EDF R&D

FLUID DYNAMICS, POWER GENERATION AND ENVIRONMENT DEPARTMENT
SINGLE PHASE THERMAL-HYDRAULICS GROUP

6, QUAI WATIER
F-78401 CHATOU CEDEX

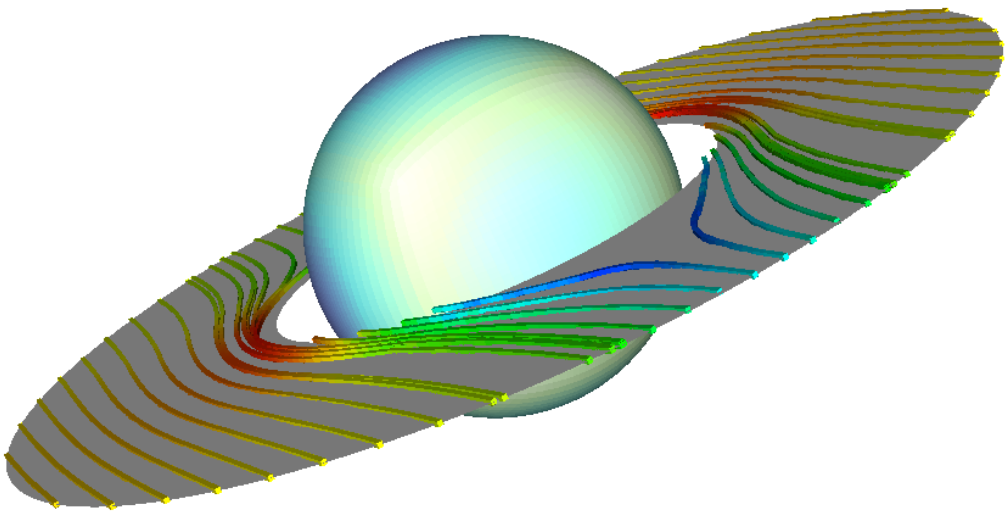
TEL: 33 1 30 87 75 40
FAX: 33 1 30 87 79 16

FEBRUARY 2011

Code_Saturne documentation

Code_Saturne version 1.3.2: guide pratique et théorique du Préprocesseur

contact: saturne-support@edf.fr



SYNTHÈSE

Le Préprocesseur de *Code_Saturne* facilite la mise en œuvre de cet outil, en permettant la lecture de maillages de plusieurs formats différents, en préparant le découpage du domaine de calcul et les structures pour la mise en œuvre du parallélisme, en permettant le recollement de maillages non conformes, et en permettant des données de type posttraitement à des fins de vérification. Cet outil combine un certain nombre d'algorithmes propres (liés notamment au recollement) avec l'encapsulation de librairies extérieures spécialisées (découpeur de graphes *Metis*, librairie *MED* fichier pour la lecture et écriture de maillages, etc.)

Le Préprocesseur est utilisé de manière transparente par les lanceurs de *Code_Saturne*. Ce document décrit des possibilités d'utilisation plus avancées de cet outil pour une préparation plus aisée d'une étude, et fournit quelques conseils d'utilisation. On y décrit brièvement les différents formats de maillage supportés, avec leurs avantages et inconvénients.

Ce document comporte aussi une partie théorique : on y décrit aussi les principes des algorithmes géométriques utilisés, permettant à l'utilisateur de mieux appréhender le fonctionnement du recollement de maillages et l'influence de certains paramètres.

SOMMAIRE

1 Présentation du Préprocesseur de *Code_Saturne*

Dans cette section, on explicitera le rôle de l'outil Préprocesseur dans *Code_Saturne*.

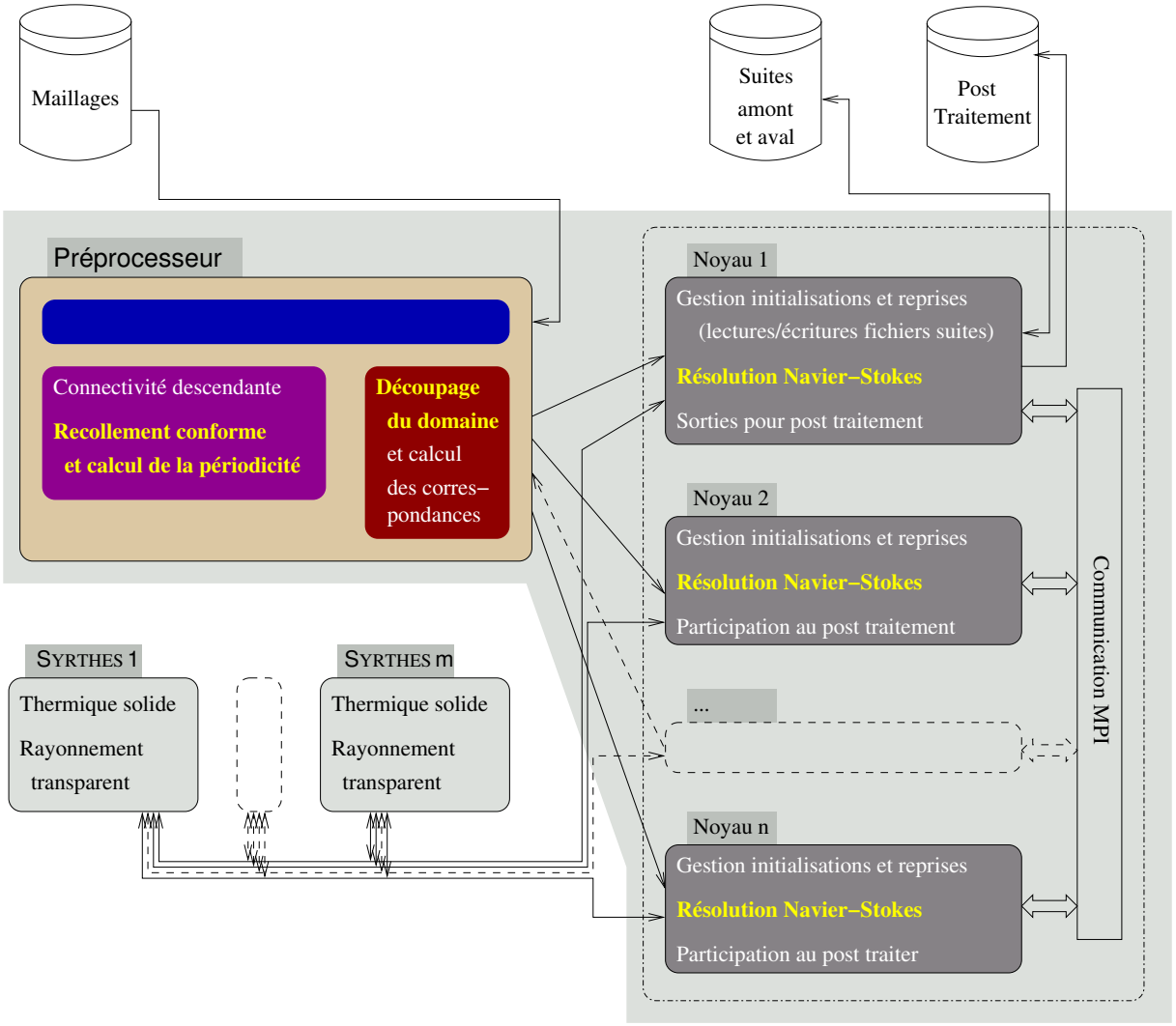


FIG. 1 – Role du Préprocesseur dans le *Code_Saturne*

La figure 1 illustre le rôle et les principales opérations réalisées par le Préprocesseur, dans le contexte d'un calcul parallèle (sur n processeurs) couplé à $0 \leq m$ instances du code SYRTHES.

On ne s'attardera pas ici sur le Noyau, qui est décrit en détail dans le guide pratique d'utilisation attaché à chaque version. On précisera simplement que la description géométrique du maillage à fournir en entrée du Noyau fait abstraction des types géométriques des éléments, considérant qu'il s'agit simplement de polyèdres à faces planes. Le plus souvent, les informations à fournir en entrée du Noyau se limitent à la donnée :

– de la connectivité « faces → cellules » ;
– de la connectivité « faces internes → sommets » ;
– de la connectivité « faces de bord → sommets » ;
– des coordonnées des sommets ;
– de familles[1] des faces de bord et des cellules.

Ce type de représentation géométrique du maillage n'est pas celle que fournissent les mailleurs : ils donnent le

[1] Description compactée des couleurs et groupes

plus souvent une description en connectivité nodale du maillage. De plus, elle n'est pas non plus adaptée aux formats usuels de post-traitement, et ne permet pas de reconstruire de manière triviale une connectivité nodale à cette fin.

La première fonctionnalité du Préprocesseur consiste donc à transformer la description géométrique du maillage issue d'un mailleur dans la représentation adaptée au Noyau. Le Préprocesseur peut lire plusieurs maillages (conformes ou non) et les concaténer automatiquement, puis réaliser à la demande de l'utilisateur leur recollement conforme aux surfaces sélectionnées (par découpage de faces non conformes en sous-faces conformes).

On notera que de plus, c'est le Préprocesseur qui assure à la demande la recherche des correspondances entre faces périodiques. Cette dernière fonctionnalité étant basée sur une extension du recollement conforme, il n'est pas nécessaire que les surfaces en vis-à-vis soit maillées de manière conforme.

En cas de calcul parallèle, le Préprocesseur fait appel à un algorithme de découpage de domaines (via une librairie dédiée à des découpages minimisant les tailles des interfaces entre domaines si disponible), et définit les correspondances entre numérotations globales et locales.

Finalement, des fonctionnalités de vérification sont disponibles, et comprennent :

– la production de maillages volumiques et de bord pour le post-traitement permettant la vérification des maillages fournis et des différents groupes ou références qu'ils définissent.
– la production de maillages surfaciques supplémentaires pour le post-traitement correspondant aux faces internes répondant à un critère de sélection défini par l'utilisateur (couleurs, groupes, ...) ;
– la production de maillages surfaciques supplémentaires pour le post-traitement afin de permettre la visualisation des faces issues de recollements conformes et de périodicités, ainsi que les faces modifiées par ces recollements et surtout les éventuelles faces sur lesquelles le recollement aurait échoué ;
– la production de maillages surfaciques pour le post-traitement correspondant au bord du domaine de calcul et aux faces isolées éventuelles (qui sont ignorées par le Noyau), ou qui appartiennent à plus de deux cellules (ce qui indique une erreur dans la connectivité) ;
– la sortie sur les maillages volumiques pour le post-traitment du numéro de domaine parallèle d'appartenance des cellules.

L'utilisation fréquente du Préprocesseur lors de la phase de mise au point d'un maillage est donc recommandée, ainsi que celle du Noyau en mode « vérification » pour obtenir le calcul de divers critères de qualité locaux (non-orthogonalité, pondération, non-planéité, ...).

2 Utilisation du Préprocesseur

Le Préprocesseur se contrôle via des arguments de ligne de commande, certains arguments acceptant des sous-arguments. Il est possible de compléter la ligne de commande par un fichier, utilisant la même syntaxe que celle des arguments de ligne de commande mais permettant aussi l'ajout de sauts à la ligne et de commentaires. Quelques variables d'environnement permettent à un développeur ou à un utilisateur expert de modifier quelques comportements du Préprocesseur ou d'obtenir un fichier trace de la gestion mémoire ou des temps CPU intermédiaires.

Les options et sous-options les plus utiles seront décrites brièvement dans ce document. Pour obtenir une liste complète et à jour des options et variables d'environnement, il suffit de taper la commande : `ecs -h` ou `ecs -help`. De nombreuses options, comme celle-ci, acceptent une version courte et une version plus explicite.

Si le chemin de l'exécutable `ecs` ne figure pas dans votre variable d'environnement `PATH`, vous pouvez utiliser le chemin complet (le Préprocesseur ne faisant pas appel à d'autres exécutables). À MFEE, le chemin « portable » est `/home/saturne/Enveloppe/ecs-{version}/bin/ecs`, ce script appelant un exécutable situé sous
`/home/saturne/Enveloppe/ecs-{version}/arch/{nom_arch}/bin/ecs`.
La première méthode est recommandée, car le script d'encapsulation de l'appel peut aussi positionner des variables d'environnement si nécessaire sous certaines architectures.[2]

[2]La plupart des compilateurs et éditeurs de liens acceptent une option du type `-Wl, -rpath -Wl,{chemin}` permettant de retrou-

2.1 Options et sous-options

Les principaux choix se font au moyen de paramètres de ligne de commande. Les paramètres correspondant à une sous-option doivent suivre directement celui activant l'option, et la définition des sous-options se termine au premier paramètre ne pouvant pas correspondre à cette sous-option. Certaines options peuvent être appliquées plusieurs fois. Par exemple, `-j` étant l'option qui déclenche le recollement conforme, `-fraction` étant la sous-option de recollement permettant de modifier les tolérances par défaut, et `-color` étant une sous-option de sélection de faces, et donc aussi une sous-option du recollement :

```
ecs -m essai.des -j -fraction 0.2 -color 2 -color 3 -j
```

signifie que l'on applique un recollement conforme aux faces de bord de couleur 2 ou 3 du maillage `essai.des` avec une tolérance de $0,2$, puis que l'on applique un second recollement à toutes les faces de bord restantes (avec les autres paramètres par défaut), alors que :

```
ecs -m essai.des -j -fraction 0.2 -color 2 -j -color 3
```

signifie que l'on applique un recollement conforme aux faces de bord de couleur 2 avec une tolérance de $0,2$, puis que l'on applique un second recollement aux faces de bord de couleur 3 (avec les autres paramètres par défaut).

2.1.1 Fichier d'options

Il est possible de compléter la ligne de commande par un fichier, utilisant la même syntaxe que celle des arguments de ligne de commande mais permettant aussi l'ajout de sauts à la ligne arbitraires et de commentaires. Tout ce qui suit un caractère # jusqu'à la fin de la ligne est ignoré dans un tel fichier. On indique l'utilisation d'un tel fichier via l'option `-i`. Si par exemple on a un fichier `ecs_cmd` contenant les lignes suivantes :

```
-ens # activation de la sortie EnSight
-med # activation de la sortie MED
```

alors la commande : `ecs -i ecs_cmd -m essai.unv`
équivaudra à : `ecs -m essai.unv -ens -med`

L'utilisation de fichiers d'options peut permettre de contourner les difficultés liées à la limitation de la longueur de la ligne de commande, plus ou moins rapidement atteinte selon les environnements. Elle permet aussi de conserver facilement des combinaisons d'options complexes et laborieuses à saisir.

2.1.2 Sélection de maillages

Toute utilisation du Préprocesseur nécessite un ou plusieurs maillages (hormis `ecs` et `ecs -h` qui affichent respectivement le num'ero de version et la liste des options). On les sélectionne au moyen de l'option `-mesh`, ou `-m`, suivi de la liste des maillages à prendre en compte. Le format de maillage est normalement déterminé automatiquement à partir de son extension (c.f. 4.2 page 13) mais cette option accepte aussi une sous-option `-format`, ou `-fmt`, permettant de forcer le choix du format des fichiers sélectionnés.

Pour les maillages au format CGNS, on peut de plus utiliser les sous-options `-grp-cel` ou `-grp-fac` suivies des mots clés `section` ou `zone` pour construire des groupes de cellules ou de faces supplémentaires selon l'organisation du maillage en sections ou zones. Ces sous-options n'ont pas d'effet sur des maillages d'autres formats.

On peut définir autant de sélections de maillages que l'on souhaite. Les maillages sont automatiquement concaténés, dans l'ordre de leur apparition sur la ligne de commande. Ainsi, la commande :

```
ecs -m fic.1 fic.2 -fmt ideas -m branche.cgns -grp-cel section
```

signifie que l'on lit les maillages des fichiers `fic.1` et `fic.2`, en précisant qu'ils sont au format *I-deas*, et

ver des librairies partagées en dehors des chemins système usuels, mais si une telle option n'est pas disponible ou que la librairie a été déplacée, on devra positionner la variable d'environnement système `LD_LIBRARY_PATH`.

que l'on ajoute le maillage décrit dans le fichier `branche.cgns`, sur lequel on construira de plus des groupes supplémentaires correspondant aux noms des sections CGNS auxquels les cellules appartiennent.

2.1.3 Sorties pour post-traitement

On n'a par défaut aucune sortie du maillage pour le post-traitement. En ajoutant une option `-ensight` (ou `-ens`), `-med`, ou `-cgns`, on provoque l'écriture du maillage concaténé au format indiqué. Il est possible d'effectuer des sorties simultanées sous plusieurs formats. On notera qu'on obtiendra des maillages surfaciques supplémentaires correspondant au bord du domaine de calcul, aux faces issues ou modifiées par les recollements ou les périodicités, etc. Les faces périodiques ne sont pas considérées comme faisant partie du bord du domaine.[3]

On considère quatre types de sorties : le maillage volumique, (avant recollements éventuels), le maillage de bord, les maillages d'information (tels que les faces issues ou modifiés par un recollement ou une périodicité, ou les faces internes sélectionnées), et les maillages correspondant à des zones avec des erreurs). Par défaut, tous les maillages sont sortis. Si l'on utilise une ou plusieurs des trois options de filtrage `-volume`, `-boundary`, et `-info`, on ne sortira que les maillages des types demandés, ainsi que les maillages correspondant à des faces zones avec des erreurs (non filtrables).

On peut de manière optionnelle provoquer le découpage des polygones en triangles en ajoutant la sous-option `-split-poly`. Ceci permet notamment de contourner des *bugs EnSight* apparaissant sur certains maillages relativement complexes, ou le non support des polygones par certains outils associés à CGNS. Dans ce cas, on construit pour chaque maillage surfacique un maillage dont les polygones sont découpés en triangles, et un maillage ne contenant que les arêtes du maillage non découpé.

Dans le cas d'une sortie au format *EnSight Gold*, on peut utiliser la sous-option `-simple` pour éviter la subdivision des maillages obtenus en plusieurs *parts* correspondant aux différents attributs (couleurs et groupes) portés par ce maillage. Dans le cas d'une sortie au format CGNS, cette même option force l'écriture du maillage en une seule section, quitte à utiliser une section de type « mixed » si nécessaire (au lieu de subdiviser le maillage en sections correspondant aux différents types d'éléments).

Finalement, dans le cas du format *EnSight Gold*, on peut forcer la sortie en mode texte via la sous-option `-text`, ou forcer la sortie binaire en mode « big endian » via la sous-option `-big-endian`. Par défaut, la sortie est de type binaire natif.

2.1.4 Sélection de faces

La sélection de faces intervient à plusieurs niveaux, notamment pour le recollement conforme.

Il est aussi possible de sélectionner un ensemble de faces internes portant des attributs (couleurs ou groupes) à seule fin de vérification au moyen de l'option `-int-face`. Le Préprocesseur affiche le nombre de faces internes portant des attributs et répondant au critère de sélection. Si de plus on a activé une sortie sur fichier pour post-traitement, un maillage surfacique correspondant aux faces sélectionnées sera généré.

On construit un critère de sélection au moyen des sous-options suivantes :

`-color` c_1 c_2 ... c_n	pour sélectionner des faces des couleurs indiquées par leurs numéros
`-groupe` g_1 g_2 ... g_n	pour sélectionner des faces des groupes indiqués par leurs noms
`-invsel`	pour prendre le complémentaire aux couleurs et groupes indiqués

Les mêmes sous-options de sélection s'appliquent pour le recollement conforme, la périodicité, et la vérification de faces internes. Dans le cas du recollement et de la périodicité, la sélection est automatiquement restreinte aux faces de bord, alors que dans le cas de la vérification de faces internes, elle est restreinte aux faces internes portant des attributs.

Ainsi, pour sélectionner toutes les faces internes du maillage `essai.des` qui portent un attribut (couleur ou groupe), mais ne sont pas de couleur 2 et n'appartiennent pas au groupe « entree », avec une sortie de *part EnSight Gold* (via `-ens`) et sans envoi de données au Noyau (via `-sc`, c.f. 5 page 19), on a la ligne de

[3] On interprète la périodicité comme étant une condition « géométrique » plutôt qu'une condition aux limites usuelle sur les variables.

commande suivante (le caractère \ marque la continuation de la ligne) :

```
ecs -m essai.des -sc -ens \
    -int-face -group entree -color 2 -invsel
```

2.1.5 Parallélisme

Pour activer le découpage de domaines nécessaire à un calcul parallèle, on utilise l'option `-parall` (ou `-p`), qui prend obligatoirement un argument entier correspondant au nombre de domaines voulus. Ainsi, pour un calcul sur 8 processeurs, on ajoutera à la ligne de commandes : `-p 8`.
On notera qu'une face sur une frontière entre deux domaines apparaîtra comme une face interne dans la définition de chacun de ces domaines, toujours avec la même orientation.

2.1.6 Recollement conforme

Le recollement conforme (cf. § 6.2, page 23) est une des fonctionnalités principales du Préprocesseur. Pour l'activer, on utilise l'option de ligne de commande `-join` ou `-j`, suivie éventuellement de critères de sélection de faces et de sous-options permettant de modifier les paramètres de tolérance associés. Pour un maillage simple, il est rarement utile de préciser des critères de sélection de faces, le recollement étant suffisamment automatisé pour détecter quelles faces peuvent effectivement être recollées. Pour un maillage plus complexe, ou comportant des parois minces que l'on veut éviter de faire disparaître au travers d'un recollement, il est conseillé de limiter les faces traitées par l'algorihme aux faces appartenant aux surfaces à recoller au moyen de sous-options de sélection de faces, ce qui a l'avantage supplémentaire de réduire le nombre de faces à traiter notamment lors de la phase de recherche d'intersections et donc améliorer les performances de l'algorithme.

On peut aussi modifier les critères de tolérance au moyen de deux sous-options :

`-fraction` r	permet d'affecter la valeur r (où $0 < r < 0,49$) au multiplicateur de distance maximale d'intersection ($0,1$ par défaut). La distance maximale d'intersection est basée sur la longueur de la plus petite arête connectée à chaque sommet, multipliée par r (cf. Figure 15, page 27) ;
`-plane` c	pour indiquer le cosinus minimal des normales pour que deux faces soient considérées coplanaires ($0,8$ par défaut) ; ce paramètre est utilisé dans la seconde phase de l'algorithme, pour la reconstruction de faces conformes (cf. Figure 11, page 26).

En pratique, on est parfois amené à augmenter la valeur du multiplicateur de distance maximale d'intersection jusqu'à $0,2$ environ dans le cas de recollements de surfaces courbes, afin que toutes les intersections soient bien détectées. Ce facteur jouant sur la simplification des faces issues du recollement mais aussi sur la déformation des faces « latérales », on ne le modifie en général que si nécessaire. Par contre, la sous-option `-plane` n'a que très rarement été nécessaire sur les maillages réels rencontrés à ce jour.

Pour recoller par exemple les faces de couleur 2 du maillage `essai.des` avec un multiplicateur de distance d'intersection de $0,15$, puis recoller toutes les faces de bord restantes pouvant l'être avec le paramètre par défaut de $0,1$, on utiliserait la ligne de commande suivante :

```
ecs -m essai.des -j -fraction 0.15 -color 2 -color 3 -j
```

Lorsque les faces à recoller comportent au moins un sommet en commun (ce qui est notamment le cas pour des éléments issus de raffinements successifs produits par des mailleurs tels que *Harpoon*), on peut utiliser la sous-option `-semi-conf`, qui ofrre une recherche beaucoup plus rapide des faces à recoller par rapport à l'algorithme usuel.

2.1.7 Périodicité

La gestion de la périodicité est basée sur une extension du recollement conforme. Le principe est décrit § 7 page 32. Toutes les sous-options relatives au recollement conforme s'appliquent donc aussi à la périodicité. De

plus, on doit définir le pas de périodicité correspondant. On notera que deux faces périodiques sont orientées de la même manière.

Pour une périodicité de translation, on utilisera la sous-option `-trans` suivie des trois composantes du vecteur translation (tx, ty, tz).

Pour une périodicité de rotation, on utilisera la sous-option `-rota`, qui doit être complétée soit par :

`-angle` α `-dir` $dx\ dy\ dz$

où α est l'angle de rotation en degrés et (dx, dy, dz) définit la direction de l'axe de rotation (dans le sens trigonométrique), soit par :

`-matrix` $m11\ m12\ m13\ m21\ m22\ m23\ m31\ m32\ m33$

où $\begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix}$ définit la matrice de rotation.

Dans les deux cas, on peut définir un point invariant différent de $(0, 0, 0)$ en ajoutant la sous-option de rotation `-invpt` $px\ py\ pz$. On peut composer une translation et une rotation ; dans ce cas, la transformation composite est toujours $R \circ T$, quel que soit l'ordre dans lequel la translation et la rotation sont définies parmi les sous-options de cette périodicité.

Le sens de la periodicité n'a pas d'importance (mais, dans le cas d'une transformation composée $R \circ T$, R et T doivent être cohérentes entre elles).

Il est conseillé comme pour le recollement de filtrer les faces de bord à traiter au moyen d'un critère de sélection. On peut construire autant de périodicités que l'on veut, tant que l'on a des faces de bord. À l'issue de la construction d'une périodicité, les faces ayant des correspondantes périodiques n'apparaissent plus comme des faces de bord, et ne seront donc plus disponibles pour la construction d'autres périodicités.

On termine avec l'exemple d'une périodicité de rotation de 90 degrés autour de l'axe de direction $(0, 0, 1)$ passant par $(1, 0, 0)$, pour les faces du groupe « perio_1 » et avec une tolérance de recollement de 20% (le caractère \ marque la continuation de la ligne) :

```
-perio -rota -angle 90 -dir 0 0 1 -invpt 1 0 0 \
       -fraction 0.2 -group perio_1
```

2.1.8 Correction de l'orientation des éléments

On peut activer la correction éventuelle de l'orientation des éléments au moyen de l'option `-reorient`.

On notera que l'on ne peut garantir dans tous les cas la correction (ou même parfois la détection) d'une mauvaise orientation. Toutes les possibilités de numérotation locale des éléments n'étant pas vérifiées, on se concentre sur les permutations de numérotation « usuelles ». Surtout, les algorithmes utilisés peuvent produire des fausses alertes ou échouer à trouver une renumérotation dans le cas d'éléments fortement non convexes. Il n'y a dans ce cas rien à faire, car sans l'hypothèse de convexité des éléments, on ne peut pas toujours choisir entre deux définitions possibles d'un élément à partir d'un nuage de points.

Avec une option de post-traitement telle que `-ensight`, on produit des maillage visualisables avec un outil de post-traitement correspondant aux éléments à l'orientation corrigée ainsi qu'à ceux qui restent mal orientés.

2.2 Variables d'environnement

Le positionnement de quelques variables d'environnement spécifiques au Préprocesseur permet de modifier certains comportements par défaut. En général, si un comportement est modifiable par variable d'environnement plutôt que par option de ligne de commande, c'est soit qu'il a peu d'intérêt pour un non-développeur, soit que sa modification est potentiellement dangereuse. On décrit brièvement l'effet des diverses variables d'environnement spécifiques au Préprocesseur :

`ECS_CONT_SUR_FPE`

Si cette variable est définie et correspond à un entier strictement positif (i.e. `ECS_CONT_SUR_FPE=1`), on autorise la poursuite sur erreur en virgule flottante(sous PC Linux uniquement à ce jour). Inutile de chercher à passer un calcul derrière, mais cela peut éventuellement permettre d'aller jusqu'au post-traitement en vérification du maillage, ce qui peut aider à comprendre quel défaut a pu mener à cette erreur.

`ECS_FIC_CHRONO`

Permet de définir un nom de fichier dans lequel les temps intermédiaires de calcul seront imprimés.

`ECS_FIC_MEM`

Permet de définir un nom de fichier dans lequel une comptabilité des allocations, redimensionnements, et libérations de mémoire sera imprimée.

`ECS_DEF_LNG_ARE_MIN`

En dessous de la longueur indiquée (10^{-15} par défaut), une arête est considérée comme dégénérée et ses sommets seront fusionnés après le passage en connectivité descendante. Les arêtes et faces dégénérées sont alors supprimées. Ainsi, l'élément post-traité ne change pas, mais le Noyau peut par exemple voir un prisme là où on avait au départ un hexaèdre avec quatre sommets confondus deux par deux (et donc une face de surface nulle). Si le Préprocesseur n'affiche aucune information relative à ce type de correction, c'est qu'elle n'a pas été nécessaire. Pour désactiver complètement cette correction automatique, on peut fournir à cette variable d'environnement une valeur négative.

`ECS_PRE_IDEAS_IGNORE_SYS_COO`

Si cette variable est définie et correspond à un entier strictement positif, on ne prendra pas en compte les systèmes de coordonnées dans les fichiers au format Universel *I-deas*. Le comportement du Préprocesseur sera alors le même que celui des versions 1.0 et 1.1. de l'Enveloppe. On notera que dans tous les cas, les systèmes de coordonnées non cartésiens ne sont pas encore traités.

`ECS_RC_MAX_FAC_DEC`

Définit le nombre de nouvelles faces issues d'une face initiale (100 par défaut) au delà duquel on considère que la reconstruction du recollement conforme est probablement entrée dans une boucle sans fin et a donc échoué. Ce critère est le dernier à intervenir dans la détection d'erreurs et n'entre en jeu que dans des cas pathologiques liés probablement à une déformation excessive des faces à recoller (et donc à un critère de tolérance trop lâche). Il ne devrait pas être nécessaire de l'augmenter pour un maillage réellement destiné à un calcul, car 100 faces correspondent grossièrement à un rapport de longeur de 10 dans la taille des cellules de part et d'autre de la surface de recollement, ce qui est largement excessif du point de vue numérique.

2.2.1 Variables d'environnement système

Certaines variables d'environnement liées au système peuvent aussi modifier le comportement du Préprocesseur. Par exemple, si l'on a compilé le Préprocesseur avec le support *MED* sur une architecture permettant l'utilisation de librairies partagées (i.e. toutes celles utilisées à MFEE, mais pas des architectures à micro-noyaux de calcul minimalistes telles qu'IBM BlueGene ou Cray XT3), on peut utiliser la variable d'environnment `LD_PRELOAD` pour fournir un chemin « prioritaire » pour les libraires *MED*-fichier ou HDF5, et ainsi expérimenter avec une autre version de ces librairies sans recompiler le Préprocesseur. Pour savoir quelles librairies partagées sont utilisées par un programme exécutable, tapez la commande `ldd {chemin_exécutable}`.

3 Fonctionnalités optionnelles

Certaines fonctionnalités s'appuient sur des librairies externes, qui ne sont pas toujours disponibles. Il est donc possible de configurer et compiler le Préprocesseur de manière à ne pas utiliser ces librairies. Au lancement du Préprocesseur, il est affiché quelles options sont supportées. Les librairies optionnelles sont les suivantes :

– Librairie CGNS. En son absence, le support du format CGNS est simplement désactivé.
– Librairie *MED*-fichier. En son absence, le support du format *MED* est simplement désactivé.
– Librairie *Metis* pour le découpage de graphes. Sans cette librairie, le découpage de domaines pour le parallélisme reste possible, mais on a recours à un algorithme simpliste qui ne minimise pas la taille des interfaces.
– Lecture de fichiers compressés avec Zlib. Avec cette option, il est possible de lire directement de fichiers de maillage compressés au format *.gz*. Cette fonctionnalité est limitée aux formats ne s'appuyant pas déjà sur un librairie propre (i.e. elle n'est pas utilisable avec des fichiers CGNS ou *MED*), et entraîne un surcoût de mémoire de de temps CPU, mais peut être d'un certain confort. Sans cette librairie, les fichiers devront être décompressés avant utilisation.

4 Formats d'entrée et post-traitement supportés

Le Préprocesseur supporte plusieurs formats de maillage différents, tous ces formats ayant été demandés par des utilisateurs ou des projets en fonction de leurs outils de maillage ou de post-traitement, et tous ayant des avantages et des inconvénients (en termes de simplicité, fonctionnalité, pérennité, et popularité) les uns par rapport aux autres. On décrit ici ces divers formats supportés par le Préprocesseur.

4.1 Remarques générales

On notera que le Préprocesseur est en général capable de lire tous les éléments de type « classique » présents dans les fichiers de maillage (triangles, quadrangles, tétraèdres, pyramides, prismes, et hexaèdres). Les éléments de type quadratiques ou cubiques sont convertis dès la lecture en leurs équivalents linéaires. Les sommets référencés par aucun élément (sommets isolés ou sommets milieux d'éléments quadratiques par exemple) sont supprimés. Les maillages sont lus dans l'ordre défini par l'utilisateur et sont concaténés, les indices des sommets et éléments étant incrémentés en conséquence. [4]

À ce stade, on trie les éléments volumiques en fonction de leur type, et le fichier de maillage du domaine fluide pour le post-traitement est généré si on l'a demandé ou qu'on lit des résultats de calcul. Le recollement conforme, qui peut transformer des cellules en polyèdres quelconques n'intervient qu'ensuite. Ceci permet de contourner la limitation de la plupart des outils de post-traitement, qui n'acceptent pas de polyèdres quelconques. On notera qu'à la conception du module Enveloppe en début 1999, et ce jusqu'à la sortie d'*EnSight* 7.4 en début 2002, aucun des outils envisagés ne permettait de gérer des polyèdres quelconques, et que peu d'entre eux le font encore à ce jour. *Il est possible de demander la sortie simultanée sous plusieurs formats.*

Sauf précision contraire, les couleurs ou groupes affectés à des sommets sont ignorés. Les repérages se font donc directement par faces et cellules. La manière de repérer des faces dépend du mailleur. Par exemple, avec *SIMAIL*, on peut référencer directement les faces des éléments volumiques, alors qu'avec *I-deas*, on doit ajouter au maillage volumique une couche d'éléments surfaciques, portant les couleurs et groupes désirés. De manière interne, le Préprocesseur considère toujours que l'on ajoute une couche d'éléments surfaciques (i.e. lors de la lecture d'un maillage *SIMAIL*, des faces supplémentaires sont générées pour porter les couleurs de faces des cellules). Lors du passage en connectivité descendante, toutes les faces de même connectivité sont fusionnées ; la présence initiale de deux couches d'éléments de peau topologiquement identiques mais portant des couleurs différentes se traduirait donc par des faces de bord du maillage de calcul fourni au Noyau portant deux couleurs.

D'autres maillages, surfaciques, peuvent alors être générés, selon les combinaisons d'options utilisées (maillage de bord et faces issues ou modifiées par les recollements conformes ou périodicités lorsque la vérification est activée, sélections de faces). Ces maillages étant extraits à partir de l'état en cours (pour recollements ou périodicité) ou final (sélections) du maillage, ils peuvent être amenés à contenir des polygones quelconques. Selon si le format le permet, on pourra post-traiter directement ces polygones, ou on devra les découper en triangles. Même pour les formats permettant l'utilisation directe de polygones, on peut forcer via une sous-option `-dec-poly` le découpage en triangles, au cas ou le logiciel de post-traitement utilisé en aval ne supporte pas les polygones quelconques. Si l'on découpe des polygones en triangles, on génère alors en supplément un

[4] Les labels éventuels des entités ne sont pas conservés, car ils risqueraient de ne pas être uniques en cas de concaténation de plusieurs maillages.

maillage contenant les arêtes des faces non découpées, afin de pouvoir visualiser leurs frontières.

4.2 Formats d'entrée

4.2.1 NOPO/Simail (INRIA/SIMULOG)

Ce format produit par *SIMAIL* est fortement utilisé à MFEE. On ne traite pas actuellement le cas des systèmes de coordonnées cylindriques ou sphériques, mais il semble que *SIMAIL* écrit toujours les maillages en coordonnées cartésiennes, même si les points ont été définis dans un autre système. La plupart des types d'éléments « classiques » sont utilisables, sauf les pyramides.

On remarquera que selon l'architecture sur laquelle *SIMAIL* a produit le fichier [5], le fichier ne pourra pas nécessairement être relu par *SIMAIL* sur une machine différente, alors que cela ne pose pas de problème au Préprocesseur, qui détecte automatiquement l'ordre des octets et effectue les permutations nécessaires.

Extension par défaut :	`.des`
Type de fichier :	binaire semi-portable « Fortran » (entiers sur 4 octets, réels IEEE sur 8 octets, octets ordonnés selon architecture)
Éléments surfaciques :	triangles, quadrangles (+ références faces des éléments volumiques)
Éléments volumiques :	tétraèdres, prismes, hexaèdres
Repérage volumique :	références systématiques (couleurs) d'éléments
Compatibilité :	tous fichiers de ce type à condition que le système de coordonnées utilisé soit cartésien et non cylindrique ou sphérique
Documentation :	Documentation utilisateur Simail ou documentation MODULEF : http ://www-rocq.inria.fr/modulef Notamment : http ://www-rocq.inria.fr/modulefDoc/FR/Guide2-14/node49.html

4.2.2 Universel *I-deas*

Ce format est encore très utilisé à MFEE. Il peut comporter un grand nombre de rubriques, relatives entre autres à la CAO, au maillage, aux matériaux, à des résultats de calcul, ou à la représentation d'une pièce. La plupart de ces rubriques sont ignorées, et seules celles relatives à la définition des sommets, des éléments, et des groupes sont prises en compte. On signale à l'utilisateur si plusieurs systèmes de coordonnées sont utilisés, mais on n'en tient pas compte, car il semble que les coordonnées exportées soient en fait définies par rapport à un repère absolu, et que les systèmes de coordonnées n'aient donc pas d'influence (d'après des essais avec *I-deas* pour tenter de comprendre le comportement associé, la documentation étant peu claire à ce sujet).

La définition du format évolue avec les versions d'*I-deas*, mais de manière limitée : certaines rubriques sont déclarées obsolètes, et sont remplacées par d'autres, mais la définition d'une rubrique donnée n'est jamais modifiée. La définition des sommets et éléments ne semble pas avoir changé ces dernières années, et seule la définition des groupes a évolué. Si on venait à lire un fichier généré avec une version ultérieure d'*I-deas* pour laquelle cette définition aurait à nouveau changé sans mettre à jour le Préprocesseur, la nouvelle rubrique étant inconnue, elle serait simplement ignorée.

On notera qu'il s'agit d'un format texte. La plupart des types d'éléments « classiques » sont utilisables, sauf les pyramides.

[5] « little endian » comme avec les processeurs de type Pentium, Athlon, ou Alpha, ou « big endian » comme avec la plupart des autres

Extension par défaut :	`.unv`
Type de fichier :	texte
Éléments surfaciques :	triangles, quadrangles
Éléments volumiques :	tétraèdres, prismes, hexaèdres
Repérage volumique :	couleurs (systématiques) et groupes d'éléments
Compatibilité :	*I-deas* (*Master Series* 5 à 9, *NX Series* 10 à 12) à minima
Documentation :	Documentation en ligne I-deas NX Series

4.2.3 GAMBIT neutral

Ce format peut être produit par le mailleur GAMBIT du code *FLUENT*. Ce premier ne proposant pas directement l'export de maillages sous d'autres formats gérés par le Préprocesseur (bien que *FLUENT* lui-même permette l'export aux formats CGNS ou universel *I-deas*), il a été jugé important d'ajouter au Préprocesseur la possibilité de lire directement des fichiers au format GAMBIT neutral.

On notera qu'il s'agit d'un format texte. Les types d'éléments « classiques » sont utilisables.

Extension par défaut :	`.neu`
Type de fichier :	texte
Éléments surfaciques :	triangles, quadrangles
Éléments volumiques :	tétraèdres, pyramides, prismes, hexaèdres
Repérage volumique :	groupes d'éléments
Documentation :	Documentation en ligne GAMBIT

4.2.4 *pro-STAR/STAR4 Polyfile*

Ce format polyédrique de la société CD-Adapco semble être utilisable à la fois avec les outils STAR-CD et Star-CCM+, et devrait pouvoir être généré par l'outil pro-STAR. Les maillages test dont on dispose ont été générés par l'outil *Comet-Design*, depuis remplacé par d'autres outils de la gamme CD-Adapco, notamment STAR-CD V4. Son support dans le Préprocesseur est peu testé, et il peut subsister des problèmes, les cas tests disponibles n'étant pas exhaustifs en termes de fonctionnalités du format (notamment en ce qui concerne la définition des descriptions), et ne provenant que d'un seul outil.

Actuellement, les numéros d'entité géométrique associée à des mailles sont convertis en numéros de couleur. Ceci tend à produire un grand nombre de couleurs de faces. Il serait utile de disposer de l'association entre les entités géométriques et les conditions aux limites, afin de pouvoir produire un maillage avec moins de couleurs distinctes.

Extension par défaut :	`.ngeom`
Type de fichier :	fichier binaire avec codage portable XDR.
Éléments surfaciques :	polygones
Éléments volumiques :	polyèdres
Repérage volumique :	
Compatibilité :	tous fichiers de ce type (mais faible retour d'expérience)
Documentation :	documentation et sources fournies par CD-adapco dans le cadre de la collaboration avec UMIST et EDF R&D/MDTT

4.2.5 *EnSight* 6

Ce format est utilisé pour les sorties du mailleur *Harpoon*, développé par la société Sharc Ltd (qui est aussi le distributeur *EnSight* pour le Royaume-Uni). C'est aussi du format de sortie par défaut d'un certain nombre d'outils développés ou utilisés à MFEE (y compris des versions 1.0 et 1.1 de *Code_Saturne* ou de *NEPTUNE_CFD* en mode « sans module Enveloppe »). Ce format peut représenter tous les éléments « classiques ».

Initialement destiné au post-traitement, il ne prévoit pas explicitement le repérage de zones surfaciques ou

volumiques, mais permet la création d'un nombre arbitraire de *parts* (i.e. de groupes d'éléments distincts ou non) s'appuyant sur une même liste de noeuds. La pratique (utilisée par Harpoon à minima) consiste alors à ajouter des éléments de peau au maillage volumique et à les groupes dans différents *parts* correspondant à différentes zone ou types de conditions aux limites. On peut aussi selon le même principe séparer le maillage volumique en *parts* pour identifier différentes zones. Les noms de *part* pouvant contenir jusqu'à 80 caractères, on ne les transforme pas en groupes (dont les noms pourraient être lourds à manipuler), et l'on se contente de convertir leurs numéros en numéro de couleur.

On remarquera que les fichiers produits par *Harpoon* contiennent des prismes mal orientés, et qu'il est donc nécessaire d'utiliser l'option `-reorient`. La correction n'est pas déclenchée automatiquement, car on espère que ce défaut sera corrigé. Les maillages correspondants contiennent aussi (par principe du mailleur) des éléments non conformes, partageant certains sommets, et correspondant à des subdivisions plus ou moins fines d'une grille initiale (avec raffinements près des frontières). Il est donc nécessaire d'utiliser avec ces maillages issus de *Harpoon* l'option `-recollement -semi-conf` (ou `-j -semi-conf`). Cette option n'est pas déclenchée automatiquement, car l'utilisateur peut vouloir préciser des zones à recoller ou surtout à ne pas recoller (i.e. pour préserver des parois minces).

Extension par défaut :	`.case`
Type de fichier :	un fichier texte (extension *.case*), et un fichier texte, binaire, ou binaire Fortran (extension *.geo*) décrivant les entiers et les réels au format IEEE sur 4 octets
Éléments surfaciques :	triangles, quadrangles
Éléments volumiques :	tétraèdres, pyramides, prismes, hexaèdres
Repérage volumique :	numéros de parts convertis en numéros de couleurs
Compatibilité :	tous fichiers de ce type
Documentation :	documentation en ligne, disponible aussi à : http ://www.ceintl.com/ensight76docs/OnlineHelp/UM-C11.pdf

4.2.6 *Gmsh*

Ce format est utilisé par l'outil libre *Gmsh* . Cet outil offre à la fois des fonctionnalités de maillage et de post traitmenent. Le Préprocesseur n'exploite que la partie maillage.

On remarquera que certains fichiers produits par *Gmsh* contiennent des éléments mal orientés, et qu'il peut donc être nécessaire d'utiliser l'option `-reorient`.

Le Préprocesseur gère les versions 1 et 2 de ce format. Dans la version 1, deux étiquettes sont associées à chaque élément : la première fournit un numéro d'entité physique, la seconde un numéro d'entité élémentaire. Avec la version 2, il est possible d'associer un nombre arbitraire d'étiquettes à chaque élément, mais les fichiers produits par *Gmsh* utilisent par défaut 2 étiquettes, avec la même signification que précédemment.

On a choisi de convertire les numéros d'entités physiques en couleurs. Il est possible de constuire un maillage avec *Gmsh* sans définir d'entités physiques (dans quel cas tous les éléments auront le même numéro de couleur), mais la documentation de l'outil fait clairement apparaitre les entités géométriques comme étant destinées à regrouper des entités élémentaires en groupes (disjoints ou non) ayant un sens « physique ».

Pour pouvoir disposer de couleurs distinctes avec un maillage issu de *Gmsh*, il est donc nécessaire pour l'utilisateur de définir des entités physiques. En contrepartie, il dispose d'un contôle relativement fin sur les couleurs associées aux zones du maillage, alors qu'avec les seules entités élémentaires, il aurait à gérer un nombre de couleurs très important.

Extension par défaut :	`.msh`
Type de fichier :	fichier texte ou binaire
Éléments surfaciques :	triangles, quadrangles
Éléments volumiques :	tétraèdres, pyramides, prismes, hexaèdres
Repérage volumique :	numéros d'entités physiques convertis en numéros de couleurs
Compatibilité :	tous fichiers de ce type
Documentation :	documentation en ligne, disponible aussi à : http ://www.geuz.org/gmsh

4.2.7 Igg HEXA (NUMECA)

Ce format est assez particulier dans la mesure où il définit un maillage hiérarchique constitué uniquement d'hexaèdres, de faces quadrangulaires, et d'arêtes, les arêtes pouvant être divisée en deux, les faces en 2 ou 4, et les cellules en 2, 4, ou 8. Deux éléments voisins ne sont donc pas toujours conformes. Le Préprocesseur ne conserve que le niveau de maillage le plus fin, et utilise les informations hiérarchiques pour reconstruire automatiquement le recollement conforme approprié (avant le traitement des recollements éventuellement demandés par l'utilisateur). Les informations de numéro de face CAO associée sont transformées en couleurs.

Remarque : le filtre est basé sur le format *Igg Hexa* tel qu'il était défini fin 2001. Depuis, le mailleur semble avoir été renommé ou avoir évolué vers *HexPress*, nous ne disposons ni d'une documentation ni d'un fichier test récent.

Extension par défaut :	`.hex`
Type de fichier :	binaire portable par défaut (entiers sur 4 octets, réels IEEE sur 8 octets, octets ordonnés dans le sens « big endian ») ou texte
Éléments surfaciques :	quadrangles
Éléments volumiques :	hexaèdres
Repérage volumique :	aucun
Compatibilité :	à vérifier (fichiers fin 2001 à minima)
Documentation :	éléments fournis par NUMECA (http ://www.numeca.be) en 2001.

4.3 Formats d'entrée ou sortie

4.3.1 *EnSight Gold*

Ce format peut représenter tous les éléments « classiques », ainsi que les polygones et les polyèdres quelconques. Le maillage principal volumique étant basé sur le maillage initial, avant traitement des recollements, on n'utilise que rarement les polyèdres quelconques.

Ce format évolue légèrement d'une version d'*EnSight* à une autre, mais en conservant une compatibilité ascendante. Par exemple, les polygones quelconques ne pouvaient pas être utilisés dans un même maillage que d'autres types d'éléments avant la version 7.4, qui a aussi ajouté le support des polyèdres convexes génériques. La version 7.6 a ajouté le support des définitions de matériaux. On notera que certaines librairies permettant le support direct de fichiers *EnSight Gold* ne traitent pas forcément toutes les possibilités offertes. Notamment *VTK* supporte à ce jour les fichiers *EnSight Gold* parallèles (i.e. découpés), mais pas les éléments polyédriques quelconques ou la définition de matériaux.

Ce format offre en outre un nombre important de possibilités non exploitées actuellement par *Code_Saturne*, telles que celles de définir des valeurs sur une partie d'un maillage uniquement (via des valeurs « indéfinies » ou des profils), la possibilité depuis le printemps 2003 d'affecter un ou plusieurs matériaux à des mailles, et surtout la possibilité de découper les maillages en plusieurs parties, pour une utilisation parallèle en mémoire distribuée.

Ce format peut être utilisé comme format d'entrée, de la même manière que le format *EnSight* 6. Par contre, on ne peut repérer des zones surfaciques que si le fichier définit les « labels » des noeuds. En effet, à la différence du format *EnSight* 6, chaque *part* contient sa propre liste ne noeuds, et on ne peut donc déterminer si des noeuds appartenant à deux *parts* distincts sont équivalents que si leurs numéros sont précisés et identiques (on évite la

détermination d'équivalence de noeuds sur des critères géométriques, car cela serait coûteux en temps CPU et surtout empêcherait l'utilisation de parois minces dans le domaine, si les noeuds de part et d'autre d'une telle paroi étaient fusionnés).

Extension par défaut :	répertoire {*nom_cas*}`.ensight`, contenant un fichier à l'extension `.case`
Type de fichier :	plusieurs fichiers texte (ou binaires en entrée)
Éléments surfaciques :	triangles, quadrangles, polygones
Éléments volumiques :	tétraèdres, pyramides, prismes, hexaèdres, polyèdres convexes
Repérage volumique :	possibilité de définir des matériaux (non encore utilisée)
Compatibilité :	fichiers lisibles par *EnSight* 7.4 à 8.2, ainsi que par des outils à base de librairie *VTK* , notamment *ParaView* (http ://www.paraview.org)
Documentation :	documentation en ligne, disponible aussi à : http ://www.ceintl.com/ensight76docs/OnlineHelp/UM-C11.pdf

4.3.2 *MED* 2.3

D'origine EDF R&D, ce format (*Modèle d'Échanges de Données*) a été défini et évolue au travers du groupe de travail *MED* constitué de membres EDF R&D et CEA (l'équipe *Code_Saturne* y est représentée). C'est le format de référence pour l'environnement *S*ALOME et le projet *P@L*. Ce format est assez riche, permettant la définition de tous les types d'éléments « classiques », en connectivité nodale ou descendante. depuis l'automne 2003, le format *MED* 2.2 permet la définition de faces polygonales et de mailles polyédriques, ainsi que la définition de maillages structurés.

Ce format, qui s'utilise via une librairie s'appuyant elle-même sur la librairie libre HDF5, permet aussi bien la lecture de maillages avec leurs attributs (couleurs et groupes d'entités classés en familles) que leur écriture, ainsi que celle de résultats de calcul, avec possibilité (non exploitée par *Code_Saturne*) de définir des variables sur une partie du support uniquement.

On notera que la librairie *MED*-fichier est disponible sous une licence de type LGPL depuis la version 2.1.5, ce qui en fait un logiciel libre (HDF5 étant aussi placé sous une licence libre).

La version 1.1 de l'Enveloppe utilisait le format *MED* 2.1 ; avec la version 1.2 de l'Enveloppe, on est passé au modèle *MED* 2.2 (le sous-ensemble de *MED* 2.3 utilisé par la version 1.3 du Préprocesseur est compatible avec *MED* 2.2).

Extension par défaut :	`.med`
Type de fichier :	binaire portable, basé sur la librairie HDF5 (http ://hdf.ncsa.uiuc.edu)
Éléments surfaciques :	triangles, quadrangles, polygones simples
Éléments volumiques :	tétraèdres, pyramides, prismes, hexaèdres, polyèdres simples
Repérage volumique :	familles (*i.e.* couleurs et groupes) d'éléments
Compatibilité :	toutes versions de *MED* 2.2 ou 2.3 (le Préprocesseur ne supporte que les maillages avec une connectivité nodale)
Documentation :	voir site *MED* EDF : http ://med.der.edf.fr

4.3.3 CGNS 2.4

Promu notamment par la NASA, Boeing, et ICEM CFD (ainsi que l'ONERA en France), ce format(*CFD General Notation System*) semble connaître un certain succès dans le monde de la CFD. Le concept est très semblable à celui de *MED*, avec de plus un accent sur la normalisation des noms de variables ou d'informations sur le calcul, et un nombre de possibilités d'utilisation encore plus important. À l'inverse de *MED*, la première version était limitée aux maillages structurés multiblocs, les maillages non structurés ayant été ajoutés par la suite.

Légèrement antérieure à *MED*, cette librairie a l'avantage d'avoir été libre depuis le départ, avec une documentation anglaise, ce qui fait qu'elle est beaucoup plus connue. Elle est beaucoup plus ciblée sur la CFD, là ou *MED* se veut plus générique. On trouve avec la distribution de CGNS un certain nombre d'utilitaires, dont un

visualiseur de maillages (qui ne traite pas les faces polygonales) et un interpolateur.

En lecture, on devrait pouvoir lire presque tous les maillages représentés via ce format, à l'exception de maillages comportant des interfaces inter-zones de type recouvrement. Les autres interfaces interzones ne sont pas traitées directement (car il existe au moins trois ou quatre manières de les décrire, et certains mailleurs ne les exportent pas[6]), mais on prévient l'utilisateur de leur existence, en lui suggérant d'utiliser un recollement conforme approprié. Les zones structurées sont converties en zones non structurées immédiatement après la lecture.

Les informations de type conditions aux limites (C.L.) sont interprétées comme étant des groupes portant le même nom. Le format ne prévoit pas encore de repérage d'éléments volumiques, seules des conditions aux limites étant prévues, définies sur des faces (en non structuré uniquement) ou des sommets. On remarquera que des conditions aux limites définies sur des sommets ne sont pas ignorées par le Préprocesseur, mais transformées en conditions sur les faces dont tous les sommets portent une même condition aux limites.[7] On a aussi ajouté la possibilité de construire des groupes de volumes ou de faces supplémentaires, en fonction de l'appartenance de ces entités à des zones ou sections du maillage. Cette fonctionnalité activable comme sous-option de la sélection de maillage permet notamment de récupérer les informations de C.L. de fichiers ne comportant pas d'informations explicites sur les C.L. mais dont les maillages sont subdivisés en zones ou sections appropriées (ce qui dépend de l'outil utilisé pour générer le maillage).

En écriture, on fournit pour le domaine volumique une connectivité non structurée, sans information de recollement ni faces supportant des couleurs ou des groupes.[8] Les variables sont affectées aux cellules (les déplacements éventuels de sommets ne sont pas encore écrits par le Préprocesseur sous ce format).

On notera que le support CGNS par des outils externes est souvent décevant, du moins pour les maillages non-structurés. Ainsi, divers éditeurs semblent utilisent des moyens légèrement différents de repérer les zones à associer aux conditions aux limites, et certains comportements sont pires ; par exemple, l'outil de visualisation ViSit considère que plusieurs maillages dans un même fichier CGNS correspondent forcément à un même maillage à plusieurs temps différents, et ne lit que la première section d'éléments d'un maillage non structuré (alors qu'on peut très bien en avoir plusieurs, le plus souvent associées à divereses zones ou types d'éléments). Quant à EnSight, dès lors qu'un maillage comporte plusieurs types d'éléments, il affiche les variables associées aux cellules sur les mauvaises cellules. Quant au support des polygones quelconques, même les outils de vérification et de visualisation livré avec la librairie CGNS ne savent pas bien les gérer (ce type d'élémént apparait bien dans la norme et dans le code source de la librairie principale CGNS 2.5.2, mais pas dans celui des outils associés).

Extension par défaut :	`.cgns`
Type de fichier :	binaire portable (utilise la librairie ADF spécifique à CGNS)
Éléments surfaciques :	triangles, quadrangles, polygones simples
Éléments volumiques :	tétraèdres, pyramides, prismes, hexaèdres
Repérage volumique :	aucun par défaut, mais il est possible de créer des groupes associés aux zones ou sections présentes dans le maillage au moyen des sous-options de sélection de maillage
Compatibilité :	en lecture, CGNS 2.0 à 2.5 à minima (le Préprocesseur ne prenant pas directement en compte les divers cas possibles d'interfaces multizones/non conformes, et l'utilisateur doit donc forcer un recollement conforme approprié) ; en écriture, la sortie étant par défaut au format CGNS 2.4, il faut une version au moins aussi récente pour la lire (sinon, on peut recompiler le Préprocesseur avec une version plus ancienne)
Documentation :	voir site CGNS : http ://www.cgns.org

[6] Par exemple, *ICEM CFD* peut recoller des maillages non conformes, mais il exporte les surfaces de recollement au format CGNS comme de simples faces frontières avec des conditions aux limites de type utilisateur.

[7] Si l'un des sommets d'une face donnée ne porte pas la condition aux limites, cette condition n'est pas reportée sur la face.

[8] Un défaut de ce format en non structuré est que qu'un champ doit systématiquement s'appliquer à tous les éléments d'une zone, et que l'ajout de faces pour la définition de groupes à une zone impliquerait donc de sortir les valeurs du champ sur ces faces, avec le risque qu'un outil de post-traitement utilisé en aval ne gère pas bien la combinaison d'éléments volumiques et surfaciques.

4.4 Méta-fichiers de mailage

Il est possible d'utiliser en lieu de fichier de maillage des fichiers texte (portant de préférence l'extension `.mesh` décrivant un ensemble de maillages et de transformations. Les lignes vides sont ignorées, et le caractère # peut être utilisé pour définir des commentaires (toute portion d'une ligne à partir de ce caractère est ignorée).

On peut demander la lecture d'autant de maillages que l'on souhaite, en utilisant à chaque fois une rubrique de la forme :

`read_mesh:` *nom fichier*

ou :

`read_mesh:` *nom fichier* `<sous options>`

Si cette rubrique apparait plusieurs fois, les maillages sont concaténés automatiquement. Au besoin, un méta-fichier de maillage peut lui-même inclure un méta-fichier de maillage parmi les maillages lus. Les sous-options éventuelles associées à la lecture d'un fichier peuvent être séparées par des virgules, point-virgules, ou espaces, et sont de la forme :

`format=`	nom du format (identique aux options de ligne de commande)
`num=`	numéro du maillage (utile lorsqu'on a plusieurs maillages dans un même fichier)
`grp_cel=`	`<section` ou `zone>`, utile uniquement pour les fichiers au format CGNS
`grp_fac=`	`<section` ou `zone>`, utile uniquement pour les fichiers au format CGNS

On peut aussi définir une transformation géométrique à appliquer au maillage, à partir d'une matrice de transformation en coordonnées homogènes (3 lignes de 4 colonnes, les 3 premières colonnes correspondant à la partie rotation/dilatation, la dernière colonne à une translation). La rubrique correspondante est la suivante (les valeurs pouvant être réparties au choix sur plusieurs lignes) :

`transformation_matrix:` $t_{11}\ t_{12}\ t_{13}\ t_{14}\ t_{21}\ t_{22}\ t_{23}\ t_{24}\ t_{31}\ t_{32}\ t_{33}\ t_{34}$

On notera que l'ordre de déclaration de lecture de maillages multiples définit l'ordre dans lequel ils sont concaténés, mais la transformation géométrique éventuelle n'est appliquée qu'à la fin (il s'agit là d'un fichier de rubriques, pas d'un langage de commandes).

4.5 Outils de maillage et formats associés

Le plus souvent, le choix du format de maillage est associé au choix de l'outil. Cependant, certains outils permettent la sortie du maillage sous plusieurs formats supportés par le Préprocesseur. C'est notamment le cas de *FLUENT* et *ICEM CFD*, qui peuvent exporter des maillages au format universel *I-deas* ou CGNS (le mailleur *GAMBIT* de *FLUENT* pouvant aussi exporter des maillages au format universel *I-deas*). En général, on a plutôt l'habitude d'exporter un fichier au format universel *I-deas*, mais ce format ne permet pas la représentation d'élements de type pyramide, souvent utilisés par ces mailleurs pour la jonction entre zones maillées en héxaèdres et zones maillées en tétraèdres. L'utilisateur est encouragé à utiliser le format CGNS, qui n'a pas cette limitation.

En fonction de l'évolution des outils liés à la plate-forme SALOME, le format *MED* devrait avoir un intérêt croissant.

5 Fichiers fournis au Noyau

Les données fournies par le Préprocesseur au Noyau sont transmises via des fichiers binaires, avec un ordonnancement de type « big endian » des octets. Ces fichiers se trouvent dans un répertoire `preprocessor_output`, et se nomment `n00001`, `n00002`, etc.

Il est possible de demander l'affichage des noms des rubriques écrites ainsi que des n premières et dernières valeurs de chaque rubrique en utilisant l'option de ligne de commande `-echo-comm [n]`.

Lors d'une utilisation autonome du Préprocesseur pour la vérification des maillages, on n'a souvent aucun besoin des données envoyées au Noyau. On peut alors utiliser l'option de ligne de commande `-sc` pour simuler simplement la communication, sans créer de fichier(s) (bien que l'écho des données qui devraient l'être fonctionne normalement).

6 Algorithmes géométriques

Dans ce chapitre, on apportera des précisions sur les algorithmes utilisés pour diverses opérations réalisées par le Préprocesseur.

6.1 Grandeurs géométriques

Les normales des faces (de longueurs égales aux surfaces) ainsi que les centres de gravité des faces et cellules sont calculés par directement par le Noyau, mais des normales de faces peuvent aussi être calculées par le Préprocesseur pour le recollement conforme.

6.1.1 Normales et centres de gravité des faces

L'algorithme utilisé pour le calcul des normales des faces est valable pour tout polygone plan, même si celui-ci est non convexe. Il consiste à prendre un point arbitraire P_a dans le plan du polygone, puis à calculer la somme des normales de tous les triangles $\{P_a, P_i, P_{i+1}\}$ où $\{P_1, P_2, ..., P_i, ..., P_n\}$ sont les sommets du polygone et l'on considère $P_{n+1} \equiv P_0$. Comme on le voit sur la figure 2, certaines normales ont une contribution « positive », et d'autres ont une contribution « négative », à condition de bien respecter l'ordre des sommets du polygone. La normale obtenue a pour longueur la surface de la face.

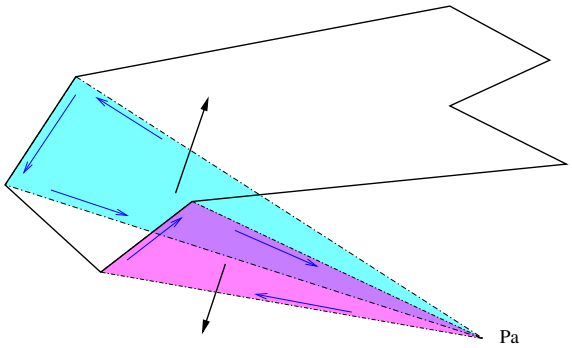


FIG. 2 – Principe du calcul des normales des faces

En pratique, on prend le point « arbitraire » P_a comme étant le centre de gravité des sommets du polygone, ceci afin de limiter les problèmes de précision dus aux erreurs de troncature et de s'assurer que le point choisi est bien dans le plan du polygone. Pour le polygone de la figure 2, on obtient les triangles suivants :

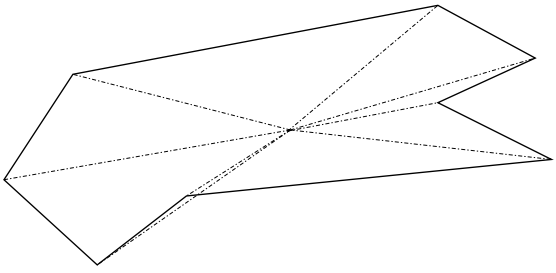


FIG. 3 – Triangles pour le calcul des grandeurs aux faces

Le centre de gravité G d'une face est égal au barycentre des triangles T_i définis par $\{P_a, P_i, P_{i+1}\}$ et dont les barycentres sont notés G_i. Soit O le centre du repère de calcul et $\overrightarrow{n_f}$ la normale associée à la face, on a :

$$\overrightarrow{OG} = \frac{\sum_{i=1}^{n} \text{surf}(T_i).\overrightarrow{OG_i}}{\sum_{i=1}^{n} \text{surf}(T_i)} \qquad \text{avec} \quad \text{surf}(T_i) = \frac{\overrightarrow{n_{T_i}}.\overrightarrow{n_f}}{\| \overrightarrow{n_f} \|}$$

Il est important de bien tenir compte du signe associé à la surface de chaque triangle pour que la formulation ci-dessous reste valide pour des faces non convexes.

On remarquera qu'en réalité, certaines faces ne sont pas parfaitement planes. Dans ce cas, le calcul comportera une légère erreur, mais il est difficile de définir ce qu'est réellement la surface d'une face polygonale lorsque toutes ses arêtes ne sont pas dans le même plan. On pourrait par exemple considérer qu'il s'agit de la plus petite surface limitée par ses arêtes (i.e. la surface que l'on obtiendrait avec une maquette des arêtes et un film d'eau savonneuse), ou bien qu'il s'agit de la surface définie par la face découpée en triangles (avec une triangulation respectant le critère de Delaunay dans un plan médian de la face, lui-même à préciser), ou choisir encore une autre définition.

Pour limiter les erreurs dues aux faces gauches, on compare la contribution aux volumes des cellules voisines de la face à la contribution (calculée à partir de la formule de Stokes) obtenue à partir des triangles $\{P_a, P_i, P_{i+1}\}$, et l'on déplace le centre de gravité initialement calculé dans l'axe de la normale à la face de manière à obtenir la même contribution.

6.1.2 Centres de gravité des cellules

Si l'on considère qu'en théorie, la méthode VF travaille sur des grandeurs constantes par maille, il n'est pas indispensable pour le Noyau de connaître avec exactitude le centre de gravité d'une cellule donnée, un point arbitraire contenu dans la cellule en question devant suffire. En fait, pour la précision, la position du point assimilé au centre de gravité est importante, car elle intervient dans le calcul des valeurs et gradients aux faces.

Considérons une cellule $\mathcal{C}$ comportant p faces de centres de gravité G_k et de surfaces de norme S_k. Soit O l'origine du repère de calcul, le centre de gravité G de $\mathcal{C}$ est défini par :

$$\overrightarrow{OG} = \frac{\sum_{k=1}^{p} S_k . \overrightarrow{OG_k}}{\sum_{k=1}^{p} S_k}$$

Si la cellule $\mathcal{C}$ comporte q sommets de coordonnées X_l, une autre possibilité de définition du centre de gravité G de $\mathcal{C}$ est la suivante :

$$\overrightarrow{OG} = \sum_{l=1}^{q} \frac{\overrightarrow{OX_l}}{q}$$

Il est conseillé d'utiliser le calcul par défaut basé sur les centres de gravité des faces ; la deuxième définition correspond à celle qui était utilisée par les versions 1.0 de *Code_Saturne*, et la possibilité d'y revenir n'a été conservée que par précaution,[9] et afin de pouvoir comparer des calculs effectués avec diverses versions de *Code_Saturne*.

Sur la figure 4, on représente le centre de gravité calculé selon les deux algorithmes sur une cellule exemple bidimensionnelle (ou 3D issue d'une extrusion). À gauche, la cellule n'est pas altérée ; à droite, on a ajouté des sommets supplémentaires sur une face de la cellule, comme si cette cellule se trouvait sur la frontière d'un recollement conforme (cf. § 6.2, page 23). On voit que la position du centre de gravité est stable selon la formulation basée sur les centres de gravité des faces, alors que ce point est décalé vers la face à laquelle on a ajouté des sommets avec la formulation basée sur le centre de gravité des sommets.

On montre l'effet possible sur la position des segments joignant les centres de gravité de plusieurs cellules dans une situation de recollement conforme sur la figure 5.

On voit ici que l'ancien algorithme basé sur les centres de gravité des sommets à tendance à accentuer les non-orthogonalités des faces, par rapport à l'algorithme actuel basé sur le barycentre des faces. C'est la principale

[9] On craignait que le calcul basé sur les centres de gravité et surfaces des faces ne puisse poser problème dans le cas de maillages extrudés de faible épaisseur ou semblables, comportant des faces de surfaces très faibles par rapport à d'autres. On pouvait aussi s'inquiéter du comportement de l'algorithme lorsque l'on a des faces gauches, et que les positions des centres de ces faces sont incertaines d'un point de vue théorique. Jusqu'ici, il semble que cet algorithme n'a jamais provoqué des difficultés, et il améliore plutôt la convergence dans certains cas comportant des faces gauches.

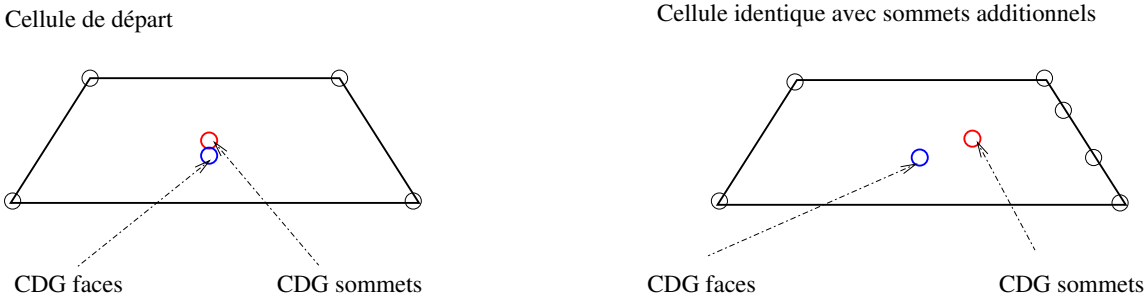


FIG. 4 – Centre de gravité d'une cellule

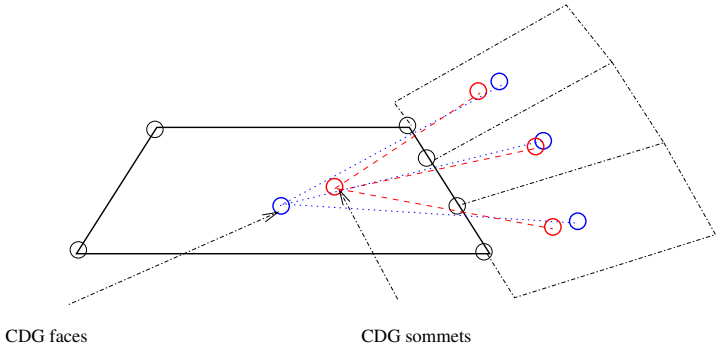


FIG. 5 – Orthogonalité des faces selon le centre de gravité

raison pour laquelle cet algorithme est déconseillé, et n'est activable que par positionnement d'une variable d'environnement, à fin d'utilisation exceptionnelle par les développeurs et experts.

Remarque : il n'y a pas de calcul d'un centre égal au centre de la sphère circonscrite, qui n'existe pas pour tous les types de cellules.

6.2 Recollement conforme

Le recollement de maillages non conformes constitue une des fonctionnalités les plus importantes du Préprocesseur, et les algorithmes associés sont les plus complexes de ce module. Le principe est de construire de nouvelles faces correspondant aux intersections des faces à recoller, et de remplacer les faces d'origine par leur recouvrement avec des faces issues de cette intersection, comme le montre la figure 6 (en coupe latérale 2D) :

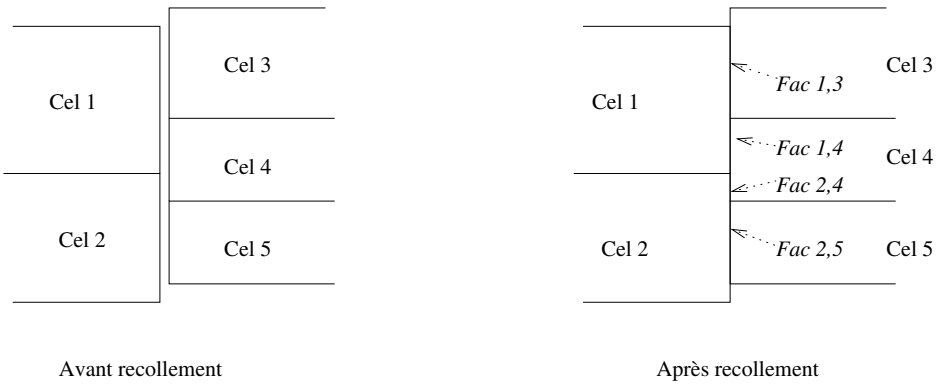
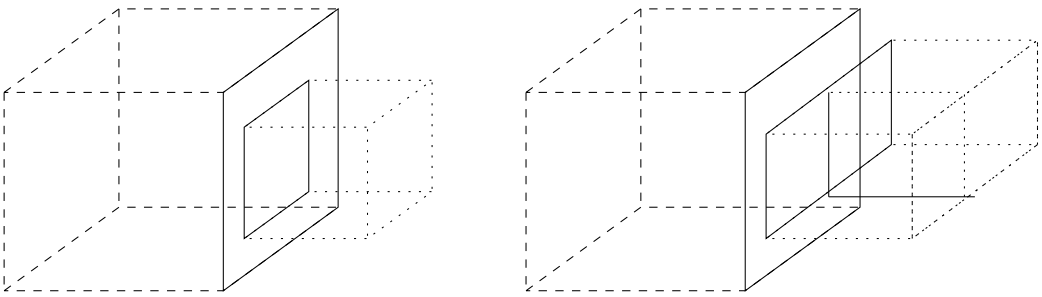


FIG. 6 – Principe du recollement conforme

On parle de *recollement conforme*, car le maillage issu de ce recollement est bien conforme, la où le maillage

d'origine ne l'était pas.

Le nombre de faces d'une cellule dont un bord a été recollé sera supérieur ou égal au nombre de faces d'origine, et rien ne garantit que les nouvelles faces issues du recollement ne soient des triangles ou des quadrangles, même si les toutes les faces d'origine étaient de l'un de ces types. C'est pourquoi le modèle de données du Préprocesseur (ainsi que celui du Noyau) doit être adapté à la représentation de faces polygonales et de cellules polyédriques quelconques, ou presque. En effet, on se limite à des faces polygonales sans trous, comme illustré par la figure 7. [10]



a) Définition impossible b) Définition possible

FIG. 7 – Cas de recollement possibles

6.2.1 Présentation des aspects influant sur la robustesse

On a cherché à faire en sorte que l'algorithme de recouvrement puisse fonctionner avec un minimum de paramètres utilisateur, et puisse traiter une grande variété de cas.

Plusieurs critères sont importants :

1. **déterminisme** : il est important de pouvoir prédire le comportement de l'algorithme. Notamment, on a cherché à construire un algorithme qui produira le même résultat indépendamment de la numérotation des mailles. Si ce critère n'était pas respecté, le recollement d'un maillage A concaténé avec un maillage B pourrait être différent du recollement de B concaténé avec A. L'algorithme final peut présenter une dépendance aux erreurs de troncature sur de rares cas, mais cet impact est très limité et l'algorithme théorique est bien indépendant de la numérotation (et donc l'ordre de parcours) des diverses entités de maillage. L'utilisateur n'a donc pas à se poser de questions sur l'ordre d'entrée des maillages par rapport au recollement conforme.[11]
2. **surfaces non planes** : On veut pouvoir recoller à la fois des surfaces courbes et des surfaces correspondant à un ensemble de plans, mais dont la normale n'est pas nécessairement une fonction continue de l'espace.
3. **jeu entre les maillages** : Les surfaces à recoller peuvent ne pas être strictement confondues. Ceci peut provenir de problèmes d'arrondis lors de la génération des maillages, de cotes prises avec un nombre de décimales différentes, ou simplement de l'approximation d'une surface courbe par un ensemble de faces planes.

6.2.2 Principe de base

Considérons deux surfaces à recoller :

[10] La connectivité faces-arêtes utilisée permettrait des faces avec trous, en supprimant simplement l'hypothèse selon laquelle le parcours des arêtes dans l'ordre de leur définition correspond au parcours du bord de la face dans le sens direct. Par contre, la connectivité faces-sommets ne pourrait plus être représentée simplement, et ces faces ne pourraient donc être visualisées sans découpage ni avec *EnSight* ni avec un outil basé sur la librairie *VTK* (et encore moins avec *I-deas*). Certains algorithmes devraient être en outre être modifiés, notamment le découpage de faces en triangles, et le calcul des normales et centres de gravité.

[11] La géométrie issue d'un recollement est en théorie indépendante de l'ordre d'entrée des maillages, mais la numérotation des mailles ne l'est pas. Un ordre d'entrée utilisé pour un calcul doit donc être conservé pour toute reprise de ce calcul.

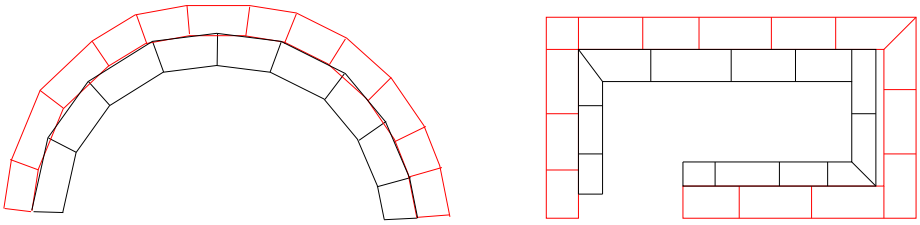


FIG. 8 – Surfaces initiales

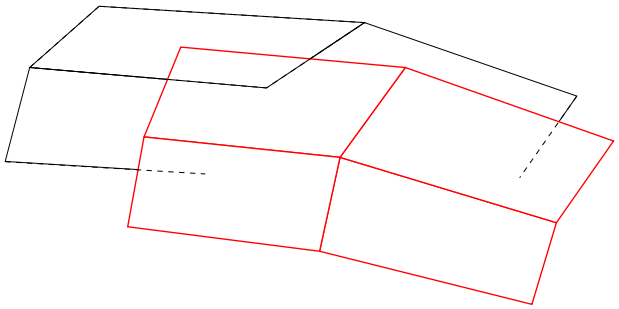


FIG. 9 – Recollement de deux surfaces

On cherchera à déterminer les intersections entre les arêtes de ces deux faces, et découper les arêtes en sous-arêtes s'appuyant sur ces intersections. On décrira par la suite ce que l'on entend par « intersection » entre arêtes, la notion utilisée ici étant relative à un voisinage de l'arête.

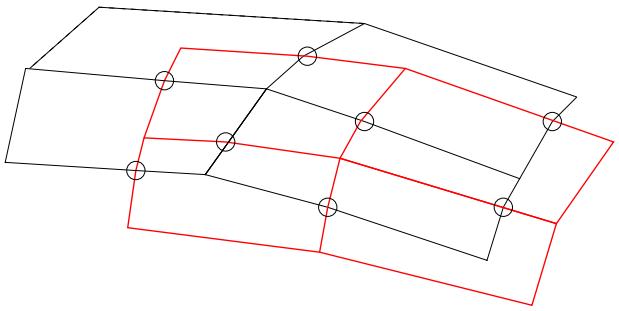


FIG. 10 – Après intersection des arêtes

Ensuite, on reconstruira des sous faces issues de chaque face initiale. Pour ce faire, en partant d'une arête d'une face d'origine, on cherchera à construire des cycles fermés, en choisissant à chaque sommet l'arête connectée à ce sommet pointant le plus à gauche (vu du plan de la face, debout sur la face normale vers le haut et en regardant dans la direction de l'arête courante), jusqu'au point de départ, de manière à construire le cycle le plus court possible en tournant dans le sens trigonométrique. Chaque face à recoller est ainsi remplacée par son recouvrement constitué de sous-faces construites de cette manière.

Lors du parcours des cycles on fera attention à rester proche du plan de la face d'origine. On ne considérera que les arêtes appartenant à des faces dont la normale a une direction voisine de la face en cours de subdivision (i.e. le produit scalaire des normales unitaires doit être proche de 1 en valeur absolue).

Une fois construites toutes les sous-faces, on devrait avoir pour deux faces initiales en vis-à-vis deux sous-faces topologiquement identiques, héritant chacune d'une des faces initiales et ainsi connectées à une cellule différente. Il suffit alors de fusionner les faces topologiquement identiques, et la face issue de la fusion sera connectée à deux cellules. Les faces de bord fusionnées sont donc remplacées par des faces internes, et le recollement est effectif.

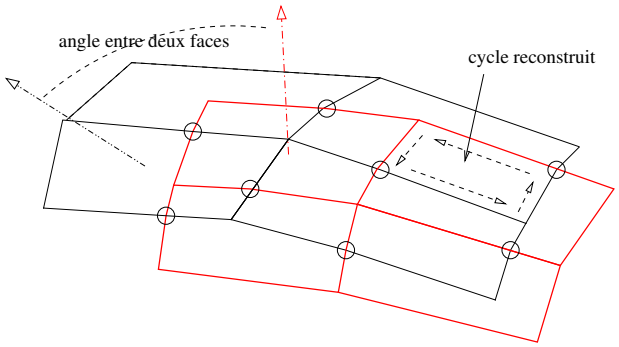


FIG. 11 – Construction de sous-faces

6.2.3 Simplification du recouvrement

Pour le *Code_Saturne*, il est préférable d'éviter d'avoir des faces de dimensions très différentes appartenant à une même cellule. Pourtant, cela peut se produire facilement lorsque l'on découpe les faces de bord non conformes afin de reconstruire une interface conforme. Sur l'exemple de recouvrement ci-dessous, on voit que si l'on découpe les faces en fonction de leur recouvrement avec d'autres faces non-conformes, on sera amené à créer des faces beaucoup plus petites que d'autres.

Il est possible de simplifier le recouvrement de faces de manière à éviter ou du moins réduire ce phénomène, en déplacant légèrement certains des nœuds des maillages de part et d'autre du recouvrement. On cherchera à déplacer les noeuds de manière à simplifier le recouvrement tout en déformant le moins possible le maillage.

Un exemple de recouvrement est représenté ci-dessous. On y montre quelques possibilités de simplification. On remarquera que toutes ces possibilités sont associées à une même arête. Les possibilités de même type associées à d'autres arêtes ne sont pas représentées afin de ne pas surcharger la figure :

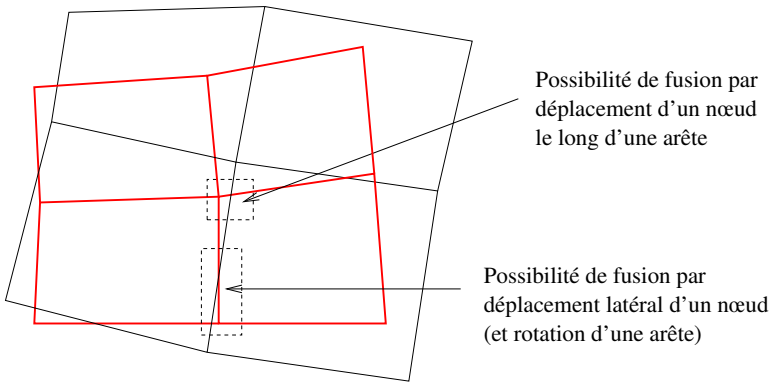


FIG. 12 – Possibilités de simplification

Après simplification, on a la situation suivante :

6.2.4 Traitement effectué

Le point de départ de l'algorithme est la recherche d'intersections entre arêtes. En 3D, on ne se limitera pas aux « vraies » intersections, mais on considérera que l'on a une intersection dès que la distance minimale entre deux arêtes passe en dessous d'un certain seuil.

À chaque sommet, on associera une distance maximale, proportionnelle à la longueur de la plus petite arête connectée à ce sommet. Cette distance sera toujours strictement inférieure à la moitié de la longueur d'une arête (la valeur maximale autorisée pour la sous-option `-fraction` étant 0.49). On considérera un voisinage d'une

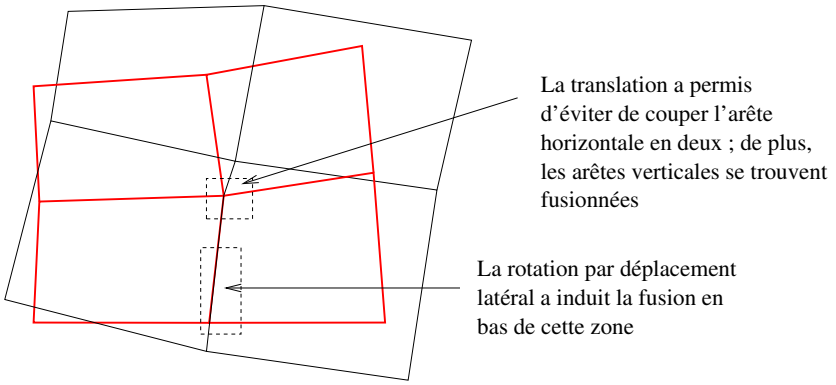


FIG. 13 – Faces après simplification

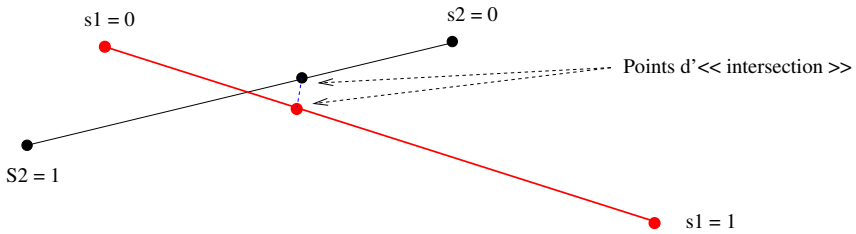


FIG. 14 – Intersection d'arêtes dans l'espace

arête constitué de l'enveloppe rejoignant les deux sphères associées. En un point de l'arête d'abscisse relative s, ce voisinage sera donc la sphère de rayon $d_{max}(s) = (1-s)d_{max}\mid_{s=0} + s.d_{max}\mid_{s=1}$.

On aura alors intersection entre arêtes $A1$ et $A2$ dès que le point de $A1$ le plus proche de $A2$ se trouve à l'intérieur du voisinage associé à $A2$, et que simultanément, le point de $A2$ le plus proche de $A1$ se trouve simultanément dans le voisinage associé à $A1$.

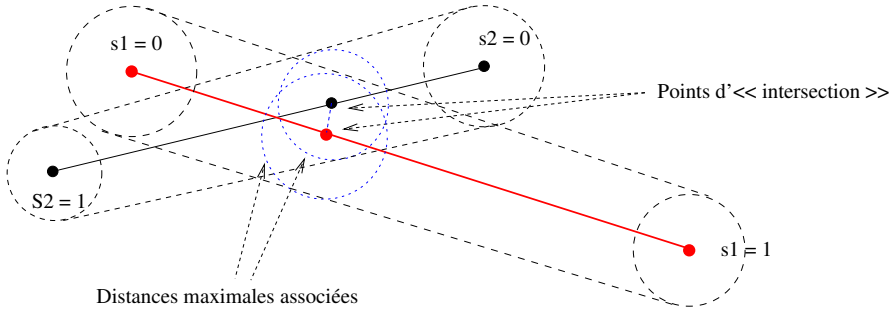


FIG. 15 – Tolérances associées aux intersections d'arêtes

Si l'arête $A2$ passe dans le voisinage associé à un sommet de l'arête $A1$, et que ce sommet se trouve dans le voisinage associé à $A2$, on prendra pour définir l'intersection ce sommet plutôt que le point de $A1$ le plus proche de $A2$. Ceci évitera de découper inutilement les arêtes par la suite. On recherchera donc en priorité les « intersections » sommet-sommet, sommet-arête, puis arête-arête. Si les voisinages associées à deux arêtes s'intersectent, mais que le critère : $\exists P1 \in A1, \exists P2 \in A2, d(P1,P2) < min(d_{max}(P1), d_{max}(P2))$ n'est pas respecté, on n'aura pas intersection. Ces cas sont représentés ici (vus de côté) :

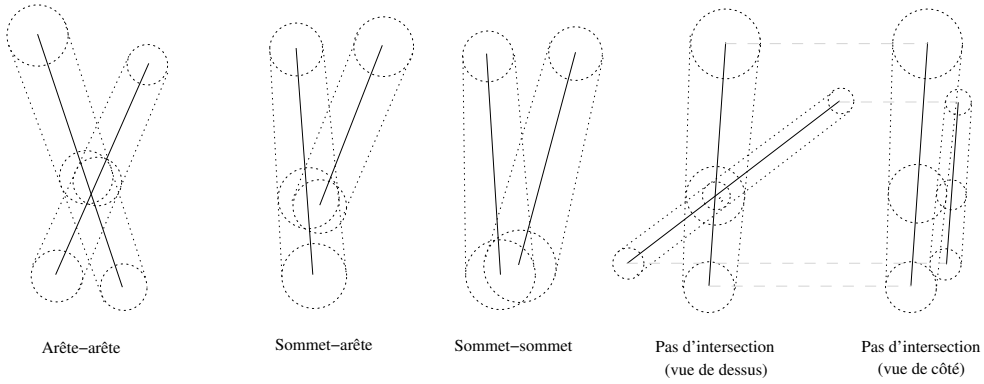


FIG. 16 – Types d'intersection

6.2.5 Problèmes associés à la fusion de sommets voisins

Si l'on a déterminé qu'un sommet S_1 doit être fusionné avec un sommet S_2, et qu'indépendamment, ce sommet S_2 doit être fusionné avec un sommet S_3, alors par transitivité, S_1 et S_3 devraient être agglomérés, même si ces sommets ne partagent aucune intersection. On illustre figure 17 des situations théoriques pouvant mener à ce problème.

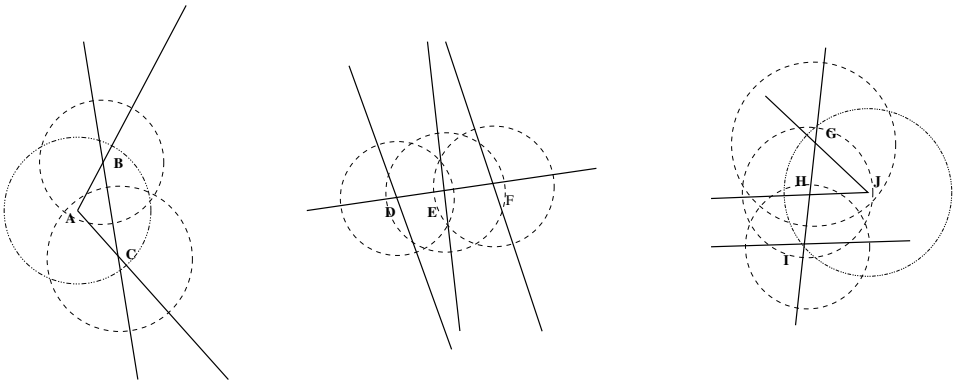


FIG. 17 – Transitivité de la fusion

Sur la figure 18, on représente des cas plus représentatifs de ce que l'on peut obtenir couramment avec des maillages réels, compte tenu de la prise en compte des tolérances locales.

La figure 19 montre l'impact d'une combinaison de fusions sur des sommets appartenant à une même arête. On voit que dans ce cas, les arêtes passant initialement par les sommets G et J sont très fortement déformées (i.e. découpées en arêtes très mal alignées). Sans transitivité, des arêtes passant par des sommets issus des seules fusions de (G, H) d'une part et (L, J) d'autre part seraient beaucoup plus proches des arêtes d'origine.

Pour éviter des simplifications excessives du maillage liées à des enchaînements de fusions, on vérifie pour chaque arête si des situations de ce type se produisent. Si c'est le cas, on calcule localement un facteur multiplicatif (inférieur à 1) des tolérances de fusion/intersection pour les sommets appartenant à cette arête de manière à déterminer un maximum de fusions entre sommets d'intersection avec cette arête sans aller jusqu'à provoquer des fusions non désirées par transitivité.

Finalement, sur la figure 20, on montre l'effet possible de la transitivité de fusions sur des sommets appartenant à plusieurs arêtes. Ici, même si la réduction de tolérance locale permet d'éviter la fusion des sommets (G, I) par combinaison des regroupements (G, H) et (H, I), cette fusion pourrait être provoquée par transitivité des agglomérations (H, I), (H, J), et (G, J). De plus, I et J seraient alors fusionnés alors qu'il ne devraient pas l'être.

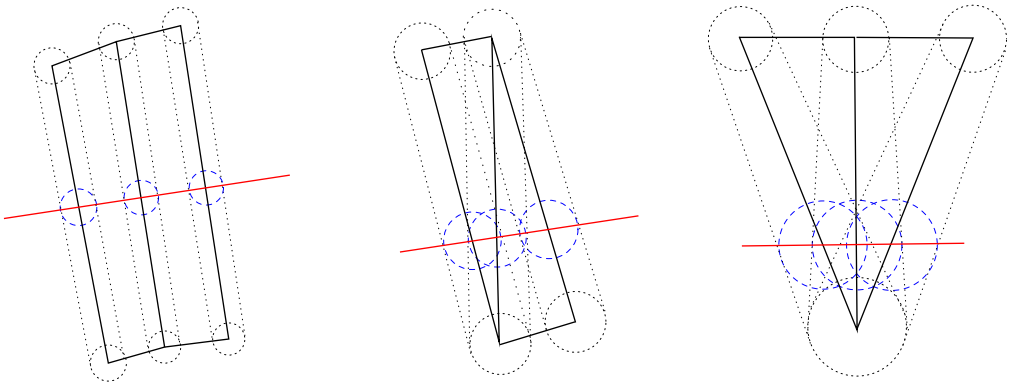


FIG. 18 – Transitivité de la fusion (cas réels)

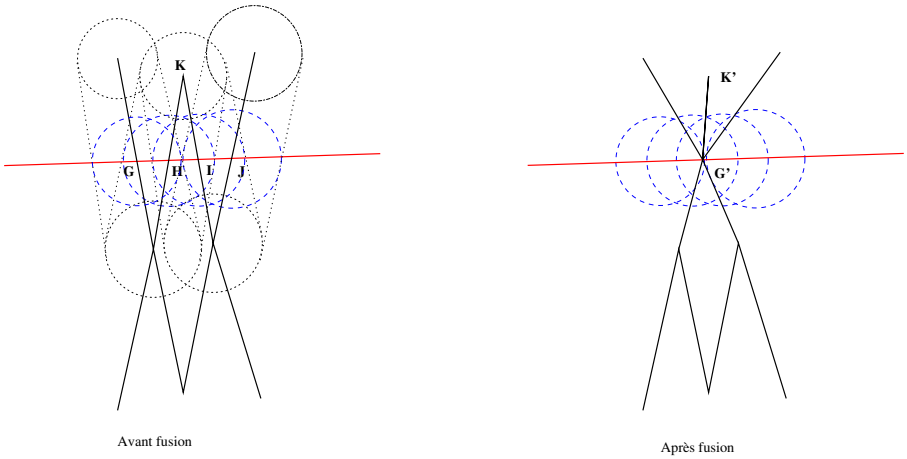


FIG. 19 – Transitivité de la fusion sur une arête

Dans cet exemple précis, la réduction locale de tolérance sur chaque arête devrait éviter la fusion (H, I), car les tolérances associées à G et H sont plus grandes que celle associée à I ; la réduction locale de tolérance devrait donc conserver la fusion (G, H) et supprimer la fusion (H, I). On n'aura donc pas d'agglomération (I, J) par transitivité, et seuls (G, H, J) seront fusionnés.

Cependant, dans la configuration de la figure 21, où les tolérances associées aux sommets G, H, et I sont différentes de celles présentées figure 20, la réduction de la tolérance locale évitera initialement la fusion (G, H) en conservant (H, I), dont la combinaison avec (H, J) provoquera en fin de compte une agglomération de (G, H, I, J) en G'.

Ce dernier cas n'est pas traité, la réduction de tolérance locale ne se faisant qu'au niveau des arêtes. Pour traiter tous les cas en maintenant l'indépendance théorique de l'algorithme vis-à-vis de l'ordre de parcours des faces et arêtes, il serait possible d'effectuer plusieurs passes, en réduisant définitivement les tolérances locales. Cependant, s'il n'est pas gênant vis-à-vis du parcours des arêtes d'une face pour la reconstruction de faces conformes de ne pas fusionner certains de ses sommets (le parcours n'étant pas fondamentalement modifié), une réduction de tolérance inter-arêtes reviendrait par contre à supprimer certaines intersections détectées. Ceci aurait un impact moins prévisible sur le résultat de l'algorithme. L'algorithme actuel n'ayant pas à ce jour posé de problèmes de robustesse qui n'aient pu être traités par un ajustement de la tolérance par défaut, on a préféré ne pas le rendre plus complexe, sachant que la non-transitivité assurée des intersections sur une même arête semble éviter la grande majorité des problèmes.

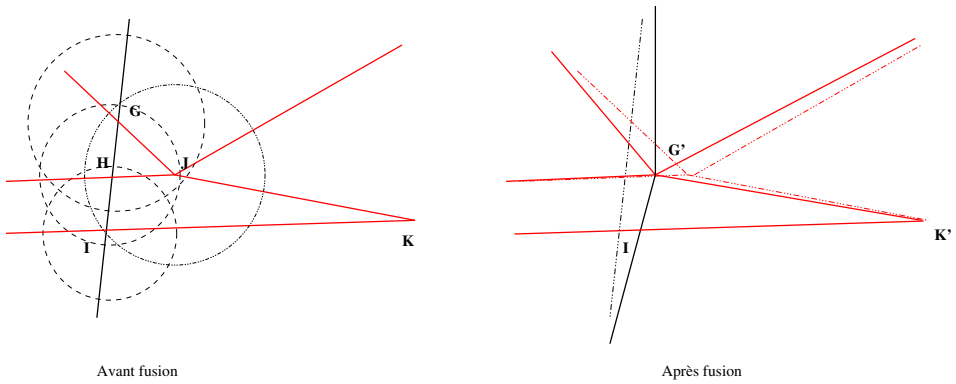


FIG. 20 – Transitivité évitée de la fusion sur plusieurs arêtes

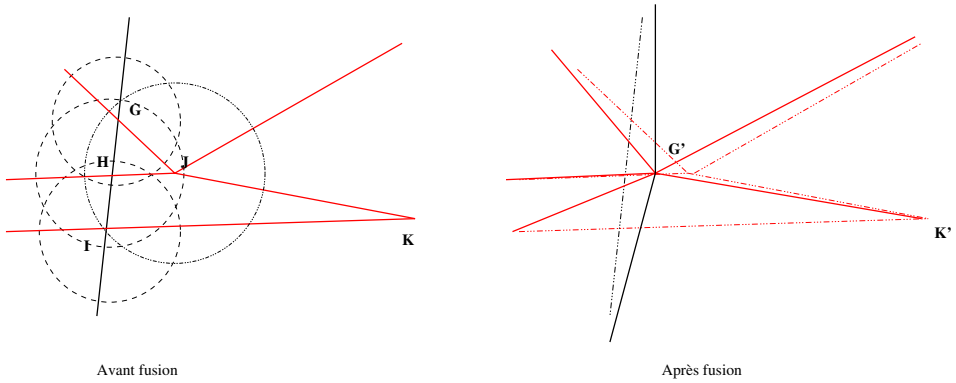


FIG. 21 – Transitivité de la fusion sur plusieurs arêtes

6.2.6 Optimisation de l'algorithme

Un certain nombre de facteurs influent sur les performances à la fois en coût calcul et mémoire. On cherchera toujours à optimiser les deux, en cherchant toujours à ne pas augmenter trop sensiblement le coût mémoire d'une exécution du Préprocesseur avec recollement par rapport à une exécution sans recollement.

Lors de la recherche d'intersections entre arêtes, on cherche à limiter au maximum le nombre d'essais. On calcule les boîtes englobantes associées aux arêtes, et on n'appelle la fonction de calcul d'intersection que si ces boîtes s'intersectent, dans quel cas la probabilité d'avoir une intersection est élevée. Ces boîtes seront alignées avec les axes, et de taille légèrement supérieures au minimum nécessaire pour contenir les arêtes et le voisinage associé à la tolérance :

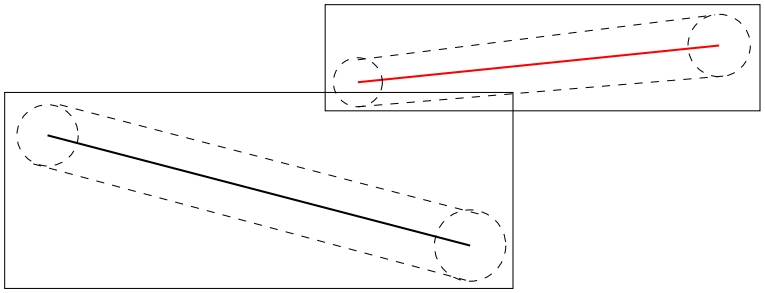


FIG. 22 – Boîtes englobantes pour les arêtes

On effectuera alors une boucle sur les arêtes des faces sélectionnées, avec un algorithme hybride dichotomie-

parcours pour rechercher les arêtes pouvant intersecter l'arête courante. Soit i l'indice de l'arête courante, on ne cherchera des intersections qu'avec des arêtes d'indice j tel que $j > i$, les autres ayant normalement dues être trouvées lorsque j était l'arête courante.

On pourrait aussi éviter de rechercher des intersections entre arêtes partageant un sommet. Par exemple, sur le schéma suivant, il n'est pas nécessaire de rechercher d'intersection entre les arêtes $A2$ et $A4$, ni entre $A3$ et $A4$, car on devra dans tous les cas détecter l'intersection de type « sommet-sommet » entre $A1$ (qui partage ce sommet avec $A2$ et $A3$) et $A4$.

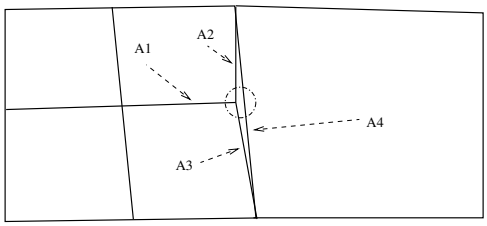


FIG. 23 – Limitation possible des tests d'intersection d'arêtes

Les intersections du même type que celle reliant le milieu de $A4$ avec un sommet de $A2$ devraient donc être détectées indirectement. Ceci devrait presque toujours être le cas, à moins de construire des cas pathologiques, tels que l'on pourrait avoir sur le schéma ci-dessus en supprimant l'arête $A1$, mais en conservant $A2$, $A3$, et $A4$. On aurait alors des faces géométriquement superposées, mais ne comportant pas le même nombre de sommets et d'arêtes.

Un cas plus probable est celui où l'on recolle des quadrangles déformés avec des triangles. Le problème ne se pose qu'aux bords des surfaces à recoller, et uniquement si des sommets sont communs aux deux surfaces (ce qui peut se produire si l'on a concaténé ces maillages sous un outil tel que *I-deas* ou *SIMAIL*, et fusionné les nœuds voisins) :

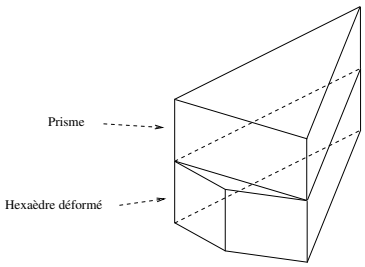


FIG. 24 – Problème avec la limitation des tests d'intersection

Pour pouvoir traiter malgré tout ce type de cas, on doit permettre la recherche d'intersections entre arêtes possédant un sommet commun. Ce type de recherche n'ést en général pas utile sur les maillages testés, mais on aussi observé que le surcoût induit par ce type de test était négligeable à l'échelle de l'ensemble des opérations liées au recollement, et on l'effectue doc systématiquement.

6.2.7 Influence sur la qualité du maillage

Il est préférable pour le noyau du *Code_Saturne* que le maillage soit le plus « orthogonal » possible (une face est parfaitement orthogonale lorsque le segment rejoignant son centre de gravité au centre d'une cellule voisine quelconque elle appartient est aligné avec sa normale). Il est tout aussi important d'éviter des faces non planes. [12]

[12] Le calcul des centres de gravité des faces comporte une correction de manière à ce que la contribution d'une face gauche au calcul du volume soit la même que celle des triangles joignant le centre de gravité des sommets aux arêtes de la face, de manière à ne pas commettre d'erreur sur le calcul des grandeurs linéaires, mais il n'est pas certain que cette correction suffise pour les grandeurs d'ordre 2.

Par principe même, des faces initialement orthogonales seront découpées en plusieurs faces non orthogonales lors d'un recollement conforme. De plus, plus la tolérance utilisée est importante, plus la fusion de sommets voisins tendra à entraîner un gauchissement des faces latérales à celles recollées.

Il est donc important d'être sensible à ces effets, et d'être conscient qu'un maillage issu du recollement de maillages héxaédriques parfaitement réguliers peut être de mauvaise qualité, notamment si l'on a utilisé un paramètre de tolérance de fusion lâche et que les raffinements des maillages recollés sont très différents. Il est donc fortement conseillé de visualiser les critères de qualité sur le maillage final, et d'éviter les recollements à l'intérieur des zones sensibles du maillage.

7 Recherche de faces périodiques

La construction de structures associées à la périodicité se fonde sur le recollement périodique. Le principe de base, décrit par la figure 25, est le suivant :

- Soit un ensemble de faces de bord sélectionné, on duplique l'ensemble de ces faces (ainsi que les arêtes et sommets associés) et l'on déplace la copie selon le pas de périodicité indiqué par l'utilisateur. On conserve un lien d'historique entre les entités dupliquées et les entités d'origine.
- On construit un recollement conforme sur l'ensemble $\{faces\ sélectionnées \cup faces\ dupliquées\}$. Ce recollement déformera légèrement le maillage de manière à pouvoir fusionner des sommets très proches les uns des autres, et découpera les faces en vis-à-vis en sous-faces conformes (si les faces périodiques ne le sont pas).
- On applique si nécessaire le découpage des faces copiées puis recollées aux faces dont elles sont la copie.
- On supprime les entités dupliquées qui n'ont pas participé au recollement.

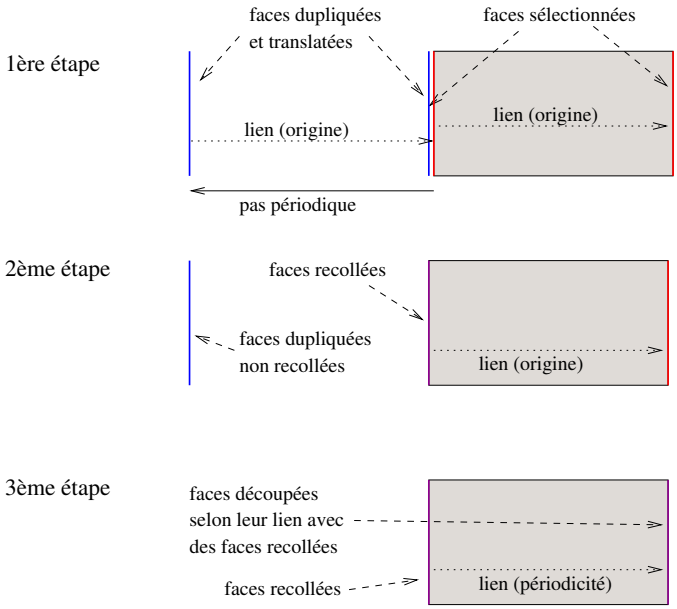


FIG. 25 – Principe du recollement périodique

Il n'est donc pas nécessaire pour utiliser des conditions de type périodicité que les surfaces périodiques soient maillées de manière identique, même s'il est préférable de ne pas abuser des possibilités de ce type au regard de la qualité du maillage de calcul.

On remarquera qu'il peut sembler plus simple de séparer les faces périodiques en un ensemble de faces « amont » et un ensemble de faces « aval », en ne dupliquant et transformant que le premier ensemble, afin d'optimiser la recherche de correspondants et de ne pas avoir à supprimer en fin de traitement des faces copiées n'ayant pas participé à la périodicité. L'algorithme utilisé a été conçu pour permettre une sélection des faces périodiques au moyen de sous-options de ligne de commande semblables à celles utilisées pour le recollement conforme. Comme pour le recollement conforme, la spécification d'un critère de sélection afin de ne traiter que les faces

effectivement périodiques ne s'avère en général pas utile sur un maillage relativement simple.

8 Découpage de faces en triangles

Le découpage de faces en triangles se fait en deux passes. La première passe est un algorithme du type *ear cutting*, qui peut découper tout type de polygone à peu près plan, convexe ou non. Le découpage généré est arbitraire, et dépend notamment du sommet du polygone choisi pour le début du parcours.

La deuxième passe consiste à permuter des arêtes deux à deux jusqu'à ce que le découpage soit de type Delaunay contraint, ce qui résulte généralement en un découpage plus régulier.

8.1 Découpage initial

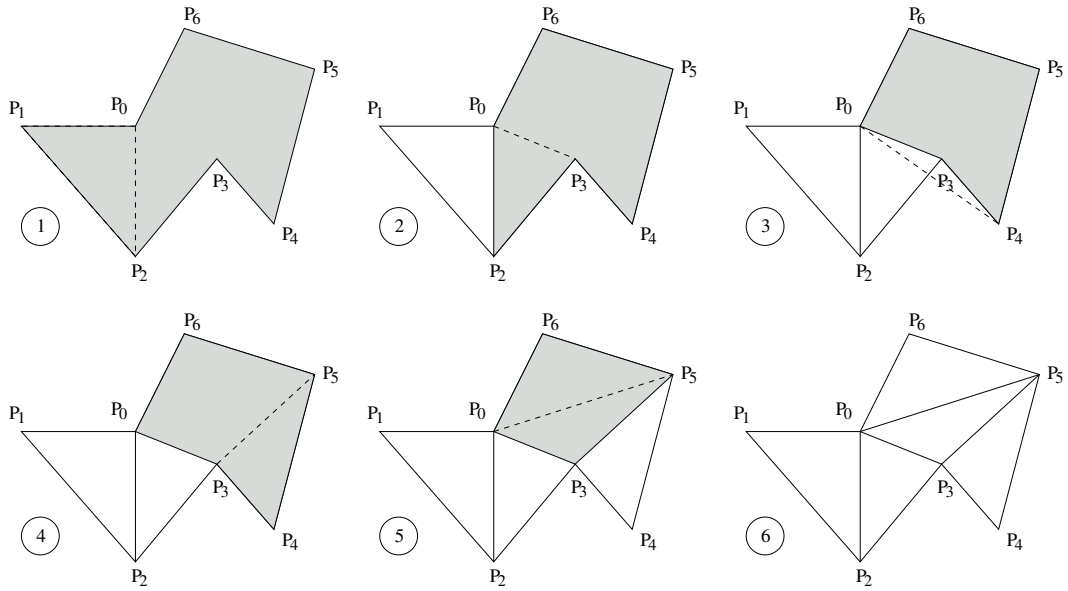


FIG. 26 – Principe du découpage de faces en triangles

L'algorithme est basé sur celui décrit dans [2]. Son principe est illustré figure 26. On commence par vérifier si le triangle constitué des premiers sommets du polygone, (P_0, P_1, P_2) est une « oreille » du polygone, c'est à dire s'il est intérieur au polygone et ne l'intersecte pas. Comme c'est le cas sur cet exemple, on obtient un premier triangle, et on doit traiter la partie restante du polygone. À l'étape suivante, le triangle constitué des premiers sommets du polygone restant, (P_0, P_2, P_3) constitue lui aussi une oreille, et peut être supprimé.

Par contre, on voit à l'étape 2 que le prochain triangle canditat au découpage, (P_0, P_3, P_4) n'est pas une oreille, car il n'est pas contenu dans le polygone restant. On décale alors les indices des sommets à considérer, et on voit étape 4 que le triangle (P_3, P_4, P_5) est bien une oreille et peut être supprimé.

L'algorithme est construit de manière telle qu'un triangle est sélectionné par rapport au dernier sommet du dernier triangle considéré (en partant du triangle (P_0, P_1, P_2)). Ainsi, on considère étape 5 le triangle se terminant par P_5, soit (P_0, P_3, P_5). Une fois ce triangle supprimé, le polygone restant est un triangle, et son traitement est immédiat.

8.2 Amélioration du découpage

On montre sur les figures 27 et 28 deux exemples du découpage produit sur deux polygones semblables mais dont les sommets sont numérotés de manière différente.

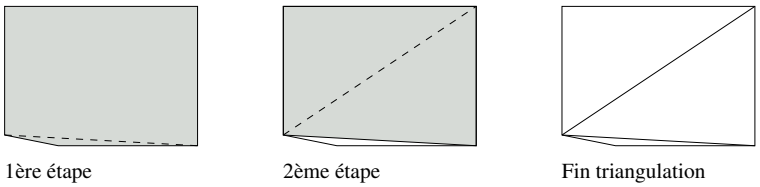


FIG. 27 – Exemple de découpage (1)

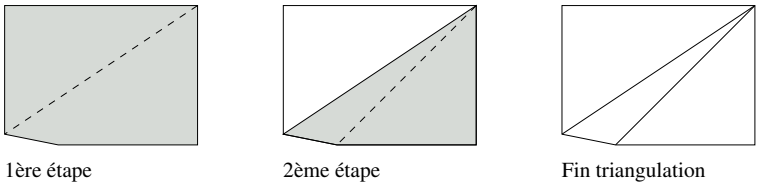


FIG. 28 – Exemple de découpage (2)

Non seulement le découpage obtenu n'est pas le même, mais il a tendance à produire des triangles très aplatis. Une fois un premier découpage obtenu, on applique donc un algorithme correctif, consistant à permuter des arêtes séparant deux triangles adjacents deux par deux de manière à respecter le critère de Delaunay.

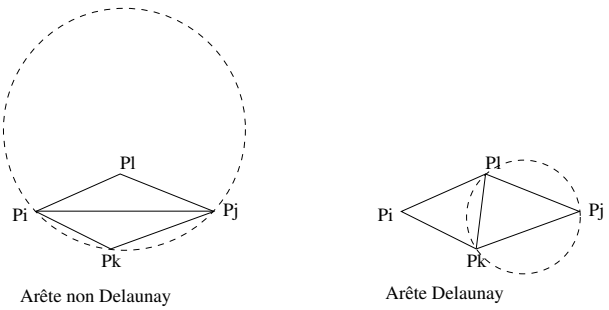


FIG. 29 – Critère de Delaunay (2)

Ce critère est illustré figure 29. Dans le premier cas, l'arête $\overline{P_i P_j}$ ne respecte pas le critère, car le sommet P_l est contenu dans le cercle passant par P_i, P_j, P_k. Dans le second cas, l'arête $\overline{P_k P_l}$ respecte bien ce critère, P_i n'étant pas contenu dans le cercle passant par P_j, P_k, P_l.

Si les triangles (P_i, P_k, P_j) et (P_i, P_j, P_l) étaient issus du découpage initial d'un même polygone en triangles, on les remplacerait donc par les triangles (P_i, P_k, P_l) et (P_l, P_k, P_j), qui respectent le critère de Delaunay. Dans le cas d'un quadrangle, on aurait terminé, mais avec un polygone de départ plus complexe, ces nouveaux triangles pourraient éventuellement être remplacés par d'autres, en fonction de leurs autres voisins appartenant au polygone.

Sur les figures 30 et 31, on illustre cet algorithme sur les mêmes exemples que précédemment (figures 27 et 28). On voit bien que le résultat final est le même. En théorie, l'algorithme de transformation de la triangulation pour respecter l'algorithme de Delaunay converge toujours. Pour éviter des problèmes dus à des erreurs de troncature, on applique une tolérance avant de décider de permuter deux arêtes. Ainsi, on accepte que le découpage final puisse simplement « presque » respecter le critère de Delaunay.

Dans certains cas, notamment celui de carrés parfaits, deux découpages en triangles différents respectent le critère de Delaunay, aux erreurs de troncature près. Pour la périodicité, on doit éviter que deux faces périodiques puissent être découpées de manière non périodique (par exemple selon les deux diagonales sur un quadrangle). On découpe donc une des faces, et on applique ce découpage à sa face périodique, plutôt que de découper les deux faces indépendamment.

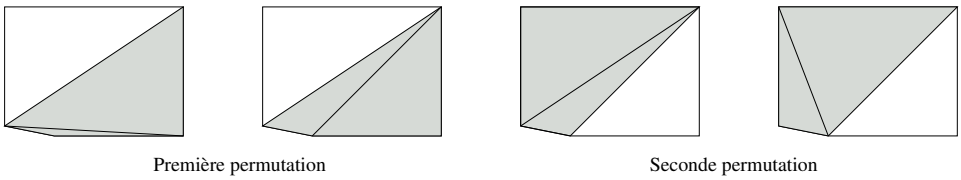


FIG. 30 – Exemple de correction Delaunay (1)

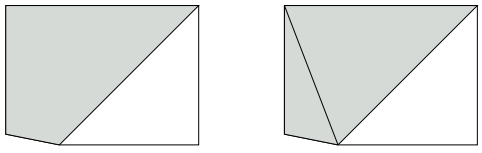


FIG. 31 – Exemple de correction Delaunay (2)

9 Perspectives d'évolution

Le Préprocesseur est issu du module Enveloppe des versions 1.0 à 1.2 de *Code_Saturne*, duquel le couplage SYRTHES et le post-traitement ont été supprimés (car reportés vers le Noyau à l'aide de la librairie FVM).

Dans les spécifications et fonctionnalités proposées initialement [1] on envisageait plusieurs façons de relier l'Enveloppe et le Noyau, ainsi que la transformation de la structure de données sous plusieurs formes. En pratique, l'Enveloppe se présentait sous la forme d'un exécutable séparé du Noyau, qui communiquait via des messages échangeables de diverses manières. Cet exécutable est séquentiel, alors que le Noyau est parallèle, et seules deux formes de la structure de données sont gérées : une connectivité nodale initiale, et une connectivité descendante principale. Surtout, cet outil participait à la parallélisation du code, mais n'a pas été lui-même parallélisé, faute de priorité avec des moyens limités.

On remarquera aussi que l'Enveloppe était envisagée surtout comme une bibliothèque, qui puisse être accessible au moyen d'un langage de commandes tel que *Python*. Il était en effet question à l'époque d'adosser le code à un outil du commerce et son environnement, et l'Enveloppe devant être le lien entre le Noyau et cet outil non encore choisi, elle aurait pu être le lien entre les commandes d'un langage de script ou de macros utilisé par un tel environnement. Cette voie a été abandonnée, et faute de moyens et surtout de besoins réels, l'Enveloppe s'est transformée petit-à-petit en simple exécutable (beaucoup moins coûteux à développer, car seules les fonctions immédiatement nécessaires au *Code_Saturne* devaient être développées). Le support *Python* expérimental et incomplet a été supprimé. On notera que l'Enveloppe (maintenant le Préprocesseur) permet d'alimenter occasionnellement certains modules associés à l'effort *MED* en cas tests. Certaines fonctionnalités clés comme le recollement conforme sortant du champ de *MED* ou *MED Memory*, un éventuel composant recollement compatible avec cette librairie pourrait être construit à partir de code issu du Préprocesseur.

Par rapport aux spécifications initiales, il est donc possible de simplifier certains ascpects du Préprocesseur, ce qui faciliterait sa maintenance. Surtout, on souhaite paralléliser progressivement cet outil, de manière à pouvoir traiter à terme des cas de très grande taille en les distribuant sur plusieurs nœds de calcul. Cela passe par le report progressif de fonctionnalités du Préprocesseur vers le Noyau, comme expliqué par la suite.

9.1 Module Enveloppe et Noyau

La séparation entre Enveloppe et Noyau est quasi totale dans la version 1.0 de *Code_Saturne*, tous les échanges se faisant par passage de messages. Dans la version 1.1 du code, certains développements côté Noyau s'appuyaient sur des principes et du code issus ou voisins de ceux de l'Enveloppe. C'est notamment le cas des fonctionnalités « coupes » pour laquelle le paramétrage par ligne de commande de l'Enveloppe n'aurait pas suffi, et qui fait donc intervenir à la fois du code spécifique Noyau et le module Enveloppe, et la gestion des fichiers de redémarrage de calcul, intégralement gérée par le Noyau (ceci afin d'eviter de devoir gérer encore de nouveaux messages). Cette partie du Noyau, écrite en langage C, et basée sur une version simplifiée (puis parallélisée) des concepts et méthodes de l'Enveloppe, pourrait en fait s'apparenter à une partie de l'Enveloppe (le spécification initiale prévoyant aussi la possibilité d'une couche Enveloppe et d'une couche Noyau dans un même exécutable). On pourrait parler pour ce code de la *couche* Enveloppe par rapport au *module* Enveloppe. Du point de vue gestion de configuration, le code correspondant est attaché au Noyau. Dans la version 1.2 du code, des fonctionnalités de bas niveau voisines ont été regroupées dans la librairie BFT, partagée par le Noyau et l'Enveloppe, et utilisée aussi par NEPTUNE_CFD). Dans la version 1.3, les fonctionnalités associées au couplage et au post-traitement ont été réécrites (sous forme parallèle) dans la librairie FVM, utilisée par les Noyau de *Code_Saturne* et NEPTUNE_CFD.

La méfiance vis-à-vis de langages autres que le Fortran 77, l'existence du prototype Noyau avant même les spécifications de l'Enveloppe, et la répartition des tâches en sous-équipes différentes ont fait que la séparation en deux exécutables était bien adaptée à ces phases de développement. Cette séparation a aussi facilité la parallélisation du code, l'Enveloppe se chargeant du découpage du domaine en restant séquentielle, et le Noyau étant totalement parallélisé. Par contre, il est plus simple pour certaines fonctionnalités de ne pas nécessiter le passage et la gestion complexe de messages entre les deux exécutables, et la tendance actelle est à associer directement au Noyau (souvent via la libriairie FVM) ce qui ne nécessite pas réellement le passage par le Préprocesseur.

On notera que les développements associés à la migration de la fonctionnalité Enveloppe vers le Noyau étant plus récents, il bénéficient du retour d'expérience, et s'appuient sur un modèle de données plus simple et surtout moins « stratifié » que celui de l'Enveloppe, dont la structure à base d'objets au contenu masqué par rapport aux objets appelants va probablement trop loin dans la logique du masquage, aucun gain réel n'étant apporté par cette approche, qui aurait eu un intérêt surtout si les objets de plus haut niveau avaient été implantés au moyen d'un langage interprété lui-même de haut niveau, tel que *Python*. De manière schématique, on peut dire que dans le Noyau, on a conservé du *module* Enveloppe l'encapsulation des fonctions système, les fonctions utilitaires principales, et l'essentiel de la puissance, en rendant le code plus accessible. Il paraît donc intéressant du point de vue maintenance de privilégier les développements à ce niveau. Le plus naturel pour la parallélisation d'une fonctionnalité de l'Enveloppe est donc de migrer cette fonctionnalité vers le Noyau (et souvent en partie vers la librairie FVM, selon les opérations et structures considérées).

9.2 Post-traitement et polyèdres quelconques

On a jusqu'ici toujours effectué le post-traitement volumique sur le maillage initial, avant recollements conformes éventuels. Ce choix était lié au fait que le maillage initial est généralement donné sous forme d'élements « classiques » avec les formats actuellement utilisés, et que beaucoup d'outils de post-traitement ne gérent pas des polyèdres quelconques. Les version récentes d'*EnSight* peuvent directement gérer les polyèdres quelconques, et une lègère modification des filtres de lecture au format *EnSight Gold* dans *ParaView* lui permet aussi de les lire. *MED* gère aussi les polyèdres. Toutefois, on rencontre sensiblement plus de « bugs » avec ces outils en utilisant des polygones ou polyèdres simples quelconques qu'avec des éléments plus classiques, donc il est important de pouvoir subdiviser ces éléments en éléments plus simples.

La possibilité de post traiter le maillage volumique complet après recollement conforme est disponible dans le Noyau via la librairie FVM. Il purrait être intéressant d'ajouter une fonctionnalité semblable au Préprocesseur, pour pouvoir visualiser l'effet de recollement sur les mailles volumiques, mais la possibilité de générer un maillage de post-traitement à partir du maillage initial reste essentielle, notamment en cas d'erreur de connectivité (dans quel cas il faut mieux pour analyse pouvoir convertir les données d'entrée en données de post-traitement avant toute manipulation risquant de les rendre encore plus incohérentes), et pour l'utilisation de raffinements successifs avec l'outil HOMARD, celui-ci pouvant raffiner des éléments classiques, mais pas des polyèdres, et devant donct travailler sur une structure de type « cellules non conformes simples + joints » que sur une structure de type « cellules conformes polyédriques ».

9.3 Simplifications

Les principales traces des anciens modes *post-traitement* et *pré-post-traitement* du module Enveloppe ont été supprimées, mais la structure globale du Préprocesseur reflète encore la complexité induite par ces anciennes fonctionnalités, et il est possible de simplifier le code. On peut envisager à terme de créer un « composant recollement conforme et périodicité » qui pourrait éventuellement être mutualisé avec d'autres codes utilisant le modèle *MED*.

9.4 Possibilités à plus long terme

Pour aller plus loin, il serait nécessaire de pouvoir effectuer un découpage de domaines au moins un redécoupage depuis le Noyau. Ceci pourrait se faire au moyen de la librairie *ParMetis*. On pourrait alors envisager la parallélisation du recollement conforme (qui représente actuellement le pic de consommation mémoire du Préprocesseur). On notera que la périodicité est une extension du recollement basée sur la recopie de frontières, le recollement entre les frontières d'origine et les recopies déplacées selon le pas de périodicité, et la gestion des liens entre faces d'origine et faces recopiées. La parallélisation se baserait sur une extension de ce principe, en recopiant des faces d'un domaine à l'autre, en effectuant un recollement « classique », et en reportant les modifications des faces recopiées sur les faces dont elles sont issues sur leurs domaines d'origine. Cela représente une certaine quantité de travail, mais les algorithmes ne devraient pas poser de problèmes.

Une fois le recollement et la construction des périodicités parallélisés (et donc migrés vers le Noyau), ne sub-

siteraient plus dans le Préprocesseur que le passage en connectivité descendante et les filtres de lecture de maillages. Leur parallélisation et migration vers le Noyau ne poserait a priori pas de difficultés particulières. À ce stade, le Préprocesseur pourrait donc disparaître, les fonctionnalités essentielles étant reprises par le Noyau, intégralement parallélisé.

10 Bibliographie

[1] Y. FOURNIER
Définition du module Enveloppe pour le Solveur Commun,
Rapport EDF HE-41/99/017/A, 1999

[2] T. THEUSSL
A Review of Two Simple Polygon Triangulation Algorithms,
cours « Special Topics in Computer Graphics 1998 »
http ://www.cg.tuwien.ac.at/ theussl/

[3] G. T. TOUSSAINT
Efficient Triangulation of Simple Polygons,
The Visual Computer, vol. 7, 1991, pp. 280-295

[4] H. ELGINDY, H. EVERETT, G. T. TOUSSAINT
Slicing an Ear Using Prune and Search,
Pattern Recognition Letters, vol. 14, September 1993, pp. 719-722.

[5] J.R. SHEWCHUK
Lecture Notes on Delaunay Mesh Generation,
Notes de cours
http ://www.cs.berkeley.edu/ jrs/meshf99/

[6] M. BERN, D. EPPSTEIN
Mesh Generation and Optimal Triangulation,
Computing in Euclidean Geometry, 2nd ed., D.-Z. Du and F.K. Hwang, eds., World Scientific, 1995, pp. 47-123
http ://www.ics.uci.edu/ eppstein/pubs/geom-tri.html