Université de Liège
Faculté des Sciences Appliquées

Rapport F.R.I.A. 1999-2000

Deuxième bourse — Première année

Formulation Arbitraire Lagrangienne-Eulérienne pour le contact lubrifié entre solides. Applications aux opérations de mise à forme.

R.Boman

24 janvier 2007

L.T.A.S.–Thermomécanique & Milieux Continus
Rue Ernest Solvay, 21
B-4000 LIEGE

Tel : +32–(0)4/366.91.89
Fax : +32–(0)4/253.25.81
E-Mail : r.boman@ulg.ac.be

Table des matières

Chapitre 1

Introduction

Ce rapport décrit en détails les développements effectués pendant ma troisième année de thèse (seconde bourse FRIA, première année). Ceux-ci ont principalement été axés sur la gestion du formalisme Arbitraire Lagrangien Eulérien (ALE) [27, 19, 3, 18, 21, 10, 13, 12] dans le code de calcul développé au laboratoire LTAS-MC&T de l'Université de Liège : METAFOR [27, 32, 29, 31].

Ce rapport n'est pas un ancien rapport mis à jour. Il décrit uniquement les nouveaux développements de l'année scolaire 1999–2000. Le lecteur ne trouvera pas, dans ce rapport, les rappels théoriques relatifs au formalisme ALE. Il est donc conseillé de se reporter aux deux précédents rapports d'activité (rapport 1997–1998 [5] et rapport 1998–1999 [6]) pour obtenir ces informations.

Rappelons que le but de ce travail de thèse est la modélisation du contact lubrifié entre solides déformables lors d'opérations de mise à forme. Pour ce faire, nous utilisons la méthode des éléments finis et une formulation Arbitraire Lagrangienne Eulérienne qui consiste à déplacer la grille de calcul indépendamment du mouvement de la matière durant la résolution du problème. Cette formulation est appliquée non seulement au solide (ce qui permet par exemple dans le cas de la simulation du laminage de ne discrétiser qu'une petite portion de la tôle), mais aussi pour le lubrifiant. Les développements informatiques sont ajoutés continuellement au code de calcul METAFOR.

Suite à nos premières recherches dans le domaine de la lubrification [5, 6, 11, 7, 14, 15], nous avons conclu que l'utilisation du formalisme Arbitraire Lagrangien Eulérien est capitale pour obtenir des résultats satisfaisants. En effet, en simplifiant les choses, on peut le comprendre de manière intuitive : la gestion du contact lubrifié entre deux solides subissant de grandes déformations fait entrer en jeu non seulement les équations régissant le mouvement et le comportement des deux solides mais aussi celles du fluide lubrifiant. Les premières sont généralement formulées dans le formalisme lagrangien pour lequel le maillage est fixé à la matière durant le calcul. Par contre, les secondes, relatives au fluide,

sont très souvent exprimées dans une grille de calcul fixe dans l'espace, c'est-à-dire dans le formalisme eulérien. On est donc intuitivement poussé à utiliser un formalisme mixte dans les zones de contact lubrifiées des deux solides. Le formalisme ALE est donc parfaitement approprié pour décrire l'interface entre deux solides.

Cependant, nous avons également vu que l'utilisation de ce formalisme ne résout pas tous les problèmes. Tout d'abord, vu la complexité de la résolution de l'équation de Reynolds sur le maillage éléments finis représentant la surface de contact, notre méthode de remaillage [1] et de transfert de données du nouveau maillage vers l'ancien doit être efficace mais aussi robuste. Il est, en effet, pénible d'essayer de comprendre certains effets numériques qui pourraient être perturbés ou camouflés derrière des approximations où des erreurs dans la gestion des grilles de calcul. Au début de cette année, nous avons estimé que l'ensemble des routines ALE de notre code de calcul ne respectaient pas ces critères. Bien qu'il n'y avait pas de réelle grosse faute de programmation, les résultats obtenus étaient parfois assez étonnants (dans le mauvais sens du terme). Il n'était pas rare, par exemple, de voir un noeud du maillage, fixé explicitement dans le jeu de données d'un problème, commencer, dans certains cas, à se déplacer lors de la résolution du problème.

Un autre grand problème, en lubrification, est la gestion de la géométrie de la zone de contact (voir par exemple [24, 6]). Premièrement, il faut pouvoir repositionner correctement les noeuds de la zone de contact pour que la longueur de la zone reste plus ou moins constante. Deuxièmement, il faut essayer de conserver dans la mesure du possible la géométrie de la zone d'entrée du lubrifiant dans le contact avant et après remaillage. En effet, l'angle formé par les deux solides en contact va conditionner toute la solution du problème de lubrification par l'intermédiaire de la valeur du débit de fluide s'écoulant entre eux. Ce problème de géométrie serait moins prononcé et beaucoup plus simple à résoudre dans un formalisme lagrangien. Cependant, dans ce formalisme, les valeurs obtenues pour la longueur de contact et l'angle d'entrée subissent des variations temporelles même si le problème résolu est stationnaire. Ceci est dû à la variation du nombre de noeuds en contact et l'entrée en contact et la sortie du contact de certains noeuds.

Pour pouvoir gérer efficacement ce genre de problème, il faut donc obligatoirement être confiant vis-à-vis des routines ALE et connaître parfaitement les possibilités et les limitations de celles-ci.

En conséquence, et pour d'autres raisons expliquées tout au long de ce rapport (voir en particulier la section 2.2, page 16), nous avons décidé de reconstruire et de reprogrammer sur de nouvelles bases solides l'ensemble des routines de remaillage et de transfert de données. Notre but, presque atteint à l'écriture de ces lignes, est l'obtention d'une librairie ALE où la majorité des hypothèses (sur le maillage, les éléments utilisés, la géométrie, la structure interne et l'origine des données) sont supprimées.

[1] Le mot "remaillage" est, dans ce rapport, un synonyme de "repositionnement de noeuds" (ou rezoning en anglais). Nous utiliserons dans la suite indifféremment ces deux expressions bien que le mot remaillage soit bien plus général.

Grâce à cette librairie, nous pourrons améliorer de manière plus efficace l'ensemble de nos développements dans le domaine de la lubrification mais aussi étendre l'utilisation de ce formalisme à tous les problèmes que METAFOR peut et pourra résoudre. Dans cette optique, nous avons étendu le formalisme ALE aux problèmes tridimensionnels et aux maillages non structurés. Dans un futur très proche, notre librairie pourra également être utilisée, sans modification interne fondamentale, dans le cas d'éléments finis de coque 2D et 3D, actuellement en développement dans notre laboratoire [29, 17].

Le développement de cette librairie s'est déroulée dans le cadre d'une amélioration de ce que nous appelons la "tête" du code. Cette nouvelle tête, qui gère principalement la préparation des données provenant du programme de pré-traitement et la gestion de la mémoire a subit une très forte évolution au cours de cette année. Grâce à ces nouvelles routines, nous avons pu porter METAFOR sur d'autres types de machines que les traditionnelles stations alpha utilisées depuis toujours au laboratoire. Nous pouvons donc maintenant travailler sur un PC traditionnel ce qui représente une avancée énorme. D'autre tâches, expliquées brièvement dans ce rapport (chapitre 4), ont permis de gérer notre code de calcul de manière professionnelle.

Ce dernier point nous a malheureusement coûté un temps précieux mais nous pensons qu'il faut voir ce temps comme un investissement dans l'avenir du code METAFOR et dans l'avenir de ce travail de thèse. Ces développements informatiques ont tout d'abord un très grand intérêt pour le laboratoire puisqu'il serait maintenant tout à fait envisageable de commercialiser notre code METAFOR. Pour notre thèse, l'intérêt est aussi non négligeable puisqu'ils assurent que notre travail informatique restera à jamais inclus dans le code. Il est en effet bien malheureux mais très courant de voir un grand nombre de travaux qui sont développés dans une version parallèle d'un code de calcul et qui ne sont jamais réintégrés à celui-ci.

Ce rapport se décompose comme suit :

- Après ce chapitre d'introduction, nous expliquons dans le chapitre 2 la méthode suivie pour obtenir une librairie de routines ALE. Les différentes méthodes utilisées pour la détection des noeuds sur les entités géométriques, pour le repositionnement de ces noeuds (remaillage) et pour le transfert des données stockées aux points de Gauss sont expliquées. Nous nous focalisons principalement sur ce qui est nouveau. Ainsi, nous ne détaillons pas la méthode de convection utilisée (volumes finis) puisque celle-ci a été expliquée et étudiée en détail dans notre premier rapport. Par contre, nous expliquons comment la plupart des restrictions des anciennes routines ont pu être éliminées.
- Le chapitre 3 montre des applications numériques aussi bien bidimensionnelles que tridimensionnelles. Nous comparons, pour divers problèmes tel que l'impact de la barre de Taylor et le problème du poinçonnement cylindrique, les solutions 2D, 3D, lagrangiennes et ALE. Nous expliquons aussi certains problèmes rencontrés et les solutions qui sont envisagées pour l'avenir.
- Le chapitre 4 décrit brièvement les tâches annexes qui ont été nécessaires au cours

de cette année. Il s'agit principalement de la mise au point d'une interface graphique permettant de visualiser le déplacement du maillage éléments finis au cours du calcul.

- Enfin, le chapitre 5 présente nos conclusions et les perspectives envisagées dans le domaine du formalisme ALE.

Chapitre 2

Mise en oeuvre d'une librairie ALE

2.1 Introduction

2.1.1 Choix du langage de programmation

Vu les difficultés rencontrées dans la gestion de la lubrification par la méthode des éléments finis, nous avons décidé de rendre les routines gérant le formalisme ALE [27] plus efficaces et plus robustes. En effet, ces routines, à l'image de la majorité du code de METAFOR, sont écrites dans un "vieux" style de programmation. Le langage utilisé est le FORTRAN 77 qui a toujours eu beaucoup de succès auprès de la communauté scientifique. Or, il est bien connu que ce type de langage possède de nombreuses limitations. Celles qui nous préoccupent le plus sont les suivantes :

- Le FORTRAN 77 ne permet pas d'allocation dynamique de mémoire. Il est cependant possible de contourner cette limitation en allouant un grand vecteur et en gérant cet espace soi-même (on appelle cela l'allocation pseudo-dynamique). Il est clair que si cette approche ne pénalise pas trop une personne travaillant sur des lois constitutives, qui a besoin de peu de vecteurs ou matrices, celles-ci étant parfaitement déterminées lors du démarrage du calcul, la programmation devient délicate lorsqu'il faut considérer un grand nombre de vecteurs comme dans le cas de l'ALE. Une gestion complètement dynamique devient vraiment nécessaire et incontournable lorsqu'on aborde le problème du remaillage complet du problème envisagé. Dans tout notre travail, nous avons toujours gardé en tête qu'une future extension de la librairie ALE inclurait un remaillage global où le nombre d'éléments peut varier en cours de calcul. Cependant, même sans envisager ce cas extrême où tous les tableaux doivent être redimensionnés, il est très confortable de pouvoir allouer de la mémoire lorsqu'on le désire. Par exemple, lorsqu'on veut déterminer les arêtes des éléments d'un maillage non structuré, il n'y a pas un moyen simple de connaître a priori leur nombre. On

gagne donc du temps en allouant dynamiquement le tableau des arêtes.
- La structuration des données est très difficile en FORTRAN 77 parce qu'il n'y a rien de prévu a cet effet. Il faut donc être extrêmement méticuleux pour obtenir un schéma de stockage de données clair et efficace. Ceci est d'autant plus complique lorsque plusieurs personnes travaillent sur le même code de calcul (nous sommes neuf développeurs METAFOR au laboratoire).
- L'absence de variables globales (dimensionnées dynamiquement) est très limitatif. Il n'est donc pas rare d'avoir plusieurs centaines d'arguments pour chaque appel de fonction. Là encore la rigueur s'impose et chaque développeur subit les erreurs des autres. De plus, modifier une routine implique souvent de modifier des dizaines d'autres pour amener une variable jusqu'à la routine concernée.

Ces gros désavantages entraînent une perte de temps assez élevée et il n'est pas rare de passer plus de temps à maintenir le code "en vie" qu'à programmer des algorithmes.

En conséquence, nous avons choisi de programmer dans le langage C. Ce langage, bien qu'assez vieux lui aussi, ne possède aucune des limitations décrites ci-dessus. Il permet une programmation structurée et propre sans trop d'effort. De plus, la maintenance du code est grandement facilitée (variables globales, structures, librairies,...).

2.1.2 Evolution actuelle de METAFOR

Il n'est pas possible de parler de notre nouvelle implémentation du formalisme ALE sans parler des évolutions récentes de METAFOR, dont nous sommes en partie responsable.

Depuis sa création lors de la thèse de J.-P. Ponthot [27], le code de calcul METAFOR a subit de très grandes modifications et améliorations suite à quelques thèses [29, 31, 32] et travaux de fin d'études (le notre, entre autres [4]). Cependant, une grande limitation était ancrée dans les nombreuses routines depuis le début : depuis toujours, METAFOR est intimement lié aux routines de BACON, logiciel de pré-traitement de SAMCEF (voir figure 2.1). Ceci est dû non seulement au fait que METAFOR était, lors de ses débuts, destiné à être inclus comme module non linéaire dans SAMCEF mais aussi à une longue tradition (SAMCEF a vu le jour au sein du LTAS de Liège). Techniquement, il n'était pas possible de créer un exécutable METAFOR sur une machine où on ne possédait pas les librairies de SAMCEF. La situation était bien pire puisque la version de BACON utilisée était une version assez ancienne et modifiée spécialement pour le laboratoire. En conséquence, nous ne bénéficiions même plus des dernières nouveautés de BACON puisque notre version était incompatible avec les nouvelles. De plus, notre programme ne fonctionnait que sur d'onéreuses stations alpha (qui ne constituent pas un standard industriel, de nombreuses industries se tournant maintenant vers le PC, bien moins cher).

Pour construire une librairie ALE, il était indispensable d'obtenir, au minimum, des informations sur les entités géométriques (points, lignes, domaines, ...) qui constituent le

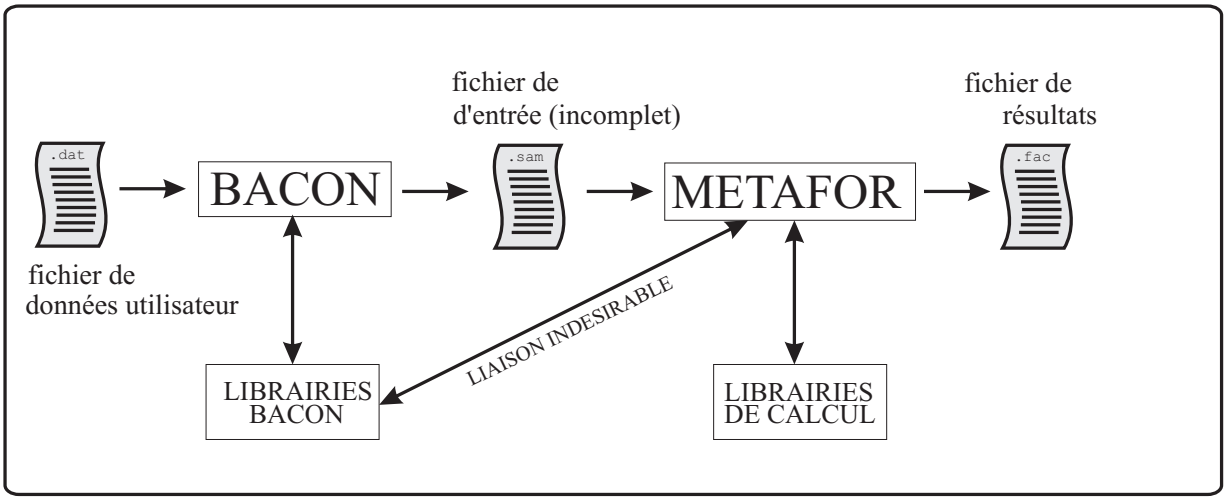

FIG. 2.1: Ancienne structure de METAFOR et ses librairies.

problème. Sans ces informations, il n'est pas possible de faire tomber toutes les limitations des anciennes routines. De plus, nous ne voulions pas perdre la possibilité future d'inclure un vrai remaillage à nombre d'éléments variable au cours du calcul. Or, dans ce dernier cas, toutes les données (matériaux, conditions aux limites, ...) doivent être formulées sur des entités géométriques et non sur des noeuds ou sur des éléments finis. En effet, comment appliquer les conditions aux limites après remaillage si les noeuds avant et après remaillage ne se correspondent pas ?

Depuis bientôt 3 ans, le projet METASTAMP [25] tente de faire une liaison solide (une interface) entre le nouveau BACON (la version 8.1) et METAFOR. Cette liaison a été programmée de telle manière à ce que la récupération des entités géométriques soit possible et complètement automatisée. Cette tâche étant très longue et n'étant pas complètement terminée, même à l'heure actuelle, nous avons dû aider à accélérer la programmation pour obtenir enfin ce que nous voulions.

D'une manière beaucoup plus judicieuse, cette nouvelle interface n'est pas située dans le code METAFOR mais dans un programme externe nommé Z-Mesh (voir figure 2.2). Ce programme est l'outil de pré-processing initialement programmé et utilisé par L. Stainier [32]. Ce programme, bien que très simplifié face à BACON, a le gros avantage que toutes ses sources sont disponibles et modifiables par quiconque. Il est donc, par exemple, extrêmement facile, d'ajouter une nouvelle commande utilisateur pour introduire de nouvelles données relatives à de nouveaux développements.

Le but a atteindre est d'obtenir un code de calcul indépendant du logiciel de prétraitement utilisé. METAFOR n'est capable de lire qu'un seul type de fichier, le fichier de sortie de Z-Mesh. Ce fichier, de type ASCII est formaté selon nos besoins et est parfaitement portable d'une machine à une autre. Z-Mesh transforme donc le fichier de sortie

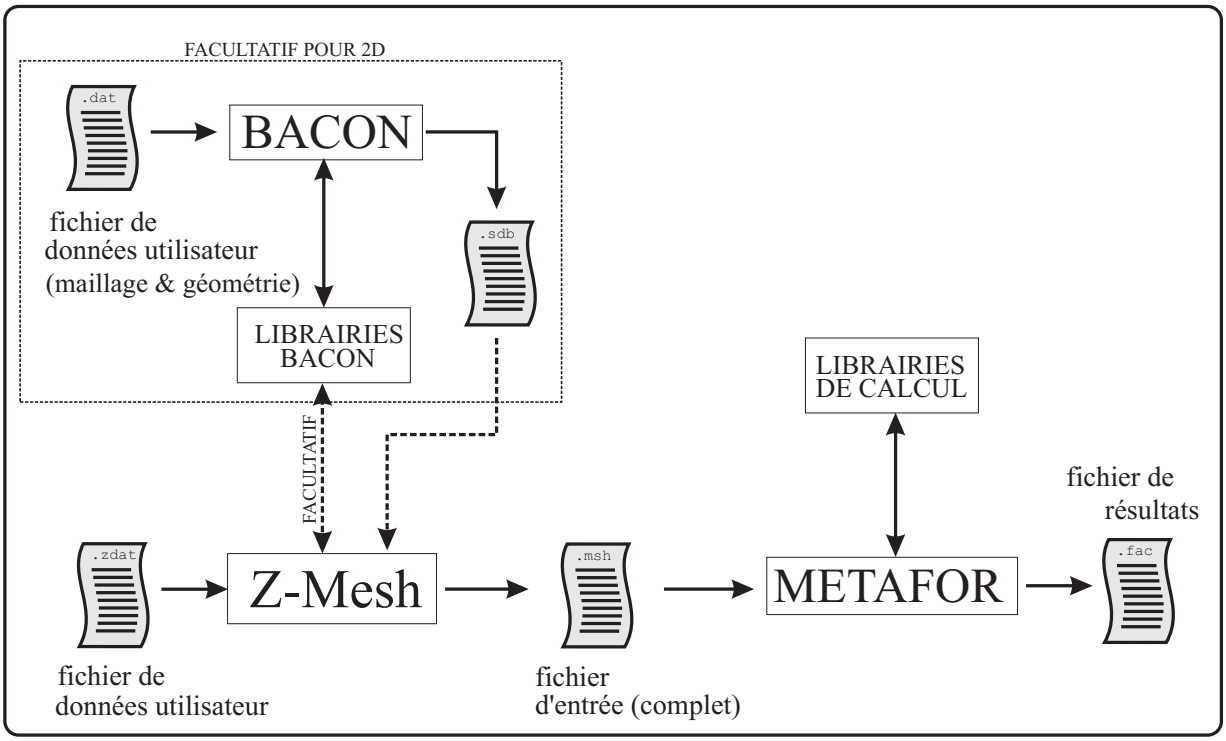

FIG. 2.2: Nouvelle structure de METAFOR et ses librairies.

de BACON en un fichier contenant uniquement les informations pertinentes pour METAFOR, le reste n'étant pas traduit. Cette manière de faire permet d'utiliser en parallèle les capacités de Z-Mesh pour introduire proprement des nouvelles données impossibles à introduire dans BACON (c'est le cas pour les données ALE). L'idéal étant de générer le maillage élément fini dans un logiciel comme BACON et introduire les données (conditions aux limites, matériaux, paramètres ALE, ...) via Z-Mesh. Dans le futur, changer de logiciel de maillage ne posera pas de problème : il "suffira" de programmer dans Z-Mesh un module de conversion du maillage et de la géométrie fournie par cet autre logiciel. Les commandes introduisant d'autres données étant toujours entrées via Z-Mesh.

Dans METAFOR, la lecture des données fournies par Z-Mesh est faite dans une série de routines rassemblées sous le nom de "nouvelle tête". Cette nouvelle tête est une branche parallèle aux anciennes routines ("ancienne tête") de lecture de la sortie du vieux BACON. Ces deux branches doivent coexister jusqu'à ce que toutes les possibilités de METAFOR soient identiques d'un côté et de l'autre (ce qui n'est pas encore le cas).

La première étape était donc de rendre METAFOR indépendant de BACON. Cette étape est maintenant presqu'entièrement achevée. Il est en effet possible (après de nombreux efforts) de compiler METAFOR sur n'importe quelle machine (a l'heure actuelle, nous avons essayé sur COMPAQ ev5, COMPAQ ev6, PC, MAC). Le code restant dans METAFOR est du code de calcul pur dont toutes les sources sont disponibles et donc parfaitement

portable. Le code a été complètement nettoyé et il n'existe plus aucun avertissement de compilation.

Z-Mesh a également beaucoup évolué en un an. Seule la manière d'entrer les données est restée inchangée. Nous avons reprogrammé toutes les commandes qui permettent d'introduire les conditions aux limites, les matériaux mais aussi la géométrie, si bien qu'il est tout à fait possible de se passer de BACON pour un problème 2D. Pour un problème 3D, BACON reste indispensable vu que Z-Mesh ne possède pas de mailleur 3D.

Pour le 2D, sur certains points, Z-Mesh est même bien plus intéressant que BACON : il possède un mailleur frontal quadrangulaire que l'on ne trouve pas dans BACON et il permet de spécifier toutes les conditions aux limites sur des entités géométriques (BACON lie uniquement ces informations aux noeuds).

Dans un second temps, nous avons introduit dans Z-Mesh les nouvelles commandes permettant l'introduction de données de manière simple et claire. Il n'y a donc plus besoin de créer plusieurs fichiers annexes, un pour le contact, un pour le remaillage, un pour le solveur, ...

Nous avons ensuite développé la librairie ALE complètement à côté de METAFOR. Nous n'utilisions uniquement la lecture du fameux fichier de sortie de Z-Mesh. C'est à ce stade que nous avons effectué le plus gros de la programmation. Cette manière de faire a permis d'être certain de l'indépendance de la librairie par rapport au code qui l'appelle et, de manière plus terre à terre, retarder le moment où la librairie serait connectée à METAFOR. En effet, lors de l'écriture des routines, la "nouvelle tête" n'était pas assez développée, même pour des simples cas-tests 2D. A l'heure actuelle, la situation est différente et seuls certains aspects du contact 3D ne sont pas disponibles.

Après avoir testé toutes les parties de la librairie, il fallait ensuite établir une connexion entre METAFOR et celle-ci. Nous avons donc programmé une série de routines d'interface. Cette façon de faire peut paraître inutilement compliquée. C'est en partie vrai si on se limite à une vision à court terme. Par contre, à long terme, les avantages sont évidents : la librairie ALE, à l'exclusion de ces quelques routines d'interface est totalement indépendante de METAFOR (et pourrait être utilisée dans un autre code sans modification). De plus, la librairie peut être facilement étudiée hors de METAFOR lors de nouveaux développements.

2.2 Limitations des anciennes routines ALE

Cette section présente d'une manière détaillée les principales raisons pour lesquelles nous avons pensé qu'il était incontournable de repenser intégralement la gestion du formalisme ALE dans METAFOR. Le contenu scientifique de la librairie n'est pas fondamentalement supérieur à celui des anciennes routines. Cependant, tous les problèmes qui pouvaient être résolus avec les anciennes routines peuvent l'être avec les nouvelles mais l'inverse n'est évidemment pas vrai.

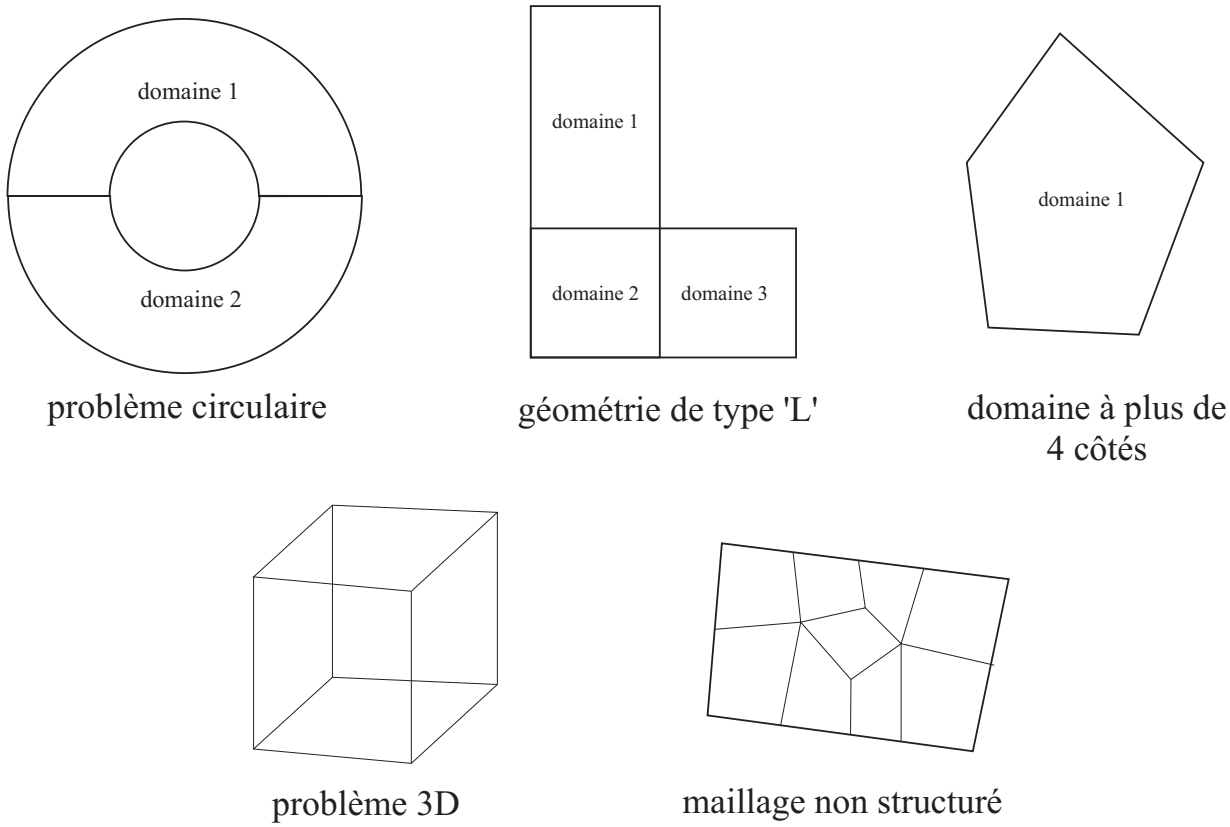

FIG. 2.3: Problèmes impossible à résoudre avec les anciennes routines.

Les anciennes routines se débrouillent avec une connaissance de la géométrie et de la topologie du maillage assez réduite voire très limitée. Ceci était dû à l'ancienne version du logiciel de pré-traitement BACON. La plupart des limitations citées ci-dessous en sont une conséquence (voir aussi figure 2.3) :

- Il est impossible de gérer des maillages non structurés. En effet, une hypothèse fondamentale est que chaque domaine à remailler est constitué de quatre lignes. Chaque domaine doit être composé de noeuds dont la numérotation est tout a fait ordonnée. On sait donc par exemple que un domaine est composé de 10 noeuds dans la direction de sa première ligne et de 5 noeuds dans l'autre direction. On peut ainsi recomposer

la topologie du maillage assez facilement. Mais le maillage doit être obligatoirement similaire à un maillage de type "transfini". Cette hypothèse est assez restrictive dans le cadre de la lubrification puisque nous pensons qu'il serait intéressant d'ajouter des mailles le long de la zone de contact mais aussi et surtout aux alentours des zones d'entrée de lubrifiant pour pouvoir mieux approximer la géométrie de cette zone très importante.

- En conséquence directe, on ne peut pas utiliser des domaines qui sont composés de plus de 4 lignes. Ceci n'est pas une grande limitation puisqu'on peut en général toujours décomposer ce type de domaine en plusieurs domaines à 4 cotes. Cependant, cette technique peut compliquer inutilement le jeu de données du problème.
- Vu la manière dont les mailles sont mémorisées, beaucoup de géométries sont purement et simplement impossible a modéliser : des géométries de type circulaire (rouleau de laminage) ou même simplement en forme de 'L' posent des problèmes dans la numérotation des noeuds. En conséquence, si on revient momentanément au point précédent (découpage de domaines à plus de 4 côtés), on comprend vite que ce type de domaines n'est pas gérable (sauf exception) avec les anciennes routines ALE.
- La gestion des fichiers de données pose également de sérieux problèmes. Tout d'abord, les données sont fractionnées dans 3 fichiers séparés (un pour le géométrie, le maillage et les données, un pour le remaillage et un pour la convection). Ce nombre de fichiers ne peut que croître puisque l'ancienne version du pré-processeur n'accepte pas de nouvelles commandes. Ensuite, le format d'entrée est totalement différent si on veut faire fonctionner un cas-test en formalisme lagrangien ou en ALE. Malgré que le formalisme lagrangien est un cas particulier du formalisme ALE, ce n'est pas le cas pour les commandes d'introduction des données. Comme nous l'avons dit, les commandes ALE de l'ancien BACON sont très limitatives sur les géométries considérées et sont très laborieuses à utiliser (il faut presque une calculatrice à côté de soi pour s'y retrouver !). Si bien que la majorité des jeux de données dont dispose le laboratoire doivent être entièrement réécrits pour fonctionner en ALE. Notre but, ici, est d'avoir exactement le même type de jeu de données quel que soit le formalisme utilisé. Seul un paramètre (la fréquence de remaillage, mise à zéro dans le cas lagrangien) décide si on utilise le formalisme ALE ou lagrangien. Ce faisant, le formalisme ALE deviendra plus accessible et (nous l'espérons) sera utilisé par d'autres chercheurs comme une "boîte noire".
- La manière dont sont programmées les anciennes routines ALE rend assez complexe la gestion des frontières eulériennes (lignes frontières qui permettent un flux de matière vers l'extérieur ou l'intérieur du maillage. Le problème se pose à deux endroits bien précis : les noeuds extrémités d'une telle ligne doivent être absolument repositionnés d'une manière spéciale pour éviter les retournements de mailles. De plus, la gestion des conditions aux limites sur ces lignes, et particulièrement les déplacements imposés, doit se faire de manière incrémentale (on utilise généralement un repositionnement total qui est bien plus précis, évitant les erreurs d'arrondis, dans le cas lagrangien). Jusqu'à présent, et ce sera toujours le cas vu la pauvreté des données à notre disposition dans les anciennes routines, on est obligé de modifier à la main le

code et l'adapter en fonction du cas traité (ce qui n'est ni efficace, ni professionnel).
- Certaines méthodes de remaillage fonctionnent de manière très limitée. Il est par exemple proposé une fonction "remaillage miroir" permettant d'obtenir le même maillage sur un côté et son opposé. Cependant, cette méthode ne fonctionne pas si les côtés en question ne sont pas parallèles aux axes x et y.
- D'un point de vue beaucoup plus technique, l'ancien ALE est en fait une routine kilométrique appelant quelques routines utilitaires et définissant plusieurs centaines de variables. Certaines parties de cette routine sont spécifiques à un ou deux cas tests, d'autres parties, héritage de plusieurs thèses, sont tellement obscures qu'on s'en demande l'utilité. Il n'est donc pas très agréable de travailler dans ce contexte.
- Les anciennes routines sont prévues pour un maillage régulier d'éléments quadrangulaires avec, au maximum 4 éléments adjacents à un noeud. Ceci est très limitatif. Outre les maillages non structurés quadrangulaires, il est très probable que d'autres types d'éléments finis soient pris en compte dans METAFOR. Les coques triangulaires ainsi que les triangles ou les tétraèdres du second degré devront facilement être utilisés en formalisme ALE. Ce qui paraît impossible dans les anciennes routines.
- Une dernière limitation est l'absence de prise en compte du 3D. A l'heure actuelle, les ordinateurs permettent d'envisager de plus en plus souvent les aspects tridimensionnels des problèmes industriels. Il serait dommage de se limiter à deux dimensions. Le gros problème des anciennes routines ALE est l'absence de structuration des données. Il serait vraiment difficile, voire impossible d'aborder des cas 3D dans ce contexte sans s'imposer à nouveau de nombreuses limitations.

Après cette énumération des problèmes rencontrés lors de l'utilisation des anciennes routines ALE, il n'est pas très difficile de comprendre pourquoi nous avons entrepris ce travail même si celui-ci paraît très important.

D'un point de vue pratique, toutes ces limitations ont été supprimées dans la nouvelle implémentation de l'ALE et, pour une utilisation quotidienne de cette librairie, il semble que la facilité toute récente d'introduction des données représente l'avancée la plus visible et la plus phénoménale mais peut être aussi la moins scientifique.

2.3 Structuration des données

Si on peut se passer d'une structuration des données dans les cas simples bidimensionnels (ce qui était fait jusqu'à présent), elle semble indispensable dans le cas de maillages non structurés et pour aborder des problèmes 3D.

La seule hypothèse de notre librairie ALE est l'existence d'une structure de données (structure au sens du langage C) et que celle-ci soit remplie avec les informations données par le logiciel de pré-traitement. La structure peut être partiellement remplie. Dans ce cas, si c'est possible, la librairie se chargera de la compléter. Par exemple, l'information d'appartenance d'un noeud à un domaine ou à une face n'est pas requise.

Cette structuration des données a été créée par L.Stainier lors de sa thèse [32]. Malheureusement son travail n'a pas été réintroduit dans METAFOR et la mise en place et le remplissage de cette structure n'a pu débuter qu'avec le projet METASTAMP. Nous avons beaucoup contribué à la mise en place de cette structure et nous avons ajouté les données essentielles pour le formalisme ALE (à l'heure actuelle, la librairie ALE est la seule partie de METAFOR utilisant directement la structure pour effectuer des calculs).

En quelques mots, nous pouvons décrire cette structure de la manière suivante (voir aussi figure 2.4) :

- La structure de donnée est, en fait, composée de deux structures distinctes mais entre lesquelles de nombreuses relations existent : une structure "maillage" et une structure "géométrie". Chacune de ces deux structures regroupent toutes les informations nécessaires pour le calcul qui proviennent du jeu de données. La première regroupe les noeuds, les mailles, les arêtes linéiques des éléments en 2D et les facettes des éléments volumiques en 3D. La seconde regroupe une liste de domaines, de faces, de contours, de lignes et de points.
- Chaque entité est elle-même une structure de données. Ainsi, un élément possède un numéro, un attribut, un type, une liste de noeuds, une liste de facettes et une liste d'arêtes. De la même manière, une ligne possède un numéro, un type, un nombre de points (2 pour une droite, 3 pour un cercle et n pour une spline), un nombre de noeuds et une liste de noeuds si elle est maillée. Grâce à l'allocation dynamique, si une information n'est pas utile pour le calcul, la mémoire n'est pas allouée. Par exemple, METAFOR n'a pas besoin de connaître les arêtes des éléments pour un calcul lagrangien traditionnel. Par contre, cette information est capitale lors de la convection par volumes finis des grandeurs aux points de Gauss en formalisme ALE.
- Toutes les entités sont intimement connectées entre elles. La première connection est simplement celle du maillage vers sa géométrie. Mais il en existe bien d'autres comme les lignes d'un contour, les facettes d'un élément 3D, les noeuds sur une face,... Toutes ces connexions sont représentées par des pointeurs (au sens du langage C) et non pas par des nombres entiers comme c'est généralement le cas en FORTRAN.
- Grâce à cette structure, il est donc très simple de parcourir les données dans tous

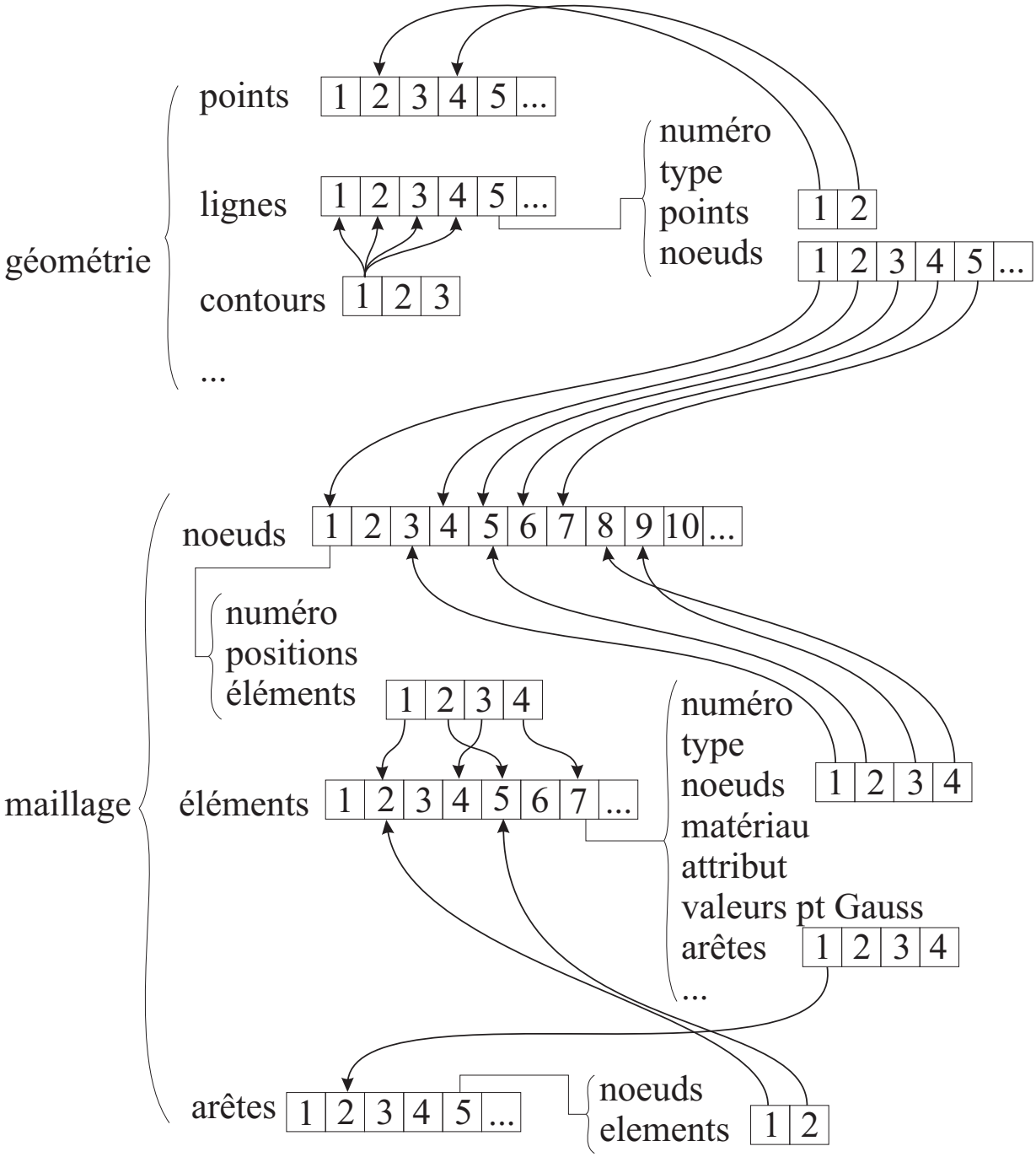

FIG. 2.4: Schéma simplifié de la structuration des données.

les sens. Par exemple, si on veut avoir la liste des noeuds se trouvant sur la ligne i du contour j du domaine k, il suffit de suivre les pointeurs. On peut également avoir accès à des informations jusque là difficiles à obtenir comme par exemple la liste des éléments surfaciques d'un maillage. On est gagnant sur deux plans : l'exécution est plus rapide puisqu'on utilise de vrais pointeurs et on y gagne en clarté (le code est extrêmement explicite).

- Une remarque très importante doit être faite ici concernant l'apparente duplication des données. En effet, on pourrait penser qu'une telle structure duplique la majorité des informations en mémoire. Ce n'est pas du tout ce qu'il se passe. Grâce aux pointeurs, il est possible de faire pointer les tableaux principaux de la structure vers leurs homologues FORTRAN. On obtient ainsi simplement une autre manière d'accéder aux données. De plus, si un tableau FORTRAN est modifié la structure est automatiquement mise à jour puisqu'il s'agit du même emplacement mémoire.

La structure telle que L.Stainier l'avait implémentée dans son code n'était pas assez complète à notre goût. En effet, n'utilisant pas le formalisme ALE mais plutôt un remaillage global effectué dans un programme séparé, il lui était superflu de connaître par exemple la liste des noeuds d'un domaine (tous les noeuds intérieurs étaient de toute façon régénérés avec d'autres numéros).

De plus, n'ayant pas fait de 3D, certaines entités étaient manquantes (les facettes, par exemple, équivalant aux arêtes en 3D).

Remarquons également que son utilisation de la structure était principalement réservée au pré-traitement des données et au remaillage. Il n'y avait donc pas de lien direct entre les inconnues du problème et la structure ; celle-ci servant principalement à agencer les données correctement en début de calcul et lors d'un remaillage global. Nous avons donc ajouté une série de liens qui permettent d'avoir accès, pour un noeud, à ses 4 positions (initiale, équilibrée, actuelle et remaillée) et, pour une maille, à ses valeurs aux points de Gauss. Cette façon de faire est indispensable si on veut utiliser cette structure de données non seulement pour le repositionnement des noeuds mais aussi pour la convection des grandeurs aux points de Gauss. Pour toutes ces grandeurs, la remarque concernant la duplication des données est toujours valable. Les positions et les valeurs aux points de Gauss ne sont en aucun cas dupliquées. La structure permet simplement d'accéder à ces valeurs d'une manière différente.

Enfin, nous avons ajouté une structure "matériau" qui permet de traiter des problèmes multi-matériaux. Dans ce cas, un lien de l'élément vers son matériau est établi, ce qui permet de savoir quelles grandeurs doivent être transférées lors de la convection.

2.4 Schéma de la librairie ALE

2.4.1 Introduction

La librairie ALE a été écrite en essayant de l'isoler au maximum de METAFOR. C'est assez facile a priori puisque les routines ALE ne sont appelées que dans la phase de pré-traitement, avant le calcul et lorsqu'un équilibre est atteint (figure 2.5). A côté de ces deux points d'entrée principaux, on peut ajouter un appel lors de la gestion des déplacements imposés qui doivent parfois être mis à jour de manière incrémentale.

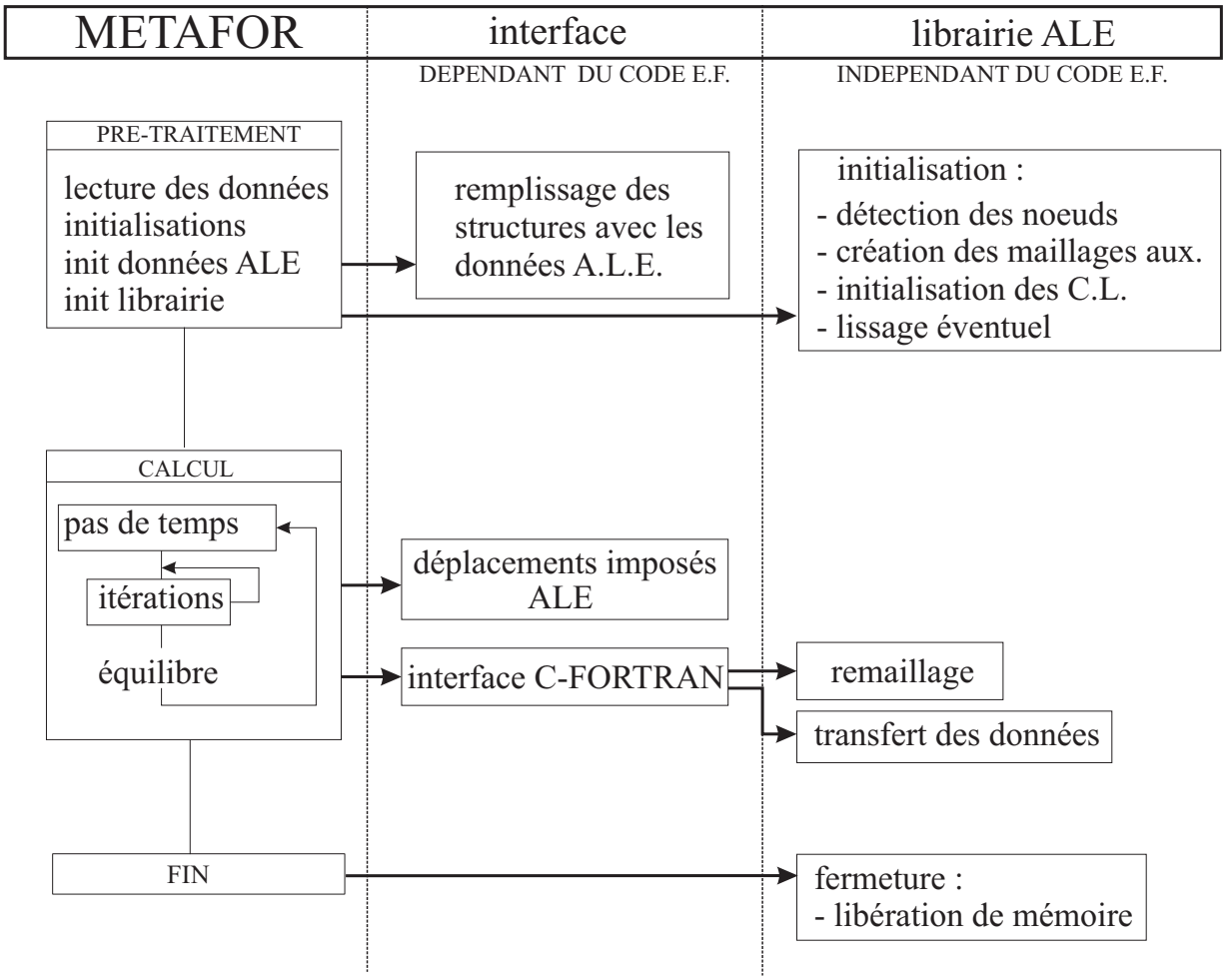

FIG. 2.5: Schéma de la librairie et de son interface avec METAFOR.

2.4.2 Pré-traitement des informations

Le premier point d'entrée fait donc partie du pré-traitement des données. Après avoir rempli la structure avec les données essentielles au calcul, on appelle la routine d'initialisation de la librairie. Le rôle de cette routine est de compléter la structure de données pour que celle-ci soient complète lors du premier remaillage.

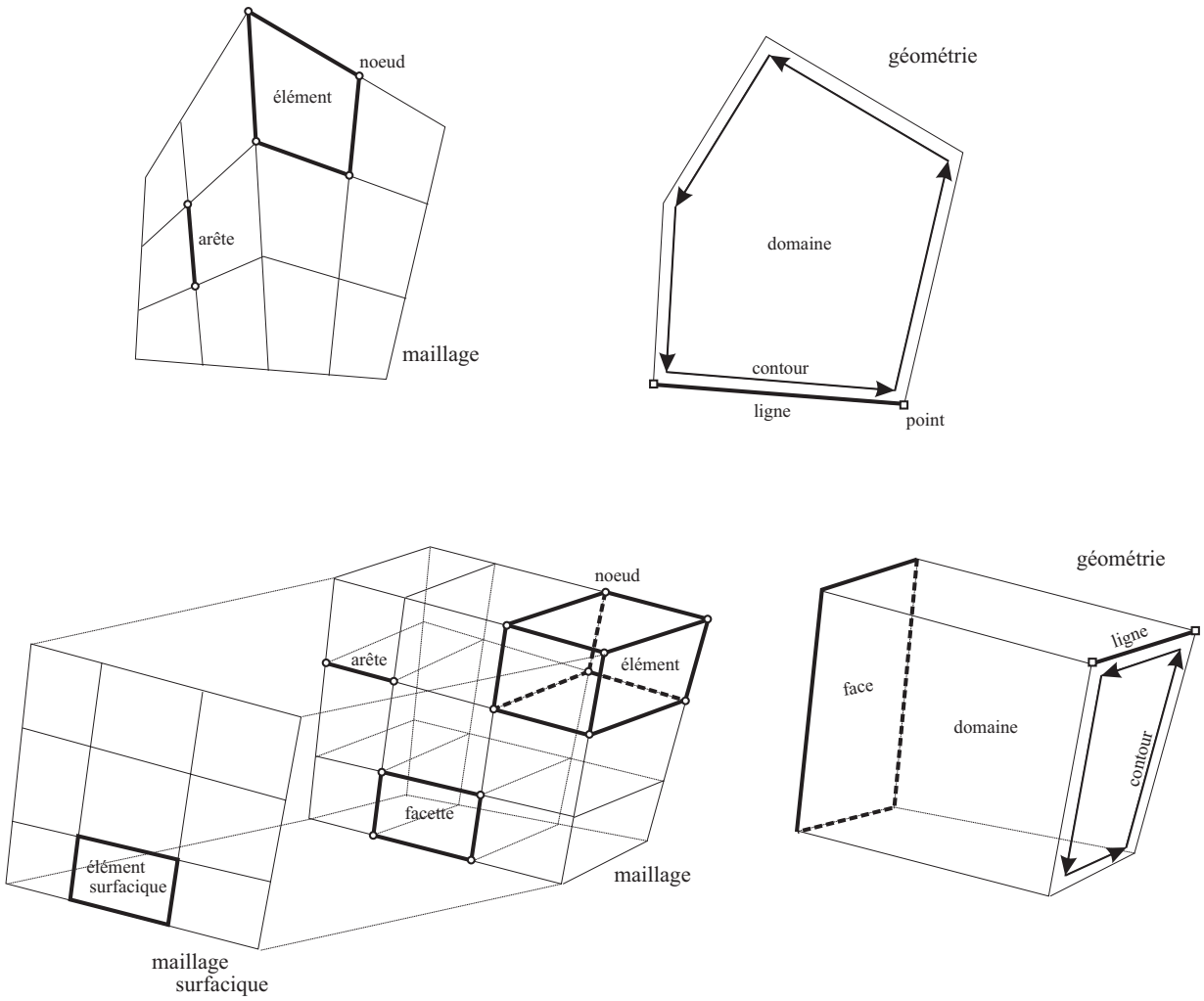

FIG. 2.6: Décomposition du maillage et de la géométrie en leurs composants respectifs

Nous avons choisi d'utiliser une méthode de remaillage hiérarchique. Autrement dit, lors d'un remaillage, chaque entité géométrique est prise en considération successivement en partant des points pour terminer par les domaines volumiques en 3D ou les domaines surfaciques en 2D. La figure 2.6 définit clairement les termes utilisés pour désigner la géométrie et le maillage.

Chaque entité géométrique possède un type de remaillage spécifié par l'utilisateur. Ainsi,

par exemple, chaque domaine géométrique de la structure étudiée peut être remaillée de manière différente. Pour l'instant, nous n'avons pas implémenté de méthode de remaillage globale qui agirait sur tous les noeuds en une seule fois. Grâce à cette façon de faire, il est facile de ne repositionner les noeuds que dans un seul domaine et laisser un autre en formalisme lagrangien traditionnel.

Cependant, contrairement aux méthodes globales qui traitent indifféremment tous les noeuds, il nous faut dans notre cas une information supplémentaire à savoir l'appartenance d'un noeud à une entité géométrique. Cette information n'est pas toujours présente. La liste des noeuds sur les lignes maillées est la seule information liant le maillage et la géométrie que nous possédons automatiquement via notre programme de pré-traitement (BACON ou Z-Mesh).

La première tâche de la routine d'initialisation est de détecter le reste des liaisons entre les noeuds et la géométrie (voir aussi section 2.5). Différentes méthodes ont été mises en oeuvre pour différents cas de figure. Ces méthodes ne sont malheureusement pas infaillibles mais nous avons réussi à ce que la détection se passe correctement dans la plupart des cas.

Si on considère un problème tridimensionnel, il est facile de se rendre compte que c'est le remaillage des surfaces courbes qui va poser le plus de problèmes (voir à ce sujet la section 2.6.5.1). Pour pouvoir traiter ce problème d'une manière similaire au remaillage d'un domaine 2D, nous construisons après la phase de détection des noeuds qui nous a donné une liste des noeuds sur chaque face, une série de maillages surfaciques. Ces maillages sont stockés exactement comme des maillages classiques en utilisant une structure de donnée similaire. En particulier, les positions des noeuds d'une face sont partagées par le maillage surfacique de cette face et le maillage global. Ainsi, une modification d'un maillage surfacique est équivalente à une modification du maillage global.

Lorsque cette étape est terminée, nous ajoutons une deuxième description du maillage en terme d'arêtes (2D) et en terme de facettes (3D). En effet, vu que nous utilisons une méthode basée sur les volumes finis pour transférer les grandeurs de la structure de l'ancien maillage vers le nouveau, il est nécessaire de pouvoir manipuler le maillage sous cette forme. Ces deux représentations du maillage pouvant être déduites l'une de l'autre, nous utilisons la représentation "noeuds–éléments" pour construire la représentation "noeuds–arêtes" ou "noeuds–facettes" suivant la dimension du problème.

Lors de notre première année de thèse, nous n'avons pas cru bon de stocker en mémoire cette représentation du maillage. Il est vrai que dans le cas qui nous concernait à l'époque (maillage 2D quadrangulaire structuré), il est relativement facile de passer d'une représentation à l'autre et de recréer la représentation "noeuds–arêtes" chaque fois qu'on en a besoin. Par contre, dans le cas de maillages non structurés, cela devient assez difficile et recréer cette représentation à chaque remaillage devient une opération coûteuse en temps de calcul. Le problème est encore plus compliqué lorsqu'on aborde des maillages tridimensionnels. Nous avons donc décidé de générer et garder en mémoire une fois pour toutes les deux représentations du maillage.

Après avoir traité le maillage global et ses surfaces, nous devons générer automatiquement le maillage auxiliaire [20, 30, 10] (voir section 2.5.8). Ce maillage est un maillage volumes finis qui du découpe chaque élément en un nombre de volumes finis égal au nombre de point de Gauss de l'élément. Chaque volume fini entoure un point de Gauss et servira à transférer les grandeurs déviatoriques de chaque élément.

Ici aussi, une structure complète est recréée de toutes pièces à partir du maillage initial de la structure. Toutes les grandeurs pouvant être liées aux deux maillages à la fois sont, bien entendu, partagées pour accélérer le déplacement d'un maillage lorsque l'autre se déplace. En conséquence, en 3D, si un noeud de la surface est repositionné via une méthode de lissage, trois structures de maillage sont simultanément et automatiquement modifiées : le maillage surfacique, le maillage 3D principal et le maillage 3D auxiliaire.

Ici aussi, nous avons décidé de créer ce maillage une fois pour toutes. Cette façon de faire est justifiée surtout en 3D où le temps de création du maillage peut prendre plusieurs secondes, voire plusieurs dizaines de secondes.

Après cette création de maillage, nous lions les conditions aux limites avec la structure. Il s'agit de positionner des éléments additionnels sur les faces (3D) ou arêtes (2D) où sont appliquées les conditions aux limites. Cela permet de traiter les échanges avec les conditions aux limites de la même manière que les échanges entre deux éléments intérieurs du maillage.

Enfin, à la demande de l'utilisateur, nous effectuons un remaillage initial pour lisser le maillage initial et pour éviter de trop grands déplacements du maillage entre le pas 0 et le pas 1.

2.4.3 Repositionnement des noeuds

Lorsque l'équilibre est atteint à la fin de chaque pas de temps, les noeuds sont repositionnés suivant les instructions de l'utilisateur.

Le repositionnement des noeuds se fait de manière hiérarchique (figure 2.7). Tout d'abord, les noeuds associés aux points sont repositionnés en fonction du type de repositionnement choisi par l'utilisateur. Ensuite, on repositionne les noeuds sur chaque lignes à l'exception des noeuds extrêmes (la liste des noeuds sur les lignes contient également leurs extrémités qui, logiquement, ont déjà été repositionnées lors du repositionnement des points). Ensuite vient le tour des domaines en 2D et des faces en 3D et enfin les volumes (ou domaines 3D).

Les méthodes utilisées seront décrites en détails dans une prochaine section (section 2.6). Il est cependant déjà intéressant de noter que certaines de ces méthodes sont itératives. Il s'agit du repositionnement d'un noeud au centre de gravité de ses voisins (lissage laplacien). Ce type de méthode n'a d'intérêt que si elle est répétée un certain nombre de fois. On ajoute donc une boucle englobant tous les remaillages pour permettre une utilisation efficace de

ce type de méthode.

2.4.4 Transfert des données aux points de Gauss

Lorsque les noeuds de la structure ont été repositionnés, il est nécessaire de transférer les grandeurs aux points de Gauss du maillage initial vers le nouveau maillage (voir aussi section 2.7). Ceci est accompli par la méthode de Godunov (volumes finis) [19, 20, 10]. La méthode S.U.P.G. [16, 10] présentée en première année de thèse n'a pas été réintroduite dans la nouvelle implémentation du formalisme ALE tout d'abord parce qu'elle ne convient que pour des maillages structurés mais aussi parce que nous avons montré qu'elle ne donne pas de meilleurs résultats que la méthode des volumes finis et qu'elle utilise un autre type de maillage auxiliaire qui ne serait pas toujours possible de construire automatiquement.

Grâce à la structure de données, l'algorithme explicite de Godunov est extrêmement compact (une page de code au lieu d'une vingtaine) et son extension au 3D a été immédiate puisqu'il suffit de calculer des flux à travers des facettes aux lieu de flux à travers des arêtes du maillage.

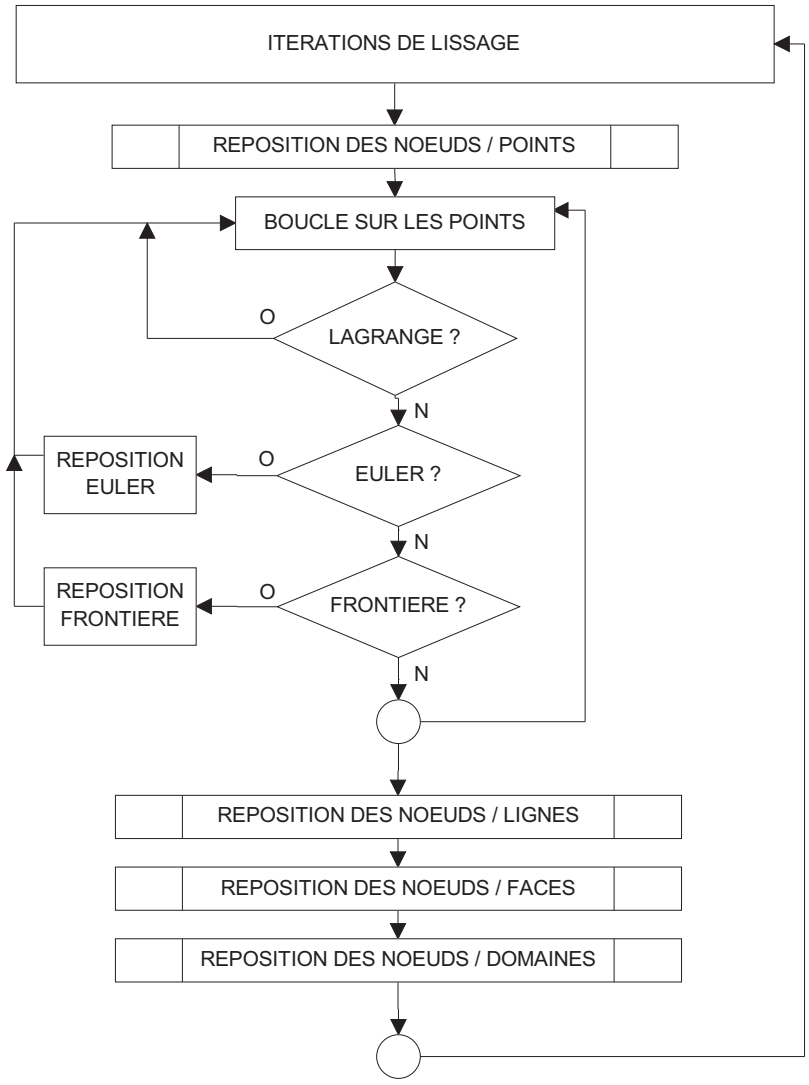

FIG. 2.7: Enchaînement des méthodes de repositionnement des noeuds

2.5 Détection des noeuds sur les entités géométriques

2.5.1 Introduction

A l'heure actuelle, la réussite du traitement ALE se résume, en grande partie, à la réussite des algorithmes d'auto-détection des noeuds sur les entités géométriques que nous avons mis en place pour compenser les lacunes de notre logiciel de pré-traitement qui refuse de nous fournir ce type d'information.

Ces routines sont tout à fait générales et pourront être utilisées dans d'autres cadres que le formalisme ALE. C'est aussi pour cette raison que nous nous sommes investi autant dans la résolution de ce problème. Deux principales applications nous viennent à l'esprit :

- Dans le cadre du contact 3D entre deux solides déformables, nos algorithmes pourraient prendre une place importante. En effet, à l'heure actuelle, l'introduction des données relatives à ce type de contact est extrêmement laborieux puisqu'il faut encoder la liste des noeuds potentiellement en contact tout en respectant un ordre bien défini pour que METAFOR puisse générer les facettes de contact "automatiquement". Inutile de dire qu'il est très fastidieux, vu ces conditions, de raffiner un cas-test 3D en contact déformable–déformable. On en arrive à passer plusieurs heures, voire une journée entière pour encoder correctement un cas-test. Il serait bien plus simple de donner comme information : la face numéro i_1 du domaine j_1 peut entrer en contact avec la surface i_2 du domaine j_2. Cette formulation a beaucoup d'avantages. Non seulement elle réduit considérablement le temps de conception d'un cas-test, elle réduit à zéro le temps pour raffiner un maillage (le jeu de données ne doit pas être modifié au niveau du contact) et elle autorise, dans le futur, un remaillage complet de la structure puisque l'information est donnée sur une entité géométrique et non sur des noeuds. La conversion des données sous cette forme simplifiée en une liste de noeuds et de facettes potentiellement en contact est immédiate via nos routines de détection de noeuds sur surfaces et de création de maillages surfaciques.
- Nous avons toujours l'espoir qu'un jour un remaillage complet de la structure étudiée (avec changement complet de topologie et du nombre de noeuds et d'éléments) sera réalisable. Dans cette optique, comme nous l'avons déjà dit, toutes les conditions aux limites (déplacement imposés, charges mortes, pressions, contact) doivent impérativement être spécifiées sur des entités géométriques. La conversion de l'entité géométrique vers le maillage nécessite des routines d'auto-détection de noeuds comme celles que nous avons implémentées dans la librairie ALE.

Bien entendu, le problème de détection étant assez compliqué, il est très probable que les méthodes employées ici subiront des modifications dans le futur. Mais déjà maintenant, beaucoup de cas ont pu être traités avec succès. Le principal étant de connaître leurs limitations respectives.

2.5.2 Détection des noeuds associés aux points

Cette détection a l'air tout a fait dérisoire et automatique mais nous en parlons quand même ici pour évoquer le problème général des tolérances employées. Celles-ci jouent un grand rôle dans toutes les méthodes qui vont suivre.

Pour détecter un noeud sur un point, on boucle simplement sur l'ensemble des noeuds du maillage et on regarde si le noeud et le point en question coincident.

Vu que nous utilisons des nombres en double précision pour toutes les valeurs réelles de METAFOR, comme les positions, le test doit se faire à une tolérance près. Si cette tolérance est trop grande, on risque de détecter plusieurs noeuds sur un point ou un noeud sur un point qui n'appartient pas au maillage (centre d'un cercle par exemple). Si celle-ci est trop petite, le test risque de ne pas être vérifié à cause, notamment, d'erreurs d'arrondis.

Nous avons donc choisi de déterminer lors de l'initialisation de la librairie ALE les grandeurs caractéristiques du problème traité et de multiplier cette grandeur par une tolérance adimensionnelle fixée à priori (par exemple 1.0e-8). Cette manière de faire permet d'avoir une détection indépendante du système d'unités choisi (ceci a peut être l'air évident mais nous avons été surpris que, par exemple, BACON ne satisfait pas ce critère).

2.5.3 Détection des noeuds associés aux lignes

La liste des noeuds associés aux lignes est la seule information qui nous est transmise du programme de pré-traitement. Nous n'avons donc rien développé pour détecter des noeuds sur des lignes.

La méthode que l'on utiliserait si c'était nécessaire serait assez simple : il suffirait de calculer la distance de chaque noeud a la courbe et de garder ensuite ceux qui fournissent une distance nulle à la tolérance près et une projection comprise entre les deux extrémités du segment. Il faudrait ensuite réarranger la liste obtenue par ordre d'abscisses curvilignes croissantes. En effet, nous verrons que les méthodes de repositionnement développées supposent que les noeuds sont ordonnés de cette façon.

2.5.4 Détection des noeuds associés aux domaines surfaciques

La détection des noeuds dans les domaines surfaciques 2D est un petit peu plus compliquée que les deux problèmes précédents. De plus, elle a beaucoup plus de risque d'échouer et donc de mener à une détection incomplète ou erronée.

Nous avons donc mis au point deux méthodes différentes (figure 2.8) permettant de couvrir la plupart des cas rencontrés en pratique. Le choix de la méthode est effectué en

fonction du nombre de côtés constituant le domaine.

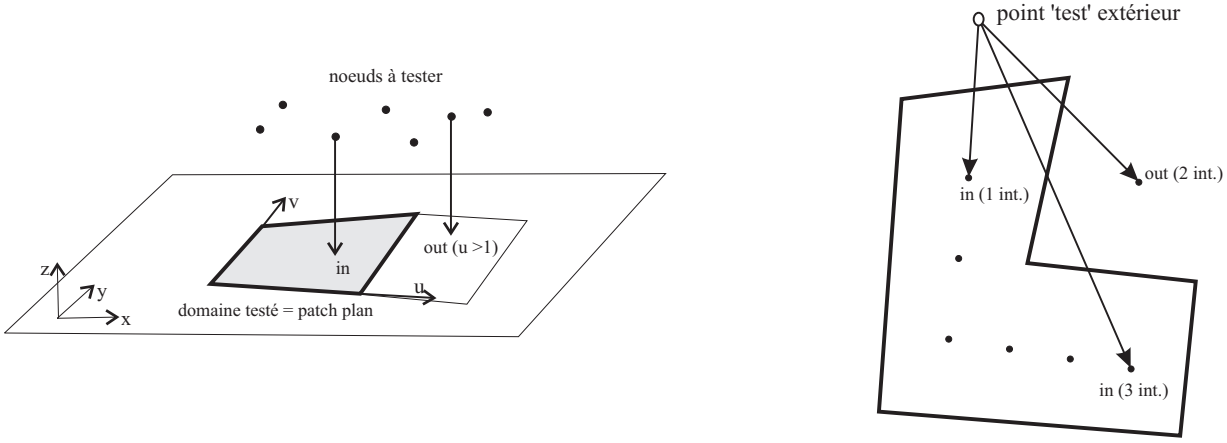

FIG. 2.8: Les deux méthodes de détection des noeuds dans les domaines 2D.

2.5.4.1 Projection sur patches de Coons

Cette méthode est employée si le domaine possède 4 cotés (c'est très souvent le cas). Dans ce cas, pour déterminer l'appartenance d'un noeud au domaine, on considère le problème comme étant tridimensionnel. On déplace le noeud à tester hors du plan (x, y) en lui donnant une coordonnée z arbitraire. On projette ensuite ce noeud sur le patch de Coons plan formé par les quatre lignes du domaine. Il est alors facile de savoir si le noeud à l'intérieur ou l'extérieur du domaine en testant les abscisses curvilignes de la projection qui doivent être comprises entre 0 et 1 pour les noeuds internes. Dans ce cas, vu qu'on ne cherche que les noeuds internes du domaine et pas ceux qui sont sur son contour, on rejette les noeuds extrêmes.

Les lignes formant le contour du domaine peuvent être des segments de droite ou des cercles.

Pour la projection sur ces Coons, nous utilisons tout d'abord un algorithme de grille pour déterminer grossièrement le point le plus proche sur la surface Coons. On utilise ensuite un algorithme de Newton-Raphson pour résoudre l'équation :

$$\vec{d}(\vec{x}(u,v)).\frac{\partial \vec{x}(u,v)}{\partial u} = 0 \quad (2.1)$$

$$\vec{d}(\vec{x}(u,v)).\frac{\partial \vec{x}(u,v)}{\partial v} = 0 \quad (2.2)$$

où $\vec{d}$ est le vecteur reliant le point à projeter et l'estimation de la projection, $\vec{x}$ est la position de la projection et u et v sont les abscisses curvilignes de la projection sur le patch.

2.5.4.2 Intersections d'une droite et du contour du domaine

Une manière beaucoup plus générale mais certainement beaucoup plus coûteuse pour détecter l'appartenance d'un noeud à un domaine est la suivante.

Si on relie le noeud à tester par un segment de droite à un point que l'on sait être en dehors du domaine, il suffit de compter le nombre d'intersections avec le contour du domaine pour être fixé sur son appartenance ou non à celui-ci. En effet, si on compte un nombre impair d'intersections, le noeud est à l'intérieur du domaine. Il est à l'extérieur sinon.

Pour compter le nombre d'intersections, on procède comme suit : on recherche l'intersection du segment de droite avec tous les éléments linéiques formant le contour du domaine et on compte simplement le nombre d'intersections obtenues. Cependant, la situation n'est pas toujours aussi simple et il peut arriver, si on n'y prend garde, qu'une intersection ne soit pas comptée ou bien, ce qui revient au même, soit comptée 2 fois. Cela arrive lorsque l'intersection tombe juste sur un noeud du contour. On pourrait naïvement penser que ça n'arrive presque jamais. Au contraire, vu qu'on traite souvent des maillages réguliers de dimensions entières, la situation est très souvent rencontrée et il faut donc pouvoir gérer correctement cette situation.

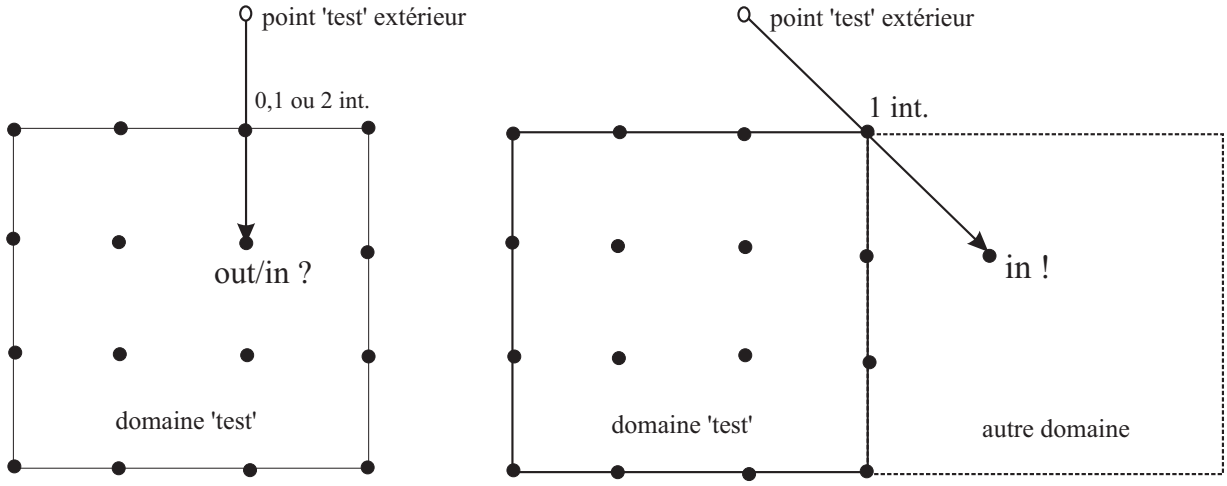

FIG. 2.9: Problème de détection des intersections.

La méthode que nous avons retenue est la suivante : si l'intersection tombe sur l'extrémité d'un élément du contour à la tolérance près, l'éventuelle intersection avec le élément linéique suivant ne sera pas comptabilisée. On arrive ainsi à obtenir un algorithme qui gère presque tous les cas de figure. Seul le cas où le segment passerait malencontreusement sur un coin du domaine à la tolérance près risque d'amener une mauvaise détection. Cependant cette situation n'est pas très courante.

Pour être complet, notons que la determination du point d'intersection des deux seg-

ments se fait par résolution du système 2x2 formé par l'égalité des équations vectorielles de ceux-ci.

2.5.4.3 Réorganisation de la liste des noeuds

Une fois les noeuds détectés dans les domaines surfaciques, il est nécessaire de réorganiser cette liste pour pouvoir effectuer à la demande de l'utilisateur un repositionnement par la méthode d'interpolation transfinie.

Si cette méthode a été choisie pour le domaine, la routine de remaillage est appelée. Cependant, celle-ci ne repositionne pas les noeuds mais elle identifie les noeuds détectés avec les positions générées par le mailleur. La liste de noeuds est alors réordonnée de telle manière à ce que les noeuds soient listés dans l'ordre du remailleur transfini de notre librairie. Grâce à cela nous pouvons utiliser la méthode d'interpolation transfinie quel que soit l'ordre avec lequel les noeuds ont été générés, c'est-à-dire quel que soit le logiciel de pré-traitement utilisé.

2.5.5 Détection des noeuds associés aux faces

C'est ici que nous introduisons une hypothèse qui peut être quelque fois restrictive : les noeuds d'une face d'un domaine 3D ne peuvent être détecté qu'à la condition que cette face soit construite sur un contour de 4 lignes (ni plus, ni moins). Cette hypothèse n'est pas simple à supprimer parce que BACON, notre mailleur 3D ne nous fournit pas directement la topologie de ces faces, c'est-à-dire la manière d'assembler les lignes du contour pour former un patch de Coons.

Par contre, si on se limite à des contours à 4 côtés, le patch de Coons peut être facilement construit et il suffit alors de projeter chaque noeud à tester sur le patch en question (figure 2.10). Si la projection tombe dans le patch (abscisses curvilignes comprises entre 0 et 1), il suffit de calculer la distance entre le noeud à tester et sa projection pour déterminer si oui ou non le noeud est sur la surface définie par la face du domaine.

2.5.6 Détection des noeuds associés aux domaines volumiques

Pour déterminer si un noeud appartient à domaine volumique 3D, on pourrait envisager une extension des deux méthodes utilisées dans le cas des volumes surfaciques 2D. Cependant, la première méthode nécessite de posséder l'équivalent d'un mailleur transfini 3D. Nous n'en n'avons malheureusement pas en notre possession et en construire un de toutes pièces est quand même un travail d'une autre ampleur qu'un mailleur transfini 2D. Nous avons donc abandonné cette méthode jusqu'à ce qu'un tel mailleur soit disponible

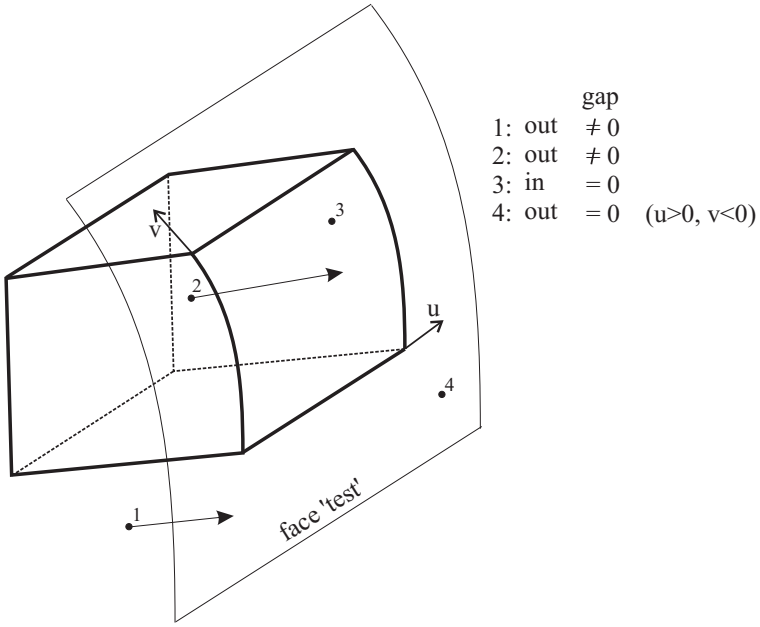

FIG. 2.10: Détection des noeuds sur une face.

dans METAFOR ou Z-Mesh. La deuxième méthode consistait à compter le nombre d'intersections entre la surface du domaine (éléments du contour en 2D et éléments surfaciques en 3D) et un segment joignant le noeud à tester et un point extérieur au domaine. Cette méthode a deux inconvénients en 3D : le premier est relatif au temps de calcul nécessaire. Une surface 3D possède généralement beaucoup plus d'éléments qu'un contour 2D. De plus ce n'est pas l'intersection de deux droites (problème très simple) mais l'intersection d'un patch et d'une droite. Ce serait donc plus long. Le deuxième inconvénient vient de la gestion des intersections avec les noeuds. Nous avons vu que dans le cas d'un maillage structuré, il est fort probable qu'une des droites "test" va avoir son intersection juste ou presque sur un noeud de la face. Dans ce cas, l'intersection risque d'être comptée plusieurs fois (au maximum un nombre de fois égal au nombre d'éléments arrivant à ce noeud). Si ce nombre est pair, l'intersection n'est pas détectée et, finalement, le noeud sera mal détecté. Mettre en oeuvre un remède à ce problème est beaucoup plus compliqué qu'en 2D puisqu'il faudrait négliger les intersections avec toutes les autres facettes contenant ce noeud (possible mais laborieux).

Nous avons donc décidé de mettre au point une autre méthode qui est rapide et qui a beaucoup de chance de réussir (figure 2.11).

Nous nous basons sur la méthode du comptage des intersections que nous appliquons sur la géométrie initiale (lignes et contours) et non sur le maillage. Chaque face est définie par un contour. Vu que nous avons fait, pour les faces, l'hypothèse que ces contours sont composés de quatre lignes (et sans cela, la méthode ne fonctionnerait pas), on construit un patch pour chaque face et on teste les intersections entre ces patch et une droite "test".

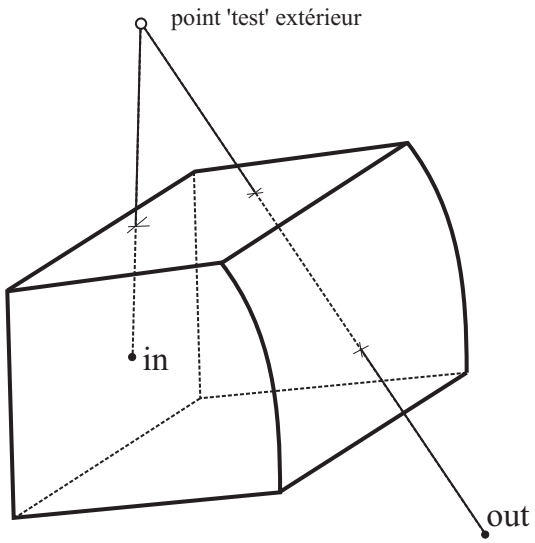

FIG. 2.11: Détection des noeuds dans un domaine volumique.

Ici, pour éviter le problème des arêtes, nous pourrions utiliser le même critère que celui utilisé en 2D mais il ne serait pas aussi fiable puisque nos patches peuvent ne pas être plans. Nous rendons notre algorithme plus robuste en changeant de point "test" pour chaque domaine et nous le plaçons centré juste au dessus de celui-ci. Nous imposons également que le noeud soit considéré dehors dès qu'une intersection est détectée sur une arête du domaine (vu qu'il n'y en a que 12 par domaine et que le noeud test est bien placé, cela mène, très souvent, à une détection totale).

2.5.7 Accélération de la détection

La détection des noeuds peut être assez lente même si les routines calculant des intersections ont été optimisées. Pour accélérer le processus global de détection, nous avons créé une liste des noeuds qui restent à détecter. On commence par réduire cette liste avec les informations contenues dans le fichier de données (noeuds sur les lignes et, en 2D, noeuds sur les points). On ne boucle ensuite que sur les noeuds qui restent pour les détections suivantes et dès qu'un noeud est détecté, il est retiré de la liste. Bien que nécessitant un peu de mémoire supplémentaire, cette méthode nous donne aussi facilement un moyen de vérifier que tous les noeuds ont été détectés (la liste doit être vide à la fin du processus de détection).

2.5.8 Génération automatique du maillage auxiliaire

Pour transférer les grandeurs stockées aux points de Gauss et, plus particulièrement les grandeurs déviatoriques (intégrées sur 4 (2D) ou 8 (3D) points de Gauss), nous avons besoin de construire un maillage auxiliaire.

Ce maillage, que nous ne construisions pas explicitement dans le cadre de nos anciennes routines vu la structure connue du maillage éléments finis, doit être gardé en mémoire lorsqu'on considère des maillages 2D non structurés ou dans tous les cas en 3D. Il serait en effet très coûteux de le reconstruire lors de chaque remaillage. Nous avons donc choisi de le construire et le garder en mémoire quel que soit le type de maillage considéré.

La création de ce maillage volumes finis est tout à fait automatisée, quel que soit le type du maillage.

En 2D, chaque élément fini est divisé en 4 volumes finis. En faisant un usage complet de la structure de donnée décrite plus haut (section 2.3), on peu facilement boucler sur chaque élément pour ajouter un noeud en son centre de gravité[1] et un noeud sur chaque arête du maillage. La construction des volumes finis devient alors assez simple si on a numéroté intelligemment les nouveaux noeuds (chaque noeud du maillage élément fini permet la construction d'un nombre de volumes finis égal au nombre d'éléments finis connectés à lui. On obtient donc un maillage auxiliaire 2 fois plus dense que le maillage initial dans chaque direction. Les arêtes du maillage auxiliaire sont également créées vu qu'elle constitue l'entité de base de la formulation volumes finis.

En 3D, la situation est légèrement plus complexe mais le principe reste le même : en plus du noeud supplémentaire au centre de gravité et sur chaque arête, on en ajoute un au centre de gravité de chaque facette. Les volumes finis sont alors générés en bouclant sur chaque noeud et les facettes,entité de base des volumes finis en 3D, en sont déduites.

Bien que les maillages éléments finis et volumes finis soient considérés comme deux structures séparées, plusieurs grandeurs sont communes. En effet, les positions des noeuds du maillage éléments finis sont liées à celles du maillage auxiliaire lorsque c'est possible. Ainsi, les deux maillages se déplacent simultanément lors du repositionnement des noeuds. Les grandeurs aux points de Gauss sont également liées. Le maillage auxiliaire est donc, pour ces valeurs, une autre manière d'accéder à ces données et il n'est donc pas nécessaire de copier les résultats obtenus par l'algorithme volumes finis du maillage auxiliaire vers le maillage éléments finis.

[1] Il serait plus correct de dire : au centre de gravité de ses 4 sommets ; cette position et le centre de gravité du quadrangle ne correspondant que dans des cas particuliers.

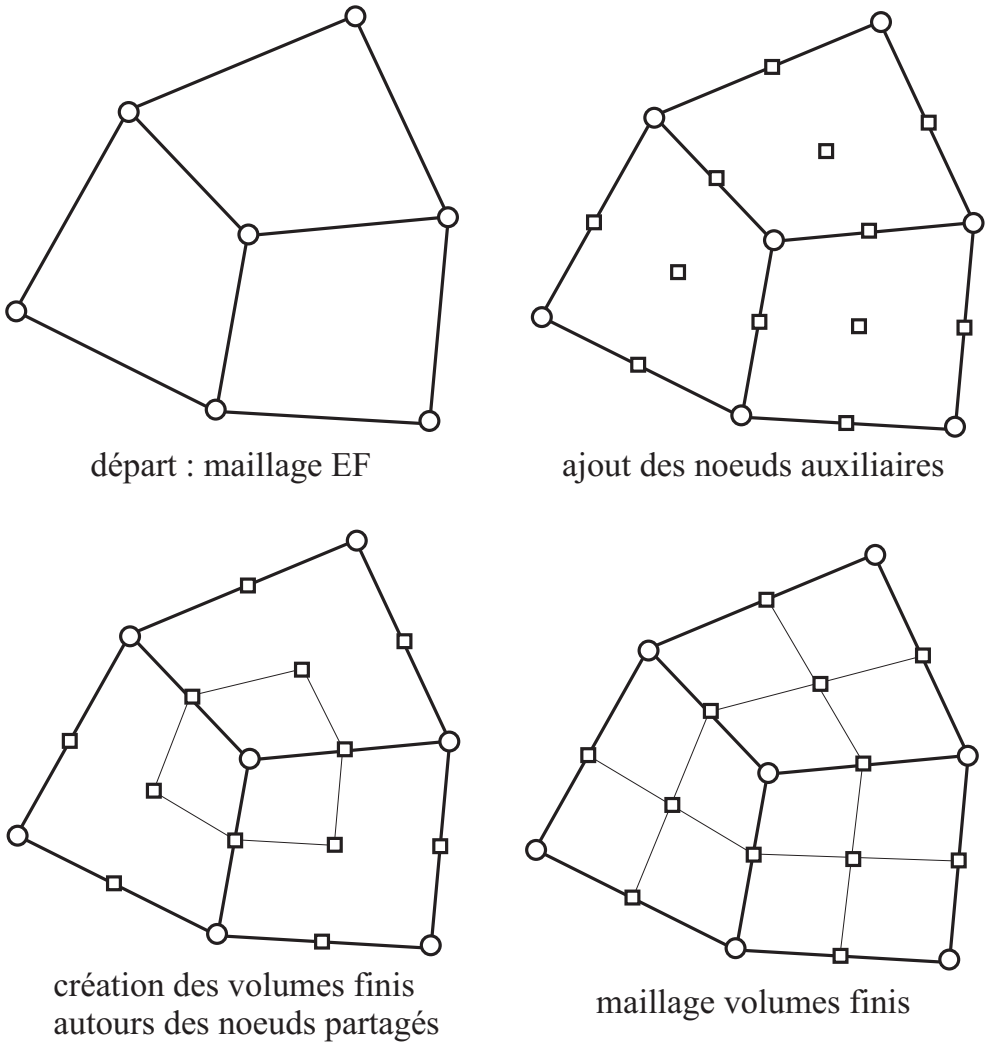

FIG. 2.12: Etapes de création du maillage auxiliaire.

2.5.9 Conclusions & perspectives relatives au pré-traitement

Cette section, bien que très descriptive et très technique est nécessaire pour comprendre le fonctionnement de la librairie. Lorsque les étapes décrites ci-dessus sont accomplies, les données sont prêtes en mémoire pour être utilisées de manière optimale.

Nous ne sommes pas arrivé du premier coup à ce résultat. La plupart des routines ont d'abord été testées séparément où la mémoire n'était pas partagée entre les structures. Certaines données étaient alors dupliquées et il fallait régulièrement mettre à jour tous les tableaux. Ceci peut être assez fastidieux. c'est le cas de la mise à jour des grandeurs aux points de Gauss puisque, a priori, chaque élément peut être orienté dans n'importe quel sens par rapport à son voisin et que les points de Gauss sont numérotés en fonction de cette orientation. Il est donc intéressant, comme nous l'avons fait, de déterminer une fois pour toutes lors de l'initialisation, les points de Gauss en vis-à-vis et leur correspondance avec les volumes finis.

Citons ici quelques améliorations possibles :

- Les routines de projection pourrait certainement être optimisées en vitesse en stockant en mémoire certaines données. Par exemple, lorsqu'on projette un point sur un patch de Coons, la structure de la surface est analysée lors de chaque évaluation de celle-ci.
- Il serait intéressant de rendre plus robuste la détection des noeuds sur les entités géométriques en trouvant une solution rapide pour les cas pathologiques (par exemple, changer la position du point extérieur lorsqu'un tel cas est rencontré).
- Il faudrait uniformiser les tolérances utilisées lors de la détection des noeuds et analyser l'influence de celles-ci.
- Un gain de performance pourrait être obtenu en évitant les allocations–déallocations successives de mémoire. En effet, cette façon de faire souvent bien pratique fragmente la mémoire et ralentit les performances du système.

2.6 Création de la configuration remaillée

2.6.1 Introduction

Un fois l'état de la mémoire préparé et les structures créées, le repositionnement des noeuds de la structure est assez simple. Chaque entité géométrique pointe vers ses noeuds et ceux-ci sont arrangés de façon optimale (dans l'ordre avec lequel le mailleur transfini parcourt la surface pour un domaine 2D et dans le sens des abscisses curvilignes croissantes pour les lignes).

Dans l'ancienne version des routines ALE, METAFOR bouclait sur les domaines et remaillait les lignes de ce domaine puis l'intérieur. Cette manière de faire n'a que des inconvénients :

- Tout d'abord les points sont oubliés. Cela veut dire que les points sont toujours considérés lagrangiens. Or, chaque ligne est délimitée par deux points, donc les lignes ne pouvaient pas laisser la matière les traverser. Il était toutefois possible d'arriver à ce résultat en fixant manuellement le maillage. Nous expliquerons plus loin (section 2.6.2) la solution automatique introduite dans notre librairie.
- En bouclant sur les domaines puis sur les lignes, on remaille plusieurs fois la même ligne si plusieurs domaines ont une frontière commune. Ce qui peut donner des résultats assez étonnants.

En conclusion, nous n'utilisons pas la même méthode qui se révèle assez peu contrôlable dans le cas d'un problème à plusieurs domaines et peu générale vu que les frontières du maillage délimitent toujours la matière.

La technique de remaillage est assez simple. Chaque entité géométrique possède un type de remaillage. Ce type dépend, bien sûr, de la nature de l'entité considérée, quoique de nombreuses méthodes sont communes. On repositionne tout d'abord les noeuds associés aux points suivant le type de repositionnement fournit par l'utilisateur. Ensuite, on traite l'entièreté des lignes, puis les domaines surfaciques en 2D, ou, en 3D, les faces puis les domaines volumiques.

Certaines méthodes étant itératives (elles dépendent de la position des noeuds voisins et, donc, doivent être appliquées plusieurs fois d'affilée), on recommence ce processus un nombre de fois déterminé par l'utilisateur en fonction du problème. Cette boucle supplémentaire, placée au dessus de toutes les boucles sur les entités géométriques permet d'appliquer ces méthodes sur plusieurs domaines, lignes et points en même temps.

2.6.2 Repositionnement des noeuds associés aux points

Dans les anciennes routines, un point ne pouvait être associé qu'à un noeud lagrangien. c'est-à-dire que le noeud n'était pas repositionné. Nous avons ajouté deux types de reposi-

tionnement supplémentaire pour les noeuds associés aux points. Ceci permet de traiter de façon automatique ce que nous appelons "les frontières eulériennes", c'est-à-dire les lignes permettant un flux de matière à travers elle.

- Le type "eulérien" (2D/3D) : le noeud est repositionné à sa position d'origine. Il sera donc fixé dans l'espace à sa position initiale pendant toute la durée du calcul.
- Le type "frontière" (2D uniquement) : le noeud est l'extrémité d'une ligne de type "frontière". Ce noeud peut coulisser selon la direction de la ligne frontière associée. Par exemple, si la ligne frontière est verticale, le noeud est repositionné sur cette verticale. Sa position précise est donnée par l'intersection du maillage avec cette droite. Ce type permet de simuler très efficacement une limite du domaine maillé par lequel la matière s'échappe. On évite ainsi un écrasement de mailles sur la frontière. Remarquons que cette fonctionnalité avait été mise en place de manière très sommaire (il fallait recompiler le code pour l'activer) dans l'ancienne version. Ici, le repositionnement est automatique et la ligne associée peut être une droite de n'importe quelle pente. Une extension au 3D devra être envisagée. Dans ce cas, les frontières eulériennes sont des plans et le noeud doit rester sur son plan frontière associé.

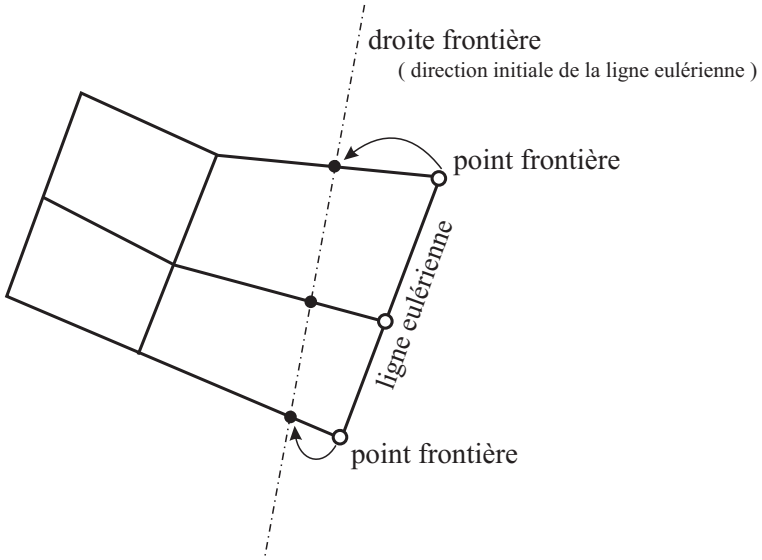

FIG. 2.13: Repositionnement d'un noeud frontière.

Trois types encore non présents dans le code sont envisagés dans un avenir proche :

- Le type "lissage volumique" (2D/3D) : le noeud est replacé au centre de gravité soit des arêtes voisines soit des éléments voisins. Ce type est important pour un point intérieur généré par une division d'un domaine complexe en une série de sous-domaines plus simples.
- Le type "lissage surfacique" (3D) : idem sur une face d'un volume complexe. On se limitera certainement à des surfaces planes. La gestion des surfaces courbes étant beaucoup plus compliquée. Les surfaces planes sont de toutes façons très utiles et généralement utilisées sur les plans de symétries éventuels du problème.

- Le type "lissage linéique" (2D/3D) : idem sur une arête. On fera l'hypothèse que le point est commun à deux segments dont la tangente unitaire est continue. Le noeud pourra dont glisser d'un segment vers l'autre.

Avec ces différents types de repositionnement, on peut aborder de manière automatique (un seul exécutable) la totalité des cas présentés dans le rapport de première année et celui-ci.

2.6.3 Repositionnement des noeuds sur les lignes

Dans l'ancienne version des routines ALE, une ligne devait délimiter obligatoirement la surface de la matière ou être eulérienne. Cependant, certaines restrictions étaient imposées : les lignes eulériennes (type "eulérien adaptatif") devaient être soit horizontales, soit verticales. De plus, les noeuds extrêmes étant repositionnés en même temps que ceux de la ligne, on pouvait parfois observer les extrémités d'une ligne eulérienne se déplacer !

Nous avons évité ce problème en évitant de repositionner les noeuds extrêmes de chaque courbe. Ceux-ci, qui sont logiquement associés à des points ont déjà été repositionnés lors de l'étape précédente.

Remarquons aussi qu'il était assez laborieux de vouloir créer un maillage non régulier sur une ligne et utiliser le formalisme ALE. En effet, dans le cas d'un maillage non régulier, les abscisses devaient toutes être spécifiées dans un fichier annexe. Ce qui n'est pas trop contraignant lorsque le maillage est grossier. Par contre, entrer les 120 valeurs pour chaque ligne du cas test de laminage (voir rapport de première année [5]) nous avait définitivement guéri de l'envie de raffiner le maillage près de la zone de contact.

Dans les nouvelles routines, les abscisses curvilignes sont considérées comme constantes et calculées à partie de la configuration initiale.

Les types de remaillage suivants sont disponibles pour les lignes en plus du type classique "lagrangien" :

- Le type "eulérien" (2D/3D) : en 3D, on repositionne les noeuds à leurs positions initiales. Dans le cas 2D, si c'est une droite, les noeuds internes de la ligne sont repositionnés sur une droite joignant les deux nouvelles positions des noeuds extrêmes en respectant les abscisses curvilignes initiales. Si un des points de cette droite est de type "frontière", la droite est une droite frontière. Dans le cas d'un cercle, les noeuds sont repositionnés à leurs positions initiales.
- Le type "spline" (2D/3D) : les noeuds sont repositionnés sur une spline cubique dont les pôles sont les noeuds de la ligne en question. Les abscisses curvilignes initiales sont conservées. La construction de la spline est décrite en détails dans l'annexe A.
- Le type "cercle" (2D) : c'est le type de remaillage qui est employé dans les anciennes routines [27]. L'algorithme a été reprogrammé vu qu'il donne de bons résultats (similaires à ceux donnés par les splines) mais semble être moins coûteux en temps de

calcul et en mémoire. Pour chaque noeud, on détermine le cercle qui passe par les trois noeuds voisins les plus proches du noeud à repositionner. La nouvelle position du noeud sera l'intersection entre le cercle et le rayon passant par le noeud. Nous n'avons pas étendu cette méthode au 3D mais cela reste possible. La gestion des cercles 3D n'était pas encore au point lors de l'écriture de cette méthode.

- Le type "lissage volumique" (2D/3D) : les noeuds sont replacés au centre de gravité de leurs voisins. Cette méthode, utile pour des lignes communes à plusieurs domaines permet le passage de matière d'un domaine à l'autre.
- Le type "lissage surfacique" (3D) : la méthode précédente est employée mais la ligne est considérée sur une face plane. Cette méthode est utile pour gérer facilement les lignes sur des plans de symétrie.

2.6.4 Repositionnement des noeuds sur les domaines surfaciques 2D

Les domaines surfaciques 2D sont les correspondants directs des domaines volumiques en 3D. Plusieurs méthodes étaient disponibles dans les anciennes routines. Nous avons reprogrammé et amélioré les plus utiles.

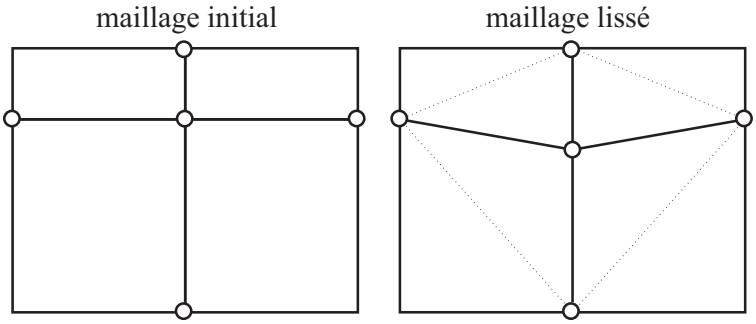

FIG. 2.14: Repositionnement au centre de gravité des noeuds voisins.

A côté du type "lagrangien", un domaine 2D peut avoir :

- Le type "eulérien" : Les noeuds sont repositionnés à leurs positions d'origine. Ce type de remaillage est assez peu utilisé.
- Le type "lissage volumique" : les noeuds sont replacés au centre de gravité de ses voisins ou des éléments voisins. C'est la méthode la plus générale qui fonctionne plus ou moins bien avec tous les types de maillage (structurés ou non). On peut cependant rencontrer des problèmes de distorsion dans le cas de maillages structurés dont les mailles n'ont pas une taille uniforme (figure 2.14). Dans ce cas, il serait intéressant (mais ce n'est pas encore fait), de pondérer la somme des positions pour retrouver, si la matière ne s'est pas déplacée, un déplacement de maillage nul.
- Le type "méthode d'interpolation transfinie" : cette méthode a été reprogrammée parce qu'elle donne de très bon résultats dans le cas de maillages structurés. Cette

méthode repositionne les noeuds en une seule passe contrairement aux méthodes de type "lissage volumique". Dans l'ancienne version du code ont était limité à 4 côtés par domaine. Nous avons fait tomber cette limitation en faisant passer une information sur la topologie du domaine maillé à partir du programme de pré-traitement vers la structure de données METAFOR. Cette information est simplement constituée de 4 listes de lignes. Dans le cas d'un domaine délimité par 4 courbes, chaque liste ne contient qu'un seul numéro.

- Le type "minimisation" : ce type (plutôt expérimental) de remaillage est basé est un lissage volumique pour lequel on replace le noeud dans la position qui minimise la longueur des arêtes aboutissant à lui. Cette méthode a l'avantage de remailler un maillage structuré non homogène correctement (les mailles restent rectangulaires). Cependant, la méthode de minimisation est une simple méthode de bisection et devrait être remplacée par une méthode analytique.

2.6.5 Repositionnement des noeuds sur les faces

Le repositionnement des noeuds sur les faces est certainement la tâche la plus compliquée de toute la librairie. Le problème est le suivant : en un instant t, la face est définie par un nuage de points (noeuds du maillage surfacique) qui ne sont pas nécessairement coplanaires. Il faut repositionner ceux-ci de telle manière à ce que les éléments surfaciques soient moins distordus tout en conservant dans la mesure du possible la forme globale de la surface.

En cherchant dans la littérature (voir par exemple [26] en mécanique des fluides), nous n'avons pas trouvé de méthode assez générale qui permette de traiter tous les cas rencontrés (la plupart des auteurs considèrent qu'un noeud a, au maximum, 4 voisins – voir par exemple [1]). D'autres considèrent que les surfaces courbes sont automatiquement lagrangiennes [28] en supprimant ce problème.

Notre but est de pouvoir traiter aussi bien des maillages surfaciques structurés que des maillages surfaciques non structurés. Nous avons donc mis au point une méthode assez générale pour pouvoir traiter tous les cas.

La méthode est basée sur la méthode du remaillage des lignes par spline et est une extension directe de celle-ci. Une surface générée par des splines est calculée à partir du nuage de point et le maillage est lissé sur celle-ci.

2.6.5.1 Création de la surface spline

Si on considérait que le maillage surfacique est structuré et généré initialement par un mailleur transfini, on posséderait un double réseaux de lignes plus ou moins parallèles. Sur chacune de celles-ci, on pourrait créer une spline et ce réseau de splines définirait

les génératrices d'une surface continue. Dans ce cas particulier, on peut même garantir facilement la continuité de la surface obtenue jusqu'à la dérivée seconde.

Pour notre librairie, nous avons besoin de pouvoir traiter non seulement ce genre de maillage (c'est celui qu'on rencontrera dans la majorité des cas) mais aussi des maillages quadrangulaires tout à fait non structurés, provenant par exemple d'un mailleur frontal.

Nous allons donc construire ce genre de surface sur un maillage quelconque. Cependant, nous ne pourront pas assurer un même degré de continuité.

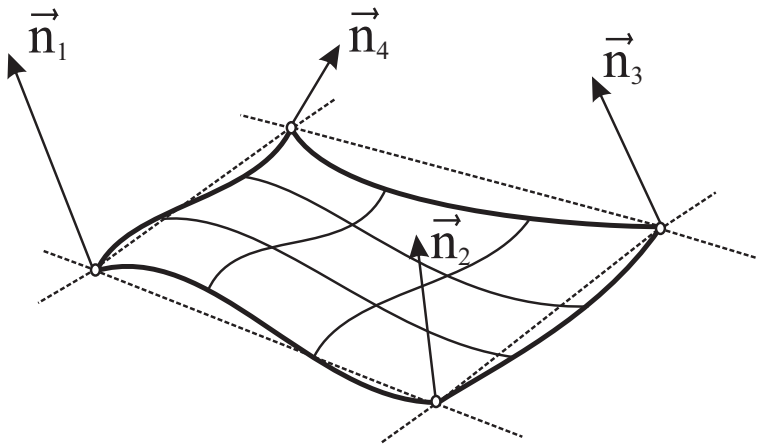

FIG. 2.15: Un élément surfacique et sa surface spline associée.

La construction de la surface se déroule en deux étapes (voir aussi annexe A) :

- Dans un premier temps, on calcule les normales moyennes en chaque noeud de la surface. Pour ce faire, pour un noeud donné, on moyenne les normales de chaque élément surfacique arrivant à ce noeud en considérant celui-ci comme un patch bi-linéaire (c'est-à-dire, la surface exacte de l'élément fini). Vu qu'on ne moyenne pas les normales unitaires, on prend automatiquement en compte l'inhomogénéité éventuelle du maillage. Ces normales sont stockées dans la structure de données comme une caractéristique de chaque noeud. Les noeuds appartenant à deux faces du maillage doivent avoir deux normales puisque, dans la plupart des cas, l'intersection de deux faces constitue une arête vive où la normale doit être maintenue discontinue. Cela a posé un problème de stockage en mémoire. En effet, si les noeuds du maillage surfacique sont partagés avec le maillage global, un noeud ne peut avoir qu'une seule normale. La solution brutale serait de ne plus partager ces noeuds entre les maillages mais il faut alors, dans ce cas, mettre à jour un maillage lorsque l'autre bouge ; ce qui est une perte de temps évidente. La solution est simple : les noeuds des maillages surfaciques sont dédoublés (donc une normale différente si on y accède via un maillage ou via un autre) mais pas leur position. Cette solution règle aussi un autre problème : lorsqu'on accède à un noeud via le maillage surfacique et qu'on cherche les éléments adjacents, on obtient les éléments surfaciques adjacents ; par contre, on obtient les

éléments volumiques si on y accède par le maillage global.

- Une fois les normale calculée, on peut générer la surface. Sur chaque arête d'un élément surfacique, on crée une spline. Les quatre splines créées forment un patch où l'interpolation se fera de manière bi-linéaire. Le fait d'avoir calculé les normales en chaque noeud garantit que, dans le cas d'un maillage structuré bien régulier, les tangentes sont continues en chaque noeud. Dans le cas général, la surface est simplement de continuité C_0. On peut cependant facilement comprendre qu'elle représente mieux une surface réelle qu'un réseau de patches bi-linéaires construit sur les arêtes des éléments surfaciques.

La surface ainsi construite peut être facilement tracée en parcourant tous les éléments de la surface. Par contre, il n'est pas évident de définir des coordonnées curvilignes globales sur la surface vu que nous avons supposé que le maillage n'était pas forcément structuré. Ceci ne nous gêne pas puisque nous pourront nous en passer lors du lissage du maillage surfacique sur cette surface.

La figure 2.16 montre un maillage "test" que nous avons choisi structuré mais qui aurait bien pu être non structuré. Quatre noeuds internes du maillage ont été déplacés de manière aléatoire. La figure 2.17 montre la surface spline obtenue par notre algorithme de génération de surface. On peut constater que le résultat obtenu ressemble beaucoup plus à une surface réelle que le maillage initial.

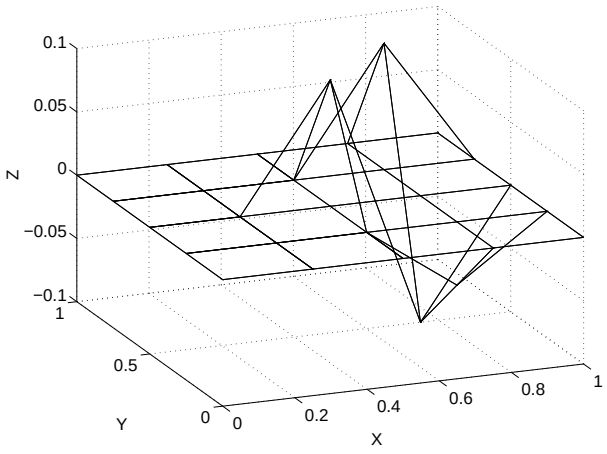

FIG. 2.16: Maillage déformé.

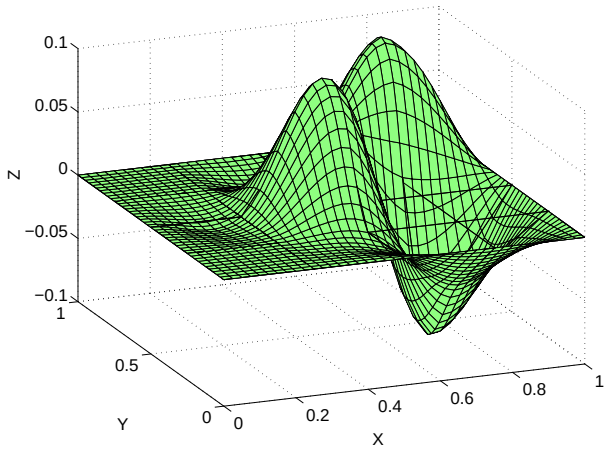

FIG. 2.17: Surface spline construite sur le maillage.

La figure 2.18 montre le même maillage où nous avons déplacé un seul noeud interne selon z et la figure 2.19 montre la surface résultante. Sur cette dernière figure, on voit bien l'influence de la perturbation dans le maillage. La position du noeud influence non seulement la surface spline des éléments directement voisins mais aussi les voisins de ces éléments (par l'intermédiaire de la normale). On peut cependant remarquer que la surface obtenue ne possède pas une symétrie de révolution.

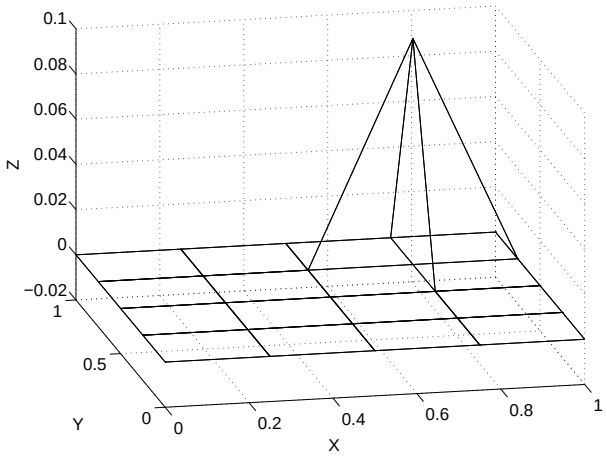

FIG. 2.18: Maillage déformé.

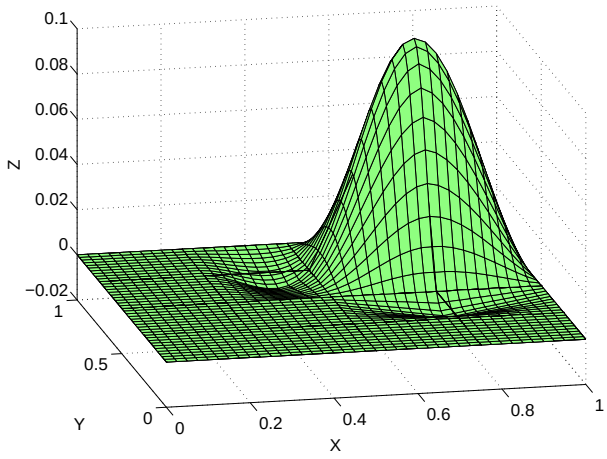

FIG. 2.19: Surface spline construite sur le maillage.

La figure 2.20 montre la surface de la figure 2.19 sous un autre angle. Ceci nous permet de faire deux remarques intéressantes :

- La dérivée de la surface n'est pas continue à travers les arêtes (voir l'angle formé par la surface sous la cloche sur la figure). Pour obtenir la continuité, nous aurions dû interpoler les 4 splines formant un patch non pas linéairement mais par des splines cubiques ; ce qui serait vraiment très coûteux en temps de calcul. Remarquons que, même aux noeuds, les dérivées ne sont continues que dans des cas très particuliers (maillage régulier non déformé).
- Les normales calculée de part et d'autre de la surface en forme de cloche de la figure 2.20 ne sont pas symétriques. En effet, la normale en $x = 0.5$ est une moyenne sur 4 éléments alors que celle en $x = 1$ est une moyenne sur 2 éléments (puisque le noeud associé est sur la frontière du maillage.

Malgré les quelques défauts de continuité et de symétrie énumérés ci-dessus, nous pensons que l'utilisation d'une telle surface d'approximation est très intéressante. En effet, par rapport à une surface composée de patches de Coons linéaires, la surface spline semble mieux représenter une surface réelle et donc éviter les pertes ou gain de volume dû au remaillage. De plus, par rapport à une surface dont la dérivée est continue, notre solution a l'avantage d'être assez simple et rapide à évaluer numériquement (n'oublions pas que nous devons aussi évaluer les 2 dérivées premières et les 4 dérivées secondes !).

2.6.5.2 Lissage sur la surface spline

Vu que nous n'avons pas accès à un système de coordonnées curviligne global sur cette surface, nous devons impérativement utiliser une méthode locale de lissage.

La méthode choisie est basée sur le lissage utilisé pour les domaines plans. On repo-

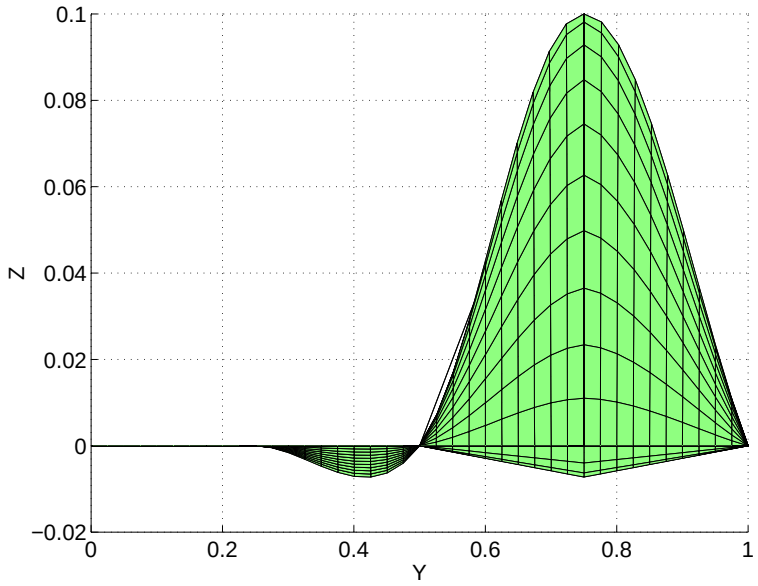

FIG. 2.20: Vue de profil de la surface créée.

sitionne tout d'abord le noeud au centre de gravité de ses voisins. Après cette opération, le noeud ne se trouve plus sur la surface à moins que la surface soit localement plane. On projette donc le noeud sur la surface spline pour obtenir sa nouvelle position compatible avec la surface.

Il faut donc projeter un noeud sur cette surface composite. La méthode employée est la même que celle utilisée pour projeter un noeud sur un patch de Coons (méthode de grille suivie d'un algorithme de Newton).

Cependant, quelques remarques s'imposent :

- Dans le cas de ces surfaces du troisième degré, l'évaluation de la dérivée première et de la dérivée seconde de la surface en un point est tout à fait indispensable pour obtenir une convergence rapide. Il arrive même, pour des surfaces assez bien déformées, que l'oubli de ces dérivées entraîne une divergence de l'algorithme de projection. Cela a pour conséquence directe d'augmenter assez fortement le temps de calcul : l'évaluation d'une dérivée est presque aussi coûteuse que la fonction elle-même et il y a 2 dérivées premières et 4 dérivées secondes !
- Vu que la surface est composite, il est impératif qu'au cours du processus de Newton-Raphson, le noeud puisse passer d'un patch à un autre sans quoi chaque noeud reste coincé sur le premier patch où il a été détecté. Nous avons donc mis en place un algorithme qui permet à l'approximation de Newton-Raphson de passer d'un patch à son voisin en changeant, si c'est nécessaire, son système de coordonnées (les éléments sont tous définis dans le même sens mais les coordonnées curvilignes ne sont pas automatiquement compatibles). Ceci est décrit figure 2.21.

L'algorithme, comme toute méthode de lissage, doit être répété plusieurs fois pour

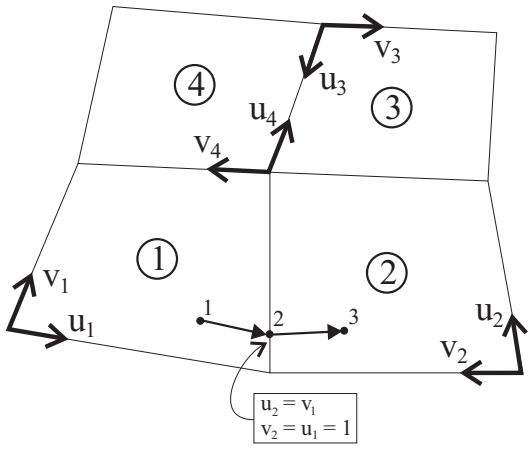

FIG. 2.21: Changement de coordonnées lors du passage d'un patch à son voisin.

obtenir un lissage satisfaisant du maillage.

2.6.5.3 Accélération du lissage

Notre première implémentation du lissage était vraiment très lente (plusieurs secondes pour lisser une surface composée de quelques centaines d'éléments). Nous avons compris rapidement qu'il fallait limiter au maximum le nombre d'évaluations de la fonction. Les évaluations de Newton sont coûteuses (puisqu'il faut calculer les dérivées) mais généralement peu nombreuses. C'est donc au niveau de la méthode de grille, c'est-à-dire dans la recherche du première approximation de Newton, que l'on peut essayer de gagner du temps.

Deux paramètres sont déterminants par la méthode de grille :

- Le premier est le nombre de patches testés. Si on teste tous les patches de la surface, on perd beaucoup de temps. En effet, il y a beaucoup de chance pour que la projection se trouve dans les éléments surfaciques voisins. Nous avons donc choisi de ne tester que ces éléments (généralement 4 pour un maillage de type transfini).
- Le second est le nombre de points testés sur chaque patches. Au départ, nous avions choisi de tester une dizaine de points dans chaque direction, ce qui est clairement trop. En essayant de réduire ce nombre, nous avons conclu que 4 évaluations par patches et dans chaque direction était un minimum pour converger dans les cas que nous avons rencontrés.

Il serait encore possible d'accélérer les évaluations en particularisant les routines d'évaluation de splines et en stockant certaines informations relatives au patch dans lequel est l'approximation de Newton-Raphson. En effet, pour l'instant, la longueur de ces splines est calculée à chaque évaluation ainsi que les tangentes déduites des normales.

Cependant, ces modifications seraient assez compliquées pour n'apporter que peu de

performances supplémentaires et cela rendrait les routines moins lisibles. Le principal est d'avoir obtenu un algorithme de projection sur surface composite dont le temps de calcul est indépendant du nombre d'éléments la composant.

2.6.5.4 Autres méthodes de repositionnement

Une autre méthode a été implémentée pour pouvoir traiter les cas particuliers de surfaces planes (principalement des plans de symétrie). Cette méthode consiste à faire un lissage identique à la méthode précédente sans projection.

2.6.6 Repositionnement des noeuds dans les domaines volumiques

Le repositionnement des noeuds dans les domaines volumiques 3D est très similaire au repositionnement dans les domaines surfaciques 2D. Nous n'avons cependant pas pu reprogrammer un mailleur transfini 3D. Il n'existe donc qu'une méthode de lissage volumique par repositionnement au centre de gravité des éléments voisins, à côté des traditionnels remaillages lagrangien et eulérien.

2.6.7 Conclusions & perspectives

Dans cette section, nous avons décrit toutes les méthodes de repositionnement de noeuds actuellement disponibles dans la librairie ALE. Celles-ci permettent de traiter tous les types de problèmes 2D et la plupart des problèmes 3D pour lesquels la matière ne traverse pas le maillage.

Pour pouvoir traiter ce genre de problème, nous avons besoin en 3D d'une gestion automatique de ce que nous avons appelé les "frontières eulériennes" comparable aux lignes eulériennes et points de type "frontière" en 2D.

Etendre ce concept au 3D est plus difficile mais cela reste de la simple géométrie analytique. On procédera de la manière suivante (voir aussi figure 2.22) :

- On définira des plans "frontières". Ces plans auront un type de remaillage "lissage sans projection". On veillera à ce que ce lissage se fasse dans le plan initial de cette surface. Les noeuds pourront donc se déplacer mais en restant contraint sur leur surface initiale. Ce type de repositionnement sera un extension du type "eulérien" et sera activé lorsqu'un plan eulérien sera délimité par des lignes de type "frontière" que nous allons décrire.
- On définira des lignes "frontières". Le remaillage de ces lignes sera assez simple : pour chaque noeud de la ligne, on repositionnera le noeud à l'intersection d'un plan et d'une droite : le plan frontière associé à la ligne et l'intersection des 2 éléments

surfaciques arrivant à ce noeud et n'appartenant pas au plan frontière. Dans le cas où le noeud possède plus de 2 éléments surfaciques voisins, une procédure spéciale devra être choisie (moyenne des intersections, par exemple).

- On définira des points "frontières" 3D dont les noeuds seront repositionnés comme les noeuds sur les lignes.

Ces nouveaux types permettront de traiter des problèmes comme le laminage en ne discrétisant qu'une petite portion de tôle.

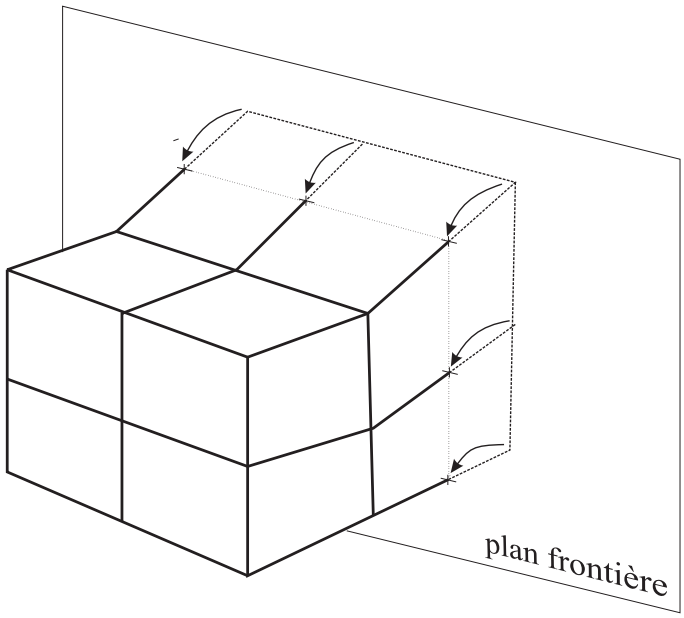

FIG. 2.22: Gestion des surfaces frontières en 3D.

2.7 Transfert des données d'un maillage à l'autre

2.7.1 Introduction

La deuxième étape importante après le remaillage de la structure dans un problème traité par le formalisme ALE est le transfert de données entre l'ancien maillage (maillage équilibré à la fin du pas de temps) et le nouveau qui vient d'être créé par les routines de repositionnement de noeuds.

2.7.2 Méthode utilisée

Cette étape se déroule en 3 phases successives quelle que soit la dimension du problème :

- Tout d'abord, le maillage auxiliaire doit être mis à jour. En effet, les noeuds communs avec le maillage éléments finis ont été déplacés lors du pas de temps et du remaillage mais ce n'est pas le cas pour les noeuds supplémentaires. Il suffit donc de repositionner ces noeuds pour qu'ils soient bien au centre de gravité de l'entité géométrique correspondante (arêtes, facettes, éléments).
- Ensuite, on effectue une convection des données déviatoriques (les composantes du déviateur du tenseur contrainte) et des variables internes à l'aide du maillage auxiliaire. Si le problème est dynamique, la masse volumique est aussi convectée. Pour l'instant, les matériaux hyperélastiques, ainsi que les grandeurs thermiques, ne sont pas (encore) transférées. Ceci est dû au fait que ces matériaux ne sont pas encore accessible aux utilisateurs de la "nouvelle tête" de METAFOR.
- Enfin, on transfère la pression à l'aide du maillage éléments finis considéré comme maillage auxiliaire volumes finis. Grâce à notre utilisation intensive de la structure, la même routine peut être utilisée pour ces deux types de transfert (ce qui n'était pas le cas dans l'ancienne version).

La méthode utilisée est la méthode de Godunov [20, 30, 10]. C'est une méthode de volumes finis traditionnelle pour la discrétisation spatiale. L'équation de convection est alors intégrée temporellement via un schéma classique explicite d'Euler du premier ordre. Il n'y a donc rien de conceptuellement nouveau dans l'étape de transfert si ce n'est que celle-ci peut se faire sur un maillage non structuré et également en 3D.

Remarquons que la méthode S.U.P.G. n'a pas été implémentée vu qu'elle n'apporte pas d'améliorations vis à vis de son homologue volumes finis. De plus elle est difficilement généralisable au cas non structuré.

2.7.3 Gestion des conditions aux limites

La gestion des conditions limites est capitale pour pouvoir simuler efficacement des problèmes où le flux de matière traverse le maillage. C'est généralement le cas pour la simulation des problèmes stationnaires tel que le laminage qui nous intéresse particulièrement.

Dans l'ancienne version du code, une gestion assez simplifiée et restrictive des conditions limites permettait de s'en sortir dans de nombreux cas. Nous avons rendu cette gestion claire et précise.

Des conditions aux limites doivent être appliquées dès qu'un flux de matière entre dans le maillage. Cependant, si on se place dans le cas 2D, lorsque l'on remaille des lignes il y a inévitablement des petits flux de matière qui entrent et sortent du maillage. Ceci est dû au fait qu'on remaille une ligne brisée par une autre ligne brisée et que la surface du domaine est approximée. Ces flux, aussi petits soient-ils, ne doivent pas donner lieu à une application de conditions aux limites. En effet, cela ne peut que modifier indésirablement la solution.

Nous décidons, en conséquence, de négliger ce genre de flux et, par extension, tous les flux à travers des lignes (2D) ou faces (3D) pour lesquelles l'utilisateur n'a pas spécifié de conditions aux limites.

Ainsi, un problème où les lignes (ou faces) délimitent exactement la matière au cours du calcul ne sera jamais perturbé par une application abusive de conditions aux limites si l'utilisateur ne spécifie rien.

Dans l'ancienne version, les conditions aux limites étaient entrées sous forme de boites dans un fichier annexe (toutes les arêtes dans cette boite avaient la même condition aux limites).

Dans la nouvelle version, les conditions aux limites sont appliquées sur la géométrie (lignes et faces) directement. C'est beaucoup plus pratique lorsque les dimensions du problème sont paramétrées (il n'était pas possible de paramétrer les boites) et c'est beaucoup plus précis (que faire quand deux lignes doivent avoir des conditions aux limites différentes et ne peuvent pas être placées dans deux boîtes différentes ?).

De plus, il est possible de spécifier une valeur limite différente pour chaque grandeur stockée aux points de Gauss du maillage. On peut donc ainsi facilement imposer les forces de traction et contre-traction lors d'une simulation de laminage, par exemple (en imposant σ_{11} à la valeur prescrite).

Numériquement, les conditions aux limites sont appliquées de manière "faible" c'est-à-dire hors du maillage (ou, d'une manière équivalente, sous forme de flux). Pour accélérer le transfert de l'information de cette valeur extérieure vers les volumes finis, nous utilisons des éléments spéciaux. Ceux-ci sont ajoutés à la structure de données et "collés" sur le maillage

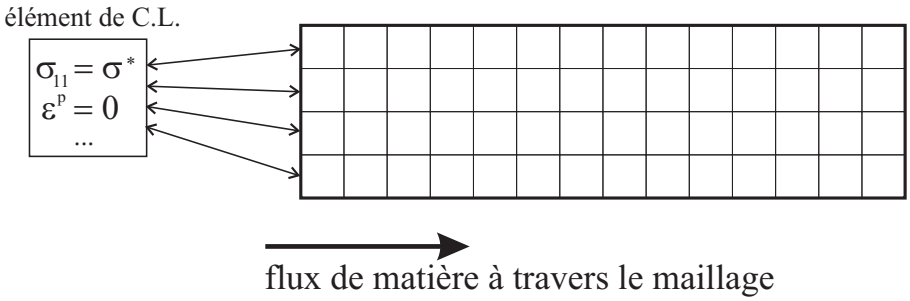

FIG. 2.23: Gestion des conditions aux limites par des éléments spéciaux.

aux arêtes correspondante (figure 2.23). Pour l'instant nous envisageons uniquement des conditions aux limites constantes sur chaque entité géométrique. Un seul élément spécial est donc assigné comme voisin à ces arêtes. Les valeurs (pression, contraintes déviatoriques, ...) sont les valeurs prescrite par l'utilisateur, ce qui permet de traiter les conditions aux limites comme un transfert entre deux volumes finis internes au maillage. L'algorithme devient alors extrêmement clair ; la difficulté étant déplacée dans la préparation des données (qui ne se fait qu'une fois par calcul, contrairement aux transferts).

Un autre avantage de cette gestion des conditions aux limites est le suivant : si une nouvelle valeur doit être transférée, comme par exemple, une ou plusieurs variables internes par exemples (back-stresses dans le cas de matériaux à écrouissage cinématique), il suffit d'ajouter ces données à la structure et aucune autre modification n'est nécessaire pour gérer ce nouveau genre de matériau.

2.8 Tâches annexes

Après transfert des données du maillage équilibré vers le nouveau, une série de calculs et ajustements doivent être effectués pour obtenir une bonne continuation du calcul.

- Certaines grandeurs doivent être recalculées. Il s'agit principalement des rayons en 2D axisymétrique, des jacobiens des nouveaux éléments et de la matrice des masses si le calcul est dynamique.
- Le transfert des grandeurs aux points de Gauss peut mener à des valeurs locales incompatibles après transfert : la déformation plastique équivalente peut être négative et les contraintes peuvent ne plus être sur la surface de charge alors qu'elles l'étaient avant le transfert. Il faut donc ajuster ces valeurs. (toute valeur négative de la déformation plastique équivalente est négligée et le déviateur des contraintes est replacé sur la surface de charge.
- Les forces internes et externes sont enfin recalculées en fonction des valeurs transférées pour déterminer le résidu d'équilibre.

2.9 Pré-processing – Introduction des données

Une très grande attention a été apportée à la manière d'introduire les données pour la librairie ALE. Vu la manière dont cela était réalisé pour l'ancienne version des routines, il n'était pas difficile de faire mieux.

Nos buts étaient assez simples :

- Il fallait que l'utilisateur n'ait aucun effort pour transformer un jeu de données traditionnel lagrangien en un jeu de données à résoudre par le formalisme ALE. En effet, l'ancienne version utilisait des commandes spécifiques très mal conçues. Ces dernières étaient utilisables également en lagrangien mais un autre jeu de commandes beaucoup plus conviviales existait pour créer un problème lagrangien. Il en résulte que l'utilisateur de METAFOR, habitué au formalisme lagrangien devait fournir un effort important pour passer d'un formalisme à l'autre (ce qu'il ne faisait pas).
- Introduire toutes les données dans un seul fichier (il en fallait 3 pour l'ancienne version : un pour le maillage et les données, un pour le remaillage et un pour la convection).
- Simplifier toutes les commandes en les faisant agir sur des entités géométriques et non sur des noeuds et des mailles. Cela permettra de faire du remaillage complet avec changement de topologie dans le futur.

Ces trois buts ont été atteints. L'utilisateur peut décider de transformer son jeu de données lagrangien en un jeu de données ALE en spécifiant la fréquence de remaillage (une fréquence de 0 indiquent que le problème est complètement lagrangien).

De nouvelles commandes spécifiques à l'ALE et dans le style des commandes METAFOR

ont été ajoutées. Elles sont tout simplement ignorées lorsque la fréquence de remaillage est nulle.

L'utilisateur peut utiliser une commande ".res" pour spécifier le type de repositionnement des noeuds associés aux entités géométriques, la commande ".acl" pour créer un groupe de conditions aux limites (un élément spécial de CL), la commande ".scl" pour attribuer un élément de conditions aux limite à une entité géométrique et ".mfg" pour déplacer manuellement le maillage pendant le calcul.

2.10 Conclusions

Nous avons présenté, de manière très détaillée le contenu de notre nouvelle librairie ALE nous en résumons ici les possibilités :

- Traitement de problèmes 2D, 2D axisymétriques et 3D, multi-domaines, multi-matériaux.
- Formulation indépendante du caractère structuré du maillage.
- 3 type de repositionnement pour les points, 6 pour les lignes, 5 pour les domaines surfaciques, 4 pour les faces et 3 pour les domaines volumiques. Gestion de contours de plus de 4 courbes en 2D.
- Repositionnement manuel du maillage par l'utilisateur si nécessaire.
- Gestion automatique des conditions aux limites lorsque la matière entre dans le maillage.
- Algorithmes de détection de la position des noeuds sur les entités géométrique.
- Algorithme de convection par volumes finis indépendant du type de maillage et d'élément fini mais limité cependant à 4 points de Gauss en 2D et 8 en 3D.

Nous insistons une nouvelle fois sur le fait que la librairie a été développée en dehors du code METAFOR ; ce qui nous fait penser qu'elle est fortement indépendante de celui-ci. Elle pourrait donc être utilisée dans d'autres codes éléments finis similaires sans grandes modifications. La structure de données du maillage est la seule l'hypothèse qui garantit le fonctionnement du nouveau code.

A propos des futurs développements, il sera intéressant de pouvoir gérer les frontières eulériennes en 3D.

Nous montrerons aussi dans le chapitre 3 que l'implémentation d'une autre méthode de repositionnement des noeuds dans les domaines 2D et 3D sera très intéressante. En effet, la méthode actuellement disponible peut provoquer des retournements de mailles lorsque la surface du maillage est fortement concave.

Pour la convection des grandeurs d'un maillage à l'autre, il faudra transférer également les grandeurs restantes comme la température mais aussi (et nous le verrons lors des applications dynamiques), le vecteur vitesse. En effet, jusqu'à présent nous nous soucions

uniquement des grandeurs stockées aux points de Gauss et nous avons laissé tombé les grandeurs nodales.

A part cela, une méthode de convection plus précise (second ordre) serait intéressante à développer.

Chapitre 3

Applications numériques

3.1 Introduction

Dans ce chapitre, nous montrons une série de résultats obtenus avec notre nouvelle implémentation du formalisme ALE.

La majorité des problèmes 2D présentés ne sont pas nouveaux. Cependant, nous avons pour certains mis au point un équivalent 3D (barre de Taylor, poinçonnement,...).

Nous nous penchons tout d'abord sur les algorithmes de convection 2D et 3D. La méthode "volumes finis" ayant été complètement reprogrammée suivant une philosophie différente, nous montrons que nous obtenons des résultats similaires à notre première implémentation. Nous montrons aussi les performances de notre algorithmes sur des maillages non structurés 2D et également en 3D.

Ensuite, nous présentons successivement différents problèmes 2D ou 3D en étudiant brièvement la qualité de la solution obtenue et en faisant, lorsque c'est possible, une comparaison en 2D, 3D et solution lagrangienne.

Au cours de cette année, nous avons également traduit l'ensemble des problèmes présentés précédemment dans le rapport de première année (extrusion, laminage) ainsi qu'une simulation . Nous ne présentons pas ici les résultats parce qu'ils sont totalement similaires. Les seules différences sont visibles dans la conception du jeu de données et l'utilisation de METAFOR.

Enfin, en fin de chapitre, nous présentons quelques conclusions et perspectives.

3.2 Test de convection 2D

3.2.1 Introduction

Dans la gestion du formalisme ALE, la gestion du repositionnement des noeuds est certainement une phase très importante. Cependant, il est aussi capital de posséder un algorithme permettant le transfert des grandeurs stockées sur l'ancien maillage vers le nouveau. Nous utilisons un algorithme de volumes finis intégré temporellement par un schéma explicite d'Euler du premier ordre. Cette section montre que notre nouvelle implémentation donne les résultats attendus.

Pour plus de détails sur la stabilité et les éventuelles oscillations de la solution, nous renvoyons le lecteur au rapport de première année de bourse [5].

3.2.2 Test symétrique 2D sur maillage non structuré ($\alpha = 1$)

Nous effectuons un test de convection 2D sur un maillage non structuré. En effet, lors de notre première implémentation, nous ne pouvions traiter que des cas structurés et nous avions fait, en ce temps, plusieurs tests de validation.

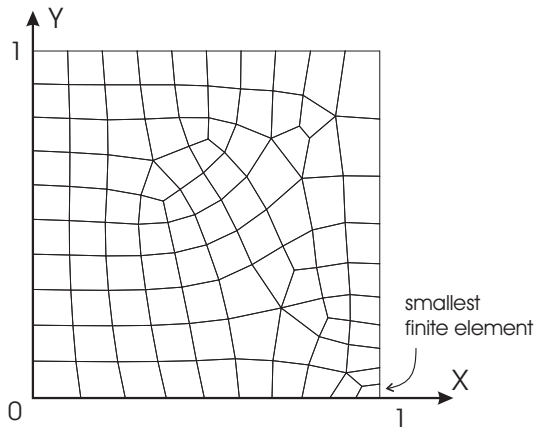

FIG. 3.1: Maillage non structuré utilisé pour le test de convection 2D.

Il s'agit d'un domaine carré de longueur unitaire (figure 3.1). Chaque côté est maillé de manière non uniforme en 10 éléments. Le maillage transfini construit sur ces côtés comporterait donc 100 mailles. Vu que nous utilisons un mailleur frontal, nous n'obtenons pas ce chiffre (le maillage obtenu comporte 98 mailles quadrangulaires).

Le premier test se déroule de la manière suivante : on considère que le mouvement du maillage est une translation à vitesse constante ($\vec{v} = (1, 1)m/s$). On choisi le pas de temps égal à 0.01 s. Le maillage est repositionné de manière eulérienne après chaque pas de temps

et des conditions aux limites devront donc être appliquées sur les côtés $x = 0$ et $y = 0$. Pour ce test, ces deux conditions aux limites sont identiques et valent 1. On observe alors la convection d'une grandeur stockée sur 4 points de Gauss par élément fini. On choisit un coefficient d'upwind égal à 1.

La figure 3.2 montre les résultats obtenus. Les deux premières images montrent respectivement le maillage élément fini de 98 mailles et son maillage auxiliaire associé aux points de Gauss de $4x98$ mailles.

On constate que le schéma de convection ne provoque pas d'oscillations mais, au contraire, assez bien de diffusion. Ceci avait déjà été constaté sur des maillages structurés lors de l'étude de stabilité du schéma de convection.

D'une manière très résumée, rappelons les résultats obtenus précédemment :

- La stabilité du schéma utilisé est conditionnelle. Cette condition s'exprime simplement par :
$$0 \leq C \leq \alpha \tag{3.1}$$
où C est le rapport entre la distance de convection locale et la taille locale du maillage (nombre CFL) et α est le coefficient d'upwind ($0 \leq \alpha \leq 1$).
- Si α est proche de C, tout en étant supérieur, le schéma est stable mais la solution montre des oscillations. Si α est grand (proche de 1), la solution est stable mais on observe beaucoup de diffusion.

Les résultats de la figure 3.2 sont en accord avec les résultats trouvés précédemment puisqu'ici la taille de la plus petite maille est 0.04 m. Le plus petit volume fini a donc un côté de 0.02 m. On a donc $C_{min} = 0.01/0.02 = 0.5$ et la condition de stabilité est largement vérifiée.

3.2.3 Test symétrique 2D sur maillage non structuré ($\alpha = 0.2$)

Nous effectuons maintenant un deuxième test sur le même maillage non structuré avec un coefficient d'upwind égal à 0.2, ce qui est en dessous de la limite de stabilité locale calculée précédemment.

La figure 3.3 montre les résultats obtenus dans ce cas. On constate la présence de fortes oscillations mais la solution ne diverge pas. On constate également que le temps de stabilisation vers la solution stationnaire (valeur 1 uniforme) est très lente (après 120 pas de temps, c'est-à-dire 1.2 s, la solutions oscille toujours).

La solution obtenue est donc oscillante mais stable malgré que la valeur de C minimum soit supérieure à la valeur du coefficient d'upwind α. Cela peut s'expliquer par le fait que l'élément le plus petit est situé sur la frontière par laquelle la matière sort du maillage. La valeur de cet élément n'influence donc pas d'autres éléments (les éléments en amont ne reçoivent pas de flux vu la valeur unitaire du coefficient d'upwind et il n'y a pas d'élément

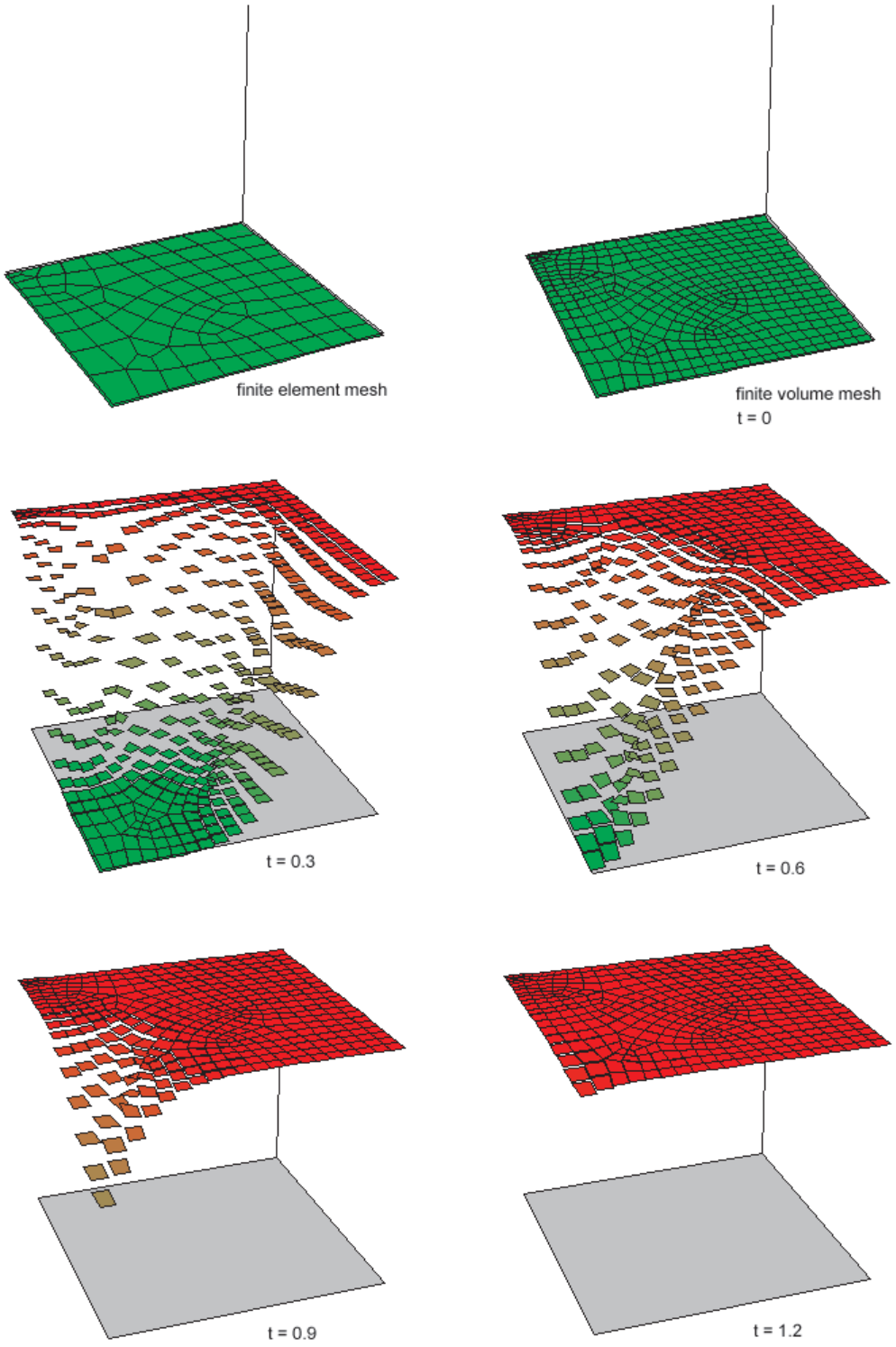

FIG. 3.2: Test symétrique 2D sur maillage non structuré ($\alpha = 1$).

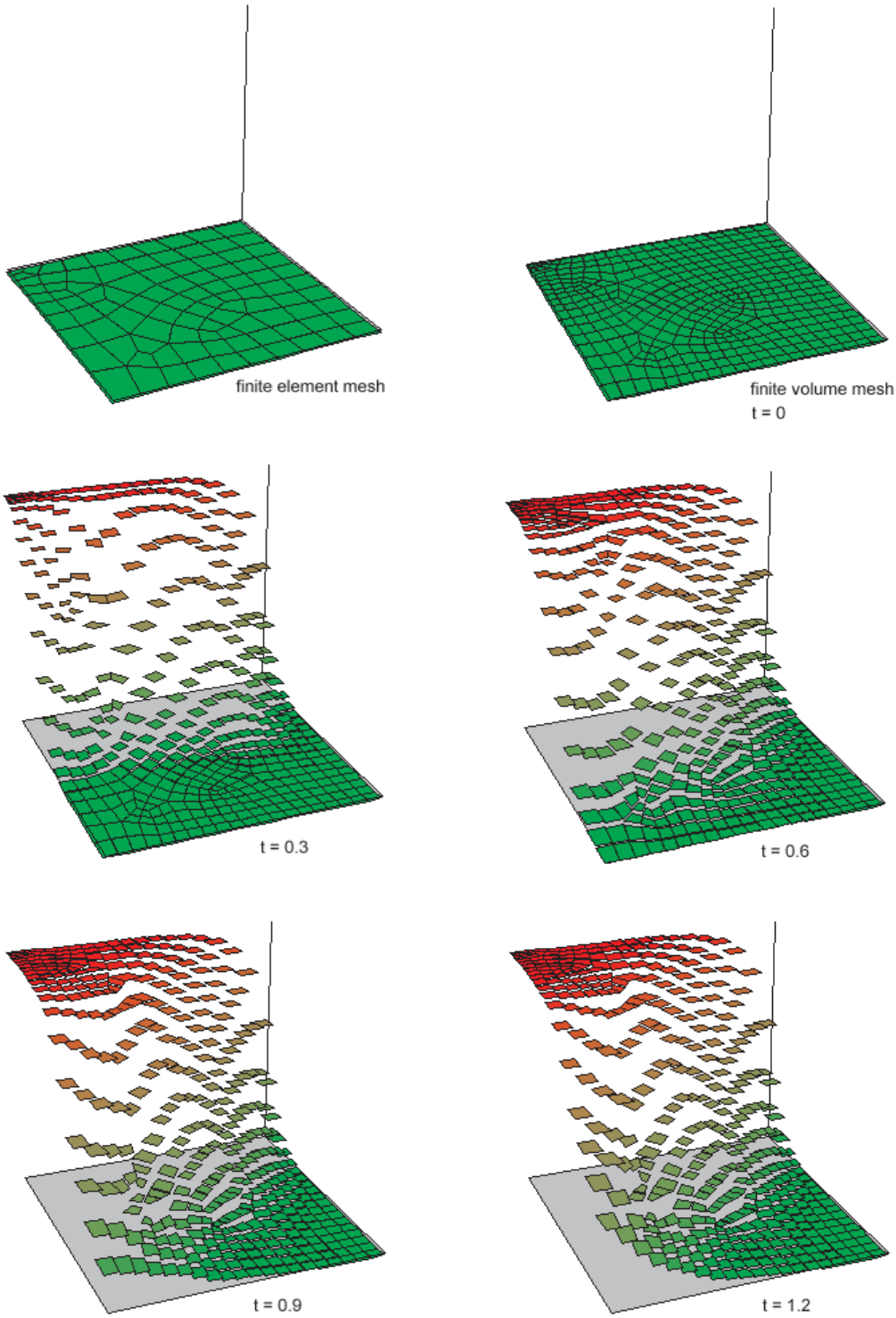

FIG. 3.3: Test symétrique 2D sur maillage non structuré ($\alpha = 0.2$).

en aval). De plus, la majorité des éléments ont une longueur caractéristique proche de 0.1. Si on considère que cette longueur est une moyenne, on a $C_{moy} = 0.01/0.05 = 0.2$, ce qui est égal à notre coefficient α. On est donc très proche de la limite de stabilité et cela explique les phénomènes observés.

3.2.4 Test non symétrique 2D sur maillage non structuré ($\alpha = 1$)

Nous effectuons un dernier test 2D pour tester la propagation d'un échelon à travers le maillage. Pour ce faire, on fixe une des deux conditions au limites à 0 (en $x = 0$). La solution stationnaire à approcher est une solution discontinue où la moitié du domaine ($x < y$) est à la valeur 1 et l'autre moitié ($x > y$) à 0 ; ces deux moitiés étant séparées par la diagonale $x = y$. On choisit d'utiliser $\alpha = 1$.

La figure 3.4 montre l'évolution de la solution au cours du temps.

Rappelons qu'un même test sur un maillage régulier (provenant d'un mailleur transfini, les côtés étant maillés régulièrement) dans les mêmes conditions (même CFL) montre beaucoup de diffusion transverse. C'était d'ailleurs pour cette raison que nous avions mis au point un algorithme SUPG qui permettait de supprimer cette diffusion transverse. Remarquons cependant que le même test effectué avec une vitesse parallèle à une direction du maillage ne montre aucune diffusion.

La diffusion transverse est donc fonction de la manière avec laquelle sont ordonnés les éléments par rapport à la direction de convection. La diffusion transverse est simplement une conséquence directe du passage du flux d'un élément à son voisin via une arête qui n'est pas parallèle à la vitesse de convection.

Dans le cas traité ici, on constate beaucoup de diffusion transverse comme on pouvait s'y attendre. A notre connaissance, il n'existe pas de solution simple pour éliminer cette diffusion transverse tout en conservant l'algorithme utilisé ici.

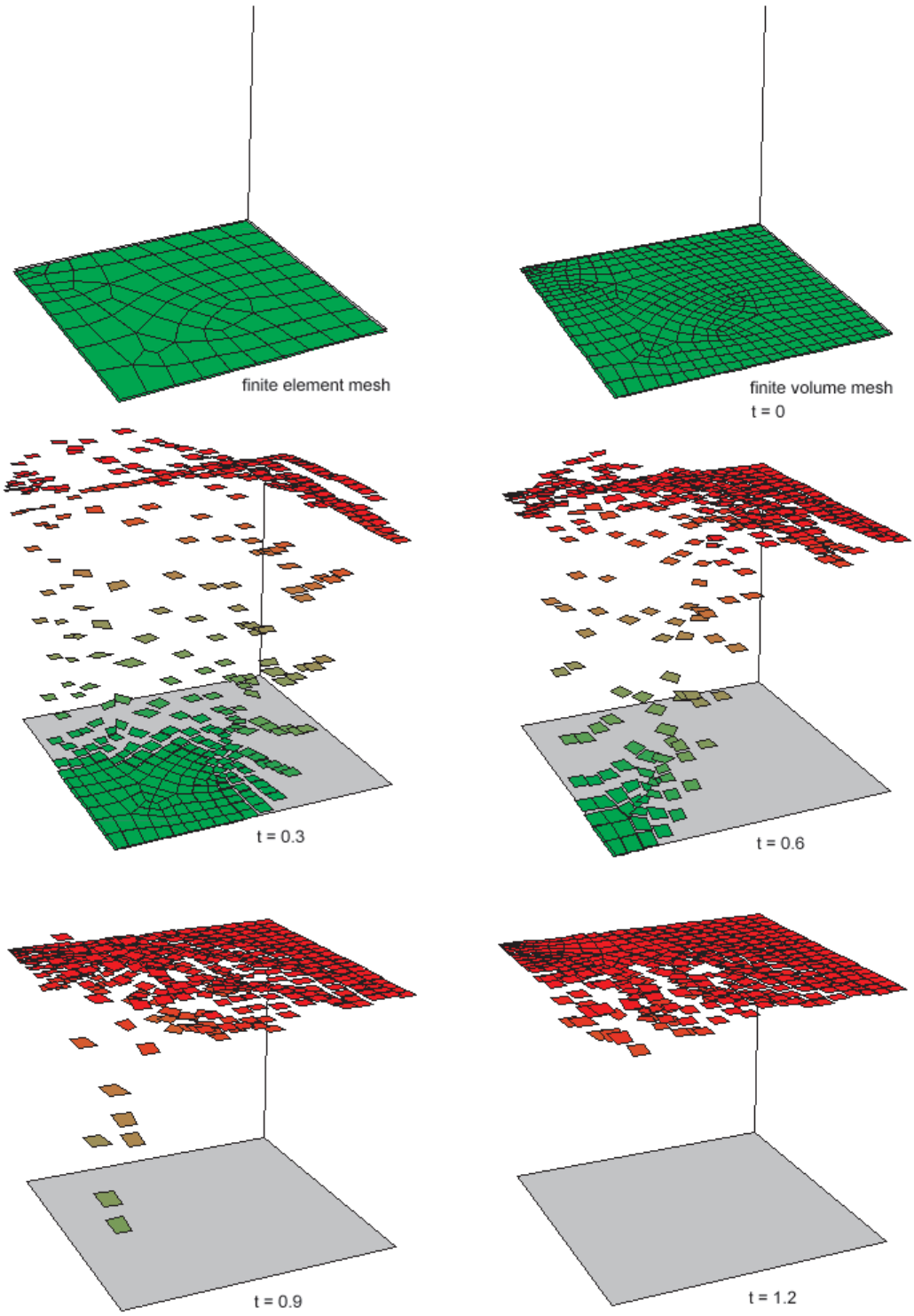

FIG. 3.4: Test non symétrique 2D sur maillage non structuré ($\alpha = 1$).

3.3 Test de convection 3D

3.3.1 Introduction

Nous avons étendu l'algorithme de volumes finis 2D au cas 3D et nous montrons, dans cette section, que celui-ci donne des résultats similaires à son homologue 2D.

On pourrait montrer, mais nous ne le ferons pas, que notre algorithme 3D possède les mêmes propriétés au point de vue de la stabilité de la solution. Ainsi, il est intuitivement facile à comprendre que, vu le caractère explicite du schéma d'intégration, le déplacement relatif maximal admissible pour garantir la stabilité est égal à la moitié de la taille d'un élément finis (c'est-à-dire la taille d'un volume finis, chaque élément finis étant divisé en 2 volumes finis selon chaque direction).

3.3.2 Test symétrique 3D sur maillage structuré ($\alpha = 1$)

Nous choisissons d'utiliser un domaine volumique cubique de longueur unitaire. Le domaine est maillé de manière régulière. On utilise ici seulement 3 éléments par côté, ce qui nous donne 27 éléments cubiques et un maillage auxiliaire de $4x27 = 108$ volumes finis.

On impose une vitesse $\vec{v} = (1, 1, 1)$ m/s à la matière et le maillage est repositionné, après chaque pas de temps de 0.01 s, de manière eulérienne. Les conditions aux limites sont similaires au cas 2D : on impose une valeur de 1 sur les faces $x = 0$, $y = 0$ et $z = 0$ et on observe la propagation de cette valeur à travers le maillage auxiliaire au cours du temps.

La figure 3.5 montre le déroulement du test. On constate que les conditions aux limites se propagent correctement dans la structure (couleur verte = 0, couleur rouge = 1). Nous observons également une diffusion assez importante dans le sens de l'écoulement vu la petitesse du nombre C vis à vis de α ($C = 0.01/(1/6) = 0.06$). Cette diffusion pourrait être réduite en diminuant le coefficient d'upwind α.

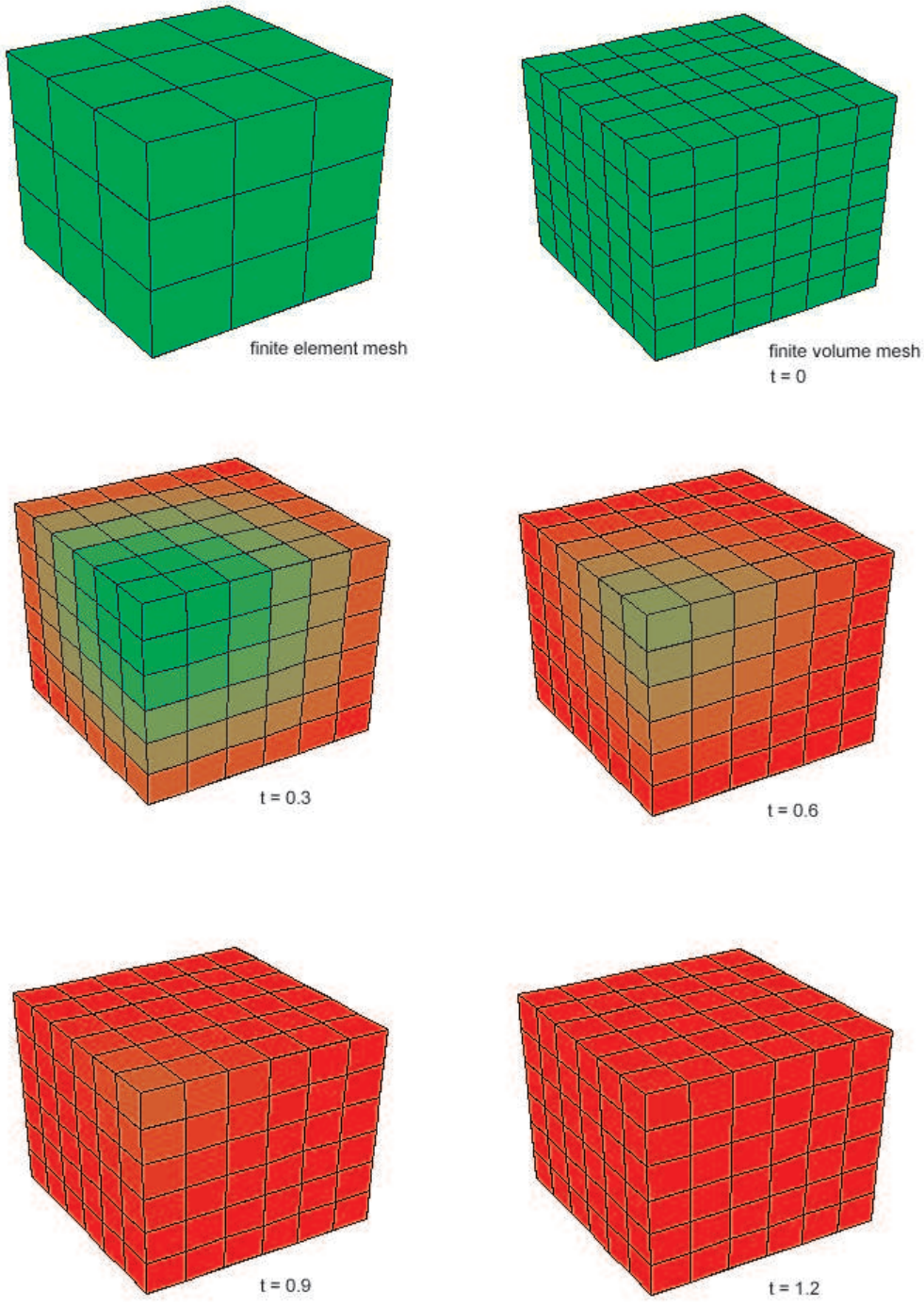

FIG. 3.5: Test symétrique 3D sur maillage structuré ($\alpha = 1$).

3.4 La barre de Taylor 2D/3D

3.4.1 Introduction

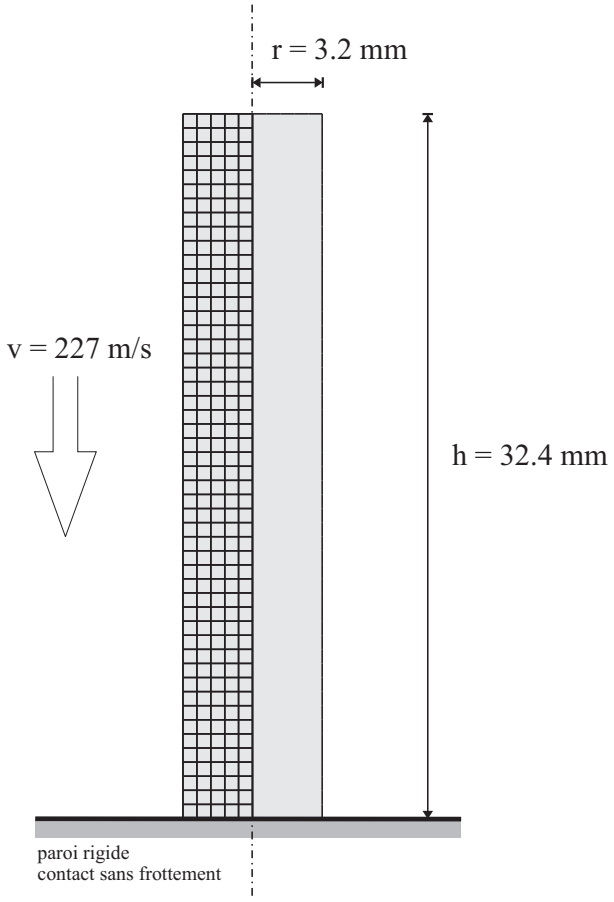

FIG. 3.6: Impact de la barre de Taylor (géométrie et maillage axisymétrique).

Le problème de la barre de Taylor est le problème le plus couramment utilisé pour valider des développements dans le domaine du formalisme ALE [30, 27, 20, 5, 28, 1]. Il s'agit de l'impact à haute vitesse d'une petite barre cylindrique sur un mur rigide. Le problème est axisymétrique et peut être étudié en 2D. Mais ce problème constitue aussi un intéressant problème de validation pour le 3D. En effet, nous allons voir que simuler l'impact de la barre de Taylor en 3D ne nécessite pas de développements compliqués. En effet, les mailles de la surface externe pourrait être traitées par le formalisme lagrangien puisqu'elles ne subissent pas de grandes distorsions. De plus, le contact avec le plan peut être simulé par des déplacements imposés nuls au niveau du maillage de la base selon la direction d'impact. Enfin, le déplacement relatif entre le maillage et la matière est faible, donc il y a très peu de convection et la plupart des algorithmes, même ceux basés sur des extrapolations et interpolations, fonctionnent assez bien. Remarquons également que ce cas

test montre aussi très nettement l'intérêt de l'ALE pour la formulation explicite. En effet, la taille des maille étant plus grande, on peut se permettre un plus grand pas de temps.

Nous allons cependant voir qu'obtenir une concordance parfaite entre la solution ALE et lagrangienne n'est pas évident. Nous expliquerons les causes lors de la présentation des résultats obtenus.

Module de Young	E	=	117	GPa
Coefficient de Poisson	ν	=	0.35	
Limite d'élasticité	σ_Y^0	=	400	MPa
Coefficient d'écrouissage	h	=	100	MPa
Masse volumique	ρ	=	8930	kg/m^3

TAB. 3.1: Propriétés matérielles pour l'impact de la barre de Taylor

Les propriétés matérielles sont reprises dans le tableau 3.4.1. Il s'agit d'une barre en cuivre. La loi élastoplastique est une loi à écrouissage isotrope. On considère qu'il n'y a pas de frottement entre la barre et la paroi rigide.

La barre possède un rayon initial de 3.3 mm et une longueur de 32.4 mm. La vitesse d'impact est de 227 m/s. La période d'observation est de 80 μs. En ce moment, la quasi totalité de l'énergie cinématique initiale est dissipée sous forme de travail plastique.

Nous utilisons un schéma d'intégration temporel explicite.

En ce qui concerne les maillages utilisés, nous avons fait en sorte que les deux cas (2D axisymétrique et 3D) soient comparables. Nous avons donc choisi dans les deux cas le même nombre d'éléments sur la hauteur et le rayon. On utilise 5 éléments sur le rayon et 50 sur la hauteur. Les éléments utilisés sont les traditionnels éléments Q4P0 en 2D (éléments à pression constante pour éviter le "locking") et leur équivalent (hexaèdres à 8 points de Gauss et 1 seul pour la pression) pour le cas 3D.

Dans le cas 3D, vu la symétrie du problème, un quart du problème est modélisé. La section circulaire est décomposée en 3 domaines (figure 3.8) pour permettre un maillage de type transfini dans ce plan (seul type de maillage de notre logiciel de pré-traitement – BACON – qui garantisse l'obtention de quadrangles en 2D). Vu que nous voulons dans les deux cas le même nombre d'éléments dans la section radiale, le nombre d'éléments sur le quart de la circonférence est automatiquement fixé et est fonction des domaines surfaciques utilisés.

La figure 3.8 montre la décomposition de la base en 3 domaines de topologies quadrangulaires et le maillage surfacique obtenu. On obtient ainsi une assez mauvaise représentation de la surface externe du cylindre mais nous verrons que les résultats obtenus sont tout de même très satisfaisants.

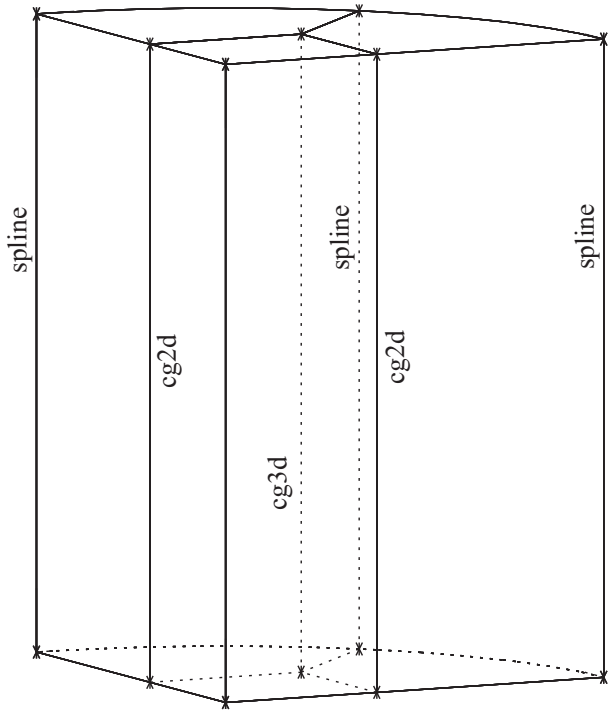

FIG. 3.7: Barre de Taylor – Géométrie de la simulation 3D.

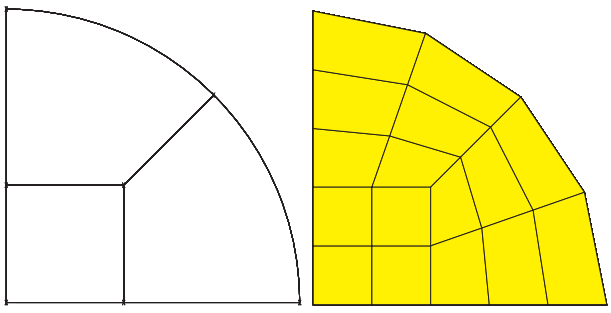

FIG. 3.8: Maillage dans le plan de la section circulaire (cas 3D).

Nous avons représenté sur la figure 3.7 les 3 domaines volumiques utilisés pour la simulation 3D.

En ce qui concerne les types de repositionnement des noeuds utilisés pour les noeuds associés aux points, aux lignes, aux faces et aux domaines, les simulations 2D et 3D sont très légèrement différentes.

Pour le cas 2D, tous les points de la géométrie (les 4 points définissant les extrémités du domaine) sont lagrangiens. Les lignes sont toutes remaillées par la méthode "spline" et nous utilisons un remaillage de domaine de type "méthode d'interpolation transfinie".

Pour le 3D, la situation est légèrement plus compliquée vu que nous sommes en présence de 3 domaines distincts.

Tous les points sont considérés lagrangiens comme en 2D. Cela mène à une différence assez importante. En effet, les 4 points qui sont au milieu d'une arête de la structure ne devraient pas être considérés comme lagrangiens puisque le noeud correspondant en 2D est repositionné lors du remaillage "spline" de la ligne sur laquelle il se trouve. Le problème, en 3D, vient du fait que le noeud associé en 3D n'est pas sur une ligne mais bien à l'intersection de plusieurs lignes. Il faudrait donc, comme nous l'avons déjà dit dans la partie théorique de ce rapport, mettre au point une méthode de remaillage des points sur les arêtes 3D. Le noeud associé serait, par exemple, repositionné au centre de gravité des deux noeuds voisins sur l'arête concernée.

Toutes les lignes sont remaillées par la méthode "spline" sauf 3 d'entre-elles (voir figure 3.7). La ligne interne (intersection des 3 domaines) est remaillée suivant la méthode du centre de gravité 3D (chaque noeud est repositionné au centre de gravité des noeuds des éléments volumiques voisins). Les deux lignes auxiliaires sur les deux plans de symétrie sont remaillées par la méthode "centre de gravité 2D" qui est une variante de la méthode 3D pour laquelle les noeuds voisins sont ceux appartenant au plan de symétrie. Cette méthode suppose que la face reste plane, ce qui est le cas ici vu les conditions de symétrie.

Pour les faces des domaines, nous utilisons plusieurs méthodes. Pour les faces internes, on utilise la méthode du centre de gravité volumique, les faces restant planes sont remaillées par la méthode du centre de gravité 2D et les 2 faces composant la surface externe du cylindre sont remaillées par notre algorithme utilisant des surfaces splines.

Nous repositionnons les noeuds internes de tous les domaines avec la méthode du centre de gravité volumique.

Remarquons qu'il serait possible de programmer des méthodes de repositionnement globales (c'est-à-dire qui s'appliquent à tous les noeuds, sans se soucier de leur appartenance à un domaine, à une face, à une ligne ou un point. Cela permettrait de simplifier l'introduction des données de remaillage. Cependant, si le remaillage volumique serait simplifié, cela entraînerait des difficultés sur les faces et les lignes. En effet, le fait d'identifier pour chaque noeud l'entité géométrique correspondante permet, par exemple, de définir 2 normales ex-

ternes en chaque noeud d'une arête du problème et donc de modéliser correctement les angles de la structure.

Pour l'algorithme de convection, nous utilisons un coefficient d'upwind α unitaire. Vu le grand nombre de pas de temps attendu (algorithme explicite), nous ne faisons appel aux routines ALE que tous les 20 pas de temps. Cela permet d'accélérer le calcul de manière significative. Nous comparerons les résultats obtenus de cette manière avec ceux obtenus en repositionnant les noeuds tous les pas de temps.

Enfin, nous utilisons 3 passes pour le remaillage de type "centre de gravité".

3.4.2 Résultats

Nous comparons dans ce rapport les résultats obtenus en 2D et en 3D ainsi que les résultats lagrangiens face à ceux obtenus par le formalisme ALE.

La figure 3.9 montre les maillages obtenus et le champ de déformations plastiques équivalentes pour les simulations 2D, 3D, lagrangiennes et ALE.

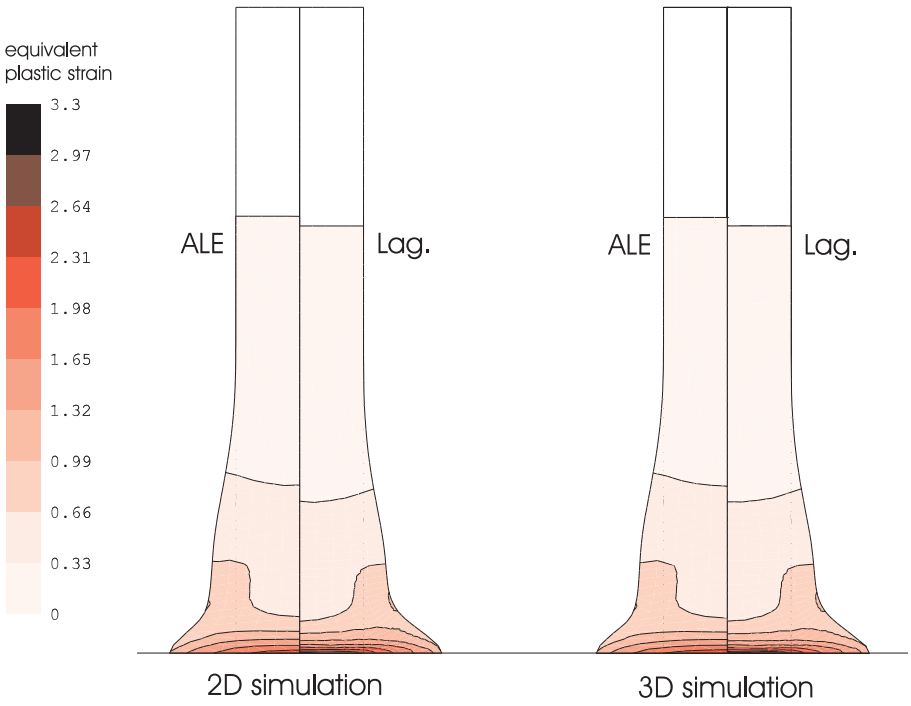

FIG. 3.9: Déformations plastiques équivalentes (Lagrangien / ALE et 2D / 3D).

Si on compare les résultats obtenus par les deux formalismes, on constate que l'on retrouve les même différences que Ponthot [27] ou J.L.F. Aymone [1] et dans notre rapport de première année [5]. Autrement dit, le rayon maximum à la fin de la déformation est

plus grand dans le cas lagrangien. De même, la hauteur finale de la barre est plus petite. Ce qui est important, c'est que ces différences sont identiques dans le cas 2D et dans le cas 3D. Il est en effet, très difficile à l'oeil nu de distinguer les solutions 2D et 3D pour un formalisme (lagrangien ou ALE) donné. Cela nous montre que notre implémentation 3D est correcte et aussi fiable que notre implémentation 2D. De plus, nous voyons que la mauvaise approximation de la géométrie cylindrique que l'on a faite en 3D n'influence pas la solution.

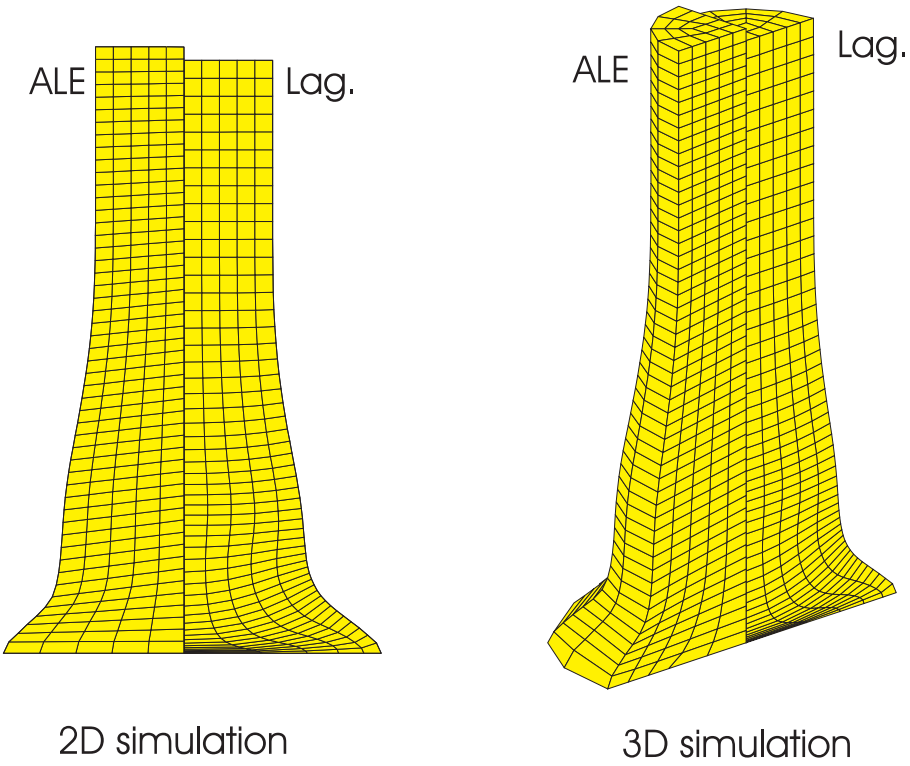

FIG. 3.10: Maillages obtenus en fin de calcul (Lagrangien / ALE et 2D / 3D).

La figure 3.10 montre les maillages obtenus dans les 4 cas (2D/3D et Lagrangien/ALE). On constate que le maillage ALE 3D présente des mailles plus écrasées que le cas ALE 2D, principalement au niveau de la base. Cela vient du type de remaillage choisi pour le noeud associé au point intermédiaire sur la base. Le type lagrangien n'est pas optimal mais nous ne disposons actuellement de rien d'autre qui convienne mieux. On constate quand même que les mailles sont nettement mieux conditionnées dans le cas du formalisme ALE que dans le cas lagrangien pur.

La figure 3.11 montre une superposition des contours obtenus dans les 4 cas considérés. On voit que la solution 3D est très semblable à la solution 2D et que les différences entre solution lagrangienne et ALE sont identiques dans les deux cas. De plus, on retrouve visuellement les mêmes différences que Ponthot [27].

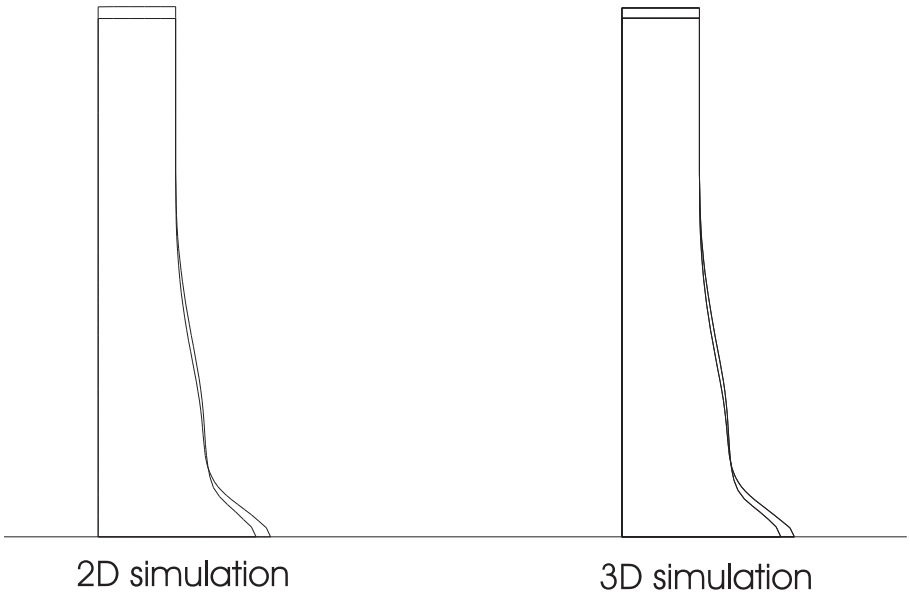

FIG. 3.11: Contours obtenus en fin de calcul (Lagrangien / ALE et 2D / 3D).

Si on compare nos résultats 3D obtenus par le formalisme ALE avec d'autres publications, on constate que les résultats obtenus sont assez bons :

S. Potapov [28] a implémenté dans sa thèse (Ecole Centrale de Paris) une méthode de remaillage 3D ainsi que le même algorithme de convection que celui utilisé ici. Ses résultats obtenus pour la barre de Taylor souffrent de l'absence de gestion des noeuds sur surfaces courbes. En effet, les mailles sur la surface extérieure sont assez bien écrasées puisque les noeuds de cette surface sont considérés lagrangiens. Nous voyons donc ici l'importance de l'utilisation d'une méthode de repositionnement de noeuds sur surface courbe. Il est en effet très limitatif de négliger cet algorithme. Néanmoins, nous allons voir que son transfert de données est plus efficace que le nôtre lors de la comparaison numérique des résultats.

J.L Farinatti Aymone [1] (Universidade Federal Do Rio Grande Do Sul) obtient des résultats très similaires aux nôtres puisqu'il possède une méthode de repositionnement des noeuds sur une surface courbe. cependant, celle-ci est différente de la nôtre (repositionnement sur patches de Coons linéaires).

Nous terminons cette brève étude de la barre de Taylor en comparant de manière plus précise les résultats obtenus dans les différents cas avec ceux publiés dans la littérature.

Le tableau 3.4.2 compare nos résultats obtenus dans le cas 2D axisymétrique avec ceux de Ponthot [27] et ceux obtenus après la première bourse [5].

Si on compare nos résultats entre eux, on constate que le calcul lagrangien est beaucoup plus lent que le calcul ALE, surtout si on effectue peu de remaillages. C'est normal puisque le pas de temps utilisé est calculé dynamiquement en fonction de la taille minimale des

	pas	H_f [mm]	R_f [mm]	ε^p_{max} (extrapolé)	CPU [s]	CPU ALE [s]
Lagrangien	11387	21.43	7.13	3.117	72	0
ALE (nrem=20)	1749	21.92	6.53	2.088	12	2.5
ALE (nrem=1)	1758	21.92	6.53	2.191	56	45.6
Ponthot (Lag)	9496	21.43	7.13	3.04	/	0
Ponthot (ALE)	1463	21.87	6.51	2.21	/	/
1ere Bourse (ALE)	1467	21.93	6.54	2.29	/	/

TAB. 3.2: Comparaison des résultats 2D

	pas	H_f [mm]	R_f [mm]	ε^p_{max} (extrapolé)	CPU [min]	CPU ALE [min]
Lagrangien	10923	21.44	7.13	3.293	22 :58	0
ALE (nrem=20)	2422	21.87	6.57	2.248	6 :26	1 :09
ALE (nrem=1)	2378	21.94	6.53	2.182	26 :04	22 :33
J.L.F. Aymone (Lag)	/	21.44	7.10	3.266	/	/
J.L.F. Aymone (ALE)	/	22.065	6.38	2.198	/	/
S.Potapov (Lag)	/	21.45	7.09	/	/	/
S.Potapov (ALE)	/	21.43	7.11	/	/	/

TAB. 3.3: Comparaison des résultats 3D

éléments du maillage. Vu que cette taille est plus grande dans le cas du formalisme ALE, le nombre de pas de temps est plus petit. On voit également qu'effectuer un remaillage à tous les pas de temps (nrem=1) est pénalisant au niveau du temps de calcul et n'apporte rien au niveau des résultats.

Si on compare nos résultats avec les anciens résultats de METAFOR (Ponthot – convection par extrapolation et développement de Taylor) on constate que les résultats sont très similaires.

On constate aussi un bon accord entre nos anciens résultats et nos nouveaux résultats. il existe cependant toujours un différence assez visible entre les résultats obtenus avec les deux formalismes (différences visibles à l'oeil nu).

Le tableau 3.4.2 compare nos résultats 3D avec des résultats obtenus par J.L Farinatti Aymone [1] et S. Potapov [28].

En comparant nos résultats entre eux, on peut tirer les mêmes conclusions que pour les simulations 2D axisymétriques. On remarque en effet une très grande similitude entre les résultats obtenus en 2D et en 3D.

Les résultats de J.L Farinatti Aymone montrent les mêmes différences entre les simulations ALE et lagrangiennes que les nôtres. Par contre, les résultats de S.Potapov sont beaucoup plus intéressants. En effet, les différences entre les deux formalismes sont très réduites et sont également très proches de nos résultats lagrangiens. On pourrait donc conclure que ses résultats sont meilleurs que les nôtres.

La seule différence entre son implémentation du formalisme ALE avec la nôtre est localisée dans la phase de convection des grandeurs. En effet nous (ainsi que Ponthot, J.L Farinatti Aymone et de nombreux auteurs) ne transférons pas le champ des vitesses d'un maillage à l'autre ; ce qui est une simplification assez forte dans ce problème-ci. Dans notre cas, nous ne transférons pas cette grandeur non seulement parce que celle-ci n'est pas stockée aux points de Gauss (cela nécessite donc une autre méthode de transfert) mais aussi parce que de nombreux auteurs, dont Ponthot, ne le font pas.

Il nous semble assez évident que c'est cette hypothèse qui est la cause des différences observées entre les résultats lagrangiens et ALE. En effet, de manière très intuitive, lors d'un remaillage, la vitesse relative de convection est dirigée dans le même sens que la vitesse d'écrasement de la barre. Les éléments finis situés au pied de la barre reçoivent donc un flux de valeurs (contraintes, pression, déformation plastique,...) inférieures à celles stockées en leurs points de Gauss (car ces éléments sont plus déformés que leurs voisins). Ainsi les valeurs stockées aux points de Gauss de ces éléments diminuent lors d'un remaillage. Par contre, la vitesse est simplement conservée d'un maillage à l'autre. cela veut dire qu'un noeud initialement situé très près du sol (et qui possède donc une vitesse proche de 0) se voit translaté dans la direction opposée de l'impact en conservant sa vitesse relativement faible. Il y a donc certainement une perte d'énergie cinétique. Or, dans ce cas-ci, la déformation

est uniquement due à cet énergie. Il n'est donc pas étonnant que la barre se déforme moins dans le cas ALE.

Il sera donc très intéressant, dans la suite de ce travail, de mettre au point un algorithme de convection du vecteur vitesse ou, d'une manière plus ou moins équivalente, de la quantité de mouvement pour pouvoir traiter plus précisement les problèmes dynamiques tels que celui-ci. Pour ce faire, on devra considérer un autre maillage auxiliaire basé sur des volumes finis centrés sur les noeuds du maillage. On effectuera ainsi une convection par le même algorithme que celui utilisé pour les grandeurs stockées aux points de Gauss.

3.5 Simulation du poinçonnement 2D/3D

3.5.1 Introduction

Le problème du poinçonnement ("coining") [30, 27, 2, 33, 19, 22, 20, 5] est aussi un cas-test intéressant à traiter par le formalisme ALE (voir figure 3.12).

Il s'agit de l'écrasement à 60% d'un lopin de matière par un poinçon. Deux variantes existent : le poinçonnement d'un lopin cylindrique par un poinçon cylindrique qui peut être traité en 2D axisymétrique ou en 3D et l'écrasement d'un lopin parallélépipédique par un poinçon de même géométrie qui peut être traité en 2D état plan de déformation ou en 3D. Pour ces deux variantes, on peut considérer un problème symétrique selon la direction de poinçonnement (écrasement entre deux poinçons) ou non symétrique (le lopin est déposé sur un plan rigide).

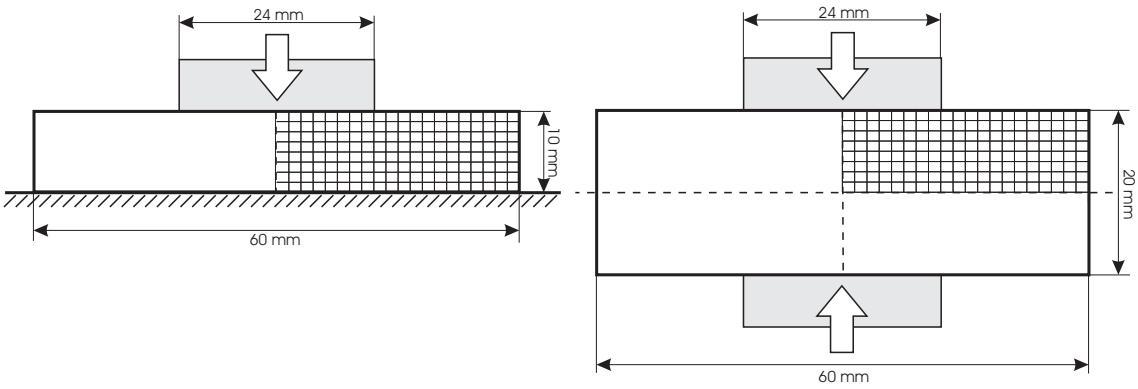

FIG. 3.12: Géométrie du problème de poinçonnement (non symétrique et symétrique).

Il est enfin possible de considérer plusieurs vitesses de poinçonnement pour observer les effets dynamiques [20, 5].

Ce type de problème est difficile à gérer par la formalisme lagrangien. En effet, le maillage doit être extrêmement raffiné sous le poinçon si on veut capter correctement la surface libre du lopin qui s'agrandit au fur et à mesure que le poinçon s'enfonce. De plus, la gestion du contact entre le poinçon et le lopin est assez délicate, surtout au niveau du bord extérieur du poinçon.

Grâce au formalisme ALE, nous n'avons pas à résoudre ces difficultés puisqu'il est possible de gérer la descente du poinçon par des déplacements imposés de noeuds du maillage. En formulation lagrangienne, l'utilisation d'une matrice de contact est inévitable puisque les noeuds sous le poinçon sont liés au mouvement de la matière et ne restent donc pas sous le poinçon au cours du calcul (sauf dans le cas ou le contact serait collant).

Nous montrons, dans cette section les résultats obtenus en 2D et en 3D pour différentes

Module de Young	E	=	200	GPa
Coefficient de Poisson	ν	=	0.3	
Limite d'élasticité	σ_Y^0	=	250	MPa
Coefficient d'écrouissage	h	=	1050	MPa

TAB. 3.4: Propriétés matérielles pour l'écrasement du lopin symétrique.

variantes du problème.

3.5.2 Poinçonnement symétrique

Nous nous occupons ici du cas symétrique. Un lopin cylindrique de rayon 30 mm est poinçonné de part et d'autre par un poinçon cylindrique de 12 mm de rayon. En plus de la symétrie selon un plan de direction normale alignée sur la direction de poinçonnement, le problème est axisymétrique. La géométrie utilisée en 3D est montrée sur la figure 3.13.

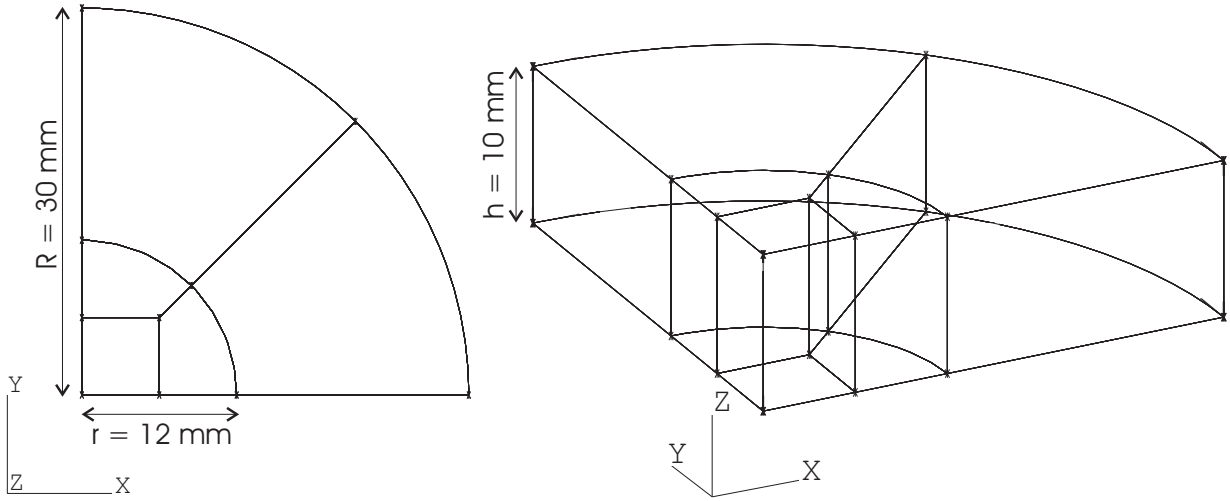

FIG. 3.13: Géométrie du problème de poinçonnement.

Contrairement au problème présenté dans le rapport de première année [5], le lopin et le poinçon sont cylindriques (le problème était traité en état plan de déformation) et il n'y a pas de frottement entre le poinçon et le lopin. Par contre, nous utilisons le même matériau. Nous avons privilégié le calcul axisymétrique parce que son homologue tridimensionnel est plus compliqué à mailler et donc plus intéressant qu'un lopin rectangulaire en état plan de déformation.

Le tableau 3.5.2 montre les propriétés matérielles utilisées. Il s'agit d'un acier classique dont l'écrouissage est supposé linéaire et isotrope. Les effets dynamiques ne seront pas étudiés ici. Nous effectuerons donc nos simulations en quasi-statique.

Nous utilisons, comme précédemment, un coefficient d'upwind α unitaire et 3 passes pour chaque remaillage par la méthode du "centre de gravité". Vu les symétries, nous ne discrétisons que la moitié du problème en 2D et un huitième en 3D.

En ce qui concerne la géométrie et le maillage, nous divisons, pour le cas 2D, le domaine complet en deux sous-domaines. Le premier sous-domaine est situé sous le poinçon. Les mailles de ce domaine ne subiront pas de déplacement selon la direction perpendiculaire du poinçonnement. Le second sous-domaine est le reste de la pièce étudiée.

Nous maillons le lopin à l'aide de 20 éléments selon la direction radiale et 8 éléments selon la direction de poinçonnement et la distance de poinçonnement est de 6 mm (60%).

Pour le repositionnement des noeuds, tous les points sont considérés lagrangiens. Cependant, la composante radiale du noeud associé au point à l'extrémité de la base du poinçon est fixée.

Les lignes sont remaillées par la méthode "spline", à l'exception de la ligne définissant le plan de symétrie ($y = 0$), juste sous le poinçon, qui est déclarée eulérienne. Les noeuds situés sur la ligne séparant les deux sous-domaines sont traités séparément : on fixe leur composante radiale à zéro.

Enfin, les deux sous-domaines sont traités par la méthode d'interpolation transfinie. Ainsi, les mailles du domaine sous le poinçon ne peuvent que s'écraser dans la direction de poinçonnement sans se dilater dans la direction radiale.

Pour le cas 3D, le problème est décomposé en 5 sous-domaines volumiques (3 pour le volume sous le poinçon et deux pour la partie extérieure – figure 3.13).

Tous les domaines sont traités par la méthode du centre de gravité. Les points sont déclarés lagrangiens mais tous ceux appartenant au volume sous le poinçon ne peuvent se déplacer que dans la direction de poinçonnement (composante x et y fixée). Les lignes sont remaillées par la méthode "spline" et les faces par la méthode du centre de gravité avec (face externe courbe) ou sans projection sur une surface spline.

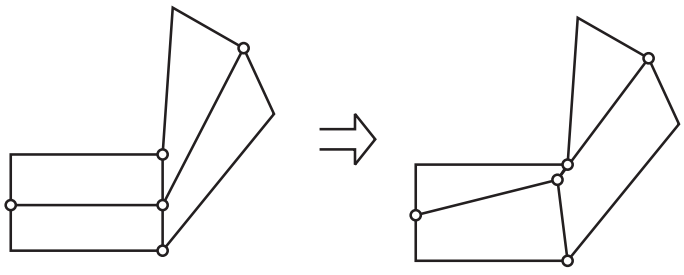

FIG. 3.14: Problème de la méthode du centre de gravité.

Nous avons tout d'abord essayé d'effectuer le calcul sans fixer tous les noeuds situés en dessous de l'extrémité circulaire du poinçon selon x et y. Cela fonctionne, mais la méthode

du centre de gravité appliquée aux trois domaines sous le poinçon ne fournit pas des mailles bien conditionnées (principalement près de l'intersection des trois domaines – voir figure 3.14). Nous avons donc fixé selon x et y tous les noeuds situés sous le poinçon. Cette dernière façon de faire permet d'obtenir la même situation qu'en 2D, c'est-à-dire des noeuds sous le poinçon qui ne peuvent se déplacer que dans la direction de poinçonnement.

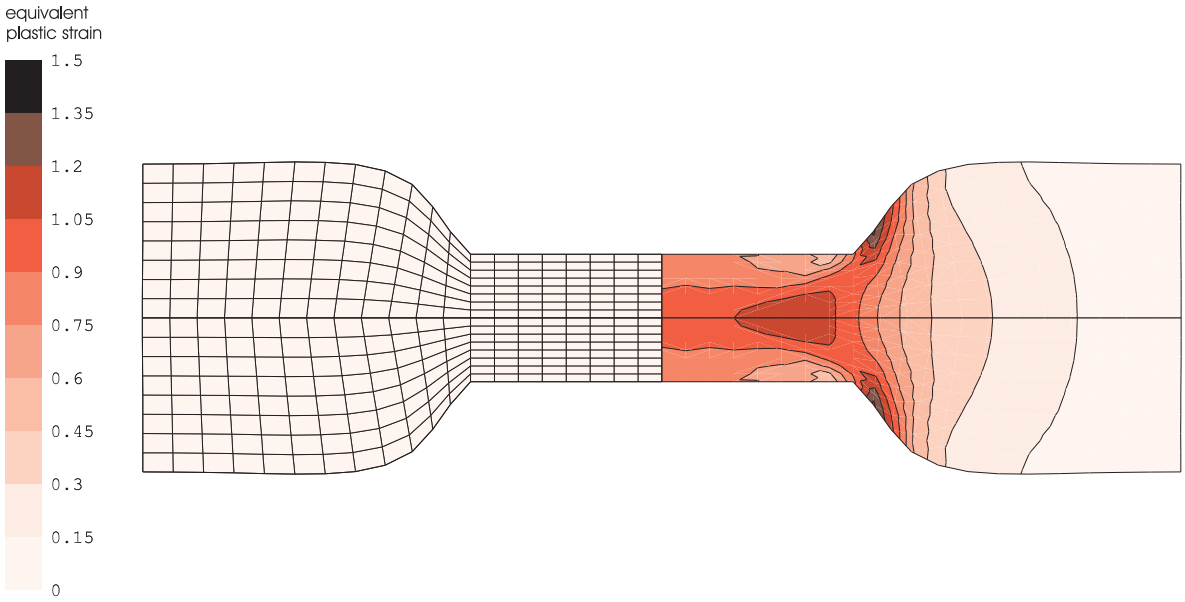

FIG. 3.15: Poinçonnement symétrique – solution 2D / axisymétrique – déformation plastique équivalente.

La figure 3.15 montre les résultats obtenus lors de la simulation 2D axisymétrique. On remarque que, comme prévu, les noeuds sous le poinçon ne se sont pas déplacé radialement. Nous voyons également que le maillage obtenu est très régulier.

La figure 3.16 montre deux simulations 3D du même problème. La première est une version possédant moins de mailles sur la section que la solution 2D et la seconde possède le même nombre de mailles sur la section.

En comparant le cas 2D et le cas 3D, on constate que les solutions obtenues pour un même nombre de mailles sur la section sont très similaires. Le champ de déformation plastique équivalente est identique ; ce qui montre une fois de plus le bon fonctionnement de nos nouveaux algorithmes 3D. Remarquons que le maillage grossier de la première solution ne permet pas d'obtenir une si bonne concordance. De plus, la solution obtenue n'est pas tout à fait axisymétrique (sur les deux plans de symétrie) ; ce qui n'est pas étonnant vu que le maillage n'est pas non plus axisymétrique et assez grossier.

Par contre, dans le cas similaire au problème 2D (maillage 3D raffiné), la solution obtenue montre beaucoup moins de défauts d'axisymétrie. La figure 3.17 montre la solution vue sous un autre angle (selon l'axe de poinçonnement). On constate que, vu sa construction,

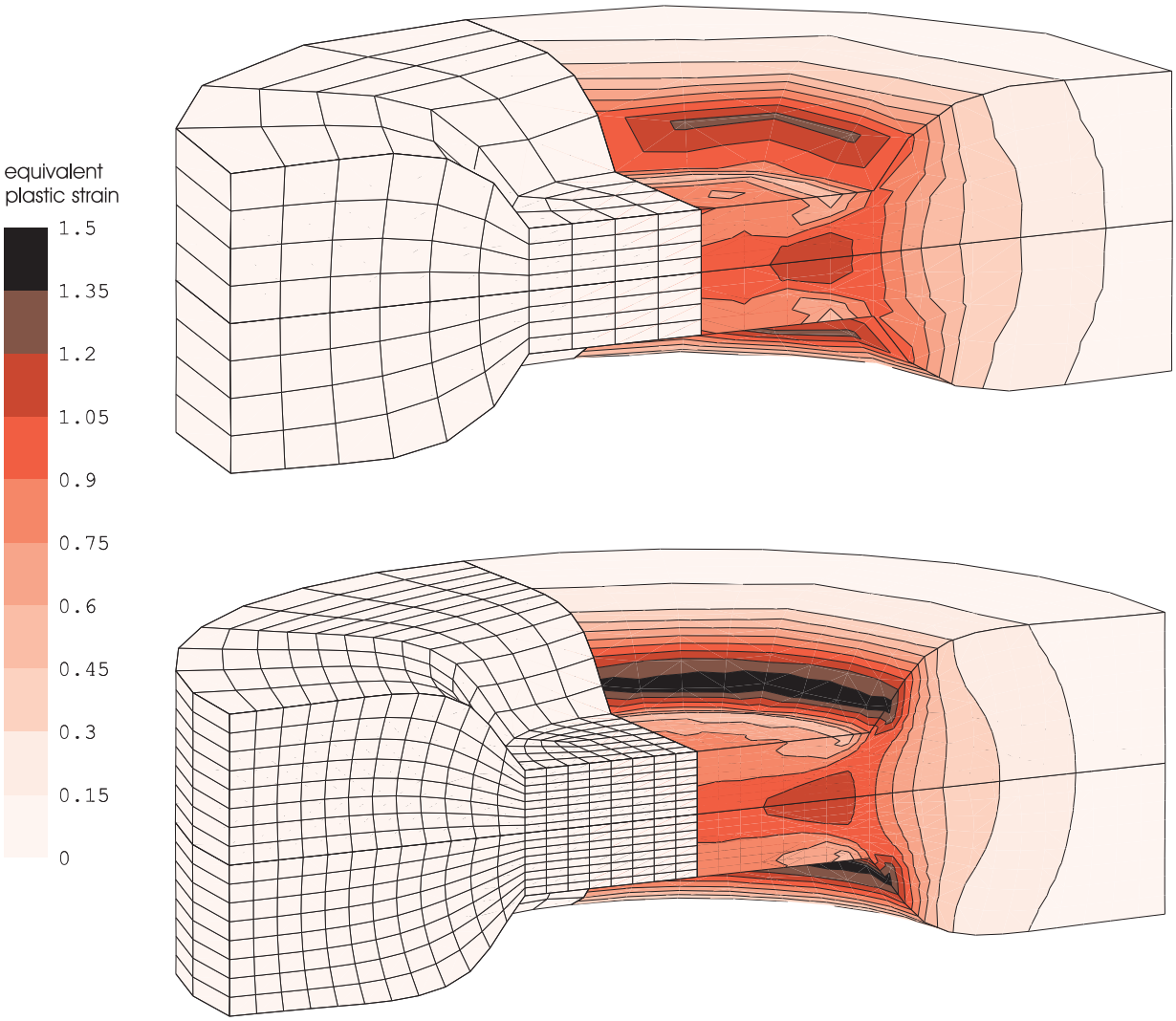

FIG. 3.16: Poinçonnement symétrique – solutions 3D – déformation plastique équivalente.

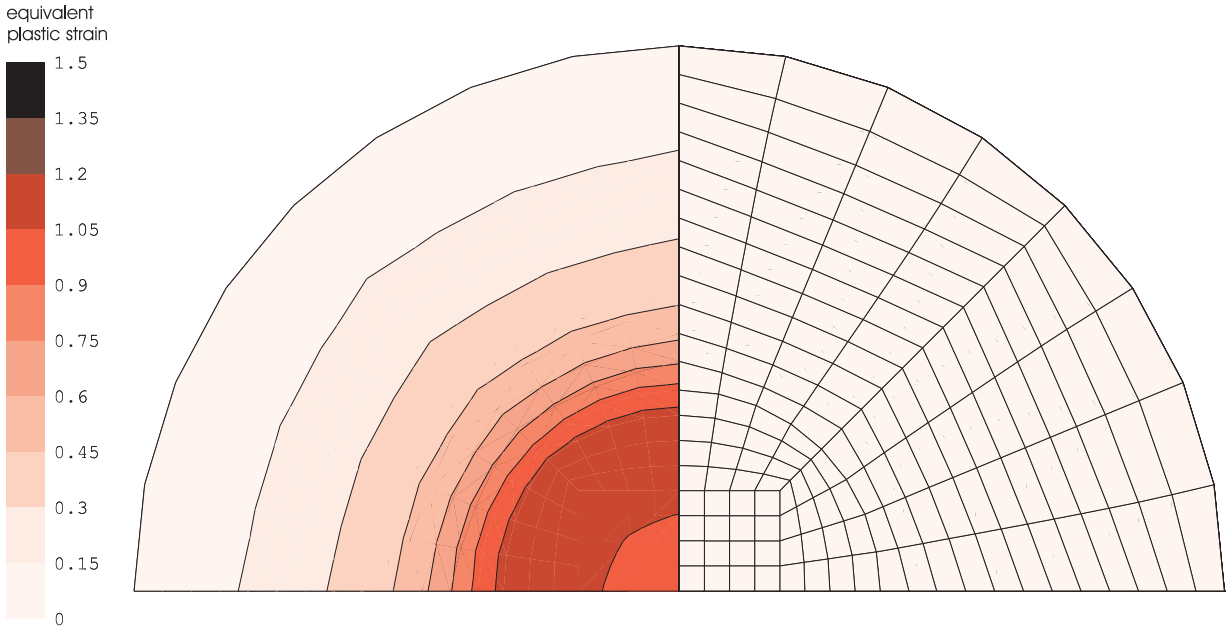

FIG. 3.17: Poinçonnement symétrique – solution 3D – déformation plastique équivalente.

le maillage n'est pas axisymétrique. Cependant, les isovaleurs obtenues sont beaucoup plus proches de cercles que ne l'est le maillage. Nous pouvons donc conclure que la solution 3D est très satisfaisante.

		2D axisym	3D grossier	3D axisym
Pas/ite		35/41	25/41	36/40
Nbre d'éléments		160	144	1152
Nbre de d.d.l.		339	557	3955
CPU	[min]	0 :02	0 :30	11 :55
CPU ALE	[min]	/	/	0 :35
Rayon final	[mm]	32.56	32.75	32.55
ε^p_{max}		1.386	1.273	1.4993

TAB. 3.5: Comparaison des résultats obtenus pour le poinçonnement symétrique

Le tableau 3.5.2 compare quelques valeurs numériques relatives aux différents problèmes envisagés. Le rayon mesuré en 2D est celui du noeud initialement sur l'intersection de la face supérieure du lopin et de la face extérieure (coordonnées $(x, y) = (R, h)$). En 3D, on prend l'équivalent sur un des plans de symétrie ($(x, y, z) = (R, 0, h)$).

On constate que les solutions obtenues sont assez proches les unes des autres, Cependant des différences relativement importantes apparaissent au niveau de la comparaison de la valeur de la déformation plastique équivalente maximale. Tout d'abord, il faut savoir que

l'on compare des valeurs extrapolées aux noeuds ; ensuite, le fait que le maillage 3D ne soit pas axisymétrique peut aussi introduire des différences (la valeur maximale n'est pas située au même endroit pour les différents problèmes).

Une dernière remarque concerne le temps de calcul. On peut voir que, dans le cas d'un schéma d'intégration temporel implicite et un problème 3D, la partie du temps de calcul concernant l'ALE est très petite (moins de 5%) pour le cas raffiné. Nous avons également essayé de réduire le temps de calcul nécessaire à la partie lagrangienne en n'utilisant plus un solveur direct de type Gauss mais bien un solveur itératif que nous avions anciennement développé [4, 9, 8] et que nous avons récemment réintroduit dans la "nouvelle tête" du code. Il nous a permis, sans faire de longues recherches sur les paramètres à utiliser, de réduire significativement le temps de calcul nécessaire au problème raffiné. Le solveur utilisé est le GMRES(60) accompagné d'un préconditionneur ILUT(50). Le temps de calcul passe alors de 11 min 35' à 7 min 06' soit une diminution d'environ 40% du temps de calcul. Lors de nos différents tests successifs pendant la période de développement de la librairie, ce gain de temps multiplié par de nombreux essais a été très appréciable.

3.5.3 Poinçonnement non symétrique

Nous considérons maintenant une variante du problème de poinçonnement de la section précédente. Il s'agit toujours du poinçonnement d'un lopin cylindrique par un poinçon cylindrique. Dans ce cas-ci, le lopin repose initialement sur un sol rigide. Le problème ne possède donc plus de symétrie suivant le plan $z = 0$. La hauteur du cylindre est la moitié de celui utilisé pour le test symétrique, ce qui nous permet de réutiliser exactement le même maillage que précédemment.

Le problème est traité en 2D et en 3D. En 2D, on utilise la formulation axisymétrique et en 3D, un quart du lopin est modélisé vu la symétrie de révolution.

Les propriétés matérielles et la géométrie sont les mêmes que celles utilisées précédemment (voir tableau 3.5.2 et figure 3.13). Le problème est traité en quasi-statique.

En ce qui concerne les paramètres de repositionnement des noeuds, nous conservons les mêmes que dans le cas symétrique que se soit en 2D ou en 3D, mis à part pour les lignes et les faces en contact avec le plan rigide et susceptibles de décoller du sol. En effet, ces faces seront courbes après déformation et nécessitent donc l'utilisation de l'algorithme de projection sur surfaces splines.

Le contact entre le plan rigide et le lopin est un contact avec frottement contrairement au contact entre le lopin et le poinçon où le frottement est nul. Nous utilisons, comme d'habitude avec METAFOR, une gestion du contact par la méthode de la pénalité. Le coefficient de frottement est fixé à 0.1 et nous essayons deux ensemble de valeurs pour les coefficients de pénalité normale (p_N) et tangentielle (p_T) pour le cas 2D : une pénalité élevée ($p_N = 10^8, p_T = 10^5$) et une pénalité plus faible ($p_N = 10^6, p_T = 10^3$). Nous verrons que

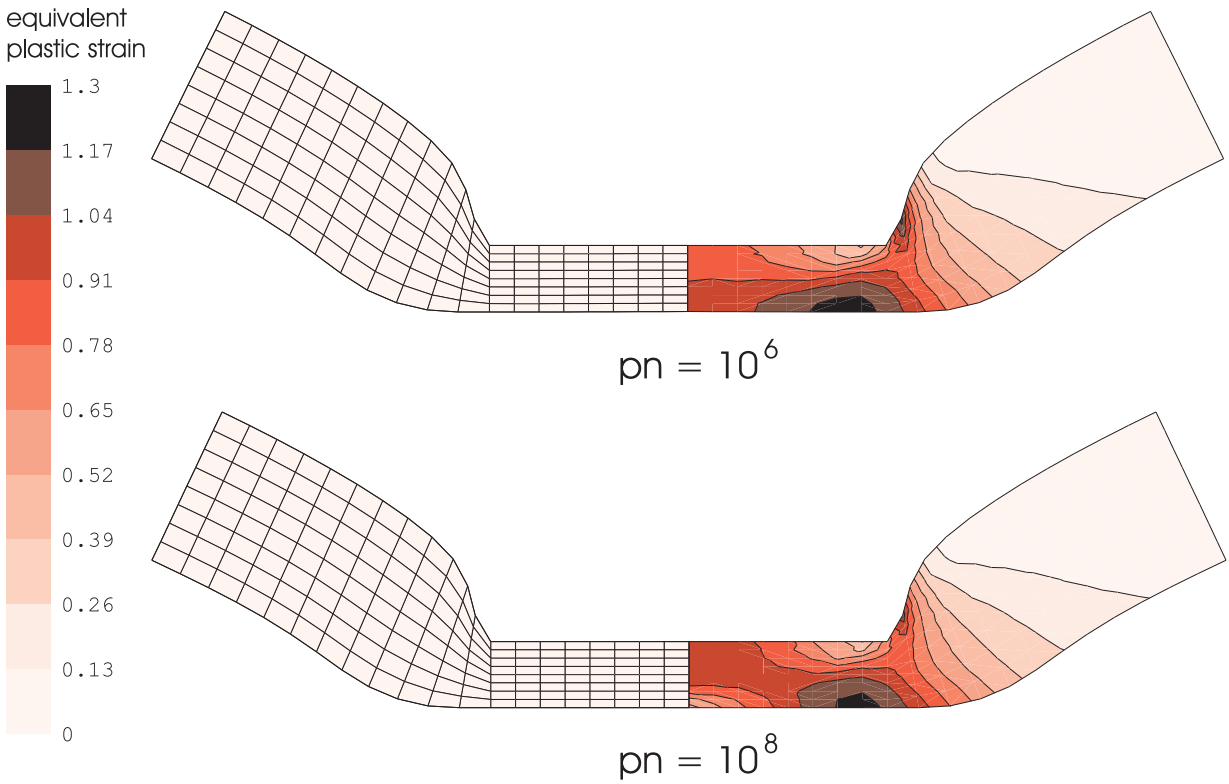

FIG. 3.18: Poinçonnement non symétrique – solution 2D / axisymétrique – déformation plastique équivalente.

ces coefficients influencent la solution. Pour le cas 3D, nous n'utilisons que les coefficients de pénalité faibles.

La figure 3.18 montre la solution 2D axisymétrique obtenue avec les deux ensembles de coefficients de pénalité. On y a représenté le maillage déformé et le champ des déformations plastiques équivalentes. On constate que les deux solutions sont identiques loin de l'axe d'axisymétrie. Lorsqu'on se rapproche de celui-ci, les solutions diffèrent légèrement. Dans le cas de l'utilisation d'une pénalité élevée, on remarque l'apparition d'une zone moins déformée plastiquement au niveau de l'axe d'axisymétrie. Cette différence sera expliquée lors de la présentation des résultats 3D.

Au niveau du maillage final, on constate que celui-ci est très régulier mais on observe cependant quelques mailles fortement cisaillées au niveau de la périphérie du poinçon. A cet endroit, le maillage forme un angle vif et on imagine que celui-ci poserait des problèmes si on envisageait une plus profonde indentation.

On pourrait penser que le problème vient du choix du remaillage de la ligne séparant les deux sous-domaines de calcul (nous avons imposé que les noeuds de celle-ci ne se déplace pas radialement tout au long du calcul). En fait, c'est la seule manière de faire aboutir le calcul. En effet, si on laisse plus de liberté aux noeuds de cette ligne, par exemple en leur appliquant une méthode du centre de gravité, les mailles deviennent de plus en plus écrasées au fur et à mesure que l'angle vif se rapproche d'un angle droit. On arrive rapidement à une situation où le centre de gravité de certaines mailles se retrouve en dehors du maillage (figure 3.14) !

Il semble donc nécessaire, dans la suite de ce travail, de programmer une méthode de repositionnement différente de la méthode du centre de gravité pour les noeuds intérieurs d'un maillage. La méthode envisagée est l'algorithme de Giuliani [28]. C'est un algorithme tout à fait général permettant de gérer des maillages 2D ou 3D, structurés ou non. En résumant très rapidement la méthode, on peut dire qu'il s'agit d'une méthode de minimisation locale du cisaillement (c'est-à-dire de la distorsion) des éléments adjacents à un noeud. Bien que cette méthode soit bien plus compliquée à programmer que le simple lissage par repositionnement au centre de gravité des mailles voisines, nous pensons qu'elle permettra de gérer correctement les angles vifs pouvant apparaître à la surface d'un matériau déformé lors de grandes déformations.

La figure 3.19 montre trois solutions obtenues pour différents raffinement de maillage en 3D. Insistons tout d'abord sur le fait que le maillage moyennement raffiné représenté n'a pas permis d'atteindre la profondeur de poinçonnement d voulue ($d = 5.7$ mm au lieu de $d = 6$ mm).

Le tableau 3.5.3 rassemble quelques caractéristiques numériques (taille du problème, temps de calcul,...) pour les 3 maillages utilisés.

La compréhension du problème de convergence observé nous paraît fondamentale pour

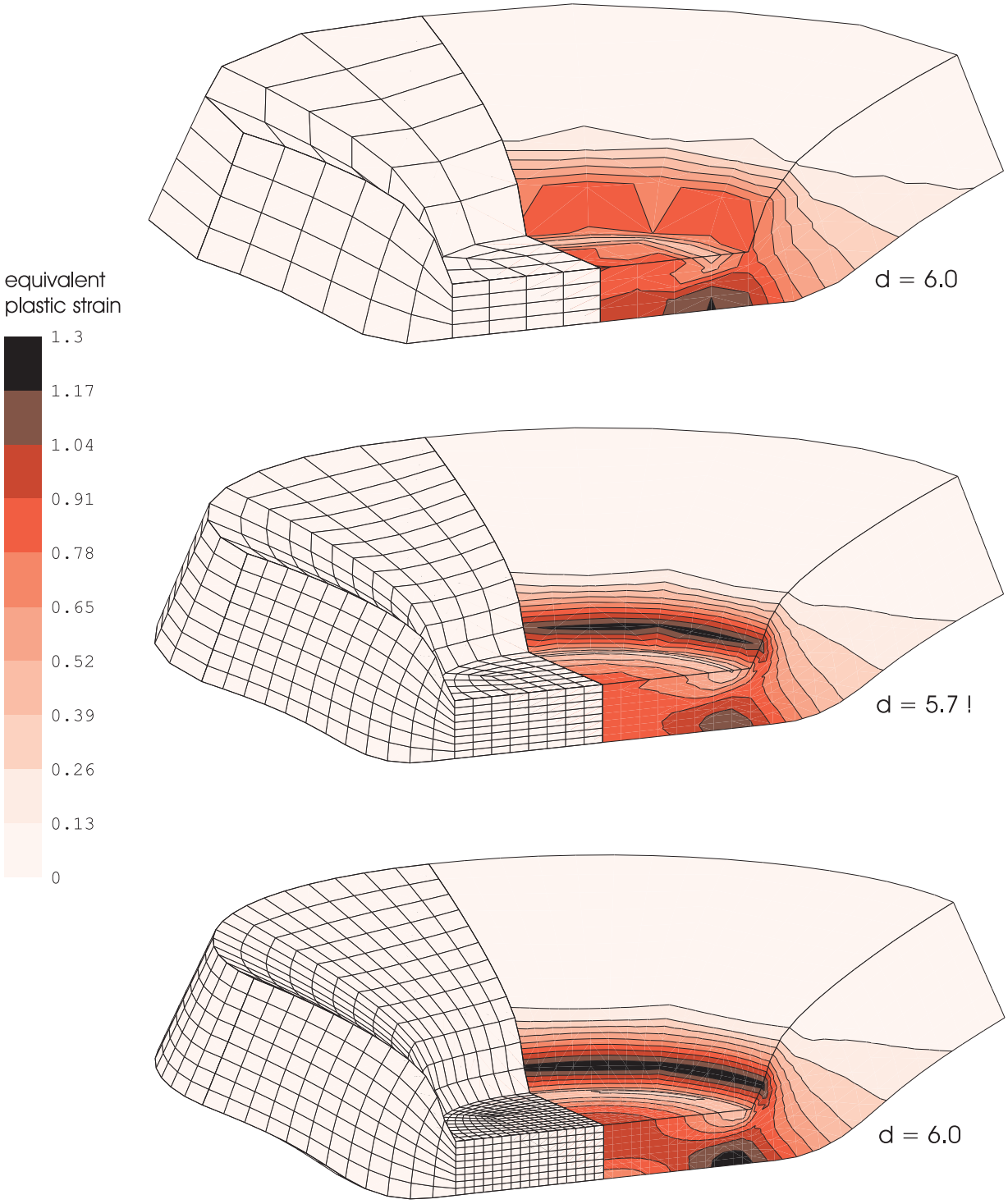

FIG. 3.19: Poinçonnement non symétrique – solutions 3D – déformation plastique équivalente.

		2D axisym	3D grossier	3D raff. 1	3D raff. 2
Pas/ite		51/79	48/75	(49/81)	49/75
Nbre d'éléments		160	144	1152	3072
Nbre de d.d.l.		360	606	4124	10628
CPU	[min]	0 :03	0 :58	14 :03	47 :40
CPU ALE	[min]	/	0 :08	/	2 :23

TAB. 3.6: Comparaison des résultats obtenus pour le poinçonnement non symétrique

l'avenir de notre librairie dans METAFOR. En effet, nous voulons que les limitations de nos méthodes soient parfaitement connues.

Le problème est similaire à celui observé en 2D. Nous avons cependant essayé de comprendre pourquoi ce problème de distorsion des éléments apparaît plus rapidement en 3D qu'en 2D (le maillage qui ne fonctionne pas en 3D possède le même nombre de mailles sur la section que le maillage 2D présenté précédemment). Pourtant, on constate que les maillages surfaciques ont l'air d'être bien conditionnés. Le problème est donc situé à l'intérieur du maillage.

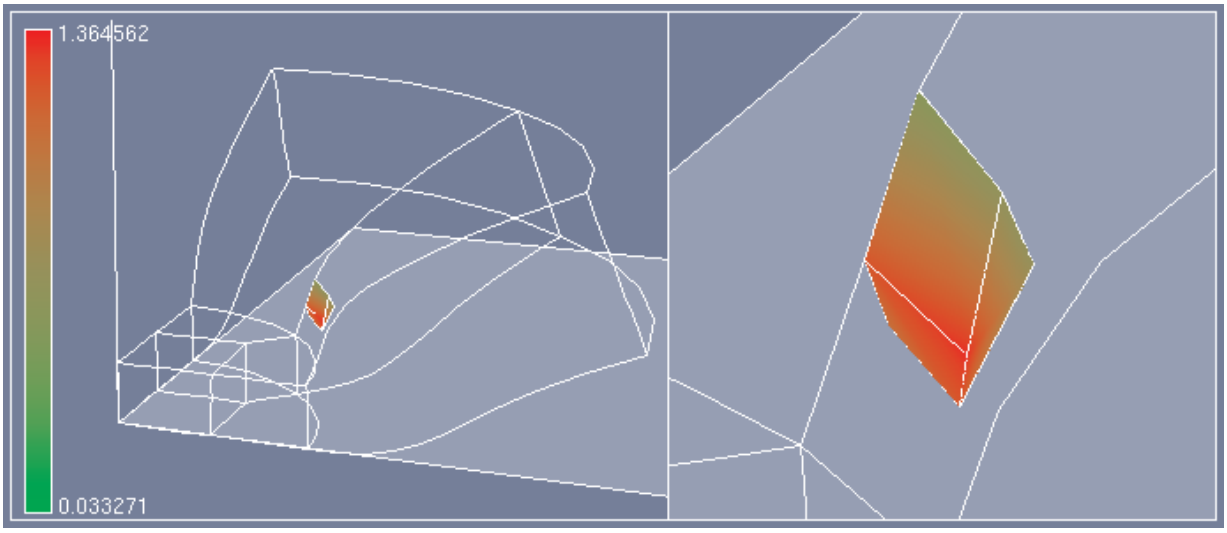

FIG. 3.20: Poinçonnement non symétrique – mauvais remaillage interne.

Pour comprendre ce problème, nous avons développé une interface graphique permettant de visualiser la déformation des mailles internes au maillage 3D pendant le calcul (voir développement auxiliaires au chapitre 4 et figure 3.20) Le problème vient de la courbure concave de la surface supérieure du lopin. Cette courbure provoque des problèmes similaires à ceux provoqués par un angle vif que l'on a expliqués en 2D. Le centre de gravité calculé pour repositionner certains noeuds internes proches de la surface du lopin tend à se rapprocher de la surface vu le sens de la forte courbure de la surface (ce problème n'apparaîtrait

pas pour une surface convexe). Si on essaye d'indenter le lopin de plus de 5.7 mm, certains noeuds internes se retrouvent à l'extérieur du maillage ; ce qui provoque des retournement de mailles et des valeurs de jacobiens négatifs.

C'est pour vérifier notre explication du problème que nous avons mis au point un autre maillage plus raffiné. Nous avons raffiné le problème de telle manière à ce que les mailles surfaciques de la face supérieure du lopin soient plus petites dans la direction circonférentielle. On diminue ainsi l'influence des noeuds voisins dans cette direction lors de l'évaluation du centre de gravité et, en conséquence, l'influence de la concavité dans cette direction. Nous constatons que le problème ainsi raffiné permet d'atteindre la profondeur d'indentation souhaitée ($d = 6$ mm).

Pour résoudre ce problème en conservant le maillage équivalent au maillage 2D, nous devons envisager une méthode alternative à la méthode du centre de gravité pour repositionner les noeuds internes d'un maillage 3D. La méthode que nous pensons programmer est la même que celle décrite pour le cas 2D : l'algorithme de Giuliani.

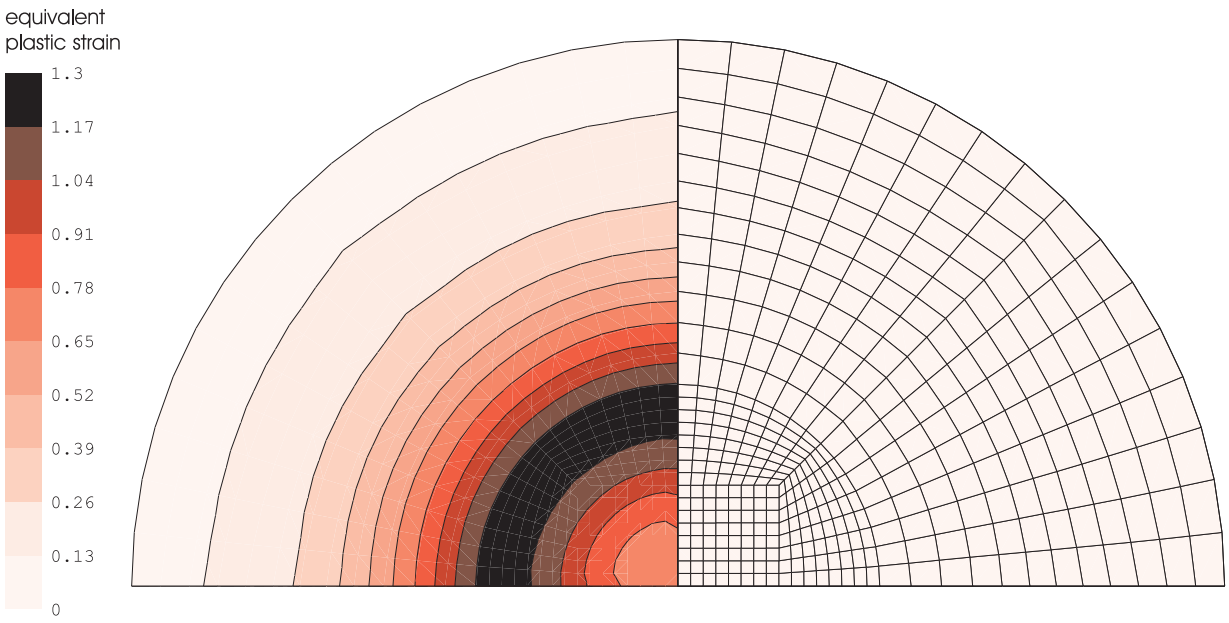

FIG. 3.21: Poinçonnement non symétrique – solution 3D – déformation plastique équivalente.

La figure 3.21 montre la solution raffinée suivant l'axe de poinçonnement. Nous voyons que, malgré le fait que le maillage utilisé n'est pas axisymétrique, nous retrouvons une solution axisymétrique. Ceci nous permet de penser que notre algorithme de repositionnement des noeuds sur des surfaces courbes par projection sur surfaces splines composites est très satisfaisant et nous pensons qu'il n'est pas nécessaire de l'améliorer pour l'instant.

Si nous revenons maintenant à l'influence de la pénalité normale sur le problème axisymétrique, on constate que la solution 3D raffinée pour laquelle on a utilisé une pénalité

de 10^6 ressemble à la solution axisymétrique où on a utilisé une pénalité normale de 10^8. C'est assez facile à comprendre : dans le cas 2D axisymétrique, il y a peu de noeuds en contact. En fait, chaque force de contact nodale est la résultante 3D des forces de contact sur un angle de 1 radian. Donc, les forces obtenues en 2D sont plus grandes que celles en 3D. Elles sont d'autant plus grandes que le noeud où elles s'appliquent est éloigné de l'axe d'axisymétrie. En conséquence, les pénétrations obtenus par la méthode de la pénalité sont plus grandes en 2D qu'en 3D (puisque proportionnelles à la force) et, par ce fait, le lopin est moins contraint. Il faut donc utiliser une valeur de pénalité normale plus élevée en 2D qu'en 3D pour obtenir des résultats identiques.

3.6 Indentation d'un lopin par un cylindre

3.6.1 Introduction

Nous présentons, dans cette section, deux cas-tests d'indentation d'un lopin de matière par un cylindre. La première variante est un calcul 3D qu'un problème en état plan de déformation. Ceci permet de comparer facilement les résultats obtenus par une simulation 2D. Le deuxième variante est l'extension 3D de ce problème. On peut alors voir fonctionner notre méthode de repositionnement de noeuds sur surfaces courbes.

Ce cas test a été initialement proposé par José Luis Farinatti Aymone [1] de l'Universidade Federal Do Rio Grande Do Sul au Brésil pour valider son implémentation du formalisme ALE.

3.6.2 Solution en état plan de déformation (EPD)

Notre premier problème est un simple problème d'indentation d'un lopin rectangulaire en état plan de déformation. Le lopin fait 4 cm de long, 1 cm de haut et 1 cm de large. Le poinçon a un rayon de 0.8 cm et on envisage d'indenter le lopin d'une profondeur de 0.8 cm. Vu la symétrie du problème, on ne considère que la moitié. Nous considérons le problème en 3D bien qu'il s'agisse d'un problème 2D. Cela nous permet de tester nos algorithme 3D ainsi que la gestion du contact en présence de remaillage. La demi géométrie est représentée sur la figure 3.22.

Le contact entre le cylindre et le lopin est sans frottement et on utilise une pénalité normale de 1000 N/cm. Le problème est traité en quasi-statique.

Module de Young	E	=	1000	N/cm^2
Coefficient de Poisson	ν	=	0.3	
Limite d'élasticité	σ_Y^0	=	10	N/cm^2
Coefficient d'écrouissage	h	=	5	N/cm^2

TAB. 3.7: Propriétés matérielles pour l'indentation d'un lopin par un cylindre (EPD)

Les propriétés matérielles utilisées sont regroupées dans le tableau 3.6.2. La loi d'écrouissage est linéaire isotrope.

On utilise un maillage de 40 éléments sur la demi-longueur, 4 éléments sur la hauteur et seulement 1 élément sur la largeur. le problème possède 160 éléments hexaédriques et 728 d.d.l.

Les méthodes utilisées pour le repositionnement des noeuds est simple : on utilise pour

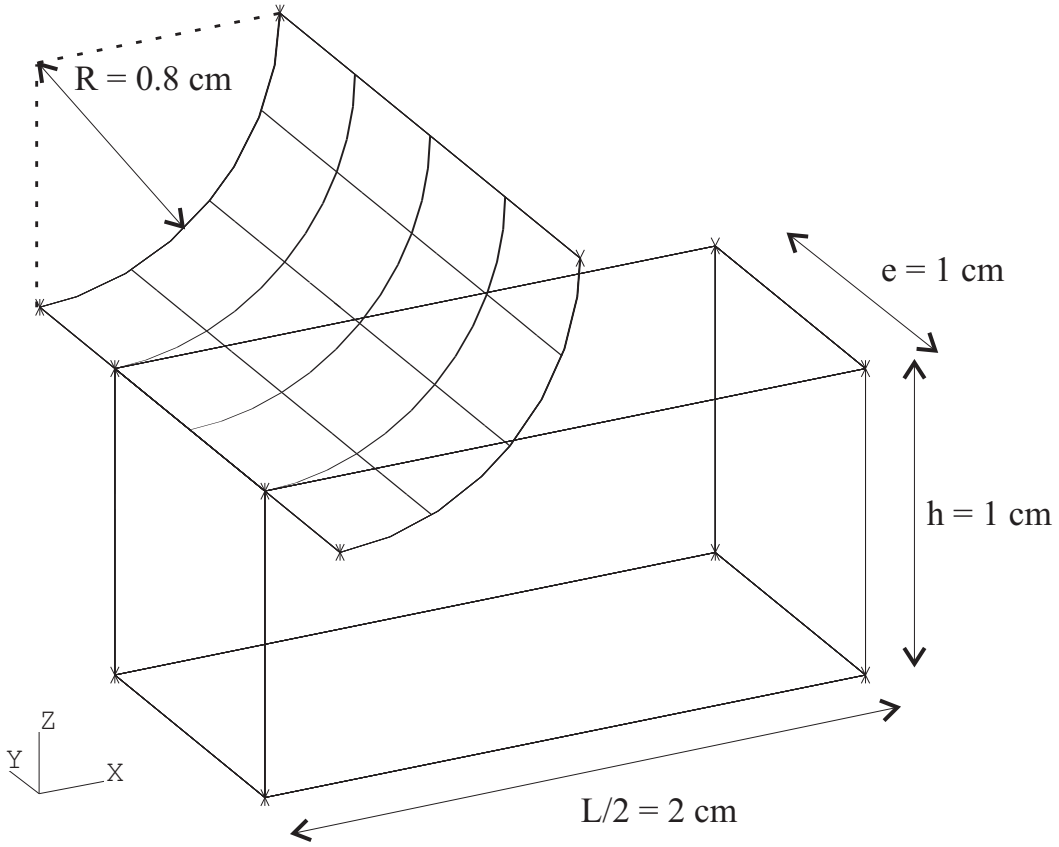

FIG. 3.22: Géométrie de l'indentation d'un lopin par un cylindre.

les lignes un remaillage sur splines et les faces sont remaillées par la méthode du centre de gravité. Vu que la surface supérieure du lopin ne comporte aucun noeud à part ceux sur son contour, on n'a pas besoin de spécifier une méthode de repositionnement sur cette face. Dans le domaine volumique, on utilise également une méthode de centre de gravité.

En ce qui concerne les paramètres ALE, nous utilisons un coefficient d'upwind α unitaire et 3 passes pour le remaillage par centre de gravité. Nous choisissons d'effectuer un repositionnement de la grille de calcul à la fin de chaque pas de temps (nrem = fréquence de remaillage = 1).

	pas/ite	CPU	CPU ALE
Lagrangien	192/384	03 :39	0
ALE (nrem=1)	370/758	10 :35	03 :22 (30%)

TAB. 3.8: Temps de calcul pour le cas EPD

Nous comparons dans la suite le calcul lagrangien avec le calcul par le formalisme ALE. Le tableau 3.6.2 montre quelques résultats numériques pour les deux calculs. On

constate que le temps de calcul est beaucoup plus élevé dans le cas du formalisme ALE (plus du triple). En effet, nous voyons que pour l'ALE, environ 1/3 du temps de calcul est passé dans les routines de repositionnement des noeuds et de convection des grandeurs aux points de Gauss et 2/3 est consacrée au calcul Lagrangien. On voit aussi que la partie du calcul lagrangien du problème ALE nécessite 2 fois plus d'itérations au total que le calcul lagrangien pur. Nous expliquons cette différence par le fait que notre implémentation de l'ALE en 3D ne gère pas différemment un noeud en contact et un noeud qui n'est pas en contact. Nous n'avons pas encore mis au point une routine qui assure que les noeuds en contact sont correctement équilibrés après remaillage. La convergence est donc plus difficile puisqu'après chaque pas de temps, il peut exister un déséquilibre assez important au niveau de ces noeuds. Nous envisageons donc, dans le futur, de mettre au point une routine de repositionnement des noeuds en contact. Nous allons voir, cependant, que la solution obtenue est correcte et même meilleure que la solution lagrangienne au niveau de la qualité du maillage obtenu.

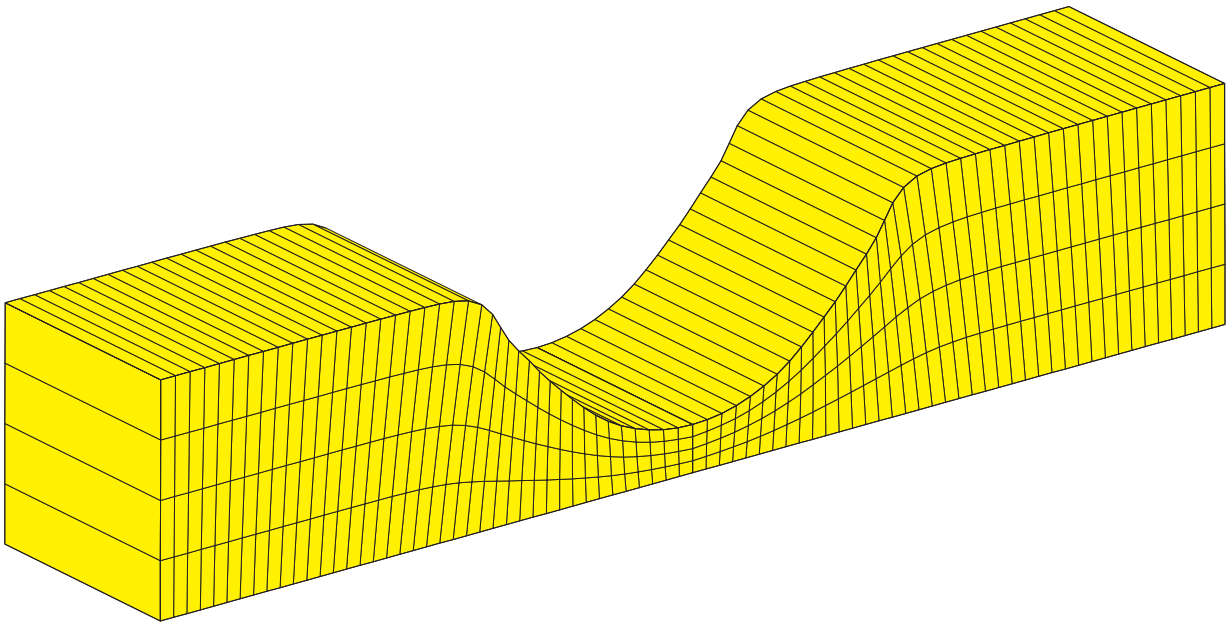

FIG. 3.23: Indentation état plan de déformation – maillage final.

La figure 3.23 montre la maillage final obtenu par le formalisme ALE. On constate que les mailles sont restées bien conditionnées durant tout le calcul. Si on compare les deux problèmes (comparaison Lagrangien et ALE sur la figure 3.24), on voit que les deux solutions sont très semblables. Le maillage lagrangien montre des écrasements de mailles juste sous le poinçon, près du plan de symétrie. Ces écrasements sont évités par le remaillage du lopin par nos routines ALE.

On remarque également que les géométries déformées ne sont pas exactement les mêmes. La figure 3.25 montre une superposition des deux déformées. On constate alors que, au

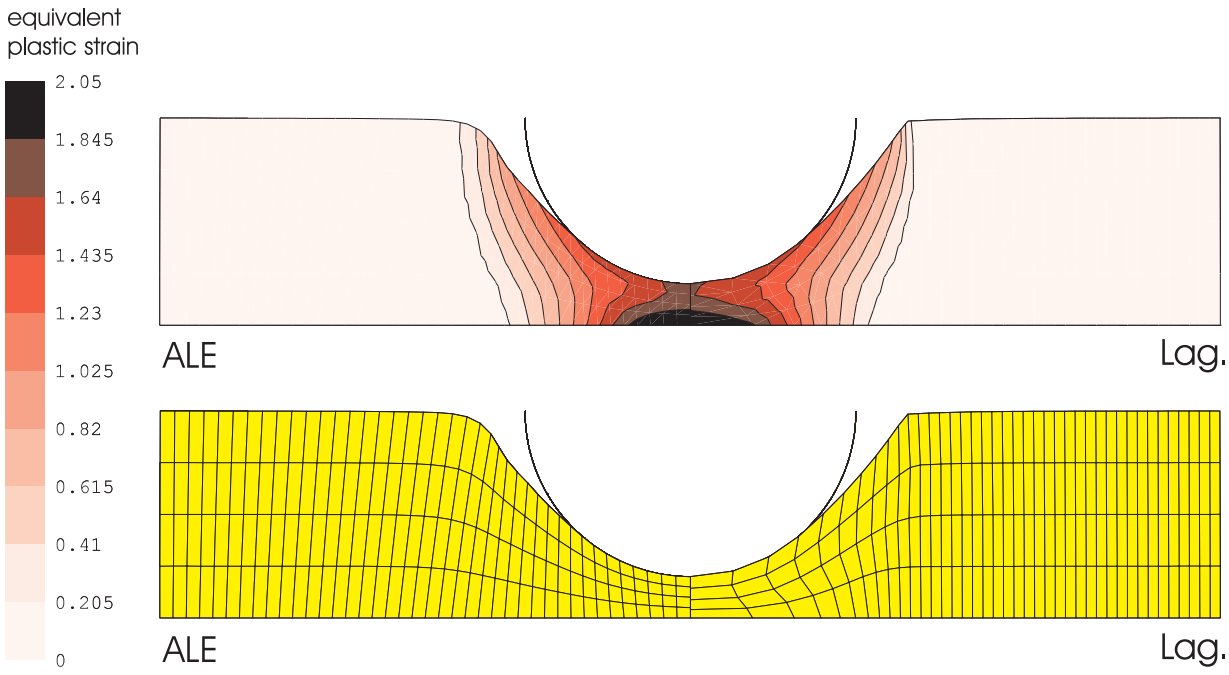

FIG. 3.24: Indentation état plan de déformation – comparaison Lagrangien / ALE.

niveau de la surface supérieure au lopin, le formalisme ALE a légèrement lissé les arêtes. C'est assez bien compréhensible puisque la solution lagrangienne présente une géométrie avec une apparente rupture de tangente. Or la méthode de repositionnement des noeuds sur les lignes basée sur des splines cubiques fait l'hypothèse que la ligne possède une tangente bien continue et régulière.

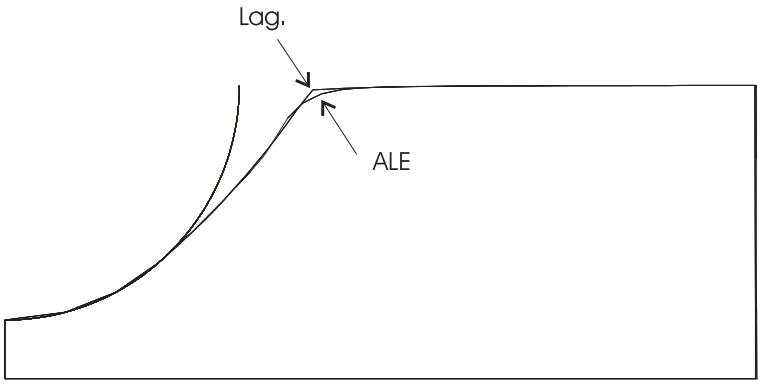

FIG. 3.25: Indentation état plan de déformation – comparaison des contours finaux obtenus en Lagrangien et ALE.

3.6.3 Solution 3D

Nous envisageons maintenant le même problème que précédemment sans fixer les faces latérales. Celles-ci peuvent alors se déformer hors de leur plan au cours de la descente du poinçon cylindrique. Nous fixons simplement un des noeuds pour éviter l'apparition d'un mode rigide en translation.

La géométrie et les propriétés matérielles et le maillage utilisées sont identiques à ceux utilisés dans le cas du problème en état plan de déformation (voir notamment le tableau 3.6.2 et la figure 3.22).

Pour le remaillage des deux faces latérales, nous utilisons cette fois-ci notre algorithme de projection sur surfaces splines. Ce problème est très intéressant pour voir si notre algorithme va permettre d'obtenir une représentation correcte des deux surfaces latérales libres.

	pas/ite	CPU	CPU ALE
Lagrangien	194/365	03 :52	0
ALE (nrem=1)	309/630	09 :37	2 :52 (32%)

TAB. 3.9: Temps de calcul pour le cas 3D

Le tableau 3.6.3 montre les temps de calcul et le nombre de pas de temps et d'itérations pour aboutir à la solution finale. On constate le même phénomène que dans le cas en état plan de déformation : le calcul lagrangien est 3 fois plus rapide que le calcul ALE. Nous attribuons cette différence d'une part au temps passé dans les routines de repositionnement de la grille de calcul et de convection et d'autre part à l'absence de traitement spécial pour les noeuds en contact.

Remarquons aussi que si on rapporte le temps passé dans les routines ALE au temps total, on constate que le fait d'utiliser notre algorithme de projection sur spline n'augmente que de 2 % la fraction de temps passée dans les routines ALE.

La figure 3.26 montre le maillage final obtenu par le formalisme ALE. Les mailles surfaciques sont restées plus ou moins cubiques lors de la déformation et les surfaces latérales sont bien courbes.

La figure 3.27 compare les maillages et le champ de déformations plastiques équivalentes dans les deux cas. on voit bien l'avantage de l'ALE en ce qui concerne la qualité du maillage obtenu. Les valeurs finales sont très similaires et on observe, comme dans le cas en état plan de déformation, un léger lissage de la surface supérieure du lopin. Ce lissage peut être comparé d'une manière plus précise sur la figure 3.28 où les deux contours déformés ont été superposés.

La figure 3.29 montre les deux maillages vus du dessous. Cela permet de voir facilement

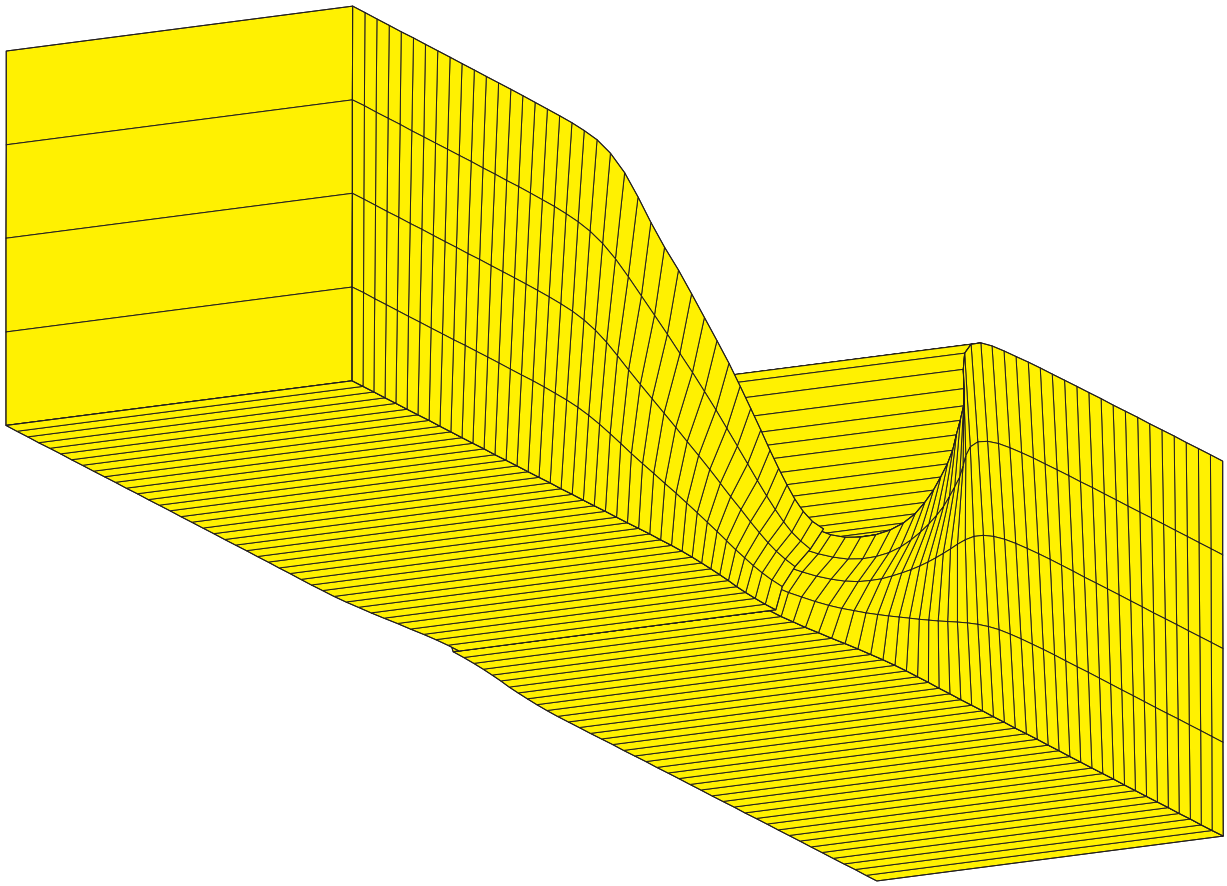

FIG. 3.26: Indentation 3D – maillage final.

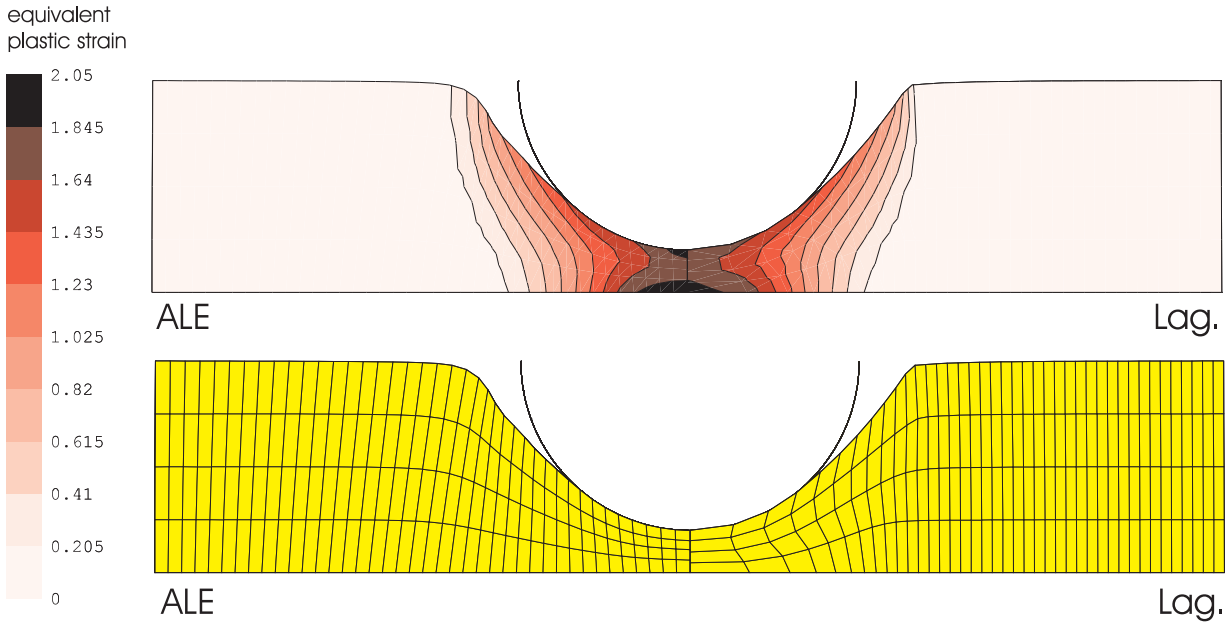

FIG. 3.27: Indentation 3D – comparaison Lagrangien / ALE.

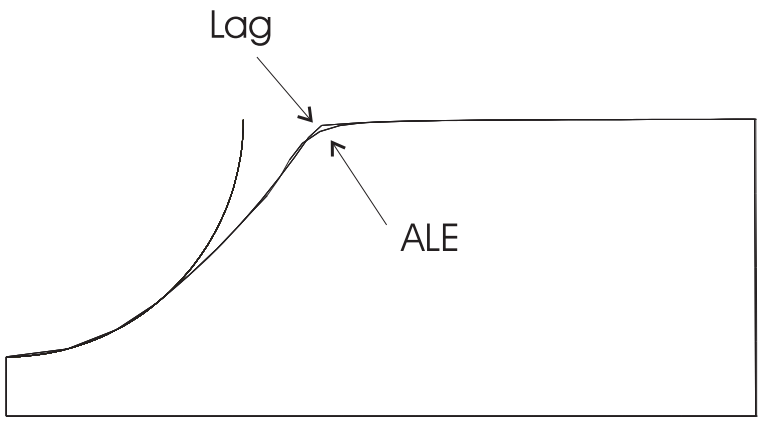

FIG. 3.28: Indentation 3D – comparaison des contours finaux obtenus en Lagrangien et ALE.

la différence de régularité entre les deux maillages obtenus par les deux formalismes.

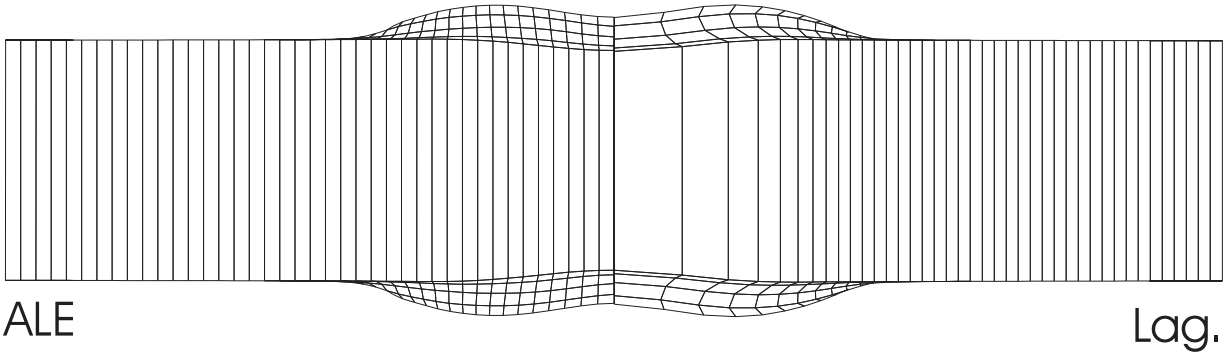

FIG. 3.29: Indentation 3D – comparaison des maillages finaux obtenus en Lagrangien et ALE.

Enfin, la figure 3.30 superpose les deux contours obtenus pour permettre une meilleure comparaison. Les différences, qui sont très faibles montrent tout de même que la solution ALE montre des tangentes perpendiculaires au plan de symétrie du problème simplement parce que le nombre de noeuds étant plus important dans la région écrasée, la géométrie finale est mieux approximée.

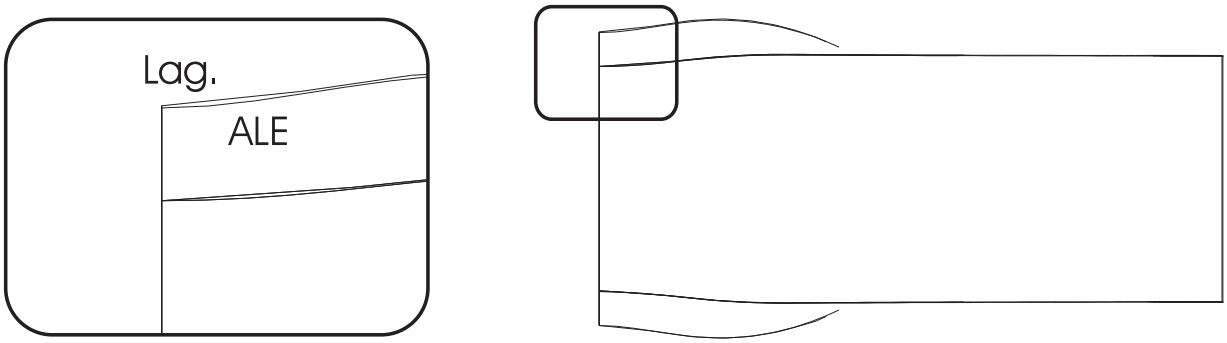

FIG. 3.30: Indentation 3D – comparaison des contours finaux obtenus en Lagrangien et ALE.

En conclusion, nous avons montré, grâce à ce problème, que notre algorithme de gestion des noeuds sur des surfaces courbes donne de bons résultats. Cependant, il serait intéressant de repositionner différemment les noeuds en contact avec le poinçon. Il nous semble que cela permettrait d'effectuer de plus grands pas de temps et, en conséquence, réduire le temps de calcul pour tendre vers les 2/3 du temps de calcul actuel.

3.7 Conclusions et perspectives

La ré-écriture de la gestion du formalisme ALE dans METAFOR nous a mobilisé une grande partie de cette année de travail. Est-ce du temps perdu ? Il est évident que non. On peut même dire que c'était une étape incontournable de ce travail de thèse.

Au début de cette année, nous avons dû effectuer un choix qui allait influencer le reste de notre thèse : fallait-il mettre à jour notre travail informatique pour pouvoir les intégrer dans la nouvelle "tête" de METAFOR ou, au contraire, rester dans l'ancienne version et continuer notre recherche sur la lubrification ? La première solution, celle que nous avons choisie, impliquait un investissement informatique important mais garantissait que notre thèse serait conservée au fil des différentes versions de METAFOR. La deuxième permettait, par contre, d'avancer dans la modélisation de la lubrification, tout en sachant que le formalisme ALE d'alors montrait de sérieuses limitations et compliquait inutilement la résolution des équations de lubrification.

Nous avons donc choisi logiquement le premier choix. En effet, grâce à ce long travail informatique aujourd'hui presque terminé :

- nous n'avons plus aucune limitations sur les géométries envisageables avec le formalisme ALE.
- le code est extrêmement clair vu l'utilisation d'une méthode de programmation claire, efficace, dynamique et structurée.
- nous sommes certain que nos développements seront toujours utilisables après notre thèse.
- nous savons que notre librairie est indépendante de la machine utilisée, du programme de pré-traitement utilisé et même du code éléments finis utilisé !
- nous abordons la résolution des problèmes de lubrification en étant confiant dans les possibilités et en connaissant les limitations du formalisme ALE. Nous n'aurons donc pas de surprises au niveau des résultats qui sortent des nouvelles routines.

Dans un avenir proche, nous mettrons au point une méthode de gestion des surfaces frontières en 3D (c'est à dire des surfaces à travers lesquelles la matière peut s'écouler). A ma connaissance, ce genre de possibilité n'est pas courante dans les codes éléments finis actuels. Cela permettra de simuler des problèmes de laminage ou d'étendre l'étude des jonc au cas tri-dimensionnel.

Nous avons vu également la nécessité de transférer la quantité de mouvement pour les problèmes en dynamique rapide comme l'impact de la barre de Taylor. Nous utiliserons, pour ce faire une méthode volumes finis basée sur un nouveau maillage auxiliaire dont les cellules sont centrées sur les noeuds du maillage éléments finis.

Il faudra aussi envisager une méthode alternative pour repositionner les noeuds intérieurs d'un maillage 2D ou 3D. En effet, nous avons vu, dans le cas du poinçonnement que les fortes variations de normales sur une surface peut provoquer les problèmes lors du repositionnement des noeuds internes proches de la surfaces. Il arrive même que ceux-ci soient

repositionnés à l'extérieur au maillage, ce qui provoque l'arrêt prématuré de la résolution du problème.

Nous devrons aussi programmer une méthode de repositionnement des noeuds en contact. Ceci permettra d'obtenir un meilleur équilibre de la configuration remaillée et une diminution du temps de calcul.

Enfin, METAFOR va se voir doté de nouveaux éléments finis. Il s'agit principalement de coques 2D (des lignes) et 3D (des triangles). Il sera alors intéressant d'adapter les quelques routines spécifiques aux éléments quadrangulaires et hexaédriques (calcul du volume, de la surface, division de l'élément en volumes finis, ...) de la librairie à ce type d'élément pour pouvoir traiter des problèmes de coques par le formalisme ALE.

Chapitre 4

Développements auxiliaires

4.1 Introduction

A côté de la réécriture des routines ALE, nous avons effectué plusieurs petites tâches annexes qui ont chacune plus ou moins d'importance. Nous présentons brièvement dans ce chapitre chacune d'elles ainsi que leur intérêt dans le cadre de ce travail de thèse.

4.2 Pilotage en force 3D des matrices rigides

Lors de notre première année de bourse FRIA, nous avions mis au point un algorithme de pilotage en force des matrices de contact 2D. Ceci avait pour but de simuler le phénomène de stick-slip sur un problème plan-plan où le patin devait serrer la tôle avec une force constante et non un déplacement constant.

Nous avons étendu cet algorithme au cas 3D. Cette tâche était assez simple puisque nous n'ajoutons que des degrés de liberté en translation à chaque matrice pilotée en force. Il n'est donc pas encore possible d'appliquer un couple imposé sur une matrice.

L'extension 3D s'est donc résumée à ajouter un degré de liberté supplémentaire possible en translation (translation selon z).

Quelques cas-tests très simples ont été effectués pour valider l'algorithme et il semble que celui-ci fonctionne aussi bien que son homologue 2D. Récemment, l'algorithme a même permis de simuler de plus gros problèmes tels que le S-Rail (emboutissage d'une tôle par un poinçon en forme de "S"). Il faut dire que ce genre de cas-test possède un nombre très important de noeuds de contact, ce qui peut rendre le mouvement de la matrice très non-linéaire.

4.3 Développement d'une interface de visualisation

4.3.1 Introduction

Lors de nos développements de la librairie ALE et, principalement pendant la création de la patrie 3D, nous avions besoin d'une manière simple et conviviale de visualiser les résultats obtenus. Or, le seul outil de visualisation que nous possédons au laboratoire est l'outil de post-traitement de SAMCEF : BACON.

Si BACON permet assez facilement de vérifier visuellement la qualité d'un maillage en 2D, il reste assez difficile et peu convivial à utiliser en 3D. En effet, effectuer des rotations de la pièce étudiée se fait à partir du clavier et il est temps d'avoir une perception 3D très développée pour fournir les bonnes valeurs de rotation du premier coup. Le zoom et les translations sont gérées de la même manière.

De plus, il peut arriver qu'un maillage 3D apparemment régulier, c'est-à-dire dont les maillages surfaciques sont réguliers, soit en fait totalement distordu à l'intérieur de la pièce. Ainsi, en essayant de simuler le problème du poinçonnement, nous obtenions des mailles retournées (jacobiens négatifs) à l'intérieur du maillage alors que les maillages surfaciques étaient corrects. Comment visualiser facilement ce qu'il se passe dans ce cas particulier sans programmer soi-même un outil de visualisation ?

Nous nous sommes lancé dans ces développements en se disant qu'avoir un tel outil serait indispensable lors de la résolution de problèmes plus complexes que ceux que nous avons envisagés jusqu'aujourd'hui par le formalisme ALE. Il est donc possible, grâce à cet outil, de visualiser l'intérieur du maillage et, plus particulièrement, les mailles posant des problèmes. On peut alors essayer de comprendre pourquoi ces retournements de mailles apparaissent (il s'agit le plus souvent d'une méthode de remaillage inappropriée) et de résoudre le problème soit en changeant de méthode de remaillage, soit en reprogrammant une nouvelle méthode.

Ayant déjà programmé lors de nos loisirs des petites interfaces graphiques, le travail a été relativement rapide.

Le résultat est assez complet. L'aspect 3D a été privilégié mais il est tout de même possible d'utiliser l'interface en 2D (le zoom n'est pas au point en 2D).

4.3.2 Présentation des possibilités

Notre interface est destinée à être utilisée principalement sous Linux (PC) mais pourra être utilisée sous la majorité des systèmes Unix.

Nous avons choisi d'utiliser la librairie GTK+ pour gérer les fenêtres, boutons et autres

“widgets”. Cette librairie devient de plus en plus populaire puisqu'elle est la base de la majorité des applications “gnome” (système d'exploitation libre couramment utilisé sous Linux). Nous avons compilé cette librairie avec succès sous Digital Unix (les stations de travail du laboratoire).

Pour la visualisation graphique 2D ou 3D, nous avons choisi d'utiliser la librairie OpenGL très répandue à l'heure actuelle. Cette librairie graphique permet de dessiner en 3D aussi simplement qu'en 2D.

Au départ, nous n'étions intéressé que par la visualisation d'un élément sélectionné dans le maillage pour pouvoir observer sa déformation au cours du calcul mais, très vite, nous nous sommes rendu compte qu'ajouter d'autres fonctionnalités comme la gestion de l'éclairage ou le prise en compte des symétries du problème était un jeu d'enfant une fois que la gestion de la fenêtre de visualisation a été implémentée.

Il était aussi très important de pouvoir piloter METAFOR (c'est-à-dire le lancer, l'arrêter temporairement, reprendre le calcul,...) à partir de l'interface graphique. Nous nous sommes donc dirigé vers une solution où le programme principal (METAFOR) lance un processus auxiliaire (appelé “thread”) gérant la fenêtre graphique. Les deux processus peuvent communiquer entre eux car ils partagent le même espace en mémoire. Nous utilisons donc une série de variables globales rassemblées dans une structure pour envoyer des ordres de METAFOR vers la fenêtre graphique (comme par exemple : le pas de temps est fini, il faut mettre à jour la géométrie) ou de la fenêtre vers METAFOR (comme par exemple : l'utilisateur veut arrêter le calcul, il a poussé sur le bouton “stop”).

Lorsque l'utilisateur lance METAFOR pour effectuer un calcul, il peut choisir de spécifier l'option “viz=1” en ligne de commande. Cette option permet l'ouverture de la fenêtre graphique. Un aperçu de la fenêtre graphique est montré sur la figure 4.1. Le numéro du pas de temps courant et d'itération ainsi que le temps total est affiché et actualisé en permanence. En dessous se trouve la zone graphique proprement dite où est affiché le maillage. A l'aide de la souris, l'utilisateur peut effectuer des rotations (bouton gauche), des translations (bouton du milieu) et des zooms (bouton droit).

Différents boutons permettent, à partir de cette fenêtre de démarrer, arrêter, continuer jusqu'à l'itération suivante et jusqu'au pas de temps suivant. On peut aussi choisir la grandeur visualisée sur le maillage (contraintes, déformation plastique équivalente, densité,...). Le bouton “config” ouvre la fenêtre représentée sur la figure 4.2. Cette fenêtre assez complexe permet de gérer tous les paramètres d'affichage.

Trois onglets permettent de configurer l'éclairage, les paramètres de visualisation des résultats et l'affichage OpenGL. La figure 4.2 montre la configuration de l'éclairage. Trois lampes sont disponibles. On peut choisir leurs couleurs, leurs positions et un bouton permet de les désactiver. On peut aussi définir la couleur et l'intensité de la lumière ambiante.

Le deuxième onglet permet d'afficher les paramètres de visualisation des résultats (figure

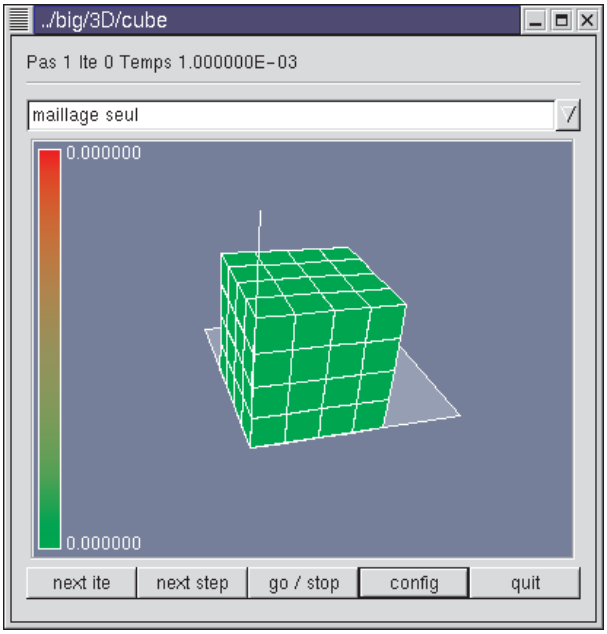

FIG. 4.1: La fenêtre de visualisation.

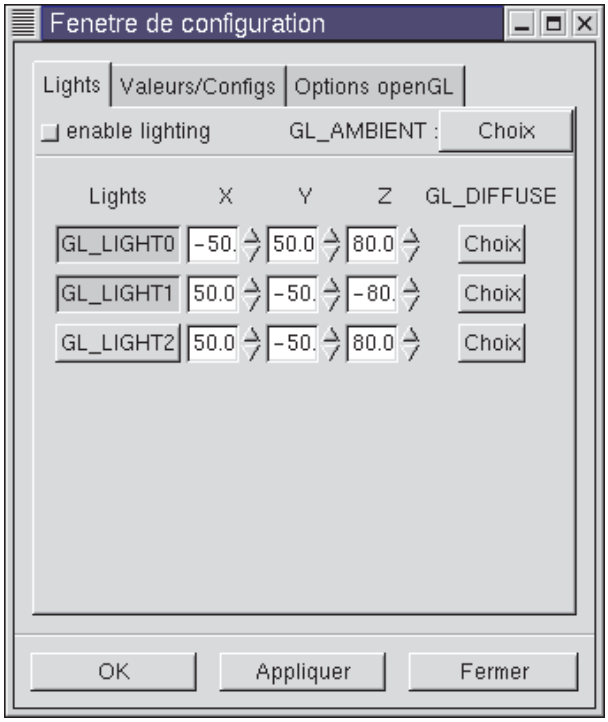

FIG. 4.2: La fenêtre de configuration de l'éclairage.

4.3). Dans cette fenêtre on peut régler les valeurs minimum et maximum affichées ou laisser le programme adapter les valeurs au cours du calcul (option "auto"). On peut aussi choisir quelle configuration (initiale, équilibrée, remaillée ou actuelle) on désire voir. Il est possible d'afficher plusieurs configurations ensemble, cependant, les configuration supplémentaires sont vues en fil de fer.

Le troisième onglet permet de configurer la librairie OpenGL et certaines autres options. Ainsi, on peut choisir d'afficher la structure en fils de fer ou avec des facettes pleines, d'afficher le maillage (grille), le contour, les axes, les noeuds, ... Les lignes peuvent subir de l'anti-aliasing pour supprimer les effets d'escalier, on peut enclencher l'algorithme automatique de vus-cachés, ...

En dessous de ces différentes options, il est possible de sélectionner un élément par son numéro et de le visualiser (il faut alors dessiner le maillage en fils de fer, bien entendu). C'est certainement l'option la plus intéressante en ce qui concerne le développement de l'ALE. Nous avons pu comprendre et résoudre, avec cette option, plusieurs problèmes de repositionnement des noeuds internes du maillage. Nous pensons que cette option est un véritable outil qui permettra aux futurs utilisateurs de nos routines, de comprendre les différentes méthodes ALE et de mieux les choisir en fonction du problème.

Enfin, cette fenêtre propose une option qui permet de dupliquer le maillage étudié

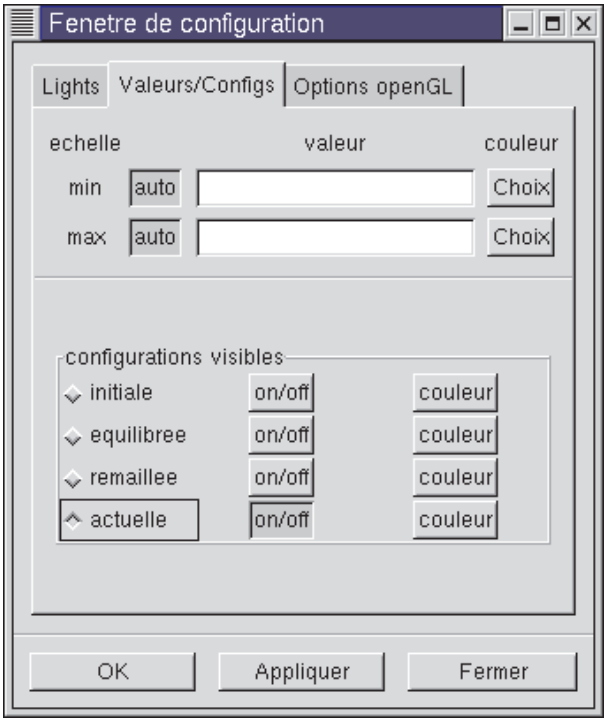

FIG. 4.3: La fenêtre de configuration de METAFOR.

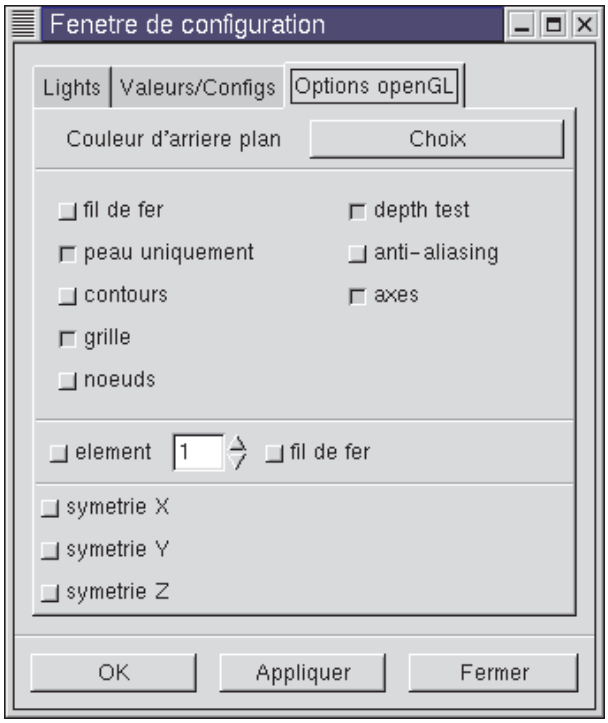

FIG. 4.4: La fenêtre de configuration d'OpenGL.

selon les 3 directions de l'espace pour visualiser la structure complète lorsqu'on traite des problèmes symétriques. Cette option permet de générer des dessins qui étaient jusqu'à lors impossible de tracer avec notre programme de post-traitement BACON.

Les figures suivantes montre les différents effets des options et le résultat obtenu dans la fenêtre graphique.

La figure 4.5 montre le cube de la figure 4.1 après déformation (on a juste poussé une fois sur le bouton "go/stop" et on a activé la visualisation des déformations plastiques équivalentes). La figure 4.6 montre le même maillage avec l'éclairage (option "enable lighting").

La figure 4.7 montre la visualisation d'un élément intérieur au maillage. Nous avons choisi de ne tracer que les contours du cube déformé.

Les figures suivantes montre un autre problème (le poinçonnement symétrique) et l'utilisation des symétries. La figure 4.8 représente le maillage étudié, c'est-à-dire un huitième de la structure réelle. Les figures 4.9, 4.10 et 4.11 montrent les symétries selon X, Y et Z. On obtient, après l'application successives de ces 3 symétries, une représentation de la pièce complète.

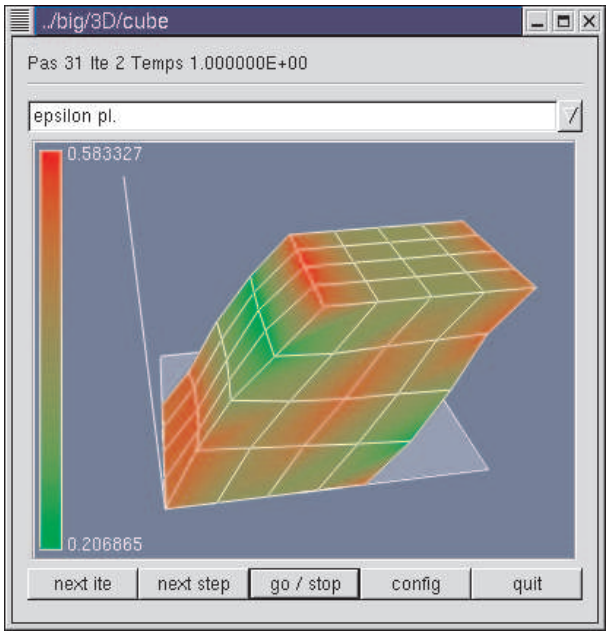

FIG. 4.5: Vue du cube après déformation.

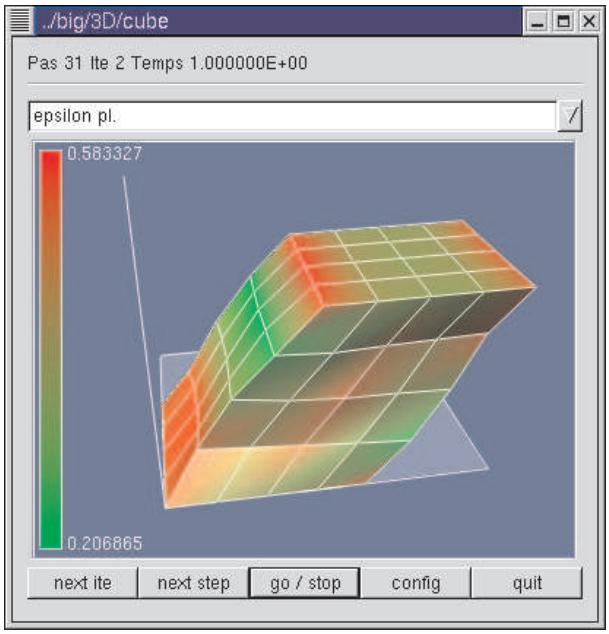

FIG. 4.6: Même vue avec éclairage.

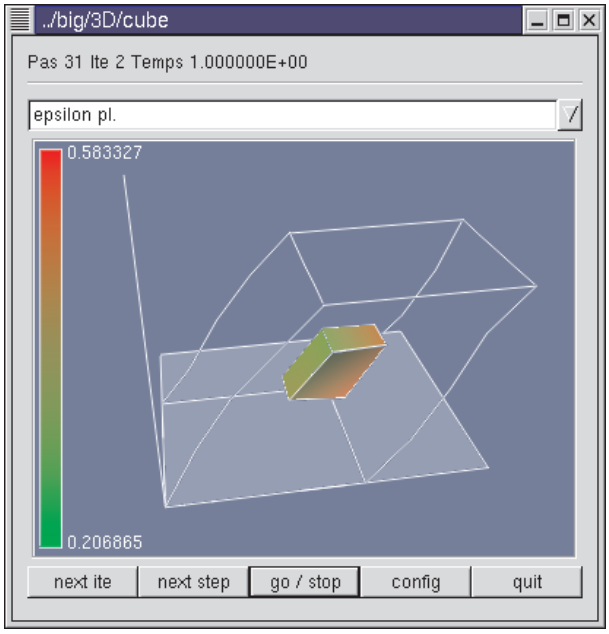

FIG. 4.7: Analyse de la déformation d'un élément intérieur.

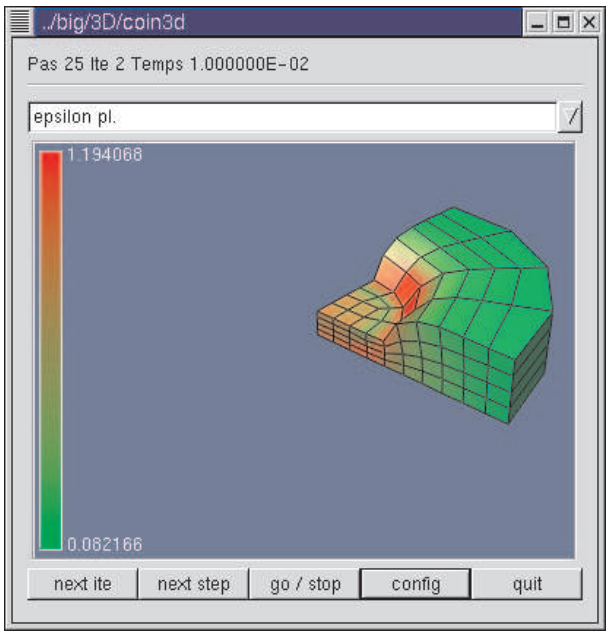

FIG. 4.8: Maillage du poinçonnement symétrique.

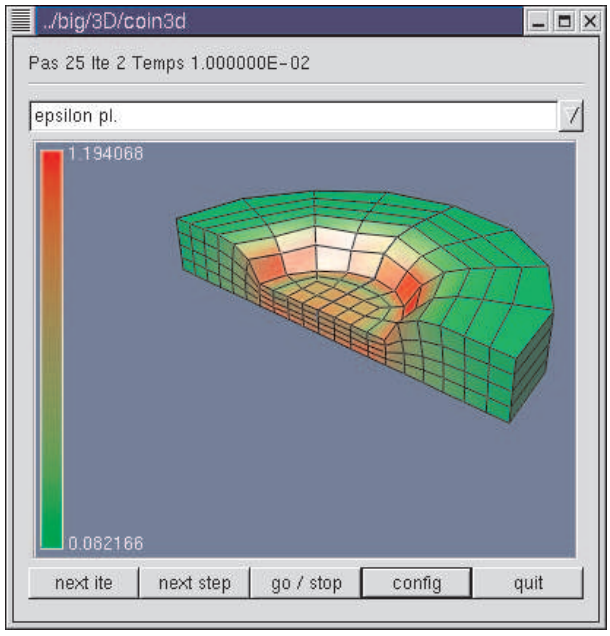

FIG. 4.9: Symétrie selon X.

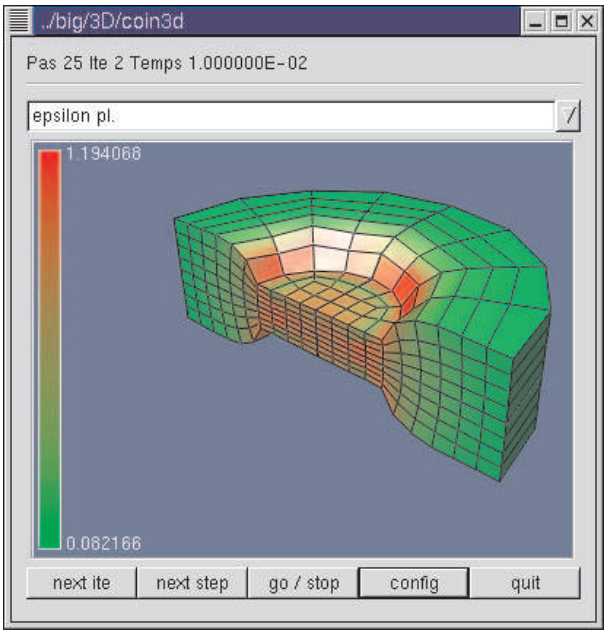

FIG. 4.10: Symétrie selon Y.

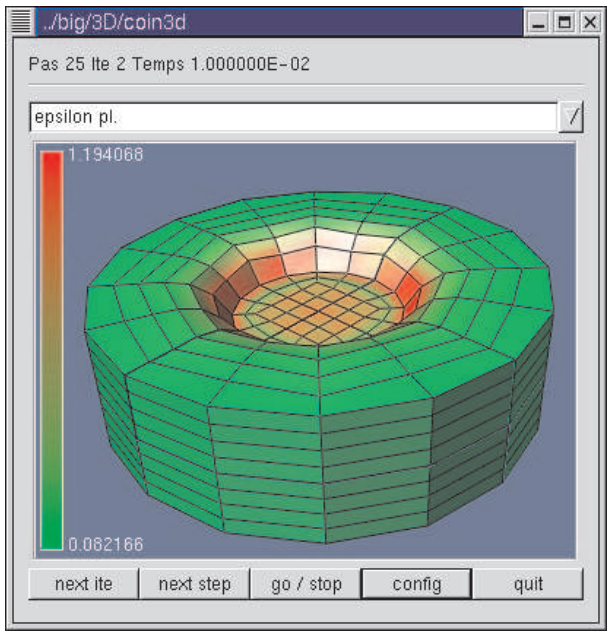

FIG. 4.11: Symétrie selon Z.

4.3.3 Perspectives

La fenêtre de visualisation que nous avons programmée est déjà très utile non seulement pour développer des algorithmes dans METAFOR et résoudre des problèmes ALE ou autres, mais aussi pour effectuer des démonstration "live" des possibilités de notre code de calcul.

Bien que nous ne travaillerons certainement plus sur ce module de visualisation qui remplit déjà maintenant tous les critères que nous trouvions indispensables, nous citons, ci-dessous, quelques améliorations possibles :

- Extension au 2D. En effet, bien que les problèmes 2D puissent être visualisés comme un problème 3D se déroulant dans un plan, la visualisation a été surtout développée dans une optique 3D. Ainsi, les fonctions de zoom et de translation ne réagissent pas comme il faudrait.
- Utilisation du mode "palette". Pour l'instant, nous utilisons un mode où toutes les couleurs sont accessibles. L'absence de palette de couleur empêche le tracé de dessins comme le fait BACON, c'est-à-dire avec des dégradés de couleurs beaucoup plus complexes. Dans le cas du mode que nous utilisons, seuls les dégradés linéaires d'une couleur à une autre sont autorisés.
- Ajout de textures. Cela permettrait de créer des rendus réalistes beaucoup plus convaincants.
- Ajout des outils en transparence. Il serait en effet intéressant de pouvoir visualiser les outils, c'est-à-dire les matrices de contact rigides, au cours du calcul. Cela permettrait de voir les problèmes relatifs au contact.
- Création d'animations automatique. Cela permettrait de générer des animations utiles pour des démonstrations sans investir beaucoup de temps.

4.4 Développements relatifs la nouvelle tête de METAFOR

4.4.1 Introduction

Notre code de calcul METAFOR a subit de très grandes modifications au cours de cette année. Toutes ces modifications ont un seul but : permettre une unification des différentes version du code un une version stable, robuste, testée, portable sur d'autres machines et incluant tous les développements effectués au sein du laboratoire.

4.4.2 Portage du code sur d'autres plate-formes

Une première étape, après un nettoyage complet de l'ensemble des routines, était la suppressions de toutes les causes qui empêchaient d'utiliser METAFOR sur d'autres machines que des stations alpha. Notre but était de faire tourner METAFOR sur un simple PC de bureau. Nous avons donc mis au point une procédure de compilation professionnelle de METAFOR basée sur des "Makefiles" assez sophistiqués. Ceux-ci sont automatiquement générés par le programme "autoconf". Ce programme permet de détecter notamment l'emplacement des différentes librairies nécessaires à METAFOR sur le système utilisé. Le développeur de METAFOR utilisant "autoconf" peut également choisir très simplement d'inclure différentes librairies facultatives du code (par exemple le module de visualisation décrit ci-dessous ou les routines de BACON).

On ne choisissant aucune des librairies facultatives, on peut théoriquement compiler METAFOR sur n'importe quel type de machine (de type UNIX) puisque le code à compiler n'est que du FORTRAN ou du C traditionnel auquel on a ajouté les fonctions mathématiques de base.

Grâce à cette compatibilité, nous pouvons maintenant faire fonctionner METAFOR sous Linux (PC) lors du développement de petits problèmes. Les stations alpha étant alors réservées au gros calculs.

4.4.3 Gestion de la paramétrisation des jeux de données

Nous voulions également permettre à l'utilisateur de pouvoir se passer du programme commercial BACON lors du développement de jeux de données. Nous n'y sommes pas encore arrivé totalement puisque BACON est toujours nécessaire pour créer et surtout mailler un problème 3D. Pour tous les problèmes 2D, nous avons, avec l'aide précieuse des chercheurs du projet METASTAMP, adapté le logiciel de pré-traitement Z-Mesh pour qu'on puisse l'utiliser pour entrer toutes les commandes de pré-traitement de BACON.

Le seul problème restant était l'absence de moyen de paramétrisation des jeux de données Z-Mesh. Nous avons donc mis au point un système d'abréviation (bien plus fiable que celui de BACON) et permettant des évaluations récursives. Ainsi, si on défini $a = 2$, $b = a$, l'évaluation de $c = a + b$ donnera la réponse attendue. Nous avons introduit plusieurs fonctions trigonométriques (sinus, cosinus,..), l'extraction de racine en plus est traditionnels opérateurs +, -, *, / et exposant.

L'utilisation des nouvelles routines ALE et la paramétrisation du jeu de données permet de concevoir des variantes des problèmes de manière immédiate (rappelons que les anciennes commandes ALE étaient très difficilement paramétrables).

4.5 Conclusions & perspectives

Nous avons dû, cette année investir une grande partie de notre temps à des tâches auxiliaires qui ne sont d'ailleurs pas toutes terminées à l'heure actuelle. Cependant, nous pensons que ce passage était obligé pour garantir que nos développements puissent être utilisés dans le futur (sur d'autres machines, par d'autres personnes). Nous pensons que cet investissement a été non seulement très bénéfique au laboratoire puisque notre code possède maintenant une présentation et une gestion professionnelle mais aussi puisque nous avons fait d'énormes progrès dans notre manière (plus rigoureuse, plus systématique et plus générale) d'aborder la programmation d'algorithmes.

Chapitre 5

Conclusions

Ce rapport résume les divers développements effectués au cours de cette troisième année de bourse FRIA (première année de seconde bourse).

L'étude des phénomènes de lubrification et l'implémentation d'un algorithme de résolution des équations du mouvement du lubrifiant couplées aux équations d'équilibres des solides en contact ont montrés que l'utilisation du formalisme Arbitraire Lagrangien Eulérien est incontournable pour obtenir des résultats numériques satisfaisants.

Or, nous trouvions que l'implémentation de ce formalisme dans notre code de calcul METAFOR ne nous convenait pas. En effet, de nombreux cas ne pouvaient pas être traités (maillages non structurés, problèmes tri-dimensionnels) et l'introduction des données pour définir les maillage, la géométrie et les paramètres d'un problème étaient très laborieuse. Ainsi, le simple fait de vouloir changer le nombre de mailles du problème nécessitait la réécriture d'une grande partie du jeu de données. Si bien que l'on passait plus de temps à mettre au point des jeux de données et les vérifier qu'à tester les algorithmes.

Nous avons aussi montré que METAFOR a évolué d'une manière phénoménale cette année grâce à une réécriture complète des routines de lecture et de pré-traitement des données. C'était donc l'occasion de reconstruire une librairie de routines ALE autour de ces nouvelles routines de pré-traitement. Grâce à ces nouvelles routines, de nombreuses informations anciennement inaccessibles sont devenues disponibles (il s'agit principalement des informations géométriques – domaines, faces, contours, ...) et nous avons pu en tirer parti pour obtenir un ensemble d'algorithmes beaucoup plus généraux que les précédents. Ainsi, nous avons fait tomber l'ensemble des hypothèses des anciennes routines et l'élaboration d'un problème ALE devient extrêmement simplifiée (il suffit de prendre un jeu de donnée lagrangien et de fixer une fréquence de remaillage).

Après un bref chapitre introductif expliquant les choix que nous avons fait en début d'année (passage d'une programmation basique en FORTRAN à une programmation structurée en C, volonté de généralisation de tous les algorithmes,...) et la logique suivie dans le

cadre de notre thèse de doctorat sur la lubrification (le formalisme ALE est indispensable), nous avons décrit, en détail dans le deuxième chapitre les nouvelles possibilités de notre nouvelle implémentation de l'ALE. Nous nous sommes attardés à montrer comment nous avons généralisé les algorithmes et comment la programmation du formalisme ALE en 3D a été une simple extension du 2D.

Le chapitre suivant a été consacré à une série de problèmes qui nous ont permis de montrer la fiabilité de nos algorithmes. Cependant, nous avons aussi insisté sur les problèmes que nous avons rencontrés au cours de nos tests et sur la manière dont nous envisageons de les résoudre dans un avenir proche.

Dans le troisième chapitre nous avons décrit les différentes taches annexes qui nous ont occupées pendant cette année. Nous avons essayé d'expliquer le plus clairement possible pourquoi il nous semblait nécessaire de se consacrer à ces tâches. Nous avons notamment présenté un outil de visualisation que nous avons programmé dans le but d'aider l'utilisateur de METAFOR et le développeur pour comprendre le déroulement d'un calcul par le formalisme ALE. Cet outil nous a permis de comprendre certaines faiblesses de nos algorithmes et nous a aidé à trouver des manières d'y remédier.

En conclusion, nous pensons que nous possédons un ensemble de routines permettant de gérer le formalisme ALE beaucoup plus efficaces et beaucoup plus robustes qu'auparavant. Les possibilités offertes sont beaucoup plus larges et permettent d'envisager des applications complexes auparavant impossibles à modéliser autrement qu'avec le traditionnel formalisme lagrangien.

Nous allons donc maintenant utiliser le résultat du travail de cette année pour continuer nos recherches dans le domaine du contact lubrifié entre solides déformables.

Annexe A

Gestion des splines et surfaces splines

A.1 Introduction

Ce chapitre décrit de manière mathématique les approximations utilisées pour générer, à partir d'une liste de points, une courbe continue de type C^1, soit, à partir d'un nuage de points, une surface composite continue. Ces développement sont à la base du remaillage unidimensionnel des courbes en 2D et 3D et du lissage des maillages surfaciques en 3D.

A.2 Splines cubiques

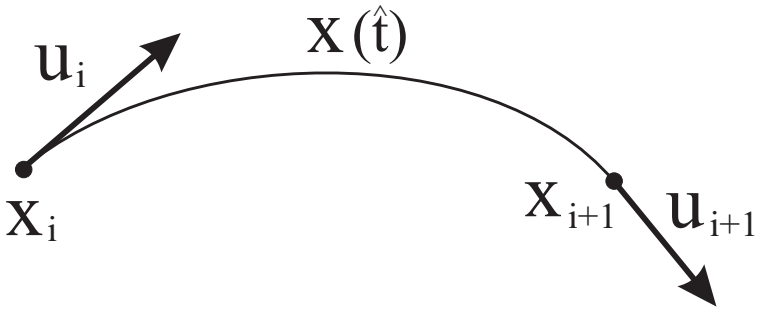

FIG. A.1: Spline cubique.

Soient N points $\mathbf{x_i}$ $(i = 1, ...N)$ sur lesquels on veut construire une courbe. Supposons également connus en tous les points le vecteur tangent $\mathbf{u_i}$ $(i = 1, ...N)$. Considérons le segment $[\mathbf{x_i}, \mathbf{x_{i+1}}]$, paramétrisé localement par $\hat{t} \in [0, 1]$. On peut construire le morceau de courbe suivant entre ces deux points :

$$\mathbf{x}(\hat{\mathbf{t}}) = \mu_{\mathbf{0}}(\hat{\mathbf{t}})\,\mathbf{x_i} + \mu_{\mathbf{1}}(\hat{\mathbf{t}})\,\mathbf{x_{i+1}} + \mu_{\mathbf{2}}(\hat{\mathbf{t}})\,\mathbf{u_i} + \mu_{\mathbf{3}}(\hat{\mathbf{t}})\,\mathbf{u_{i+1}} \tag{A.1}$$

où les μ_i sont les fonctions d'Hermite :

$$\mu_0(\hat{t}) = (1+2\,\hat{t})\,(1-\hat{t})^2 \quad (A.2)$$
$$\mu_1(\hat{t}) = (3-2\,\hat{t})\,\hat{t}^2 \quad (A.3)$$
$$\mu_2(\hat{t}) = \hat{t}\,(1-\hat{t})^2 \quad (A.4)$$
$$\mu_3(\hat{t}) = (\hat{t}-1)\,\hat{t}^2 \quad (A.5)$$

L'unicité des la tangente aux points $\mathbf{x_i}$ assure la continuité C^1 de la courbe assemblée. Si on veut maintenant travailler avec une abscisse curviligne $t \in [0, T]$ plutôt qu'avec l'abscisse réduite $\hat{t}$, on peut utiliser l'approximation de MacConalogue [23] :

$$\mathbf{x(t)} = \mu_\mathbf{0}(\frac{\mathbf{t}}{\mathbf{T}})\,\mathbf{x_i} + \mu_\mathbf{1}(\frac{\mathbf{t}}{\mathbf{T}})\,\mathbf{x_{i+1}} + \mu_\mathbf{2}(\frac{\mathbf{t}}{\mathbf{T}})\,\mathbf{T}\,\mathbf{u_i} + \mu_\mathbf{3}(\frac{\mathbf{t}}{\mathbf{T}})\,\mathbf{T}\,\mathbf{u_{i+1}} \quad (A.6)$$

où

$$T = \frac{3}{e}\left[\sqrt{f^2 + 2\,g\,e} - f\right] \quad (A.7)$$

avec

$$f = (\mathbf{x_{i+1}} - \mathbf{x_i}).(\mathbf{u_{i+1}} - \mathbf{u_i}) \quad (A.8)$$
$$g = ||\mathbf{x_{i+1}} - \mathbf{x_i}||^\mathbf{2} \quad (A.9)$$
$$e = g - \frac{1}{2}||\mathbf{u_{i+1}} + \mathbf{u_i}||^\mathbf{2} \quad (A.10)$$

Cette expression donne en fait une très bonne approximation de la longueur du segment de spline $[\mathbf{x_i}, \mathbf{x_{i+1}}]$.

Les vecteurs tangents en chaque point peuvent être obtenus par un schéma à 3 points :

$$\mathbf{u_i} = \frac{\mathbf{v_i}}{||\mathbf{v_i}||} \qquad \mathbf{v_i} = (\mathbf{1}-\alpha)\,\mathbf{\Delta x_{i-1}} + \alpha\,\mathbf{\Delta x_i} \quad (A.11)$$

avec

$$\alpha = \frac{||\Delta\,\mathbf{x_i}||^\mathbf{2}}{||\Delta\,\mathbf{x_{i-1}}||^\mathbf{2} + ||\mathbf{\Delta}\,\mathbf{x_i}||^\mathbf{2}} \quad (A.12)$$

où

$$\Delta\,\mathbf{x_i} = \mathbf{x_{i+1}} - \mathbf{x_i} \quad (A.13)$$

En début et en fin de courbe, on utilise un schéma décentré, avec :

$$\mathbf{v_0} = (\mathbf{1}-\alpha_\mathbf{0})\,(\mathbf{x_1} - \mathbf{x_0}) + \alpha_\mathbf{0}\,(\mathbf{x_2} - \mathbf{x_1}) \quad (A.14)$$

et

$$\alpha_0 = \frac{||\Delta\,\mathbf{x_0}||^\mathbf{2}}{(||\Delta\,\mathbf{x_0}|| + ||\mathbf{\Delta}\,\mathbf{x_1}||)^\mathbf{2} + ||\mathbf{\Delta}\,\mathbf{x_0}||^\mathbf{2}} \quad (A.15)$$

pour le premier noeud et

$$\mathbf{v_N} = (\mathbf{1} - \alpha_\mathbf{N})\,(\mathbf{x_N} - \mathbf{x_{N-1}}) + \alpha_\mathbf{N}\,(\mathbf{x_N} - \mathbf{x_{N-2}}) \tag{A.16}$$

et

$$\alpha_N = \frac{||\Delta\,\mathbf{x_{N-1}}||^\mathbf{2}}{(||\Delta\,\mathbf{x_{N-2}}|| + ||\mathbf{\Delta}\,\mathbf{x_{N-1}}||)^\mathbf{2} + ||\mathbf{\Delta}\,\mathbf{x_{N-1}}||^\mathbf{2}} \tag{A.17}$$

pour le dernier noeud. On peut ainsi reconstruire sur la ligne du maillage équilibré une courbe continue sur laquelle on peut repositionner les noeuds.

A.3 Surfaces splines

A.3.1 Evaluation de la surface

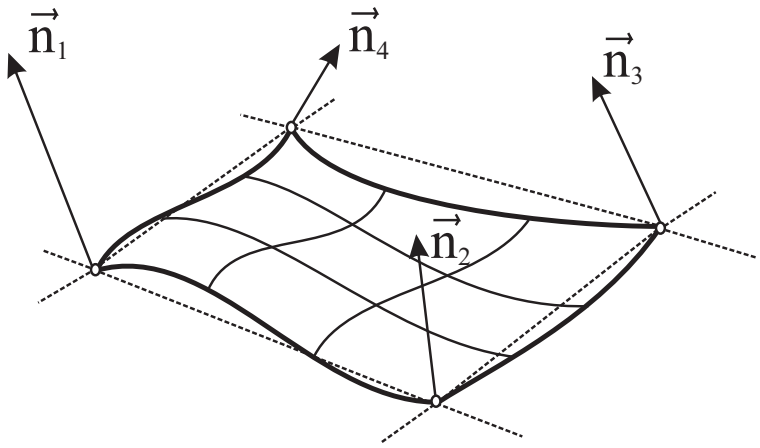

FIG. A.2: Patch formé de 4 splines cubiques.

Soit un patch bilinéaire formé à partir des 4 côtés d'un élément surfacique quadrangulaire et 4 vecteurs unitaires $\mathbf{n_i}$ associés à ses 4 noeuds, définissant la normale à la surface spline voulue (figure A.2).

L'évaluation de la surface aux coordonnées réduites (u, v) avec $u \in [0, 1]$ et $v \in [0, 1]$ se résume à :

- évaluer les côtés 1 et 3 à la coordonnée réduite u et les côtés 2 et 4 à la coordonnée réduite v par la formule A.1. Pour chaque côté, si les deux points extrêmes sont numérotés 1 et 2, l'évaluation des tangentes $\mathbf{u_1}$ et $\mathbf{u_2}$ se fait par projection du segment $\mathbf{x_2} - \mathbf{x_1}$ sur les 2 plans définis par les normales $\mathbf{n_1}$ et $\mathbf{n_2}$:

$$\mathbf{u_1} = (\mathbf{x_2} - \mathbf{x_1}) - ((\mathbf{x_2} - \mathbf{x_1}).\mathbf{n_1})\,\mathbf{n_1} \tag{A.18}$$

  Ce vecteur est ensuite normé.

- appliquer la formule d'interpolation des patches bi-linéaires :

$$\begin{aligned}\mathbf{x}(\mathbf{u},\mathbf{v}) &= (1-u)\,\mathbf{C_4}(\mathbf{v}) + \mathbf{u}\,\mathbf{C_2}(\mathbf{v}) + (\mathbf{1}-\mathbf{v})\,\mathbf{C_1}(\mathbf{u}) + \mathbf{v}\,\mathbf{C_3}(\mathbf{u}) \\ &\quad - ((1-u)(1-v)\,\mathbf{x_1} + \mathbf{u}(\mathbf{1}-\mathbf{v})\,\mathbf{x_2} + \mathbf{uv}\,\mathbf{x_3} + (\mathbf{1}-\mathbf{u})\mathbf{v}\,\mathbf{x_4}) \quad \text{(A.19)}\end{aligned}$$

L'évaluation des tangentes et des courbures s'obtient directement en dérivant l'équation (/refeq :spline :02).

Bibliographie

[1] J.L.F. Aymone. *Remanejo de malhas em problemas 3D de grandes deformaçoes*. PhD thesis, Universidade federal do Rio Grande do Sul, Brasil, 2000.

[2] H. N. Bayoumi, M. S. Gadala, and J. Wang. Numerical simulation of metal forming processes. In Huétink & Baaijens, editor, *Simulation of Materials Processing : Therory, Methods and Applications*, pages 103–108, Rotterdam, The Netherlands, 1998.

[3] D. J. Benson. Momentum advection on a staggered mesh. *Journal of Computational Physics*, 100 :143–162, 1992.

[4] R. Boman. Méthodes itératives pour la résolution de systèmes non linéaires). Technical Report Travail de fin d'études, Université de Liège, Belgium, 1997.

[5] R. Boman. Formalisme eulérien lagrangien pour le contact lubrifié entre solides (rapport d'activité FRIA 1ère bourse - 1ère année — in French). Technical Report TF-55, Université de Liège, Belgium, 1998.

[6] R. Boman. Formalisme eulérien lagrangien pour le contact lubrifié entre solides (rapport d'activité FRIA 2ème bourse - 2ème année — in French). Technical report, Université de Liège, Belgium, 1999.

[7] R. Boman. Eléments finis de lubrification pour la prise en compte du frottement dans la simulation de processus de formage des matériaux. In M. Raous P. Chabrand, M. Jean, editor, *Proceedings of MECAMAT'2000*, page /, Aussois, France, 2000.

[8] R. Boman, L. Colantonio, and J.-P. Ponthot. Effective iterative solvers for highly nonlinear & large deformation pressure dependant problems. In M. Hogge R. Van Keer, B. Verhegghe, editor, *Proceedings of ACOMEN'98*, pages 133–140, Ghent, Belgium, 1998.

[9] R. Boman, L. Colantonio, and J.-P. Ponthot. Iterative solvers for large deformation pressure dependent problems. In Huétink & Baaijens, editor, *Simulation of Materials Processing : Therory, Methods and Applications*, pages 225–231, Rotterdam, The Netherlands, 1998.

[10] R. Boman and J.-P. Ponthot. ALE methods for stationary solutions of metal forming processes. In J.A. Cavas, editor, *Second ESAFORM conference on Material Forming*, pages 585–589, Guimaraes, Portugal, 1998.

[11] R. Boman and J.-P. Ponthot. Numerical methods for the lubricated contact between solids in metal forming processes. In W. Wunderlich, editor, *Proceedings of ECCM'99*, pages on CD–Rom, Munchen, Germany, 1999.

[12] R. Boman and J.-P. Ponthot. ALE methods for determining stationary solutions of metal forming processes. In To appear, editor, *Proceedings of ECCOMAS 2000*, page /, Barcelona, Spain, 2000.

[13] R. Boman and J.-P. Ponthot. ALE update procedures for stationary solutions of metal forming processes. In UCL, editor, *Proceedings of MECA'2000, the fifth national congress on theoretical and applied mechanics*, pages 195–197, Louvain-la-Neuve, Belgium, 2000.

[14] R. Boman and J.-P. Ponthot. Arbitrary lagrangian eulerian method for the simulation of lubricated contact between solids in metal forming processes. In To appear, editor, *Proceedins of METAL FORMING 2000*, page /, Krakow, Poland, 2000.

[15] R. Boman and J.-P. Ponthot. Finite elements for the lubricated contact between solids in metal forming processes. *Acta Metallurgica Sinica (English Letters)*, 13(1) :319–327, 2000.

[16] A. N. Brooks and T. J. R. Hughes. Streamline upwind/petrov-galerkin formulations for convection dominated flows with particular emphasis on the incompressible navier-stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32 :199–259, 1982.

[17] X. Collard, S. Gohy, and L. Papeleux. Rapport metastamp. Technical report, Université de Liège, Belgium, 1998.

[18] Y-K. Hu and W. K. Liu. Simulation of ring rolling process by arbitrary lagrangian eulerian finite element method. *AMD*, 20 :225–240, 1990.

[19] A. Huerta. Ale stress update in transient plasticity problems. In E. Hinton D.R.J. Owen, E. Onate, editor, *Computational Plasticity*, pages 1865–1871. 1995.

[20] A. Huerta, F. Casadei, and J. Donéa. An arbitrary lagrangian-eulerian stress update procedure for coining simulations. In Wood & Zienkiewicz Chenot, editor, *Numerical Methods in Industrial Forming Processes*, pages 261–266, Valbonne, France, 1992.

[21] H. Huétink. *On the Simulation of Thermo-Mechanical Forming Processes. a Mixed Eulerian-Lagrangian Finite Element Method.* PhD thesis, University of Twente, The Netherlands, 1986.

[22] J. Huétink and P. T. Vreede. Progress in mixed eulerian-lagrangian finite element simulation of forming processes. *International Journal of Numerical Methods in Engineering*, 30 :1441–1457, 1990.

[23] D.J. MacConalogue. The quasi-intrinsic scheme for passing a smooth curve through a discrete set of points. *Computer Journal*, 13, 1970.

[24] N. Marsault. *Modélisation du régime de lubrification mixte en laminage à froid (in French).* PhD thesis, Ecole nationale supérieure des mines de Paris, France, 1998.

[25] L. Papeleux, J.-P. Ponthot, S. Gohy, and X. Collard. Numerical simulation of spring-back in sheet metal forming using linear shell element. In E. Onate M. Papadrakakis, A. Samartin, editor, *Computational methods for shell and spatial structures*, pages on CD–rom, Athens, Greece, 2000.

[26] J. Peraire and J. Peiro. Adaptive remeshing for three-dimensional compressible flow computations. *Journal of Computational Physics*, 103(2) :269–285, 1992.

[27] J-P. Ponthot. *Traitement unifié de la Mécanique des Milieux Continus solides en grandes transformations par la méthode des éléments finis (in French)*. PhD thesis, Université de Liège, Liège, Belgium, 1995.

[28] S. Potapov. *Un algorithme ALE en dynamique rapide basé sur une approche mixte éléments finis – volumes finis*. PhD thesis, Ecole Centrale de Paris, France, 1997.

[29] D. Quoirin. *Modélisation des grandes déformations de corps minces. Application à la mise à forme*. PhD thesis, Université de Liège, Belgium, 1996.

[30] A. Rodriguez-Ferran, F. Casadei, and A. Huerta. Ale stress update for transient and quasistatic processes. *International Journal of Numerical Methods in Engineering*, 42 : ?– ?, 1998.

[31] D. Rozenwald. *Modélisation thermomécanique des grandes déformations. Application aux problèmes de mise à forme à haute température, aux élastomères et aux structures mixtes acier-élastomères*. PhD thesis, Université de Liège, Belgium, 1996.

[32] L. Stainier. *Modélisation numérique du comportement irréversible des métaux ductiles soumis à grandes déformations avec endomagement*. PhD thesis, Université de Liège, Belgium, 1996.

[33] T. Yamada and F. Kikuchi. An arbitrary lagrangian-eulerian finite element method for incompressible hyperelasticity. *Computer Methods in Applied Mechanics and Engineering*, 102 :149–177, 1993.