Structures de Données liées à X-FEM

Résumé

Ce document décrit les structures de données (SD) liées à la méthode X-FEM. Ces objets se trouvent dans plusieurs SD :

1) `sd_fiss_xfem`, produite par l'opérateur `DEFI_FISS_XFEM` [U4.82.08] et utilisée pour un calcul de mécanique de la rupture dans le cadre de la méthode X-FEM [R7.02.12]. Cette structure de donnée est aussi utilisée et produite par l'opérateur `PROPA_FISS` [U4.82.11],
2) `sd_modele_xfem`, produite par l'opérateur `MODI_MODELE_XFEM` [U4.41.11],
3) SD créées pour la résolution du contact..

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

1 Généralités

Lors d'un calcul X-FEM, les informations concernant la fissure n'étant pas contenues dans le maillage, il est nécessaire de les stocker dans une structure de données (SD) spécifique. Ensuite, l'enrichissement des éléments conduit à de nouveaux éléments finis, dont certaines caractéristiques seront stockées dans le modèle. Ensuite, si le contact est défini sur les lèvres de la fissure, une charge de contact est alors créée, qui contient des informations spécifiques à X-FEM. Lors de la résolution du problème de contact une SD spécifique est créée puis détruite durant l'exécution de STAT_NON_LINE . Ainsi, les objets relatifs à la méthode X-FEM sont répartis dans 4 SD différentes : sd_fiss_xfem , modèle , charge et SD créée lors de la résolution du problème de contact-frottant . La SD définissant la charge de contact, produite par DEFI_CONTACT , est décrite intégralement dans [D4.06.14]. Les objets de cette SD spécifiques à la formulation X-FEM sont décrits dans [D4.06.14, §8]. L es trois autres SD font l'objet des trois grandes parties de ce document.

Un objet de type sd_fiss_xfem décrit une fissure contenue dans un maillage 2D ou 3D, sans que ce maillage ne soit forcément conforme à la géométrie de la fissure. La structure de données sd_fiss_xfem contient la description de la fissure par le biais de level sets, les bases locales associées, des données liées à l'enrichissement des éléments finis autour de la fissure, des informations topologiques concernant le sous-découpage des éléments finis enrichis, et des données liées au contact entre les lèvres de la fissure. De plus, cette structure est enrichie par l'ajout de données nécessaires à la propagation de la fissure.

Le modèle enrichi permet de globaliser les informations locales à chaque fissure sur le maillage global. Il contient aussi des informations sur les fissures du modèle. Ce document ne décrit que les objets du modèle spécifiques à X-FEM (pour la description de la sd_modele, voir [D4.06.02]).

Si le contact est défini sur les lèvres, la SD de contact contient des informations spécifiques à X-FEM, notamment concernant l'algorithme de construction des multiplicateurs de Lagrange pour satisfaire la condition LBB : listes de relations à imposer,… Ce document ne décrit que les objets du contact spécifiques à X-FEM et stockées dans la sd_modele ou dans la SD créée lors de la résolution du contact (pour la description des SD liées au contact frottement, voir [D4.06.14]).

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

2 Arborescence

```
sd_fiss_xfem      (K8)  .:=record

   ♦ '.INFO'                              OJB                    S
     V          K16
      '.MAILLAGE'                      OJB                    S          V
              K8

   ◊ '.CHAMPS.LVS'                        OJB                    S
     V          L

   ♦ '.LTNO'                              CHAM_NO
      '.LNNO'                           CHAM_NO
      '.GRLTNO'                              CHAM_NO
      '.GRLNNO'                              CHAM_NO
      '.BASLOC'                              CHAM_NO
      '.STNO'                           CHAM_NO
      '.STNOR'                          CHAM_NO

   ♦ '.GROUP_MA_ENRI'              OJB                    S          V
          I
      '.GROUP_NO_ENRI'                  OJB                    S          V
              I
      '.CARAFOND'                       OJB                    S          V
              R

   ◊  % si la structure contient un fond de fissure
      '.FONDFISS'                       OJB                    S          V
              R
      '.BASEFOND'                       OJB                    S          V
              R
      '.FONDMULT'                       OJB                    S          V
              I
      '.LTNS'                           OJB                    S          V
              K24
      '.LTNT'                           OJB                    S          V
              K16
      '.FOND.TAILLE_R'                  OJB                    S          V
              R
      '. NOFACPTFON '                   OJB                    S          V
               I


   ◊  % si des mailles sont entièrement coupées par la fissure
      '.MAILFISS.HEAV'                  OJB              S          V
        I

   ◊  % si des mailles contiennent le fond de fissure
      '.MAILFISS.CTIP'                  OJB              S          V
        I

   ◊  % si des mailles sont coupées par la fissure et contiennent le fond de fissure
      '.MAILFISS.HECT'                  OJB              S          V
        I

   ◊  % si du contact est défini sur certaines fissures du modèle
```


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

```
    '.MAILFISS.CONT'                OJB             S          V
      I
```

◊ % si la fissure est issue d'une propagation par l'opérateur PROPA_FISS

```
    '.PRO.RAYON_TORE'               OJB         S          V          R
    '.PRO.NOEUD_TORE'               OJB         S          V          L
    '.PRO.NOEUD_PROJ'               OJB         S          V          L
```

◊ % si une grille est associée à la fissure par DEFI_FISS_XFEM

```
    '.GRI.MAILLAGE'                 OJB         S          V
K8
    '.GRI.LNNO'                                 CHAM_NO
    '.GRI.LTNO'                                 CHAM_NO
    '.GRI.GRLNNO'                               CHAM_NO
    '.GRI.GRLTNO'                               CHAM_NO
```

◊ % si la grille contient un fond de fissure

```
    '.FONDFISG'                     OJB                    S          V
            R
```

◊ % si des relations d'égalité entre inconnues de contact sont imposées

```
    '.LISEQ'                                OJB                    S
         V           I
```

◊ % si des relations d'égalité entre inconnues de contact sont imposées et si il y a des éléments multi-Heaviside

```
    '.LISEQ_LAGR'                           OJB                    S
         V           I
```

◊ % si il y a des groupes d'arêtes vitales connectées

```
    '.CNCTE'                                OJB                    S
         V           I
```

◊ % si le mot clé JONCTION a été utilisé

```
    '.JONFISS'                              OJB                    S
         V           K8
    '.JONCOEF'                              OJB                    S
         V           I
```


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

```
sd_modele_xfem    (K8)  .:=record

   ♦ '.MODELE_SAIN'                      OJB                    S
     V          K8

   ♦ '.TOPOSE.PIN'                    CHAM_ELEM
      '.TOPOSE.PMI'                   CHAM_ELEM
      '.TOPOSE.CNS'                   CHAM_ELEM
      '.TOPOSE.HEA'                   CHAM_ELEM
      '.TOPOSE.LON'                   CHAM_ELEM
      '.TOPOSE.PAI'                   CHAM_ELEM
      '.TOPOSE.PJO'                   CHAM_ELEM


   ♦ '.TOPONO.NHO'                      CHAM_ELNO
      '.TOPONO.HSE'                   CHAM_ELEM

   ◊       % si il y a des éléments multi-Heaviside
         '.TOPONO.HFA'                      CHAM_ELEM

   ♦      '.TOPOFAC.PI'                     CHAM_ELEM
      '.TOPOFAC.AI'                   CHAM_ELEM
      '.TOPOFAC.CF'                   CHAM_ELEM
      '.TOPOFAC.LO'                   CHAM_ELEM
      '.TOPOFAC.BA'                   CHAM_ELEM
      '.TOPOFAC.OE'                   CHAM_ELEM
      '.TOPOFAC.GE'                   CHAM_ELEM
      '.TOPOFAC.GM'                   CHAM_ELEM

   ◊       % si il y a des éléments multi-Heaviside
         '.TOPOFAC.HE'                      CHAM_ELEM

   ♦ '.LTNO'                            CHAM_ELNO
      '.LNNO'                         CHAM_ELNO
      '.BASLOC'                             CHAM_ELNO
      '.STNO'                         CHAM_ELNO
      '.NOXFEM'                             CHAM_ELNO

   ◊   % si des éléments stockent l'information de plusieurs fissures
      '.FISSNO'                             CHAM_ELNO
      '.HEAVNO'                             CHAM_ELNO

   ♦ '.XFEM_CONT'                       OJB                    S
     V          I

   ♦ '.NFIS'                            OJB                    S
     V          I
      '.FISS'                         OJB                  S          V
               K8
      '.XMAFIS'                             CHAM_ELEM
      '.PRE_COND'                     OJB                  S          V
               K8


sd_contact_xfem   (K16)       .:=record

   ♦ '.CARACF'                   OJB                  S          V
         R
```


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

```
'.ECPDON'                    OJB                  S          V
        I
'.METHCO'                    OJB                  S          V
        I

'.XFIMAI'                    OJB                  S          V
        K8
'.XNRELL'                    OJB                  S          V
        K24
```

◊ % cartes de contact pour l'approche grand glissement

```
'.XFPO'                  CARTE
'.XFST'                  CARTE
'.XFPI'                  CARTE
'.XFAI'                  CARTE
'.XFCF'                  CARTE
'.XFHF'                  CARTE
'.XFPL'                  CARTE
```


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

3 Contenu des objets de la sd_fiss_xfem

Remarque préliminaire sur la distinction entre 2D et 3D :

La présence de certains objets dans la structure de données sd_fiss_xfem dépend parfois de la dimension du maillage sur lequel est représenté la fissure. Ainsi, la distinction entre 2D et 3D se rapporte à la dimension du maillage contenu dans le modèle donné par l'utilisateur sous le mot-clé MODELE de la commande DEFI_FISS_XFEM.

3.1 Objets généraux

3.1.1 .INFO

Le concept .INFO est un vecteur de $K16$ de longueur 3 contenant des informations générales.
La 1ère composante de ce vecteur renseigne sur le type de discontinuité relative à la sd_fiss_xfem : soit une fissure, soit une interface. La 2ème composante indique le champ discontinu : soit le champ de déplacement, soit le champ de contrainte. La 3ème composante précise si le fond est fermé ou ouvert.
.INFO(1) = 'FISSURE' ou 'INTERFACE'
.INFO(2) = 'DEPL' ou 'SIGM'
.INFO(3) = 'OUVERT' ou 'FERME'
On accède aux valeurs de ce vecteur uniquement par la routine dismoi.f.
Exemple :

```
CALL DISMOI('F','TYPE_DISCONTINUITE',FISS,'FISS_XFEM',IBID,TYPDIS,IRET)
CALL DISMOI('F','CHAM_DISCONTINUITE',FISS,'FISS_XFEM',IBID,TYPCHA,IRET)
CALL DISMOI('F','TYPE_FOND',FISS,'FISS_XFEM',IBID,TYPFON,IRET)
```

3.1.2 .MAILLAGE

Le concept .MAILLAGE est un vecteur de K8 de longueur 1 contenant le nom du maillage sur lequel est construite la fissure (valeur du mot-clé MAILLAGE en entrée de DEFI_FISS_XFEM). En cas de propagation, ce vecteur est dupliqué de l'ancienne fissure à la nouvelle fissure.
On accède à ce vecteur uniquement par la routine dismoi.f.
Exemple :

```
CALL DISMOI('F','NOM_MAILLAGE',FISS,'FISS_XFEM',IBID,MODELE,IRET)
```

3.2 Objets relatifs aux level sets

3.2.1 .CHAMPS.LVS

Le concept .CHAMPS.LVS est un vecteur de booléens de longueur 1. Si ce vecteur est présent (et sa valeur est alors toujours égale à TRUE), les champs level set de la fissure courante ont été donnés directement par l'utilisateur (par DEFI_FISS_XFEM en utilisant les mots-clé CHAMP_NO_LSN et CHAMP_NO_LST) et ils n'ont pas été calculés.
Si les deux champs ont été extraits d'une fissure propagée par PROPA_FISS, les informations sur la localisation du domaine (.PRO.RAYON_TORE (§3.4.1) et .PRO.NOEUD_TORE (§3.4.2)) et sur l'utilisation d'une grille auxiliaire pour la fissure propagée (.GRI.MAILLAGE (§3.4.4), .GRI.L*NO (§3.4.5) et .GRI.GRL*NO (§3.4.6)) ont été perdues et la SD de la fissure courante ne les contient pas. En effet ces informations sont stockées au niveau de la SD fiss_xfem et pas au niveau de la SD champ_no. Cela pose problème dans le cas où la fissure courante est propagée par PROPA_FISS parce que l'on pourrait obtenir des faux résultats. En effet dans les cas où la localisation du domaine ou la grille auxiliaire ont été utilisés, les level sets ont été mises à jours seulement dans un petit domaine autour du fond de la fissure: dans le premier cas cela est évident, et dans le deuxième cas les level sets ont été projetées sur le maillage physique avec un domaine de projection localisé indépendamment de l'utilisation de la localisation d'un domaine de calcul en fond de fissure sur la

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

grille. En dehors du domaine de localisation/projection la valeur des levels set est la même que celle calculée pour définir la fissure: les level sets ne sont donc pas continues à la limite du domaine de localisation/projection et la description de la fissure n'est donc pas correcte sur tout le maillage. Si les informations sur le domaine de localisation et sur la grille auxiliaire ne sont pas stockées dans la SD de la nouvelle fissure, PROPA_FISS propage la fissure en considérant que les level sets sont correctes sur tout le maillage. Quand le fond de la fissure propagé sort du domaine de localisation/projection utilisé pour la fissure initiale avant propagation, on obtient des résultats faux et on a des problèmes de convergence dans la réinitialisation et la réorthogonalisation des level set.
Dans le cas où les level sets données pour définir la nouvelle fissure n'ont pas été extraites d'une fissure propagée par PROPA_FISS, ou bien que la localisation du domaine ou de la grille auxiliaire n'ont pas été utilisées avec PROPA_FISS, les champs level sets donnés représentent correctement la fissure sur tout le maillage et la propagation de la nouvelle fissure peut être calculée sans problème par PROPA_FISS.
Si ce vecteur est présent dans la SD de la fissure, PROPA_FISS émet une alarme pour que l'utilisateur vérifie s'il n'est pas en présence d'un champ extrait d'une fissure propagée avec PROPA_FISS qu'il utiliserait avec localisation du domaine de calcul ou bien avec une grille auxiliaire.

3.2.2 .LTNO et .LNNO

Le concept .LTNO (resp. .LNNO) est un champ aux nœuds (CHAM_NO) scalaire qui contient pour chaque nœud du maillage la valeur réelle de la level set tangente (resp. normale) à la fissure.

3.2.3 .GRLTNO et .GRLNNO

Le concept .GRLTNO (resp. .GRLNNO) est un champ aux nœuds (CHAM_NO) à 3 composantes réelles. Il contient pour chaque nœud du maillage les valeurs du gradient de la level set tangente (resp. normale) dans les 3 directions de l'espace.

Soit i le $i^{ème}$ nœud du maillage,

```
V =.GRLTNO(i);
```

V(1)	Valeur du gradient suivant x de la level set tangente, calculé au $i^{ème}$ nœud
V(2)	Valeur du gradient suivant y de la level set tangente, calculé au $i^{ème}$ nœud
V(3)	Valeur du gradient suivant z de la level set tangente, calculé au $i^{ème}$ nœud

En 2D, on a seulement 2 composantes suivant x et y.

3.2.4 .BASLOC

Le concept .BASLOC est un champ aux nœuds (CHAM_NO) à 9 composantes réelles (en 3D). Il contient l'origine et les vecteurs de la BASe LOCale au fond de fissure. Pour chaque nœud, les trois premières composantes sont les coordonnées du projeté du nœud sur le fond, qui correspond à l'origine de la base locale. Les trois composantes suivantes sont les coordonnées du 1er vecteur (GRLT) de la base. Les trois dernières composantes sont les cordonnées du 2ème vecteur (GRLN) de la base. Le 3ème vecteur de la base n'est pas stocké, car il se détermine facilement comme étant le produit vectoriel des 2 premiers vecteurs.

Soit i le $i^{ème}$ nœud du maillage,

```
V =.BASLOC(i);
```

V(1)	Coordonnée suivant x du projeté du nœud i sur le fond
V(2)	Coordonnée suivant y du projeté du nœud i sur le fond
V(3)	Coordonnée suivant z du projeté du nœud i sur le fond
V(4)	Coordonnée suivant x du 1er vecteur de la base locale
V(5)	Coordonnée suivant y du 1er vecteur de la base locale

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

V(6)	Coordonnée suivant z du 1er vecteur de la base locale
V(7)	Coordonnée suivant x du 2ème vecteur de la base locale
V(8)	Coordonnée suivant y du 2ème vecteur de la base locale
V(9)	Coordonnée suivant z du 2ème vecteur de la base locale

En 2D, on a seulement 2 composantes suivant x et y, soient 6 composantes pour BASLOC. Rappelons qu'en 2D, soit il n'y a qu'un seul fond de fissure, qui est alors un point (cas des fissures 2D débouchantes) et dans ce cas tous les nœuds du maillage possèdent le même projeté sur le fond de fissure, soit il y a deux fonds de fissure, donc deux points (cas des fissures 2D non débouchantes) et dans ce cas, une partie des nœuds du maillage se projette sur le 1er fond de fissure et une autre partie des nœuds du maillage se projète sur le 2ème fond de fissure.

Remarques :

- Pour un front de fissure discontinu, la projection utilisée ci-dessus doit être effectuée avec précaution. Il convient de distinguer deux types de discontinuités :
  - soit le front est continu mais coupé par une discontinuité matérielle (trou dans un pièce) ce qui génère un front discontinu dans la matière, mais prolongeable par continuité hors de la matière,
  - soit le front est réellement discontinu dans la définition de la fissure à propager.

  Avant projection, il faut au préalable tester si le front est virtuellement continu ou s'il est réellement discontinu. Pour détecter alors la discontinuité du front de fissure, on s'appuie sur un critère simple qui consiste à tester la colinéarité les vecteurs de propagation (1er vecteur de la base locale) aux extrémités des fronts de fissure entre deux fronts consécutifs:

$$acos\left(\overrightarrow{V_{P_i}}.\overrightarrow{V_{P_{i+1}}}\right)<10°$$

  Si le seuil angulaire ci-dessus est dépassé (10° d'angle), on traite le front se terminant en P_i et le front commençant en P_{i+1} comme deux fronts séparés lors de la projection de chaque nœud sur le front de fissure global.

- En deux dimensions, il est commode de « retourner » le premier vecteur de la base locale (vecteur de propagation) pour modifier le signe du mode de cisaillement plan (ou mode 2) et obtenir un signe toujours positif. Cette modification n'est effectuée localement qu'à des fins de post-traitement et n'a pas d'incidence sur la structure de données décrite dans ce paragraphe.

3.2.5 .FONDFISS

Le vecteur .FONDFISS est un vecteur de réels contenant les coordonnées des points du fond de fissure. Les points sont ordonnés suivant la méthode décrite dans le document [R7.02.12], de manière à ce qu'une abscisse curviligne puisse être définie.
Si NFON est le nombre de points du fond de fissure, alors la longueur du vecteur .FONDFISS est 4xNFON. Pour chaque point du fond de fissure, les 3 premières composantes correspondent aux 3 coordonnées (en 3D) du point, et la quatrième composante est son abscisse curviligne.
Lorsque le fond est fermé, le dernier point est égal au premier. Les 4 derniers termes du vecteur .FONDFISS sont alors identiques aux 4 premiers.

Cette structure n'est pas modifiée en 2D. Cependant on utilise uniquement les 2 premières composantes, car ni l'abscisse curviligne ni la dernière composante géométrique ne sont pertinentes en 2D.

Cet objet n'est créé que si il existe au moins un point du fond de fissure.

3.2.6 .BASEFOND

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

Le vecteur .BASEFOND est un vecteur de réels contenant la base locale associée à chaque point du fond de fissure.
Si NFON est le nombre de points du fond de fissure et si NDIM est la dimension du modèle (2 en 2D et 3 en 3D), alors la longueur du vecteur .BASEFOND est $2 \times NDIM \times NFON$. En fait ce vecteur contient, successivement pour tous les points du fond, les composantes (2 en 2D et 3 en 3D) du vecteur normal au plan de la fissure et de la direction de propagation (vecteur tangent au plan de la fissure).
En 3D, lorsque le fond est fermé, le dernier point est égal au premier. Les 4 derniers termes du vecteur .BASEFOND sont alors identiques aux 4 premiers.

3.2.7 .FONDMULT

En 3D, le fond de fissure est une ligne soit fermée (fissure non débouchante), soit ouverte (fissure débouchante). Dans la plupart des cas, le fond de fissure est une ligne continue, comme celui de la fissure circulaire représentée sur la Figure 3.2.7-1.

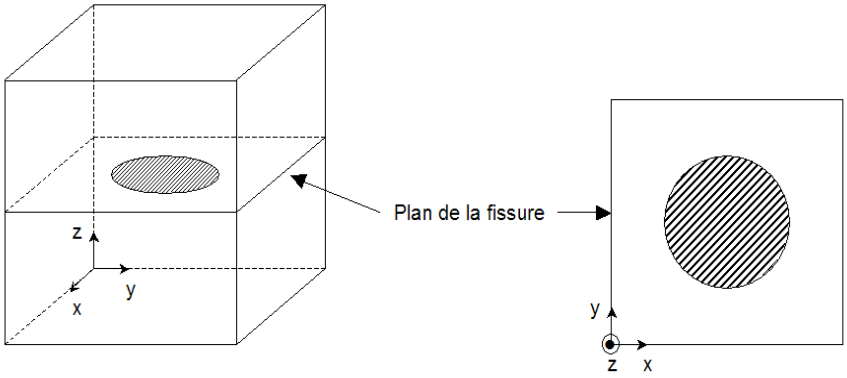


Figure 3.2.7-1 Cas d'un fond de fissure continu.

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

Cependant, il peut arriver que le fond de fissure soit en fait composé de plusieurs morceaux discontinus. C'est le cas par exemple de la fissure circulaire représentée sur la Figure 3.2.7-2. Dans ce cas, on parle toujours du fond de fissure, comme l'ensemble des morceaux du fond. On dit que le fond de fissure est un fond multiple. Sur l'exemple de la Figure 3.2.7-2, le fond de fissure est composé des lignes courbes (BC), (DE), (FG) et (HA).

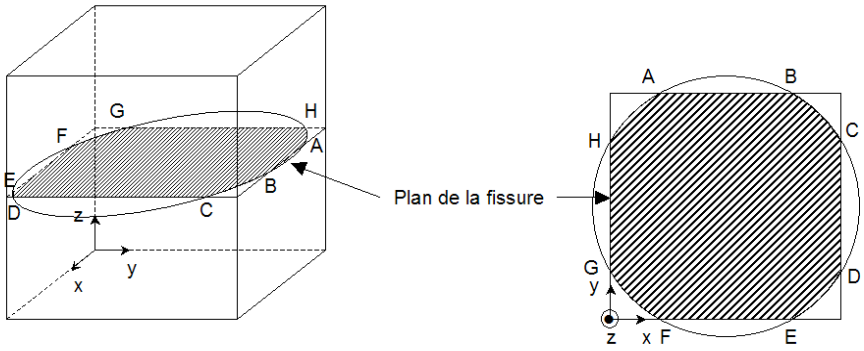


Figure 3.2.7-2 Cas d'un fond de fissure multiple.

Le vecteur .FONDMULT est un vecteur d'entiers contenant les indices des points de départ et d'arrivée des différents morceaux du fond de fissure dans le .FONDFISS. Si NFONFI est le nombre de morceaux du fond de fissure, alors le vecteur .FONDMULT est de dimension 2x NFONFI.

Par exemple :
Si FONDMULT = (1 ; 8 ; 9 ; 12)
Alors Morceau n°1 du fond de fissure : du point 1 au point 8 du . FONDFISS
Morceau n°2 du fond de fissure : du point 9 au point 12 du . FONDFISS

En 2D, chaque point du fond de fissure constitue à lui seul un fond, le vecteur .FONDMULT est néanmoins créé. Par exemple pour 2 fonds de fissures, il vaudra : FONDMULT = (1; 1; 2; 2)

L'objet est créé sous réserve qu'il existe au moins un point du fond de fissure.

3.2.8 .CARAFOND

Le vecteur .CARAFOND est un vecteur de réels de dimension 2. Il contient les paramètres utilisateurs concernant le fond de fissure. La première composante est le rayon d'enrichissement des éléments et la deuxième composante est le nombre de couches d'éléments à enrichir.

Ainsi :

CARAFOND(1)= RAYON_ENRI si RAYON_ENRI est fourni par l'utilisateur,
0 sinon.
CARAFOND(2)= NB_COUCHES si NB_COUCHES est fourni par l'utilisateur,
0 sinon.

Ces données sont celles fournies par l'utilisateur dans la commande DEFI_FISS_XFEM. Elles sont stockées car elles servent de nouveau dans la commande PROPA_FISS pour générer une fissure propagée.

3.2.9 .LTNS

Le vecteur .LTNS est un vecteur de chaines de caractères contenant les deux tables associées à la fissure. Ces tables sont accessibles via la commande RECU_TABLE.

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

La première table nommée FOND_FISS contient les coordonnées des points de fonds de fissure.
En 2D, celle-ci est constituée du numéro de fond de fissure et des coordonnées x et y. Ainsi, l'affichage de la table sera:

NUME_FOND COOR_X COOR_Y

En 3D, celle-ci décrit chaque fond de fissure. Ainsi, pour chaque fond de fissure i, la table fournira l'abscisse curviligne et les coordonnées des points en fond de fissure i. L'affichage de la table sera :

NUME_FOND NUME_PT ABS_CURV COOR_X COOR_Y COOR_Z

La seconde table nommée NB_FOND_FISS contient le nombre de fonds de fissure.

La construction de ces tables repose sur l'utilisation des vecteurs .FONDFISS et .FONDMULT.

3.2.10. LTNT

Le vecteur .LTNT est un vecteur de chaines de caractères contenant les noms des deux tables associées à la fissure à fournir à la commande RECU_TABLE. Il s'agit de FOND_FISS et de NB_FOND_FISS.

3.2.11. FOND.TAILLE_R

Ce vecteur contient pour chacun des nœuds du fond, une estimation de la taille maximale suivant la direction de propagation, des mailles qui leur sont connectées. Ces tailles sont ordonnées suivant l'ordre des nœuds donné dans le vecteur .FONDFISS.

On note $\overrightarrow{V_{P_i}}$ le vecteur de propagation de la base locale au nœud du fond N_i et $\vec{a}_{ijk}$ la k[ème] arête de la j[ème] maille à laquelle appartient le nœud N_i.
Pour chaque nœud du fond N_i, on projette les arêtes $\vec{a}_{ijk}$ sur le vecteur de direction de propagation $\overrightarrow{V_{P_i}}$. La taille maximale T_i des mailles connectées à N_i est la valeur maximale des valeurs absolues de ses projections. Autrement dit, la taille T_i est égale à

$$T_i = \max_{\substack{1 \le j \le Nb_{mailles,i} \\ 1 \le k \le Nb_{arêtes,j}}} \left(\left| \vec{a}_{ijk} . \overrightarrow{V_{P_i}} \right| \right) ,$$

où $Nb_{mailles,i}$ est le nombre de mailles connectées au nœud N_i et $Nb_{arêtes,j}$ est le nombre d'arêtes de la j[ème] maille connectée au nœud N_i.

3.2.12. NOFACPTFON

'.NOFACPTFON ' : S V I

Cet objet contient les numéros des nœuds sommets (numérotation globale du maillage) des faces des éléments parents contenant les points du fond de fissure.

Soit :

V = .NOFACPTFON
NFON = nombre de points du fond de fissure
n = LONG(V) = 4xNFON

pour i = 1, NFON :

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

V(4*(i-1)+1)	numéro global du 1er nœud sommet de la face contenant le point du fond numéro i
V(4*(i-1)+2)	numéro global du 2ème nœud sommet de la face contenant le point du fond numéro i
V(4*(i-1)+3)	numéro global du 3ème nœud sommet de la face contenant le point du fond numéro i
V(4*(i-1)+4)	• numéro global du 4 ème nœud sommet de la face contenant le point du fond numéro i si la face est quadrangulaire • 0 si la face est triangulaire

Remarques :

- *Pour chaque point du fond de fissure, la face dont les numéros de s nœuds sont stockés dans .NOFACPTFON n'est pas nécessairement unique (par exemple cas dégénéré d'un point du fond coïncidant avec une arête ou un nœud) : il s'agit de la dernière face trouvée par la routine intfac dans xptfon (phase de recherche des points du fond de fissure dans l'opérateur DEFI_FISS_XFEM)*
- *L'objet .NOFACPTFON est créé uniquement si la structure contient un fond de fissure.*

3.3 Objets relatifs à l'enrichissement

3.3.1 .GROUP_MA_ENRI

Le vecteur .GROUP_MA_ENRI contient la liste des numéros de mailles de la zone à enrichir.

3.3.2 .GROUP_NO_ENRI

Le vecteur .GROUP_NO_ENRI contient la liste des numéros de nœuds de la zone à enrichir.
Notons que ce vecteur peut se déduire de .GROUP_MA_ENRI.

Remarque :
Ces deux derniers vecteurs sont stockés car on souhaite réutiliser dans PROPA_FISS les informations liées au mot-clé GROUP_MA_ENRI de DEFI_FISS_XFEM. Et la seule possibilité de transmettre l'information est de stocker les numéros des mailles et des nœuds des mailles de GROUP_MA_ENRI.

3.3.3 .STNO

Le concept .STNO est un champ aux nœuds (CHAM_NO) à une composante entière, correspondant au statut du nœud en question.

- Si le nœud a son support entièrement coupé par la fissure, son statut vaut 1.
- Si le nœud est dans une zone « proche » (cette notion est définie suivant un critère défini par l'utilisateur) du fond de fissure, son statut vaut 2.
- Si le nœud satisfait les deux conditions précédentes, son statut vaut 3.
- Dans tous les autre cas, le statut vaut 0.

3.3.4 .STNOR

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

Le concept .STNOR est exactement le même champ que .STNO mais avec des composantes réelles. Il pourra donc être utilisé pour la visualisation avec salome par exemple.

3.3.5 .MAILFISS.HEAV

Le vecteur .MAILFISS.HEAV est un vecteur d'entiers contenant la liste des numéros des mailles enrichies de type Heaviside (maille « RONDE »).

3.3.6 .MAILFISS.CTIP

Le vecteur .MAILFISS.CTIP est un vecteur d'entiers contenant la liste des numéros des mailles enrichies de type Crack-Tip (maille « CARREE »).

3.3.7 .MAILFISS.HECT

Le vecteur .MAILFISS.HECT est un vecteur d'entiers contenant la liste des numéros des mailles enrichies de type Heaviside-Crack-Tip (maille « RONDE-CARREE »).

3.3.8 .MAILFISS.CONT

Le vecteur .MAILFISS.CONT est un vecteur d'entiers contenant la liste des numéros des mailles sur lesquelles du contact est défini pour la fissure. Il s'agit des mailles:

- Soit strictement intersectées par la fissure (type 1 sur la fig.3.3.8-1)
- Soit présentant une arête (en 2D) ou une face (en 3D) confondue avec la fissure, et située du côté esclave (type 2 sur la fig.3.3.8-1)
- Soit, en multi-fissuration, des mailles présentant un enrichissement Heaviside pour la fissure considérée et qui entrent dans une des deux premières catégories pour une autre fissure (type 3 sur la fig.3.3.8-1).

En effet, pour limiter l'introduction de nouveaux éléments lors de l'implémentation du contact avec multi-fissuration, le nombre de degrés de liberté de contact est toujours égal au nombre de degrés de liberté de Heaviside, quitte à mettre ensuite les degrés de liberté de contact superflus à 0. D'où la nécessité d'inclure les mailles de ce troisième type dans le groupe des mailles « en contact » (mais leurs degrés de liberté seront mis à 0).

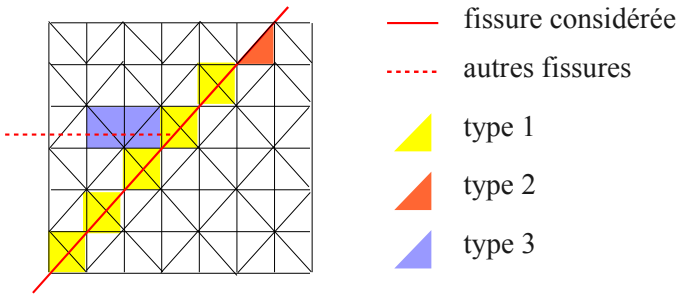


Figure 3.3.8-1 exemple du groupe .MAILFISS.CONT d'une fissure.

3.3.9 .JONFISS

Le vecteur .JONFISS contient la liste des fissures mères sur lesquelles se branche la fissure. Il est notamment utilisé dans l'opérateur MODI_MODELE_XFEM pour le découpage et l'enrichissement multiple. Ce vecteur est créé si l'utilisateur fait appel au mot clé JONCTION dans DEFI_FISS_XFEM.

3.3.10 .JONCOEF

Le vecteur .JONCOEF est de même longueur que .JONFISS. Il contient la liste des coefficients à associer aux level sets normales des fissures de .JONFISS pour retrouver le domaine dans lequel la

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

fissure branchée est définie. Ces coefficients valent $+1$ ou -1 et sont calculés automatiquement. La fissure branchée est alors définie dans la zone où pour toutes les fissures mères i, les produits `JONCOEF(i)*lsn(JONFISS(i))` sont négatifs.

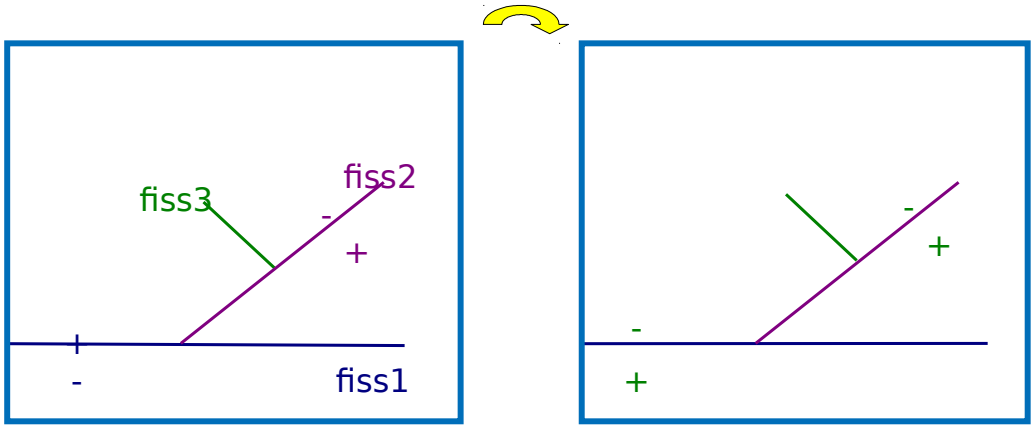


Figure 3.3.10-1 signe des levels set des fissures mères à gauche, signe obtenu après multiplication par JONCOEF à droite, et définition de la fissure branchée du côté moins.

On assure ainsi que la fissure branchée est dans la zone « moins » par rapport aux fissures mères. La figure 3.3.10-1 montre un exemple d'utilisation de `.JONCOEF` et de `.JONFISS` afin de définir la fissure 3. Sur cette figure la fissure 3 est branchée sur la fissure 2, elle même branchée sur la fissure 1, `.JONFISS` de la fissure 3 contient d onc les fissures 1 et 2. Les signes des level sets des fissures 1 et 2 sont donnés sur la figure de gauche :on en déduit les valeurs de `JONCOEF` $(-1,1)$ ainsi que la répartiti on des signe de $coef(1)\times lsn(1)$ et $coef(2)\times lsn(2)$ sur la figure de droite . La fissure 3 est définie dans la zone où les 2 signes sont moins.
En conclusion `.JONFISS` et `. J ONCOEF` permettent de définir la fissure 3 avec ses level sets normale et tangente habituelles ainsi que les level set normales des fissures mère s, auxquelles on a accès de manière élémentaire.

3.4 Objets relatifs à la propagation

3.4.1 .PRO.RAYON_TORE

Vecteur de réels de longueur égale à 1.
Dans le cas où ce vecteur existe dans la SD, le calcul de la propagation de la fissure a été fait en utilisant la localisation du domaine de mise à jour des level sets. Le domaine coïncide avec un tore en 3D ou un cercle en 2D construits autour du fond de la fissure. Le rayon de ce tore ou de ce cercle est stocké dans cet objet.
Dans le cas où ce vecteur n'existe pas dans la SD, tout le modèle a été utilisé pour le calcul de la propagation de la fissure.

3.4.2 .PRO.NOEUD_TORE

Vecteur de booléens de longueur égale au nombre de nœuds du maillage sur lequel la fissure a été défini ou, dans le cas où une grille est associée à la fissure, au nombre de nœuds de la grille.
Dans le cas où ce vecteur existe dans la SD, le calcul de la propagation de la fissure a été fait en utilisant la localisation du domaine de mise à jour des level sets. Pour chaque nœud du maillage (ou de la grille), une valeur `TRUE` signifie que le nœud est compris dans le domaine utilisé pour le calcul. Par exemple, si la valeur de la $i^{ème}$ composante du vecteur est `TRUE`, le $i^{ème}$ nœud du maillage (ou de la grille) est compris dans le domaine; par contre, si la valeur de la $i^{ème}$ composante du vecteur est `FALSE`, le $i^{ème}$ nœud du maillage (ou de la grille) est en dehors du domaine.

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

Dans le cas où ce vecteur n'existe pas dans la SD, tout le modèle (ou toute la grille) a été utilisé pour le calcul de la propagation de la fissure.

3.4.3 .PRO.NOEUD_PROJ

Vecteur de booléens de longueur égale au nombre de nœuds du maillage sur lequel la fissure a été définie.
Dans le cas où ce vecteur existe dans la SD, le calcul de la propagation de la fissure a été fait sur la grille auxiliaire associée à la fissure et les level sets calculées ont été projetées sur le maillage. Pour chaque nœud du maillage, une valeur TRUE signifie que le nœud est compris dans le domaine de projection. Par exemple, si la valeur de la i ème composante du vecteur est TRUE, le i ème nœud du maillage est compris dans le domaine; par contre, si la valeur de la i ème composante du vecteur est FALSE, le i ème nœud du maillage est en dehors du domaine et les valeurs des level sets portées par ce nœud n'ont pas été mises à jour.
Dans le cas où ce vecteur n'existe pas dans la SD, le calcul de la propagation de la fissure a été fait directement sur le maillage.

3.4.4 .GRI.MAILLAGE

Vecteur de K8 de longueur égale à 1.
Nom du maillage de la grille associée à la fissure. Si cet objet n'existe pas, aucune grille n'est associée à la fissure.

3.4.5 .GRI.LTNO et .GRI.LNNO

Le concept .GRI.LTNO (resp. .GRI.LNNO) est un champ aux nœuds (CHAM_NO) scalaire qui contient pour chaque nœud de la grille la valeur réelle de la level set tangente (resp. normale) à la fissure. Si cet objet n'existe pas, aucune grille n'est associée à la fissure.

3.4.6 .GRI.GRLTNO et .GRLNNO

Le concept .GRI.GRLTNO (resp. .GRI.GRLNNO) est un champ aux nœuds (CHAM_NO) à 3 composantes réelles. Il contient pour chaque nœud de la grille les valeurs du gradient de la level set tangente (resp. normale) dans les 3 directions de l'espace.

Soit i le i -ème nœud de la grille,

```
V =.GRI.GRLTNO(i);
```

V(1) Valeur du gradient suivant x de la level set tangente, calculé au i ème nœud
V(2) Valeur du gradient suivant y de la level set tangente, calculé au i ème nœud
V(3) Valeur du gradient suivant z de la level set tangente, calculé au i ème nœud

En 2D, on a seulement 2 composantes suivant x et y .
Si cet objet n'existe pas, aucune grille n'est associée à la fissure.

3.4.7 .FONDFISG

Le vecteur .FONDFISG est un vecteur de réels contenant les coordonnées des points du fond de fissure construit sur la grille auxiliaire. Les points sont ordonnés suivant la même méthode que celle décrite dans le document [R7.02.12] pour la création du vecteur des points du fond de fissure sur le maillage réel .FONDFISS, de manière à ce qu'une abscisse curviligne puisse être définie.
Si NFON est le nombre de points du fond de fissure sur la grille, alors la longueur du vecteur .FONDFISG est 4xNFON. Pour chaque point du fond de fissure sur la grille, les 3 premières

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

composantes correspondent aux 3 coordonnées (en 3D) du point, et la quatrième composante est son abscisse curviligne.
Cet objet est créé uniquement en 3D dans le cadre de la propagation de fissure utilisant les méthodes Upwind ou Simplexe comme outils de mise à jour des level sets.

3.5 Objets relatifs au contact

Ces objets sont créés dans la sd_fiss_xfem par DEFI_CONTACT.

3.5.1 .BASCO

Le concept .BASCO est un champ aux nœuds (CHAM_NO) à 12 composantes réelles (en 2D et en 3D). Il contient l'origine et les vecteurs de la BASe COvariante des facettes de contact. Pour chaque nœud, les trois premières composantes sont les coordonnées du point de contact associé à ce nœud, qui correspond à l'origine de la base covariante. Les trois composantes suivantes sont les coordonnées du 1er vecteur de la base. Les trois composantes suivantes sont les cordonnées du 2ème vecteur de la base. Les trois dernières composantes sont les cordonnées du 3ème vecteur de la base.

Soit i le $i^{\text{ème}}$ nœud du maillage,

```
V =.BASCO(i);
```

V(1) Coordonnée suivant x du point de contact associé au nœud i
V(2) Coordonnée suivant y du point de contact associé au nœud i
V(3) Coordonnée suivant z du point de contact associé au nœud i
V(4) Coordonnée suivant x du 1er vecteur de la base au point de contact associé au nœud i
V(5) Coordonnée suivant y du 1er vecteur de la base au point de contact associé au nœud i
V(6) Coordonnée suivant z du 1er vecteur de la base au point de contact associé au nœud i
V(7) Coordonnée suivant x du 2ème vecteur de la base au point de contact associé au nœud i
V(8) Coordonnée suivant y du 2ème vecteur de la base au point de contact associé au nœud i
V(9) Coordonnée suivant z du 2ème vecteur de la base au point de contact associé au nœud i
V(10) Coordonnée suivant x du 3ème vecteur de la base au point de contact associé au nœud i
V(11) Coordonnée suivant y du 3ème vecteur de la base au point de contact associé au nœud i
V(12) Coordonnée suivant z du 3ème vecteur de la base au point de contact associé au nœud i

En 2D, les composantes suivant z sont nulles.

3.5.2 .LISEQ (adresse JLIS1)

C'est la LISte des relations d'égalité (EQuality) entre les inconnues de contact.
Vecteur d'entiers de longueur NRELEQ*2 où NRELEQ est le nombre de relations d'égalité.
Pour chaque relation d'égalité, on stocke les numéros des 2 nœuds faisant partie de l'égalité :
Soit IE la $i^{\text{ème}}$ relation d'égalité,
ZI(JLIS1-1+2*(IE-1)+1) est le numéro du 1er nœud faisant partie de l'égalité.
ZI(JLIS1-1+2*(IE-1)+2) est le numéro du 2ème nœud faisant partie de l'égalité.

3.5.3 .LISEQ_LAGR (adresse JLISLA)

C'est la LISte des numéros de LAGRange associées aux relations d'égalité (EQuality) entre les inconnues de contact. Cette liste est construite dans le cas des mailles multi-Heaviside. Elle permet

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

de retrouver le numéro de ddl de Lagrange à associer au numéro d'un nœud pris dans une relation d'égalité.

`ZI(JLISLA-1+2*(IE-1)+1)` est le numéro de Lagrange à associer au 1er nœud de la relation d'égalité.
`ZI(JLISLA-1+2*(IE-1)+2)` est le numéro de Lagrange à associer au 2ème nœud de la relation d'égalité.

Par exemple si il faut imposer la relation `LAGS_C=LAG3_C` entre les nœuds 'N26' et 'N32' pour la ième relation d'égalité, on aurra :

```
ZI(JLIS1-1+2*(IE-1)+1) = 26
ZI(JLIS1-1+2*(IE-1)+2) = 32
ZI(JLISLA-1+2*(IE-1)+1) = 1
ZI(JLISLA-1+2*(IE-1)+2) = 3
```

3.5.4 . CNCTE (adresse JCNTES)

Il s'agit des groupes d'arêtes vitales connectées. Pour chaque numéro local d'arête dans une maille, qui appartient à un groupe, on incrémente NCTE et on stocke :

`ZI(JCNTES-1+4*(NCTE-1)+1)` le numéro de l'arête dans le groupe,
`ZI(JCNTES-1+4*(NCTE-1)+2)` le numéro de groupe,
`ZI(JCNTES-1+4*(NCTE-1)+3)` le numéro de maille,
`ZI(JCNTES-1+4*(NCTE-1)+4)` le numéro local de l'arête dans la maille.

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

4 Contenu des objets de la sd_modele_xfem

4.1 Objets généraux

4.1.1 .MODELE_SAIN

Le concept .MODELE_SAIN est un vecteur de K8 de longueur 1 contenant le nom du modèle sur lequel est construite lie modèle enrichie (valeur du mot-clé MODELE_IN en entrée de MODI_MODELE_XFEM).

4.2 Contenu des objets relatifs à la concaténation

4.2.1 .LNNO, .LTNO, .BASLOC et .STNO

Attention, ces objets n'ont pas la même structure que les objets du même nom contenus dans la sd_fiss_xfem ([§3.2.2], [§3.2.4] et [§3.3.3]). Ce sont des champs élémentaires de type ELNO et non pas des CHAM_NO. La principale différence est que les CHAM_NO stockent l'information sur les noeuds du maillage (une valeur par nœud) alors que les CHAM_ELNO stockent l'information par noeud sur l'élément. Un nœud étant connecté à N éléments, la valeur sera stockée N fois. Ce choix a pour finalité le stockage des informations de plusieurs fissures dans un élément, on utilise alors la notion de sous point.
Si un élément est contenu dans la SD fiss(i)//'.MAILFISS.HEAV', on dit qu'il est vu par la fissure i: il faut alors stocker dans le modèle les informations relatives à cette fissure. Si l'élément voit 2 fissures ou plus, il faut stocker dans le modèle les informations de toutes les fissures. En définissant le nombre de sous points $NBSP$ des CHAM_ELNO comme le nombre de fissures vues par l'élément, on duplique $NBSP$ fois l'espace mémoire relatif à la SD pour l'élément qui voit $NBSP$ fissures. Ainsi on peut stocker les informations de toutes les fissures dans cet élément.
La figure 4.2.1-1 montre un exemple de concaténation de 2 fissures dans le modèle. On représente sur cette figure les SD .LNNO des 2 fissures et celle du modèle. L'élément du centre voit 2 fissures, le champ MODELE contient donc 2 composantes par noeud dans cet élément.

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

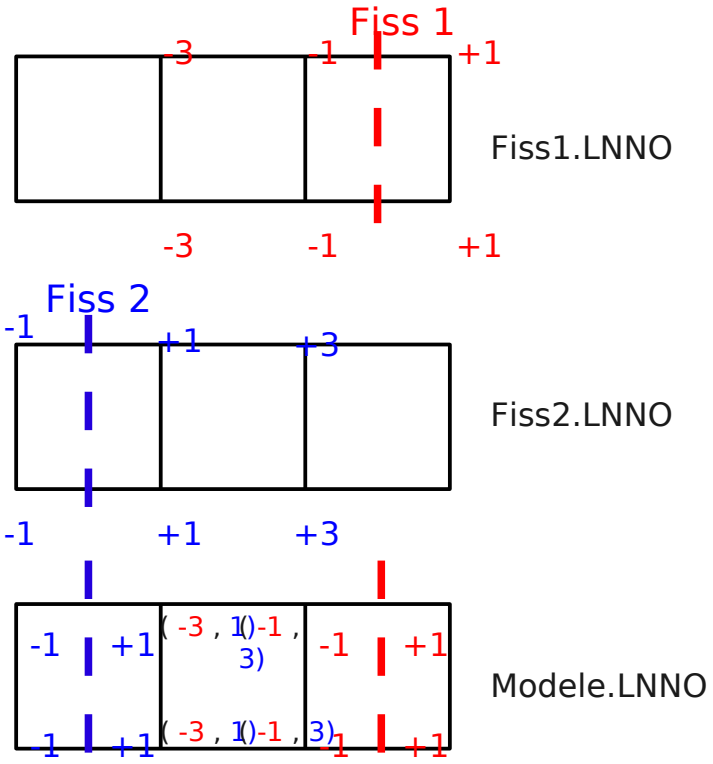


Figure 4.2.1-1 Exemple de stockage de SD multiple dans le modèle.

4.2.2 .NOXFEM

Il s'agit d'un CHAM_NO à 2 composantes. Les composantes sont affectées si le nœud possède un enrichissement X-FEM (i.e. il existe une fissure `fiss(i)` pour laquelle le champ `fiss(i)//'.STNO'` n'est pas nul en ce nœud). La première composante correspond au numéro de maille X-FEM contenant le noeud. La deuxième composante est le numéro local du nœud dans cette maille X-FEM. Cette structure est utile pour l'imposition de conditions limites de type Dirichlet.

4.2.3 .FISSNO

Il s'agit d'un CHAM_ELNO. Cette SD est créée lorsqu'au moins un élément de la structure doit stocker les informations de plusieurs fissures. Pour l'introduction de plusieurs degrés de liberté Heaviside, on a fait le choix d'incrémenter séquentiellement par nœuds les numéros de degrés de liberté. Par exemple figure 4.2.3-1, on introduit deux fissures. On affecte pour la première fissure (en bleu à gauche) les ddl $H1$. Pour la deuxième fissure (en rouge au centre) on attribue des degrés de liberté $H1$ aux nœuds qui n'ont pas déjà de degrés de liberté $H1$ affectés précédemment, et des degrés de liberté $H2$ pour les nœuds possédant déjà des degrés de liberté $H1$. Ce choix est fait dans le but de ne pas alourdir les catalogues élémentaires. En effet cette stratégie ne nécessite pas d'élément ne possédant que des $H2$.

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

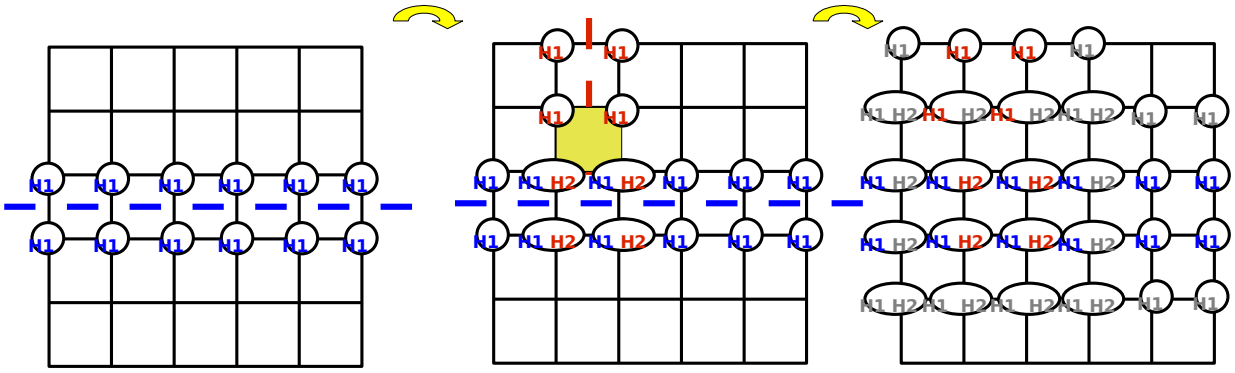


Figure 4.2.3-1 Degrés de liberté Heavisides, pour deux fissures qui se chevauchent

Cependant il faut faire attention, car dans certains éléments, l'incrémentation des fissures n'assure pas un lien direct entre le numéro de degré de liberté Heaviside global et le numéro de fissure (Dans l'exemple, $H1$ peut correspondre à la fissure 1 ou à la fissure 2 dans l'élément jaune). Lors de l'assemblage dans les TE, il faut donc pouvoir associer les degrés de liberté $H_{IFH}[X,Y,Z]$ (avec IFH le numéro de degré de liberté Heaviside global) du nœud INO au bon numéro de fissure. De plus les éléments doivent être complétés par des degrés de liberté Heaviside qui sont ensuite éliminés (degrés de liberté en gris sur la figure de droite). Il faut pour éliminer les degrés de liberté $H_{IFH}[X,Y,Z]$ du nœud INO avoir le statut de la bonne fissure en INO.
La SD `MODELE//'.FISSNO'` permet de faire cela. Elle retourne le numéro de fissure $IFISS$ vu localement au nœud INO pour les degrés de liberté $H_{IFH}[X,Y,Z]$. On a $IFISS = FISSNO(INO, IFH)$. On récupère alors le bon pointeur pour les SD `MODELE//'[.LNNO, .STNO .TOPOSE.HEA]`.

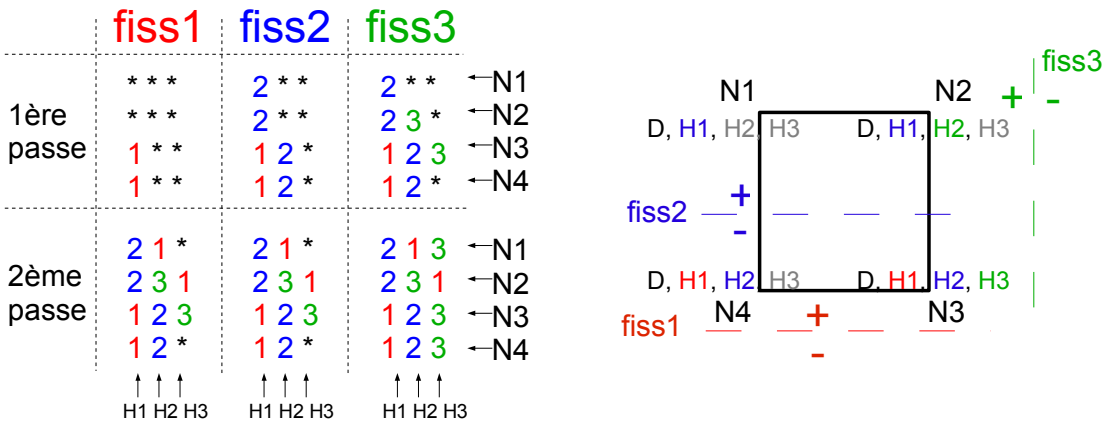


Figure 4.2.3-2 Exemple de construction de la SD Modèle.FISSNO

La figure 4.2.3-2 illustre la construction du tableau `.FISSNO`. Les lignes correspondent aux numéros de nœud, et les colonne aux numéros de degré de liberté Heaviside. On remplit le tableau de manière séquentielle en 2 passes. Lors de la première passe, on regarde le statut (`MODELE.STNO`) du noeud INO pour la fissure $IFISS$. Si le statut n'est pas nul, on remplit la première case par la donnée IFH dans leur ordre de parcours au INO qui n'a pas déjà été affectée avec $IFISS$. On complète le tableau si nécessaire lors de la deuxième passe en procédant comme pour la première passe mais en remplissant les case par la donnée IFH en INO si le statut est nul.

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

Pour illustrer l'utilisation de cette SD, on reprend l'exemple plus simple de la figure 4.2.1-1 sur la figure 4.2.3-3 . On a alors construit `Modele.FISSNO` pour l'élément du milieu. Si on veut (lors de l'assemblage par exemple) récupérer la fonction Heaviside associée au nœud $N1$ pour les degrés de liberté $H_1[X,Y]$. On récupère le numéro de fissure via $IFISS=FISSNO(INO=1,IFH=1)=2$. On a alors la fonction Heaviside qui vaut $HE(IFISS=2)=+1$. Pour le nœud $N4$, on aura $FISSNO(4,1)=1$ et donc $HE(IFISS=1)=-1$.

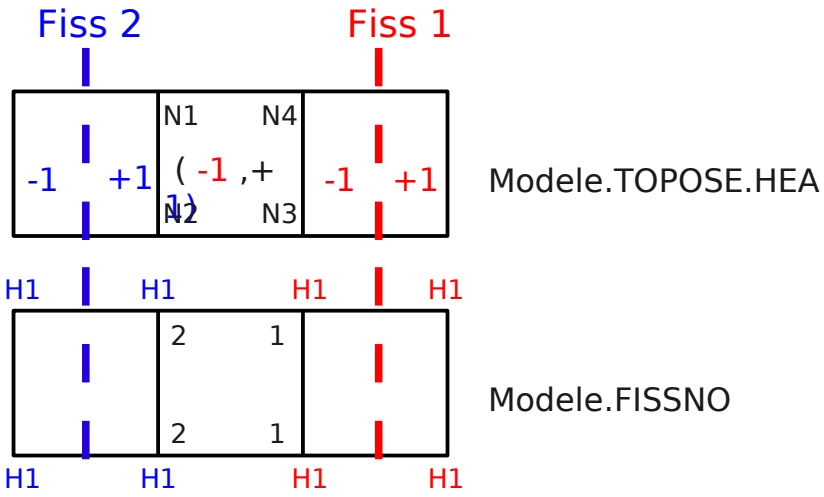


Figure 4.2.3-3 Exemple pour la SD Modèle.FISSNO

Plus généralement, la fonction Heaviside dans un sous élément d'intégration SE , pour un nœud INO et un degré de liberté Heaviside IFH s'obtient de la manière suivante :

$$HE(SE,INO,IFH)=HE(SE,IFISS(INO,IFH)) .$$

Cette structure de données permet aussi de supprimer les degrés de liberté Heaviside en trop dans les TE . On récupère le statut (`MODELE.STNO`) du nœud INO pour le degré de liberté Heaviside IFH , alors que cette structure est stockée par fissure :

$$STATUT(INO,IFH)=STATUT(INO,IFISS(INO,IFH)) .$$

Cette structure est aussi utilisée pour le post-traitement X-FEM, lors de l'interpolation des déplacements sur les lèvres de la fissure.

4.2.4 .HEAVNO

Il s'agit de la structure inverse de `.FISSNO`. Cette SD est créée lorsqu'au moins un élément de la structure doit stocker les informations de plusieurs fissures. Elle est construite en même temps que `.FISSNO` et retourne le numéro de degré de liberté Heaviside associé au nœud INO et à la fissure $IFISS$. En gardant les notations du paragraphe précédent on a :

$$IFH=HEAVNO(INO,IFISS)$$

Dans le cas du contact multi-Heaviside, le numéro de degré de liberté de Lagrange associé à une fissure $IFISS$ correspond au numéro de degré de liberté Heaviside IFH correspondant. Ainsi cette

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

structure de données permet-elle de retrouver le nom du degré de liberté de Lagrange (et sa position dans la matrice/vecteur élémentaire) à partir du numéro de fissure. Si on traite le contact sur une facette de contact associée à la fissure $IFISS$, le Lagrangien qui travaille au nœud INO porte le numéro :

$$ILAG=HEAVNO(INO,IFISS)$$

4.3 Contenu des objets relatifs à la topologie des sous-éléments

Notes aux développeurs :

- *certaines valeurs* ma *sont en dur dan le fortran (* `te0514`, `xbsigm`, `xmel3d` *...)*
- *on autorise 4 fissures au maximum pour les modélisations multiheaviside*
- *les longueurs des structures de données sont donc multipliées par 4 du mono-heaviside au multi-heaviside*

Rappel sur la manière dont sont découpés les éléments X-FEM (Document R7.02.12 partie 3.3 Sous-découpage) :

	3D	2D
Découpage(1) Phase préalable de découpe	Chaque élément (hexaèdre, pentaèdre) est virtuellement découpé en tétraèdres 6 tétraèdres au maximum	Chaque élément (quadrangle) est virtuellement découpé en triangles 2 triangles au maximum
Découpage(2) Sous-découpage	Chaque tétraèdre est à son tour découpé en sous-tétraèdres 6 sous-tétraèdres au maximum	Chaque triangle est à son tour découpé en sous-triangles 3 sous-triangles au maximum
Points d'intersection	Points d'intersection entre les arêtes des tétraèdres issus du découpage(1) et la surface de la fissure 11 points d'intersection au maximum	Points d'intersection entre les arêtes des triangles issus du découpage(1) et la courbe de la fissure 3 points d'intersection au maximum
Points milieux	Points au milieu des arêtes des triangles issus du découpage(2) [à condition de ne coïncider avec aucun nœud de l'élément initial complet] 66 points au maximum (59 points milieux au maximum distincts des nœuds centraux et 7 points centraux au maximum)	Points au milieu des arêtes des triangles issus du découpage(2) [à condition de ne coïncider avec aucun nœud de l'élément initial] 11 points milieux au maximum (10 points milieux au maximum distincts des nœuds centraux et 1 point central au maximum)
Point central du quadrangle quadratique	Stockage des points milieux des faces quadrangle du polyèdre quadratique. 7 points centraux au maximum	Point d'intersection des deux diagonales droites d'un quadrangle quadratique 1 point central au maximum
Processus de découpage(1) et sous-découpage(2)	1 élément X-FEM 3D tétraèdre n°1 6 sous-tétraèdres au max. (2) tétraèdre n°2	1 élément X-FEM 2D triangle n°1 3 sous-triangles au max. triangle n°2 (2)

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

	6 sous-tétraèdres au max. ...	3 sous-triangles au max.

Remarque : le nombre de sous- éléments n'est pas modifié par l'introduction de mailles quadratiques.

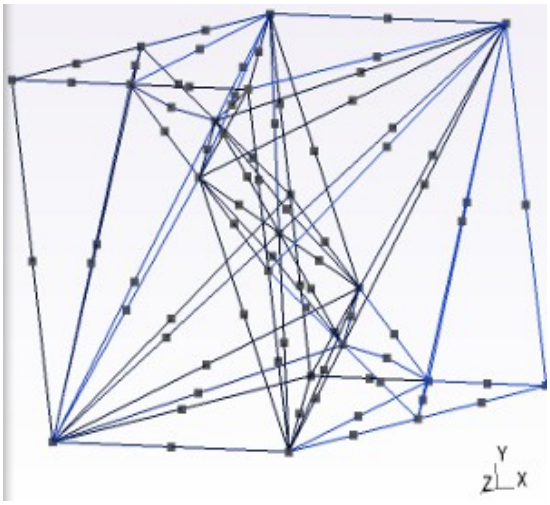


Figure 4.3-1 : configuration générant le maximum de points d'intersection / points milieux

4.3.1 .TOPOSE.PIN

Renseigne sur les POints d'INtersection.
C'est un champ constant par élément (CHAM_ELEM) à 33 composantes réelles.
Pour chaque élément, ce champ comporte les coordonnées des points d'intersection (au sens précédemment défini). Comme le nombre maximal de tels points est 11, et vu que l'on travaille en dimension 3, le nombre de composantes par élément est dimensionné à 33. Pour chaque élément, les composantes sont ordonnées comme ceci :
$\{x_1, y_1, z_1, x_2, y_2, z_2, ..., x_{11}, y_{11}, z_{11}\}$ où x_i, y_i, z_i sont les coordonnées du $i^{\text{ème}}$ point d'intersection sur l'élément.
On notera que si l'élément n'est pas fissuré, les composantes sont nulles.

En 2D, on a au maximum 3 points d'intersection dinstincts des nœuds sommets, avec 2 composantes chacun, donc 6 composantes au total.

Pour le multiheaviside :
En 3D, le champ nécessite $33 \times 4 = 132$ composantes réelles.
En 2D, le champ nécessite $6 \times 4 = 24$ composantes réelles.

4.3.2 .TOPOSE.PMI

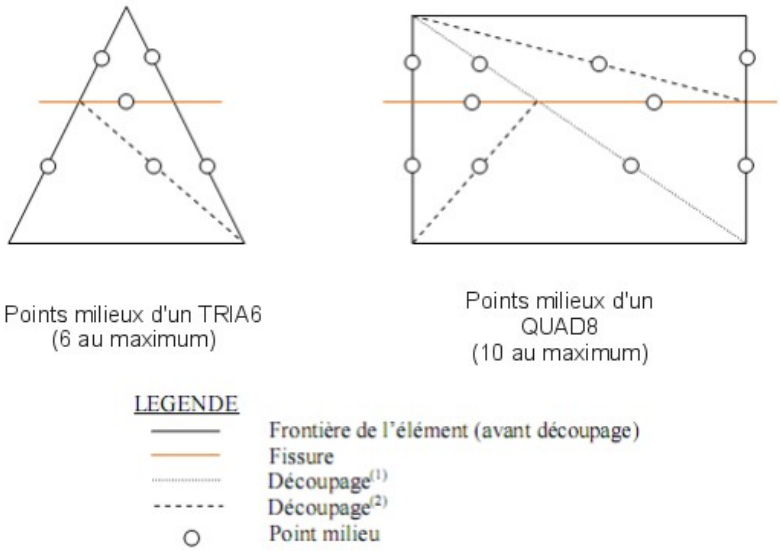
Renseigne sur les POints Milieux en présence de mailles quadratiques.

4.3.2.1. 2D

C'est un champ constant par élément (CHAM_ELEM) à 22 composantes réelles. Pour chaque élément, ce champ comporte :

- les coordonnées des points milieux

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



Remarque : o n rappelle que les coordonnées des points milieux qui coïncident avec un nœud de l'élément parent ne sont pas stockées dans cette structure de données.

- les coordonnées du point central introduit lors du découpage(1) d'un QUAD8 :

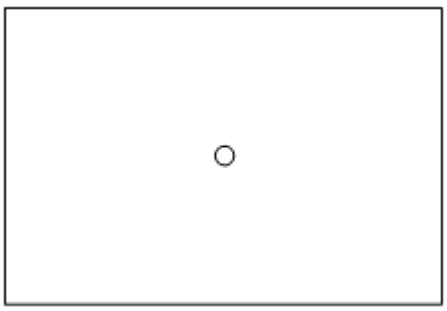


Figure 4.3.2.1-1: Point central d'un QUAD8(1 au maximum)

Cela fait un total de 11 points au maximum. Comme on se place en 2D, on peut avoir $11\times 2=22$ composantes au maximum. On dimensionne ainsi le champ à 22.

Pour chaque élément, les composantes sont ordonnées comme ceci :

$$\{x_1, y_1, x_2, y_2, \ldots, x_{11}, y_{11}\}$$

où $(x_i, y_i)_{1\leqslant i\leqslant 10}$ sont les coordonnées du i ème point milieu sur l'élément et où (x_{11}, y_{11}) sont les coordonnées du nœud central d'un QUAD8.

4.3.2.2. 3D

C'est un champ constant par élément (CHAM_ELEM) à 198 composantes réelles. Pour chaque élément, ce champ comporte :

- les coordonnées des points milieux (66 points au maximum),

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

- les coordonnées des points centraux introduits lors du découpage de l'élément quadratique 3D (7 points centraux supplémentaires au maximum, par exemple, lors du découpage d'un HEXA20 via l'élément complet HEXA27) .

Pour chaque élément, les composantes sont ordonnées comme ceci :

$$\{x_1, y_1, z_1, x_2, y_2, z_2, ..., x_{65}, y_{65}, z_{65}\}$$

4.3.3 .TOPOSE.CNS

Renseigne sur la connectivité des sous-éléments.

4.3.3.1. 3D

C'est un champ constant par élément (CHAM_ELEM) à 320 composantes entières . Ce champ est dimensionné par rapport au sous-découpage des éléments quadratiques en TETRA10. Pour, un sous-découpage linéaire, il est donc sur-dimensionné, sans conséquence notable sur le calcul et les performances.

Pour chaque élément, ce champ contient la connectivité des sous-éléments de l'élément. Par convention, tous les sous-éléments sont des tétraèdres à 10 nœuds (TETRA10), et il peut y en avoir au maximum 32 par élément X-FEM (pour une explication détaillée de ce résultat, voir la documentation de référence sur la méthode X-FEM [R7.02.12]). Il y a au maximum 6 tétraèdres et 6 sous-éléments par tétraèdre, mais il n'y a aucune configuration avec 6 sous-éléments pour chaque tétraèdre, c'est pourquoi le maximum est de 32. Les composantes sont rangées 4 par 4. L'objet .TOPOSE.CNS fait référence à 3 types de nœuds : soit des nœuds déjà existants du maillage, soit des points d'intersection, soit des nœuds milieux. Ainsi, pour chaque élément :
- si un sommet d'un sous-élément est un nœud de l'élément X-FEM en question, alors ce nœud est repéré par son numéro local dans l'élément;
- si un sommet est un point d'intersection, alors ce point est repéré par un nombre supérieur à 1000, où le chiffre des unités renvoie à son numéro local dans la liste des points d'intersection de l'élément.
– concernant les nœuds milieux : si le nœud milieu est un nœud de l'élément parent, on stocke le numéro local du nœud dans l'élément parent ; s'il s'agit d'un nouveau nœud milieu généré par la découpe, le point est alors repéré par un nombre supérieur à 2000. Si le point milieu est aussi est un point central de l'élément parent, il est alors repéré par un nombre supérieur à 3000.

Exemple :

Pour l'élément ie :

CNS(ie) =

1001	1005	6	1002	2001	12	2005	3012	3021	2004	1003	6	...

1er sous-tétra : nœud n°1 : 1001 -> renvoie au 1er point de PIN(ie)
nœud n°2 : 1005 -> renvoie au 5ème point de PIN(ie)
nœud n°3 : 6 -> renvoie au 6ème nœud de l'élément ie
nœud n°4 : 1002 -> renvoie au 2ème point de PIN(ie)
nœud n°5 : 2001 -> renvoie au 1ème point de MIL(ie)
nœud n°6 : 12 -> renvoie au 12ème nœud de l'élément ie
nœud n°7 : 2005 -> renvoie au 5ème point de MIL(ie)
nœud n°8 : 3012 -> renvoie au 12ème point de MIL(ie)
nœud n°9 : 6 -> renvoie au 6ème nœud de l'élément ie
nœud n°10 : 3021 -> renvoie au 21ème point de MIL(ie)

2ème sous-tétra : sommet n°1 : 1003 -> renvoie au 3ème point de PIN(ie)
sommet n°2 : 6 -> renvoie au 6ème nœud de l'élément ie
. . .

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

4.3.3.2. 2D

C'est un champ constant par élément (CHAM_ELEM) à 36 composantes entières. Ce champ est dimensionné par rapport au sous-découpage des éléments quadratiques en TRIA6.

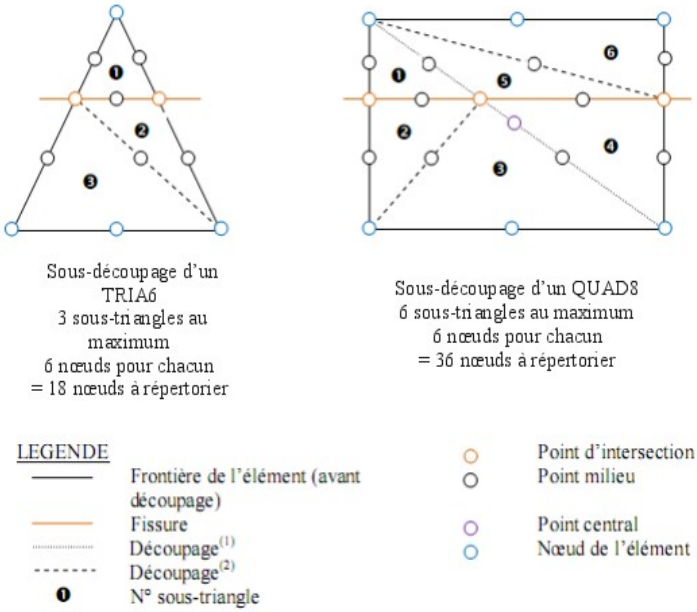
Pour chaque élément, il contient la connectivité des sous- éléments de l'élément.

On dénombre :

– 6 sous-triangles au maximum (atteint pour une maille quadrangle)

– 6 nœuds au maximum pour chaque sous-triangle (atteint pour une maille quadratique)

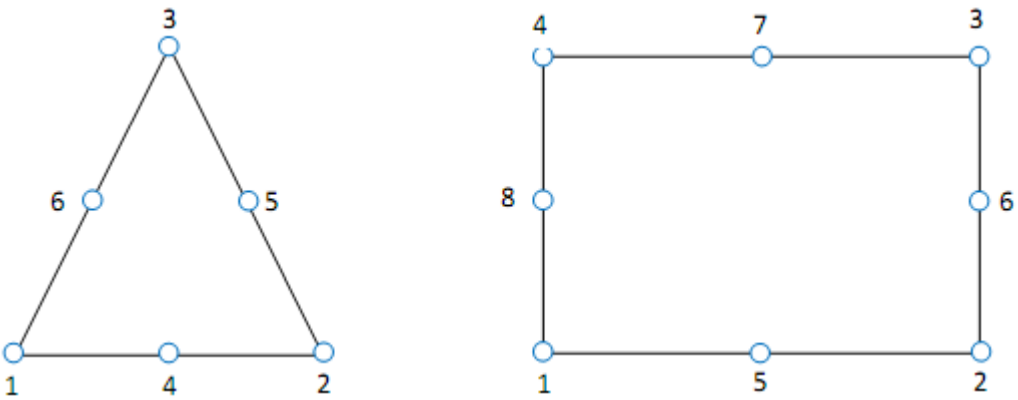
Ce qui fait $6 \times 6 = 36$ composantes au maximum. Le champ est donc dimensionné à 36 et l'on range ces dernières 6 par 6.



L'objet .TOPOSE.CNS fait référence à 4 types de points :

- Type 1 : Nœud de l'élément (appartenant au maillage)

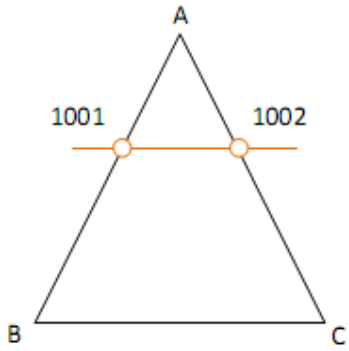
Il est repéré par son numéro local dans l'élément

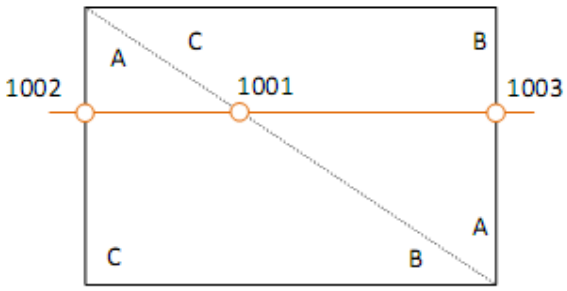


Document diffuse sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

○ Type 2 : Point d'intersection

Il est repéré par un nombre $1000+u$, où u renvoie à son numéro local dans la liste des points d'intersection de l'élément.



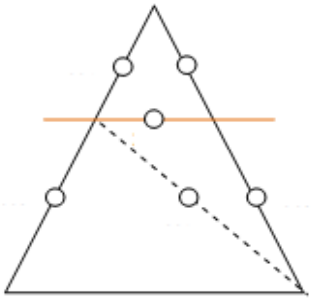


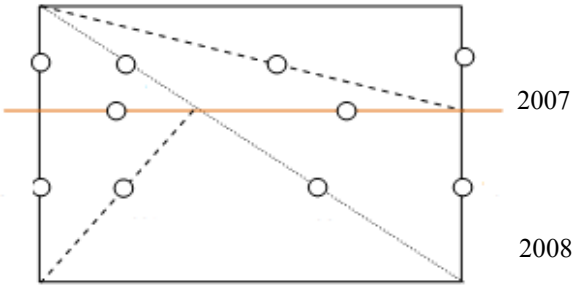
100**1** → 1er point stocké dans PINTTO

1002 → 2eme point stocké dans PINTTO…

○ Type 3 : Point milieu

Il est repéré par un nombre $2000+u$, où u renvoie à son numéro local dans la liste des points milieux de l'élément.



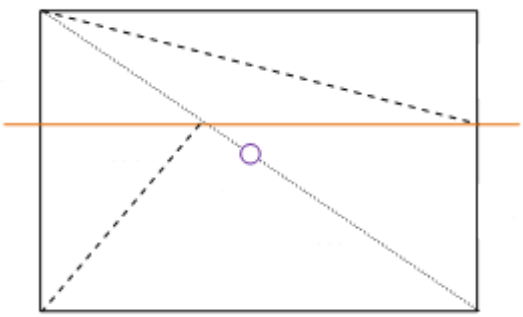


200 **1** → 1er point stocké dans PMILTO

200 **2** → 2ème point stocké dans PMILTO, etc.

○ Type 4 : Point milieu quadrangle

Il est repéré par un nombre $3000+u$, où u renvoie à son numéro local dans la liste des points milieux de l'élément.



30**11** → 11ème point stocké dans PMILTO

4.3.3.3. Cas multiheaviside

Pour les modélisations multiheaviside, on n'autorise qu'un sous-découpage linéaire. On utilise alors les dimensions des tableaux de connectivité du découpage linéaire (3D : TETRA4 au lieu de TETRA10 / 2D : TRIA3 au lieu de TRIA6).

Les dimensions des tableaux de connectivité monoheaviside sont alors multipliées par le nombre maximum de fissures :
En 3D, $128\times4=512$ composantes entières.
En 2D, $18\times4=72$ composantes entières.

4.3.4 .TOPOSE.HEA

Renseigne sur la valeur de la fonction HEAviside.
C'est un CHAM_ELEM constant par sous-élément à 32 composantes entières correspondant à la valeur de la fonction Heaviside sur les sous-tétraèdres (+1 ou –1), rangées successivement.

Il peut y avoir des 0 si le nombre total de sous-éléments est < 32 (nombre max de sous-tétraèdres).

En 2D, les sous-éléments sont au nombre maximum de 6, le champ TOPOSE.HEA est donc dimensionné à 6.

Dans le cas ou l'élément voit plusieurs fissures, ce champ est dupliqué autant de fois qu'il y a de fissures. On utilise alors le nombre de sous points $NBSP$ définit au [§4.2.1]. Le champ sera de $32\times NBSP$ composantes en 3D et de $6\times NBSP$ composantes en 2D.

4.3.5 .TOPOSE.LON

Renseigne sur la valeur de la LONgueur de champs utilisés pour définir les sous-éléments.
C'est un champ constant par élément (CHAM_ELEM) à une composante entière, en 2D comme en 3D.
Cette composante correspond au nombre de sous-éléments contenus dans l'élément X-FEM.

4.3.6 .TOPOSE.PAI

Renseigne sur les arêtes intersectées. Cette structure de données est passée à l'option TOPOFA afin qu'elle ne soit pas recalculée pour générer la structure de données .TOPOFAC.AI
Sa structure est similaire à .TOPOFAC.AI :
C'est un champ constant par élément (CHAM_ELEM) à 15 ou 55 composantes réelles contenant des informations sur les arêtes intersectées.
Pour chacun des points d'intersection de chaque élément :
La 1ère composante est le numéro local de l'arête correspondante (0 si c'est un nœud sommet).
La 2ème composante est le numéro local du nœud si c'est un nœud sommet (0 sinon).
La 3ème composante est la longueur de l'arête.
La 4ème composante est la position du point d'intersection sur l'arête, avec un sens arbitraire qui dépend de la maille. Ce réel est strictement compris entre 0 et 1, et vaut 0 si le point d'intersection est un nœud sommet.
La 5 ème composante permet de savoir si l'arête intersectée est vitale ou non. Si elle est vitale, la composante vaut 1, si elle ne l'est pas, la composante vaut 0.

Le nombre de composantes par point d'intersection (5) est accessible dans le fortran par un appel à la fonction XXMMVD('ZXAIN')

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

Soit le ème élément du maillage,

```
V =.TOPOFAC.AI(i);
```

V(1)	Numéro local de l'arête correspondant au 1er point d'intersection
V(2)	Numéro local du nœud correspondant au 1er point d'intersection
V(3)	Longueur de l'arête correspondant au 1er point d'intersection
V(4)	Position du 1er point d'intersection sur l'arête
V(5)	1er point d'intersection: si sur arête vitale (1) sinon (0)
V(6)	Numéro local de l'arête correspondant au 2ème point d'intersection
…	…
V(55)	11ème point d'intersection vitale (1) ou non (0)

Si le nombre de points d'intersection est strictement inférieur au nombre maximal de points d'intersection (3 ou 11 suivant les cas), le vecteur est complété par des 0.

En 2D, on a au maximum 3 points avec de la même façon qu'en 3D 5 informations (numéro de l'arête, numéro du nœud, longueur de l'arête et position du point d'intersection, arête intersectée vitale ou non). Le champ TOPOFAC.AI a donc 15 composantes par élément. En 3D on a 55 composantes.

4.3.7 .TOPOSE.PJO

Cette SD n'est utilisée que pour les éléments multi-Heaviside HM-XFEM cohésifs. Il s'agit d'un CHAM_ELEM . qui comporte une composante entière par nœud de l'élément. Il a donc jusqu'à 20 composantes. Cette composante vaut 0 si le nœud ne contient pas de jonction de fissure sur son support et 1 sinon. Cette information est ensuite utilisée pour imposer la continuité de la pression de fluide dans les branches d'interfaces au niveau d'une jonction de fissures hydrauliques (confer [R7.02.18]).

4.4 Objets relatifs à la topologie des facettes de contact

Les champs .TOPOFAC.PI, .TOPOFAC.AI, .TOPOFAC.CF, .TOPOFAC.LO et .TOPOFAC.BA, sont dupliqués par élément autant de fois que le nombre de fissures vues par la maille. On utilise pour cela le nombre de sous-points qui correspond à n_{fiss} .

4.4.1 .TOPOFAC.LO

Renseigne sur la LOngueur des champs utilisés pour le contact.
C'est un champ constant par élément (CHAM_ELEM) à 3 composantes entières, contenant le nombre de points d'intersection (NINTER), le nombre de facettes triangulaires de contact (NFACE) ainsi que le nombre de points (nœuds sommets et nœuds milieux) par facette triangulaire de contact (NPTF).

4.4.2 .TOPOFAC.PI

Renseigne sur les Points d'Intersection.
C'est un champ constant par élément (CHAM_ELEM) qui comporte jusqu'à 102 composantes réelles, qui sont les coordonnées **dans l'espace de référence parent** des points d'intersection des arêtes de l'élément avec la fissure.
En 2D, on a jusqu'à 7 points d'intersection (en comptant les nœuds milieux et les nœuds sommets) donc le champ est dimensionné à 14. En 3D, on en a jusqu'à 34 donc le champ est dimensionné à 102.
Pour les éléments Heaviside, il y a au maximum 18 points d'intersection en 3D, donc le champ est dimensionné à 54, et 8 points d'intersection en 2D, donc le champ est dimensionné à 16.

Soit i le i ème élément du maillage,

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

```
V =.TOPOFAC.PI(i);
```

V(1)	Coordonnée de référence suivant x du 1er point d'intersection
V(2)	Coordonnée de référence suivant y du 1er point d'intersection
V(3)	Coordonnée de référence suivant z du 1er point d'intersection
V(4)	Coordonnée de référence suivant x du 2ème point d'intersection
V(5)	Coordonnée de référence suivant y du 2ème point d'intersection
…	…
V(102)	Coordonnée de référence suivant z du 34ème point d'intersection

Si le nombre de points d'intersection est strictement inférieur au nombre maximal de points d'intersection (7, 8, 18 ou 34 suivant les cas), le vecteur est complété par des 0. Le nombre de points d'intersection est contenu dans le vecteur .TOPOFAC.LO (voir §4.4.1).

4.4.3 .TOPOFAC.AI

Renseigne sur les Arêtes Intersectées.
C'est un champ constant par élément (CHAM_ELEM) à 35 ou 170 composantes réelles contenant des informations sur les arêtes intersectées.
Pour chacun des 7 ou 34 points d'intersection de chaque élément :
La 1ère composante est le numéro local de l'arête correspondante (0 si c'est un nœud sommet).
La 2ème composante est le numéro local du nœud si c'est un nœud sommet (0 sinon).
La 3ème composante est la longueur de l'arête.
La 4ème composante est la position du point d'intersection sur l'arête, avec un sens arbitraire qui dépend de la maille. Ce réel est strictement compris entre 0 et 1, et vaut 0 si le point d'intersection est un nœud sommet.
La 5 ème composante permet de savoir si l'arête intersectée est vitale ou non. Si elle est vitale, la composante vaut 1, si elle ne l'est pas, la composante vaut 0.

Le nombre de composantes par point d'intersection (5) est accessible dans le fortran par un appel à la fonction XXMMVD('ZXAIN')

Soit i le i ème élément du maillage,

```
V =.TOPOFAC.AI(i);
```

V(1)	Numéro local de l'arête correspondant au 1er point d'intersection
V(2)	Numéro local du nœud correspondant au 1er point d'intersection
V(3)	Longueur de l'arête correspondant au 1er point d'intersection
V(4)	Position du 1er point d'intersection sur l'arête
V(5)	1er point d'intersection: si sur arête vitale (1) sinon (0)
V(6)	Numéro local de l'arête correspondant au 2ème point d'intersection
…	…
V(170)	34ème point d'intersection vitale (1) ou non (0)

Si le nombre de points d'intersection est strictement inférieur au nombre maximal de points d'intersection (7 ou 34 suivant les cas), le vecteur est complété par des 0.

En 2D, on a au maximum 7 points (4 points d'intersection et 4 points milieu) avec de la même façon qu'en 3D 5 informations (numéro de l'arête, numéro du nœud, longueur de l'arête et position du point d'intersection, arête intersectée vitale ou non). Le champ TOPOFAC.AI a donc 35 composantes par élément. En 3D, on a jusqu'à 34 points d'intersection, le champ TOPOFAC.AI a donc 170 composantes par élément.

4.4.4 .TOPOFAC.CF

Renseigne sur la Connectivité des Facettes de contact.

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

C'est un champ constant par élément (CHAM_ELEM) à 9 ou 90 composantes entières, contenant la connectivité des sommets des facettes triangulaires. Il y a au maximum 15 facettes par élément.

Soit i le $i^{\text{ème}}$ élément du maillage,

```
V =.TOPOFAC.CF(i);
```

V(1)	Numéro local du 1er sommet de la 1ère facette triangulaire
V(2)	Numéro local du 2ème sommet de la 1ère facette triangulaire
V(3)	Numéro local du 3ème sommet de la 1ère facette triangulaire
V(4)	Numéro local du 1er sommet de la 2ème facette triangulaire
V(5)	Numéro local du 2ème sommet de la 2ème facette triangulaire
…	…
V(90)	Numéro local du 3ème sommet de la 15ème facette triangulaire

Si le nombre de facettes triangulaires est strictement inférieur à 15, le vecteur est complété par des 0.

En 2D linéaire, on a jusqu'à 3 facettes de contact, qui sont des segments avec deux sommets et éventuellement un nœud milieu. Le champ TOPOFAC.CF a donc 9 composantes par élément. En 3D, on a jusqu'à 15 facettes qui sont des triangles avec 3 nœuds sommets et éventuellement 3 nœuds milieux. Le champ TOPOFAC.CF a donc 90 composantes par élément.

4.4.5 .TOPOFAC.BA

Renseigne sur la BAse covariante des facettes de contact.
C'est un champ par élément (CHAM_ELEM) à 28 ou 306 composantes réelles qui sont les coordonnées des vecteurs de la base covariante (n, τ_1, τ_2) pour chacun des 7 ou 34 points d'intersection.
Pour chaque élément (l'exposant faisant référence au numéro du point d'intersection) :

$$\left\{n_x^1, n_y^1, n_z^1, \tau_{1x}^1, \tau_{1y}^1, \tau_{1z}^1, \tau_{2x}^1, \tau_{2y}^1, \tau_{2z}^1, n_x^{2,} \ldots, \tau_{2z}^7\right\}$$

Soit i le $i^{\text{ème}}$ élément du maillage, et j le numéro du point d'intersection.

```
V =.TOPOFAC.BA(i);
```

V(9*(j-1)+1)	Coordonnée suivant x du 1er vecteur de la base au $j^{\text{ème}}$ point d'intersection
V(9*(j-1)+2)	Coordonnée suivant y du 1er vecteur de la base au $j^{\text{ème}}$ point d'intersection
V(9*(j-1)+3)	Coordonnée suivant z du 1er vecteur de la base au $j^{\text{ème}}$ point d'intersection
V(9*(j-1)+4)	Coordonnée suivant x du 2ème vecteur de la base au $j^{\text{ème}}$ point d'intersection
V(9*(j-1)+5)	Coordonnée suivant y du 2ème vecteur de la base au $j^{\text{ème}}$ point d'intersection
V(9*(j-1)+6)	Coordonnée suivant z du 2ème vecteur de la base au $j^{\text{ème}}$ point d'intersection
V(9*(j-1)+7)	Coordonnée suivant x du 3ème vecteur de la base au $j^{\text{ème}}$ point d'intersection
V(9*(j-1)+8)	Coordonnée suivant y du 3ème vecteur de la base au $j^{\text{ème}}$ point d'intersection
V(9*(j-1)+9)	Coordonnée suivant z du 3ème vecteur de la base au $j^{\text{ème}}$ point d'intersection

Si le nombre de points d'intersection est strictement inférieur au nombre maximal de points d'intersection (7 ou 34 suivant les cas), le vecteur est complété par des 0.

En 2D, la base covariante est constituée de 2 vecteurs (n, τ_1) ayant chacun 2 composantes et il y a jusqu'à 7 points d'intersection. Le nombre de composantes est donc 28. En 3D, la base covariante est

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

constituée de 3 vecteurs (n, τ_1, τ_2) ayant chacun 3 composantes et il y a jusqu'à 34 points d'intersection. Le nombre de composantes est 306.

4.4.6 TOPOFAC.HE

Renseigne sur la valeur des fonctions HEaviside de part et d'autre de la facette de contact. Cette SD n'est utile que dans le cas d'éléments multi-Heaviside. Elle n'est pas créée si le modèle ne possède pas de tels éléments. Soit n_{fiss} le nombre de fissures vues par la maille. Le nombre de sous-points pour cet objet est alors n_{fiss}^2 (pour chaque fissure, on va vouloir connaître les valeurs venant des autres fissures). Le nombre de composantes est de $2 \times n_{face}$ avec n_{face} le nombre maximum prédéfini de facettes de contact soit $n_{face}=1$ en 2D et $n_{face}=5$ en 3D.

Soit i le $i^{\text{ème}}$ élément du maillage, j le numéro de facette pour la fissure k

V =.TOPOFAC.HE(i);

V(2*nface*(nfiss*(k-1)+1-1)+2*(j-1)+1)	Valeur de la fonction Heaviside de la fissure 1 du côté esclave de la facette
V(2*nface*(nfiss*(k-1)+1-1)+2*(j-1)+2)	Valeur de la fonction Heaviside de la fissure 1 du côté maître de la facette
V(2*nface*(nfiss*(k-1)+2-1)+2*(j-1)+1)	Valeur de la fonction Heaviside de la fissure 2 du côté esclave de la facette
V(2*nface*(nfiss*(k-1)+2-1)+2*(j-1)+2)	Valeur de la fonction Heaviside de la fissure 2 du côté maître de la facette
...	
V(2*nface*(nfiss*(k-1)+n_{fiss}-1)+2*(j-1)+1)	Valeur de la fonction Heaviside de la fissure n_{fiss} du côté esclave de la facette
V(2*nface*(nfiss*(k-1)+n_{fiss}-1)+2*(j-1)+2)	Valeur de la fonction Heaviside de la fissure n_{fiss} du côté maître de la facette

Cette SD est utile pour calculer le saut de déplacement entre les côtés maître et esclave dans le cas des éléments multi-Heaviside. Ultérieurement, cette SD sera transformée en TOPONO.HFA pour faciliter l'identification des 2 domaines jouxtant les sous-facettes. Cette identification permettra alors l'évaluation des fonctions caractéristiques de domaines associées à la nouvelle approximation du saut de déplacement, comme préciser dans la documentation de référence [R7.02.12].

4.4.7 .TOPOFAC.OE

Il s'agit d'un champ dont la structure informatique est rigoureusement la même que TOPOFAC.PI. Comme ce dernier il contient les coordonnées des points d'intersections dans l'espace réel, mais **dans la configuration initiale**. Ce champ sert dans le cadre de l'approche grands glissements avec X-FEM.

4.4.8 .TOPOFAC.GE

Il s'agit d'un champ dont la structure informatique est rigoureusement la même que TOPOFAC.OE. Comme ce dernier il contient les coordonnées des points d'intersections dans l'espace réel, mais **actualisées selon la cinématique des facettes esclaves**. Ce champ sert dans le cadre de l'approche grands glissements avec X-FEM.

4.4.9 .TOPOFAC.GM

Il s'agit d'un champ dont la structure informatique est rigoureusement la même que TOPOFAC.OE. Comme ce dernier il contient les coordonnées des points d'intersections dans l'espace réel, mais **actualisées selon la cinématique des facettes maîtres**. Ce champ sert dans le cadre de l'approche grands glissements avec X-FEM.

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

4.5 Objets relatifs à la définition des fonctions caractéristiques de domaines

4.5.1 TOPONO.HNO

Ce champ ELNO contient les codes « entiers » représentant les numéros identifiants de domaine par nœud et par élément. Chaque nœud enrichi Heaviside peut représenter jusqu'à 4 fissures au maximum par élément. On montre de manière incrémentale, que 4 fissures/jonctions définissent au maximum 5 domaines par élément.

Pour décrire donc l'environnement de chaque nœud X-FEM (Heaviside), on alloue donc 5 emplacements par nœud et par élément, dans le tableau TOPONO.HNO(NNO*5) , rempli de la manière suivante :

```
V =.TOPONO.HNO(i);
```

V(1)	Code entier représentant le 1er domaine vu par le 1er nœud de l'élément
V(2)	Code entier représentant le 2ème domaine vu par le 1er nœud de l'élément
V(3)	Code entier représentant le 3ème domaine vu par le 1er nœud de l'élément
V(4)	Code entier représentant le 4ème domaine vu par le 1er nœud de l'élément
V(5)	Code entier représentant le 5ème domaine vu par le 1er nœud de l'élément
V(6)	Code entier représentant le 1er domaine vu par le 1er nœud de l'élément
V(7)	Code entier représentant le 2ème domaine vu par le 2ème nœud de l'élément
V(8)	Code entier représentant le 3ème domaine vu par le 2ème nœud de l'élément
V(9)	Code entier représentant le 4ème domaine vu par le 2ème nœud de l'élément
V(10)	Code entier représentant le 5ème domaine vu par le 2ème nœud de l'élément
…	…
V(5*N)	Code entier représentant le 5ème domaine vu par le Nème nœud de l'élément

Par ailleurs, concernant l'organisation du stockage des 5 domaines vus par chaque nœud: les 4 premiers emplacements sont réservés aux domaines « complémentaires » du domaine auquel appartient le nœud, le dernier emplacement est réservé au domaine auquel appartient le nœud.

Quand la fissure passe par le nœud, la sélection du domaine auquel appartient le nœud, s'opère automatiquement grâce à la fonction « signe ». Par convention on a : $He(lsn=0)=sign(0)=1$. Cette convention permet de prolonger par continuité le codage des domaines sur le bord des éléments et par conséquent aux nœuds.

Rappelons que le codage des domaines se fait par projection en base 4 des fonctions signe Heaviside au point **P** courant (soit un nœud, soit un point de gauss):

$$code(\underline{P})=\sum_{ifiss=1}^{nfiss} 4^{nfiss-ifiss}\left(He(\underline{P},ifiss)+2\right)$$

A l'aide des SD .LNNO et .TOPOSE.HEA , pour chaque nœud « Heaviside », on calcule alors le code du domaine auquel appartient le nœud et le code des domaines vus par le nœud (partitionnés dans les sous-éléments).

Par ailleurs, étant donnée la construction locale du champ concaténé .LNNO , un même domaine peut avoir plusieurs codes d'un élément à un autre, car un nœud peut voir un nombre différents de fissures d'un élément à un autre (voire figure 4.2.3-3). Cela entraîne un défi notable dans l'organisation du champ TOPONO.HNO pour que le codage garde une signification non locale, en clair, pour que l'information TOPONO.HNO(jno,i) représente toujours l'intersection du support du nœud n ° jno , avec le domaine Ω_i quel que soit l'élément considéré.

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

Pour contourner cette difficulté, nous définissons plutôt des domaines polymorphes. En d'autres mots, un même domaine pourra avoir un code différent d'un élément Heaviside à un autre. Du coup, le code TOPONO.HNO(jno,i) représente l'intersection du support du nœud n ° jno avec un domaine Ω_i , «vu de l'élément courant» .

Cette construction « élémentaire » est alors en adéquation avec l'ensemble des calculs élémentaires, y compris pour les mailles tardives [R5.03.53].

4.5.2 TOPONO.HSE

Ce champ élémentaire contient un code « entier » par sous-élément X-FEM. Son dimensionnement s'adosse alors sur le nombre de sous-élément issus du sous-découpage cf. TOPOSE.LON .

Cette SD représente le codage du champ de signe Heaviside par sous-élément .TOPOSE.HEA . Elle permet le calcul des fonctions caractéristiques de domaines aux points de Gauss volumiques.
En effet, chaque point de Gauss XFEM est localisé dans un sous-élément. Le sous-élément est lui-même une sous-partition d'un domaine. Assigner alors un code par sous-élément suffit pour caractériser toutes les partitions de domaines sur l'élément courant.

On alloue alors autant d'emplacements entiers qu'il y a de sous-élément dans l'élément courant TOPONO.HSE(NSE) .

```
V =.TOPONO.HSE(i);
```

V(1)	Code entier représentant la partition de domaine auquel appartient le 1er sous-élément
V(2)	Code entier représentant la partition de domaine auquel appartient le 2ème sous-élément
V(3)	Code entier représentant la partition de domaine auquel appartient le 3ème sous-élément
…	…
V(N)	Code entier représentant la partition de domaine auquel appartient le Nème sous-élément

4.5.3 TOPONO.HFA

Ce champ élémentaire contient 2 codes « entiers» par sous-facette X-FEM. Le premier code représente le domaine esclave. Le deuxième code représente le domaine maître.

Ces deux codes sont calculés avec la même formule de codage que les SD « .TOPONO » précédentes. En revanche, le codage ici s'appuie sur le champ de signe Heaviside TOPOFAC.HE . Comme ce dernier champ n'est présent qu'en multi-Heaviside, le champ TOPONO.HFA n'est créé uniquement qu'en multi-Heaviside. Étendre ce champ en mono-Heavisisde ne présente pas de difficulté majeure et améliorerait la lisibilité du code. Pour le moment, dans ce cas, le code TOPONO.HFA est calculé à partir du signe de la fonction Heaviside de part et d'autre de la facette soit ±1.

Le dimensionnement du champ TOPONO.HFA s'adosse alors sur le nombre de sous-facettes issues du sous-découpage (cf. TOPOFAC.LO) et sur le nombre de fissures vues par l'élément, puisqu'une sous-facette est un partitionnement d'une fissure au sein de l'élément courant.

On a alors:

```
V =.TOPONO.HFA(i);
```

V(2*nface*(ifiss-1)+2*ifa-1) Code du domaine du côté esclave de la fissure ifiss de la

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

	sous-facette n° ifa
V(2*nface*(ifiss-1)+2*ifa)	Code du domaine du côté maître de la fissure ifiss de la sous-facette n° ifa

4.6 Autres objets

4.6.1 .XFEM_CONT

Vecteur d'entiers de longueur égale à 1. Cette valeur vaut :
– 0 si il n'y a pas de contact ;
– 1 si une relation de contact ou une relation cohésive a été définie entre les lèvres de la fissure, et que des éléments finis linéaires sont utilisés avec une formulation standard pour le contact (CONTACT='STANDARD' dans MODI_MODELE_XFEM) ;
– 2 si une relation de contact ou une relation cohésive a été définie entre les lèvres de la fissure, et que des éléments linéaires sont utilisés avec une formulation « mortier » pour le contact (CONTACT='MORTAR' dans MODI_MODELE_XFEM) ;
– 3 si une relation de contact ou une relation cohésive a été définie entre les lèvres de la fissure, et que des éléments quadratiques sont utilisés avec une formulation standard pour le contact.

4.6.2 .NFIS

Vecteur d'entiers de longueur égale à 1. Il correspond au nombre de fissures X-FEM du modèle.

4.6.3 .FISS

Vecteur de K8 de longueur égale au nombre de fissures X-FEM du modèle. Il correspond au nom de la sd_fiss_xfem de chaque fissure.

4.6.4 .XMAFIS

CHAM_ELEM de K8. Pour chaque maille, ce CHAM_ELEM correspond au nom des fissures vues par cette maille, dans l'ordre défini par l'utilisateur. Le nombre de sous-points de cette SD correspond au nombre de fissures vues par la maille.

4.6.5 .PRE_COND

Chaîne de caractères longueur 8 [K8]. Pour chaque modèle XFEM, elle renseigne sur l'activation du pré-conditionnement algébrique XFEM. Ce pré-conditionnement est indispensable avec les éléments quadratiques.

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

5 Description des objets liés à la résolution du problème de contact-frottant

5.1 Objets liés à la SD créée lors de la définition de la charge de contact-frottant (defico)

La description complète et détaillée des objets créés lors de la définition de la charge de contact, par l'opérateur DEFI_CONTACT, fait l'objet du fascicule [D4.06.14]. Afin d'aider le lecteur du présent document à appréhender le fascicule [D4.06.14], une liste des objets spécifiques à X-FEM décrits dans [D4.06.14, §8] est donnée ci-après :

- **MODELX** , cet objet stocke une référence vers le modèle enrichi dans la sd_contact ;
- **CARAXF** , cet objet sert à définir les différents paramètres pour chaque zone de contact, *i.e* . chaque fissure maître ;
- **XFIMAI** , cet objet stocke le nom de la fissure maître associée à chaque zone de contact ;
- **XNRELL** , cet objet stocke l'objet .LISEQ (cf. § 3.5.2) associé à chaque fissure maître ;
- **MAESCX** , cet objet stocke les information s relatives aux mailles esclaves dans le but de réaliser l'appariement, dans le cas de l'approche grands glissements.

5.2 Objets liés à la SD créée lors de la résolution du problème de contact-frottant, dans le cadre l'approche grands glissements (resoco)

Il s'agit des cartes de contact grands glissement X-FEM qui sont construites une fois l'appariement fait entre les points d'intégrations des éléments esclaves et maîtres. Pour chaque appariement, un élément tardif est créé dans l'opérateur STAT_NON_LINE . On stocke dans ces différents champs les informations relatives à ces éléments tardifs utiles lors de l'intégration des contributions de contact-frottement.

5.2.1 '.XFPO'

Carte de R qui stocke pour chaque élément tardif des informations sur l'appariement (voir les commentaires dans la routine XMCART pour plus de détails).

5.2.2 '.XFST'

Carte de I qui stocke dans l'ordre des nœuds de l'élément tardif les statuts d'enrichissement des nœuds des éléments esclave et maître de l'élément tardif. Elle correspond à une copie de la SD MODELE//'.STNO' pour les éléments esclave et maître de l'élément tardif.

5.2.3 '.XFPI'

Carte de R qui stocke les coordonnées locales des points d'intersection de l'élément esclave. Elle correspond à une copie de la SD MODELE//'.TOPOFAC.PI' pour l'élément esclave de l'élément tardif.

5.2.4 '.XFAI'

Carte de R qui stocke les informations sur les arêtes intersectées de l'élément esclave. Elle correspond à une copie de la SD MODELE//'.TOPOFAC.AI' pour l'élément esclave de l'élément tardif.

5.2.5 '.XFCF'

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

Carte de I qui stocke les informations sur la connectivité des facettes de l'élément esclave. Elle correspond à une copie de la SD MODELE//'.TOPOFAC.CF' pour l'élément esclave de l'élément tardif.

5.2.6 '.XFHF'

Carte de I qui stocke la valeur des fonctions Heaviside de part et d'autre de la discontinuité, dans le cas où l'élément esclave ou maître est multi-Heaviside. Elle correspond à une copie de la SD MODELE//'.TOPOFAC.HE' côté esclave de l'élément esclave et côté maître de l'élément maître pour l'élément tardif.

5.2.7 '.XFPL'

Carte de I qui stocke la place des Lagrange de contact aux nœuds de l'élément esclave, dans le cas où l'élément esclave ou maître est multi-Heaviside. Elle correspond à une copie de la SD MODELE//'.HEAVNO' pour la fissure associée à la zone de contact, pour l'élément esclave de l'élément tardif.

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