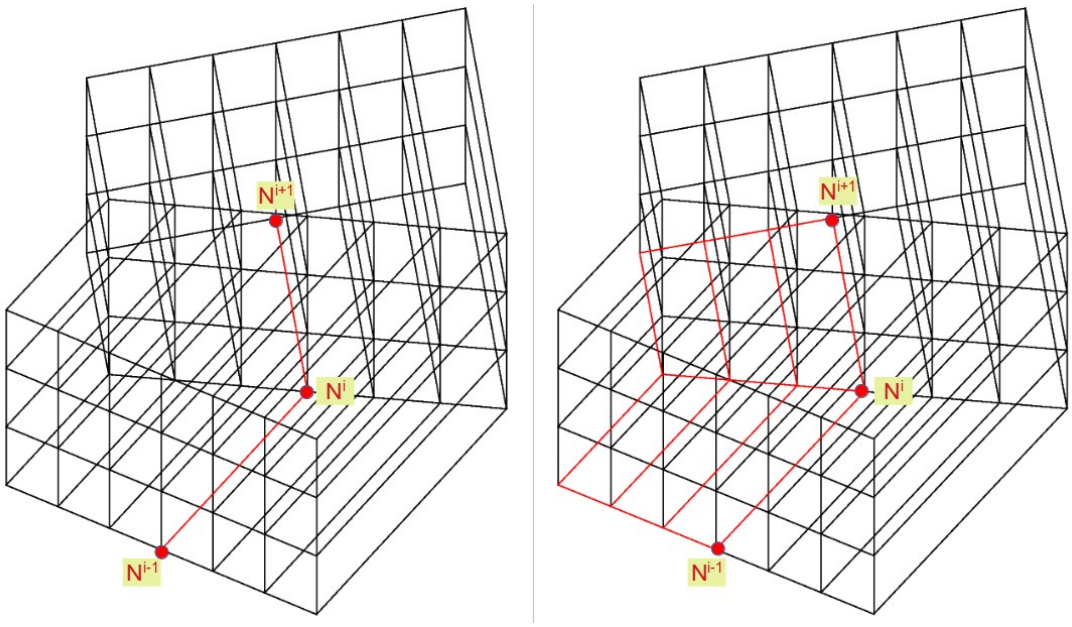


Figure 17 - Crack front nodes and the crack surfaces.

We present a simple case for calculating A_c at node N^i on the crack front using number of contours equal to 2 to illustrate the implementation. We consider a crack with crack front on which the node N^i and its neighbors N^{i-1} , N^{i+1} and its surfaces are presented Figure 17 , and more clearly in Figure 18 .

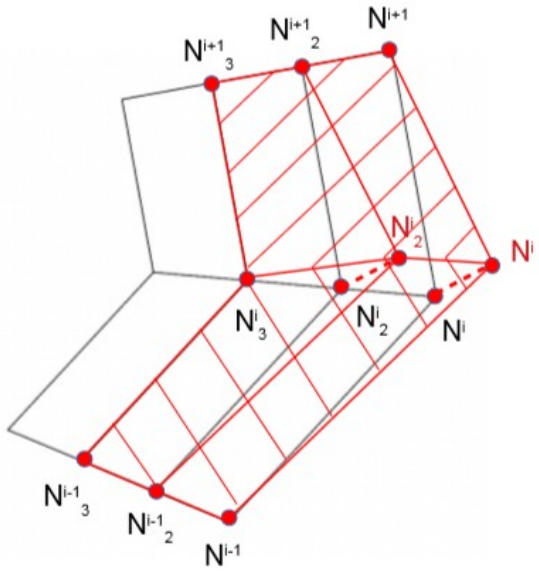


Figure 18 : Local crack advances at node Ni in the plane of crack.

- For two contours, there are only the nodes in red on the crack surfaces limited by two contours that one considers (see Figure 18). They are 9 nodes: N^{i-1} , $N^{i-1}{}_2$, $N^{i-1}{}_3$, N^i , $N^i{}_2$, $N^i{}_3$, N^{i+1} , $N^{i+1}{}_2$, $N^{i+1}{}_3$;

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- In the initial configuration, we calculate the initial area (in black) created by these 9 nodes: A_{ini} ;
- We only move two nodes N^i and $N^i{}_2$ (in black) along vector $\boldsymbol{q}_i$ (calculated in section 2.1) to the new positions (in red). It is noted that only the nodes lie on the crack surface and within the integral domain (not boundary nodes) are affected by this movement;
- Update the new surfaces (in red);
- In this configuration, we calculate the new area (in red) created by these 9 nodes: A_{fin} ;
- The increase in cracked area A_c can be calculated:

$$A_c = A_{fin} - A_{ini} \tag{31}$$

The objective of this movement is just to calculate A_c value and the other calculations are performed on the initial mesh. To calculate A_c in POST_JMOD , we use the POST_ELEM operator with INTEGRAL option by assigning a field equal to 1 on the surface which should be calculated.

2.5 Computation of gradients of deformation terms in code_aster

In the J -integral formulation, we need to manipulate the gradients of the displacement, the virtual crack extension (for the standard J) and the strain (for the modified J with or without the initial strain, both proportional and non-proportional). The discretization of these fields are presented in Equations 15, 16 and 17. The components of these fields include:

- displacement $\boldsymbol{u}$: (u_x , u_y , u_z);
- virtual crack extension $\boldsymbol{q}$: (q_x , q_y , q_z);
- strain $\boldsymbol{\varepsilon}$: (ε_{xx} , ε_{yy} , ε_{zz} , ε_{xy} , ε_{xz} , ε_{yz}).

Because of the limit of Python to read the shape functions in *code_aster* , we use an existing principle to calculate the gradient of a field. It is noted that the strain is the results of a displacement gradient field using CALC_CHAMP operator with DEFORMATION option. It can be used when:

- these fields are well determined on the nodes (TYPE_CHAM='NOEU_XXXX_R');
- the calculation is sequentially done for each component on all components of field (3 components for $\boldsymbol{u}$ and $\boldsymbol{q}$, and 6 components for $\boldsymbol{\varepsilon}$).

For example, following steps are performed to calculate the strain gradient $\varepsilon_{ij,k}$ (a non-conventional term) for the only component ε_{xx} :

$$\frac{\partial \varepsilon_{ij}}{\partial x_n} \approx \sum_{k=1}^{ng} \sum_{l=1}^{3} \frac{\partial N_k}{\partial \eta_l} \frac{\partial \eta_l}{\partial N_k} \varepsilon_{ijk}\big|_{NOEU} \tag{32}$$

1. First, the strain field which was calculated at Gauss points by defaults is changed to that at the nodal points:

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
EPS_NOEU = CREA_CHAMP(TYPE_CHAM='NOEU_EPSI_R',
                      OPERATION='DISC',
                      MODELE=MODE,
                      CHAM_GD=EPS_ELGA)
```

The strain field EPS_NOEU have 6 components: 'EPXX' , 'EPYY' , 'EPZZ' , 'EPXY' , 'EPXZ' and 'EPYZ' . We only show how to calculate the gradient of 'EPXX' component as the computations for the other five components are similar.

2. Change the strain component 'EPXX' into the displacement component 'DX' :

```
EPS_CAL = CREA_CHAMP(OPERATION='ASSE',
                     MODELE=MODE,
                     TYPE_CHAM='NOEU_DEPL_R',
                     ASSE=(_F(TOUT='OUI',
                              CHAM_GD=EPS_NOEU,
                              NOM_CMP=('EPXX',),
                              NOM_CMP_RESU=('DX',),),),)
```

3. Create a displacement field FIELD_CAL . Its value is (EPS_CAL, 0., 0.) with the components ('DX', 'DY', 'DZ') respectively;

4. Create a result from this field:

```
RFIELD = CREA_RESU(OPERATION='AFFE',
                   NOM_CHAM='DEPL',
                   TYPE_RESU='EVOL_ELAS',
                   AFFE=_F(CHAM_GD=FIELD_CAL,
                           INST=inst),)
```

5. Computing gradient with CALC_CHAMP avec DEFORMATION option of RFIELD :

```
GRAD_FIELD = CALC_CHAMP(reuse=RFIELD,
                        MODELE=MODE,
                        CHAM_MATER=MATE,
                        RESULTAT=RFIELD,
                        DEFORMATION=('EPSI_ELGA'),)

GRAD_FIELD = CREA_CHAMP(TYPE_CHAM='ELGA_EPSI_R',
                        OPERATION='EXTR',
                        RESULTAT=GRAD_FIELD,
                        NOM_CHAM='EPSI_ELGA',)
```

6. This field GRAD_FIELD have six components ('EPXX' , 'EPYY' , 'EPZZ' , 'EPXY' , 'EPXZ' , 'EPYZ') and based on the formulation: $\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i})$, we can deduce the gradient results as:

$\varepsilon_{xx,x}$ = GRAD_FIELD.EXTR_COMP('EPXX', ['MAIL',], 1)

$\frac{1}{2} * \varepsilon_{xx,y}$ = GRAD_FIELD.EXTR_COMP('EPXY', ['MAIL',], 1)

$\frac{1}{2} * \varepsilon_{xx,z}$ = GRAD_FIELD.EXTR_COMP('EPXZ', ['MAIL',], 1)

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

For a completed calculation of strain gradient $\varepsilon_{ij,k}$, we make a loop from step 1 to step 6 on all components of strain ε .

2.6 Computation of strain energy density term and its gradient

- **Computation of strain energy**

  - Without initial strain, the strain energy W can be extracted from the results of calculation STAT_NON_LINE or MECA_STATIQUE

```
W = CREA_CHAMP(TYPE_CHAM='ELGA_ENER_R',
               OPERATION='EXTR',
               RESULTAT=RESU,
               NOM_CHAM='ETOT_ELGA',
               INST=inst,)
```

  - In presence of initial strain EPS0 , the mechanical strain EPSI is first calculated, then the strain energy W is computed:

```
EPSI_TOT = CREA_CHAMP(TYPE_CHAM='ELGA_EPSI_R',
                      OPERATION='EXTR',
                      RESULTAT=RESU,
                      NOM_CHAM='EPSI_ELGA',
                      INST=inst,)

EPSI = CREA_CHAMP(TYPE_CHAM='ELGA_EPSI_R',
                  OPERATION='ASSE',
                  MODELE=MODE,
                  ASSE=(_F(CHAM_GD=EPSI_TOT, TOUT='OUI',
                           CUMUL='OUI', COEF_R=1.),
                        _F(CHAM_GD=EPS0, TOUT='OUI',
                           CUMUL='OUI', COEF_R=-1.),),)

CHW = CREA_CHAMP(OPERATION='ASSE', TYPE_CHAM='ELGA_NEUT_R',
                 MODELE=MODE, PROL_ZERO='OUI',
                 ASSE=(_F(TOUT='OUI', CHAM_GD=EPSI,
                          NOM_CMP=('EP**',), NOM_CMP_RESU=('X*',),),
                       _F(TOUT='OUI', CHAM_GD=SIGM,
                          NOM_CMP=('SI**',), NOM_CMP_RESU=('X*',),),
                       ),)

CHFMUW = FORMULE(1/2* SIGM * EPSI)

W = CREA_CHAMP(OPERATION='EVAL',
               TYPE_CHAM='ELGA_NEUT_R',
               CHAM_F=CHFMUW,
               CHAM_PARA=(CHW,),)
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- **Computation of gradient of strain energy**

In the similar way of gradient of strain in Section 2.5 , the discretization of gradient of `W` is defined in Equation 33 and was performed in *code_aster* as:

$$\frac{\partial \varepsilon_{ij}}{\partial x_n} \approx \sum_{k=1}^{ng} \sum_{l=1}^{3} \frac{\partial N_k}{\partial \eta_l} \frac{\partial \eta_l}{\partial N_k} \varepsilon_{ijk}\big|_{NOEU} \tag{33}$$

1. Change the strain energy component calculated above into the displacement component `'DX'` :

```
W_CAL = CREA_CHAMP(OPERATION='ASSE', MODELE=MODE,
                   TYPE_CHAM='NOEU_DEPL_R',
                   ASSE=(_F(TOUT='OUI',
                            CHAM_GD=__WELAS_NOEU,
                           NOM_CMP=('ETOT_ELGA',)
                           NOM_CMP_RESU=('DX',),),),)
```

2. Create a displacement field `FIELD_CAL` . Its value is `(W_CAL, 0., 0.)` with the components `('DX', 'DY', 'DZ')` respectively;

3. Create a result from this field:

```
RFIELD = CREA_RESU(OPERATION='AFFE',
                   NOM_CHAM='DEPL',
                   TYPE_RESU='EVOL_ELAS',
                   AFFE=_F(CHAM_GD=FIELD_CAL,
                           INST=inst),)
```

4. Computing gradient with `CALC_CHAMP` avec `DEFORMATION` option of `RFIELD` :

```
GRAD_FIELD = CALC_CHAMP(reuse=RFIELD,
                        MODELE=MODE,
                        CHAM_MATER=MATE,
                        RESULTAT=RFIELD,
                        DEFORMATION=('EPSI_ELGA'),)

GRAD_FIELD = CREA_CHAMP(TYPE_CHAM='ELGA_EPSI_R',
                        OPERATION='EXTR',
                        RESULTAT=GRAD_FIELD,
                        NOM_CHAM='EPSI_ELGA',)
```

5. This field `GRAD_FIELD` have six components (`'EPXX'` , `'EPYY'` , `'EPZZ'` , `'EPXY'` , `'EPXZ'` , `'EPYZ'`) and based on the formulation: $\varepsilon_{ij} = ½ (u_{i,j} + u_{j,i})$, we can deduce the gradient results as:

$W_{,x}$ = `GRAD_FIELD.EXTR_COMP('EPXX', ['MAIL',], 1)`

½ * $W_{,y}$ = `GRAD_FIELD.EXTR_COMP('EPXY', ['MAIL',], 1)`

½ * $W_{,z}$ = `GRAD_FIELD.EXTR_COMP('EPXZ', ['MAIL',], 1)`

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

3 Test cases

The keywords associated to the POST_JMOD macro-command are described within the documentation [U4.82.01]. They keywords of the operator are verified according to the following testcases:

- Validation in 2D :
  - SSNP03 : Cracked body analysis of a plate in 2D plane strain conditions with initial constraints [V6.02.003] ;
- Validation in 3D:
  - SSNV259 : Circular crack in a solid cylinder under tension [V6.04.259] ;
  - SSLV116 : Circular crack in a 3D medium with initial constraints [V3.0 4 . 116] ;

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

4 References

[1] J. R. Rice, “A path independent integral and the approximate analysis of strain concentration by notches and cracks,” *J. Appl. Mech.* , vol. 35, no. 2, pp. 379–386, 1968.

[2] Y. Lei, N. P. O’dowd, and G. A. Webster, “Fracture mechanics analysis of a crack in a residual stress field,” *Int. J. Fract.* , vol. 106, no. 3, pp. 195–216, 2000.

[3] Y. Lei, “J-integral evaluation for cases involving non-proportional stressing,” *Eng. Fract. Mech.* , vol. 72, no. 4, pp. 577–596, 2005.

[4] Y. Lei, “Improvement of modified J evaluation using finite element method,” *Br. Energy Eng.* , 2006.

[5] H. G. DeLorenzi, “Energy release rate calculations by the finite element method,” *Eng. Fract. Mech.* , vol. 21, no. 1, pp. 129–143, 1985.

[6] W. K. Wilson and I.-W. Yu, “The use of the J-integral in thermal stress crack problems,” *Int. J. Fract.* , vol. 15, no. 4, pp. 377–387, 1979.

[7] Abaqus, “Abaqus 6.14 Documentation,” 2014. [Online].
Available at: https://www.sharcnet.ca/Software/Abaqus/6.14.2/v6.14/index.html

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)