MOHAMED MOUBARACK TOUKOUROU

Modélisation et simulation par la MEF du contact avec frottement dans les procédés de mise en forme des métaux

Mémoire
présenté
à la Faculté des études supérieures
de l'Université Laval
pour l'obtention
du grade de maître ès sciences (M.Sc.)

Département de génie mécanique
FACULTÉ DES SCIENCES ET GÉNIE
UNIVERSITÉ LAVAL

DÉCEMBRE, 2000

RÉSUMÉ

Cette étude est principalement basée sur une approche originale du problème particulier de la modélisation par éléments finis du contact avec frottement rencontré dans le domaine de la mise en forme de produits métalliques. Ce phénomène particulièrement sensible dans divers appareils de transformations tels que les presses, est appliqué ici au cas de matériaux hyperélastiques.
Les outils nouveaux permettant de traiter le problème sont le Visual C++ et Diffpack. Diffpack, qui est un logiciel orienté objet, présente une capacité à réduire le temps de programmation et de débuggage par l'existence d'outils puissants tels que des abstractions avancées et un environnement graphique intéressant. Ces outils ont permis d'une part, l'implantation de la loi de comportement hyperélastique compressible, applicable en 3D et en déformation plane, et incompressible, applicable en 3D, en déformation plane et en contrainte plane, suivant le modèle neo-Hookéen et d'autre part, du contact 2D pour l'élément Q4 et du contact 3D pour l'élément H8.

ABSTRACT

This study is mainly based on an original approach of the specific problem of finite element modelisation of frictional contact usually met in the metal forming field. This particular phenomenon, actually present in transforming machines like presses, is applied here for hyperelastic materials.
The new tools used to treat the problem are Visual C++ and Diffpack. Diffpack, which is an object-oriented software, present the capacity to reduce the time to write and debug programs with the existence of powerful tools such as advanced abstractions and an interesting graphical environment. Those tools made possible, on one hand the implementation of the hyperlastic constitutive law for compressible materials, applicable in 3D and plane strain, and incompressible material, applicable in 3D, plane strain and plane stress, with the neo-Hookean formulation, on the other hand, the implementation of 2D contact with element Q4 and 3D contact with element H8.

M. Moubarack TOUKOUROU Augustin GAKWAYA. Ph.D., Ing.

AVANT-PROPOS

J'aimerais remercier très sincèrement mon directeur de mémoire, Monsieur Augustin Gakwaya, pour sa grande disponibilité et son continuel encouragement tout au long de ce travail. Ses conseils et son immense expérience m'ont permis d'acquérir les connaissances théoriques indispensables au développement de mes aptitudes d'analyse en éléments finis et de programmation de simulations numériques par éléments finis.

Je ne saurais oublier toutes les personnes proches - Amir Yazdani, Kamil Mourad – qui ont permis la réalisation de ce travail de par leurs conseils et encouragements, et l'équipe du Département de Génie Mécanique qui m'a encadré tout au long de mes travaux.

Je voudrais pouvoir dédier ce travail à deux personnes chères à qui je dois ma venue sur cette terre, mon père et ma mère.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES SYMBOLES

t	Temps
T	Température
$\mathbf{t}_N$	Force de contact normal
$\mathbf{t}_T$	Force de contact tangentiel
m	Tenseur métrique
b	Tenseur de courbure
ε_N	Coefficient de pénalité normal
ε_T	Coefficient de pénalité tangentiel
ė	La densité de l'énergie interne spécifique
F	Force
$\vec{f}_s(\vec{n})$	Vecteur force de surface
$\vec{f}_v(\vec{x},t)$	Vecteur force de volume
μ_1, μ_2	Coefficients de frottement
u	Vecteur déplacement
R, r	rayons de courbure
P, p	Pression
q_0, q	Pression répartie
δ	Delta de Kroneker
λ et μ	Coefficients de Lamé
i	Tenseur identité de quatrième ordre
ρ	Masse volumique
κ	Module volumétrique
σ	Contrainte de déformation de Cauchy
$\vec{x}$	Vecteur position

$\dot{\vec{x}}$	Vecteur vitesse
$\vec{i}, \vec{j}$ et $\vec{k}$	Vecteurs directeurs
X, Y et Z	Coordonnées cartésiennes
$\vec{n}$	Direction normale
$C(t) = C$	Configuration courante
ds	élément de surface
dv	élément de volume
C^0	Configuration initiale
F	Gradient de déformation
P	Tenseur des contraintes Piola-Kirchhoff conjugué
Ψ	Potentiel élastique ou fonction d'énergie de déformation
S_u	Contour imposé en déplacement
S_f	Contour imposé en force
$\mathbf{K}_T$	Matrice tangente
I	Tenseur de l'identité
d	Différentielle
∂	Dérivée partielle
D()[u]	Dérivée directionnelle en direction u
$\vec{g}$	Gravité
g_N	Fonction de séparation normale
g_T	Fonction de séparation tangentielle
Ω	Contour d'un solide
$\partial\Omega$	Partie du contour d'un solide
B	Intérieur d'un solide
C	Tenseur de Cauchy-Green droit
C	Tenseur élastique matériel
$\hat{\mathbf{C}}$	Composante déviatorique de **C**
$\mathbf{C}_P$	Composante de pression de **C**
$\mathbf{C}_\kappa$	Composante volumétrique de **C**

ς	Tenseur élastique spatial ou Eulérien
δW	Travail virtuel
$\mathbf{I_C}, \mathbf{II_C}, \mathbf{III_C}$	Les trois invariants de $\mathbf{C}$
$\mathbf{B}, \mathbf{b}$	Tenseur de Cauchy-Green gauche
$\mathbf{E}$	Tenseur des déformations de Green-Lagrange
$\mathbf{A}$	Tenseur des déformations d'Almansi
$\mathbf{R}$	Tenseur de rotation pure correspondant au mouvement de corps rigide
$\mathbf{U}$	Tenseur de déformation pur droit
$\mathbf{V}$	Tenseur de déformation pur gauche
λ	Valeurs propres
$\bar{\mathbf{N}}$	Directions principales
$\mathbf{L}$	Tenseur Eulérien
$\mathbf{W}$	Tenseur taux de rotation
$\boldsymbol{\pi}$	Premier tenseur (non symétrique) de Piola-Kichhoff
$\mathbf{S}$	Second tenseur de Piola-Kirchhoff
$\mathbf{S}'$	Composante déviatorique du second tenseur de Piola-Kichhoff

CHAPITRE I

INTRODUCTION GÉNÉRALE

1.1 Introduction

Les procédés de mise en forme de produits métalliques peuvent être considérés comme un ensemble d'opérations simples au cours desquelles une matière première en vrac(poudre) ou préfinie (tôle ou barre) est déformée plastiquement en une forme donnée. Cependant, à cause des variations dans les conditions aux limites telles que : effet de contact ou de lubrification, effet de bord libre ou encastré, ou des variations dans les propriétés matérielles et dimensions de la pièce, une prédiction précise et une répétitivité du même procédé pour reproduire la même forme et le même état de matériau représentent une tâche très difficile. Une opération réussie de formage exige la définition d'un ensemble de paramètres qui incluent:

- la conception de l'outillage (matrices, poinçons),
- la forme initiale de la pièce (blank),
- la conception du système de fixation de la pièce et d'application de la charge,
- le choix des propriétés matérielles de la tôle et des conditions limites sur le pourtour de la pièce.

Bien que ces procédés de formage soient répandus dans l'industrie, la conception de l'outillage et la sélection du matériau font encore largement usage de procédures très dispendieuses d'essais-et-erreurs et basées sur l'expérience du concepteur; l'utilisation d'un outil de modélisation et d'analyse fiable décrivant au mieux les processus physiques impliqués peut contribuer efficacement à la réduction des coûts et à un meilleur contrôle du procédé réel de fabrication.

L'utilisation des techniques de conception assistée par ordinateur (CAO) et ingénierie assistée par ordinateur (IAO) permet de raccourcir le cycle conception-fabrication en automatisant et en éliminant certaines étapes intermédiaires: automatisation du design et élimination de certains prototypes pour l'analyse de comportement. En décrivant la procédure de chargement de façon incrémentale, les techniques de modélisation paramétrique et de simulation avancée permettent de décrire alors à l'aide d'une suite d'opérations discrètes, la façon dont une composante est formée. Les calculs se poursuivent alors jusqu'à ce que la forme finale est obtenue ou jusqu'à ce que des conditions imposées non satisfaisantes (comme le déchirement (tearing), le rétrécissement localisé...) soient indiquées.

Actuellement, pour le traitement de géométries simples (pièces axi-symétriques ou en déformations planes) cette approche a donné d'excellents résultats. Cependant, la simulation du formage de composantes de forme quelconque est encore un sujet ouvert car de nombreux problèmes tant théoriques que numériques restent encore à formuler et à résoudre, et le développement d'un logiciel général de simulation représente encore un défi réel. En effet, plusieurs aspects physiques et géométriques non linéaires importants apparaissent en cours de fabrication :

(1) les procédés de mise en forme de produits métalliques font en général intervenir des grands déplacements, grandes rotations et des grandes déformations au cours de leur processus de déformations en leur forme finale;
(2) les matériaux utilisés peuvent développer un comportement élasto-plastique avec anisotropie initiale ou induite;

(3) la pièce à fabriquer est généralement en contact avec la matrice et le poinçon et on doit donc décrire correctement les conditions de contact unilatéral et de frottement;

(4) le contact avec frottement induit l'usure des outils ce qui peut causer une perte d'efficacité de production et de qualité des produits.

Dans ce projet, nous proposons de développer et d'appliquer la méthode des éléments finis aux problèmes de contact dans un environnement éléments finis orienté objet avec comme outil de base l'environnement Diffpack qui est un logiciel en C++.
Une première validation des algorithmes implantés pourra être réalisée grâce au logiciel FLAGSHYP pour la loi de comportement hyperélastique, puis une validation des algorithmes de contact sera fait grâce aux résultats analytiques de Hertz pour le contact *normal*.

Dans le cas où les techniques CAO/IAO doivent être appliquées à un système existant (modélisation des outils) des travaux de rétro-ingénierie pourront être effectués en utilisant, par exemple, une machine CMM (pour générer le modèle géométrique des outils). De plus, comme aujourd'hui, on est capable d'identifier en ligne, à partir de mesures expérimentales, des paramètres de la loi de comportement élastoplastique des matériaux usinés, alors une combinaison de modèles numérique et expérimental est fortement conseillée en vue d'acquérir une meilleure connaissance du procédé et de mieux le contrôler. La section suivante présente l'évolution de l'étude des problèmes de contact à travers les approches de solutions.

1.2 Problèmes de contact

Les études sur les problèmes de contact datent de plusieurs centaines d'années et leur évolution peut être divisée en trois étapes, qui sont décrites dans les paragraphes suivants.

1.2.1 Études classiques des problèmes de contact

Initialement, les corps en contact étaient restreints aux corps rigides ou élastiques, en plus, seul les phénomènes globaux tels que les forces totales de contact étaient prises en compte. La troisième loi de Newton et la loi de frottement de Coulomb peuvent être considérées comme étant les résultats majeurs à cette étape. Les résultats fondamentaux développés à cette étape sont encore utilisés aujourd'hui dans certaines analyses. Comme exemple, considérons le problème décrit à la Fig. 1.1(a). Un bloc solide élastique A repose sur un autre bloc solide élastique B qui repose à son tour sur un sol rigide C. Une force P est appliquée sur le bloc A. Pour étudier son comportement global, ce simple

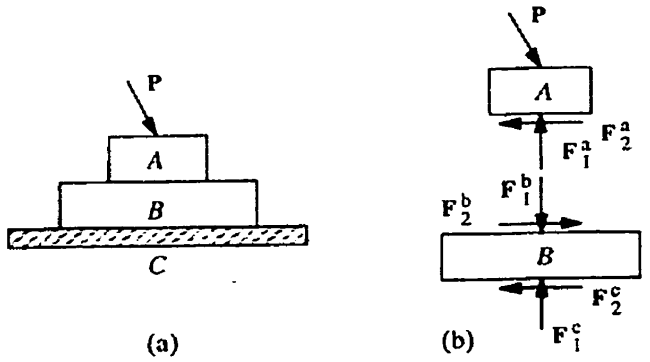


Figure 1.1 (a)Un système de contact avec 2 corps élastiques et un sol rigide, (b)Décomposition du système de contact en corps libre

système de contact peut-être décomposé en construisant des corps libres comme illustré à la Fig. 1.1(b), où les forces de contact et les forces de friction sont sujettes aux conditions de contact suivantes :

$$F_1^b = F_1^a, \qquad F_2^b = F_2^a \qquad \textit{(La troisième loi de Newton)}$$

$$F_2^a \le \mu_1 F_1^a$$

$$F_2^c \le \mu_2 F_1^c \qquad \textit{(La loi de frottement de Coulomb)}$$

où μ_1 and μ_2 sont respectivement les coefficients de frottement entre A et B et entre B et C.

Dans cet exemple, seule la force totale de contact et la force totale de frottement sont considérées, sans tenir compte de comment ces forces sont actuellement distribuées sur les surfaces de contact. Bien que cette procédure peut être utilisée pour analyser le comportement global des blocs pour certaines fins, ceci n'est pas satisfaisant pour d'autres, comme lors de l'étude de la détérioration du bloc due à la fatigue, qui demande la connaissance des distributions des contraintes de contact sur les surfaces en contact.

Le développement technologique de la Mécanique et l'augmentation des activités d'ingénierie ont permis et ont accru l'intérêt pour l'étude de problèmes mécaniques de plus en plus complexes. Des phénomènes locaux, comme les distributions de contraintes sur des surfaces en contact, ont commencé à être étudiés. Parmi tant d'autres, Hertz était un leader scientifique dans ce domaine. Son étude d'un problème de contact statique en élasticité en 1880 [Ref-24], [Ref-25] a connu un succès remarquable, et est vu comme une étape importante dans le domaine. Dans son étude, Hertz a supposé que des corps en contact peuvent être considérés comme des demi-espaces élastiques avec des petites déformations et que les surfaces en contact sont petites et, en général, elliptiques. Il a également supposé que les frontières de contact sont sans frottement.

Considérons un exemple d'un problème de contact classique de Hertz. Deux sphères élastiques en contact sont illustrées à la Fig. 1.2(a)). Selon la théorie de contact de Hertz, les déplacements des frontières de contact devraient satisfaire à la condition suivante [Ref-26]:

$$u_3^1 + u_3^2 = g - (1/2R)(r)^2 \tag{1.1}$$

où u_3^1 et u_3^2 sont les composantes respectives de déplacement dans la direction x_3 des frontières de contact des deux sphères, g est la séparation initiale des deux frontières de contact, $1/R = (1/R_1 + 1/R_2)$ où R_1 et R_2 sont les rayons de courbure respectifs des deux sphères et r est la distance entre un point donné de contact et le centre de la zone de contact, qui est un cercle dans ce cas.

À partir de l'équation (1.1), la distribution de la pression de contact sur l'aire de contact peut s'exprimer comme suit :

$$q = q_0\left[1-(r/r_0)^2\right]^{1/2} \tag{1.2}$$

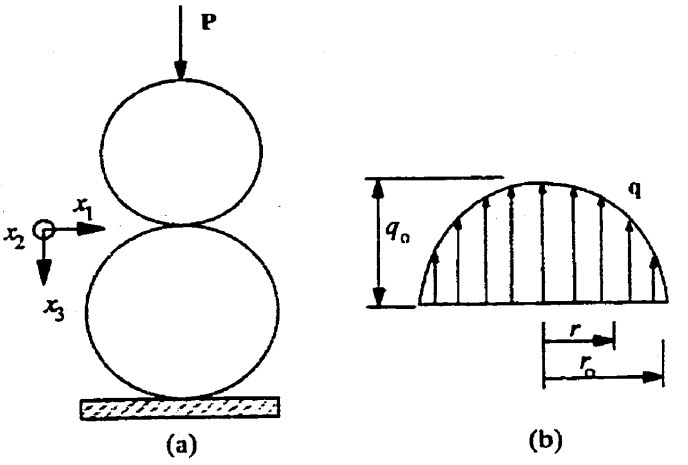


Figure 1.2 (a)Deux sphères en contact, (b)Distribution de la pression de contact entre deux sphères suivant la théorie de contact de Hertz

Où r_0 est le rayon de courbure de la zone de contact et q_0 est la pression maximum au centre. La distribution de pression dans l'équation (1.2) est illustrée à la Fig. 1.2(b).

Grâce à Hertz, l'intérêt pour les problèmes de contact s'est accru, c'est ainsi que Gladwell [Ref-27] et Johnson [Ref-26] se sont intéressés au domaine. Goldsmith [Ref-28] a quant à lui étudié les problèmes d'impact à travers des approches analytiques et expérimentales. Un point commun dans ses études est que l'on suppose que la géométrie et la déformation d'un corps en contact se comporte de façon à ce que des outils mathématiques et mécaniques puissent être utilisés afin d'obtenir une solution assez proche au problème. On pourrait classer ces études dans la seconde étape. Les développements effectués durant cette étape sont évidemment très restrictifs et ne peuvent s'appliquer qu'à des problèmes spécifiques.

1.2.2 Études des problèmes de contact avec les éléments finis

La troisième étape est axée sur l'étude numérique, c'est ainsi que diverses méthodes numériques ont été utilisées pour résoudre des problèmes de contact. Parmi ces méthodes, la méthode des éléments finis a été largement la plus utilisée. Avec la méthode des éléments finis (Voir Zienkiewicz [Ref-29], [Ref-30], [Ref-31]), Oden [Ref-32], Bathe [Ref-33], Hughes [Ref-34] et Zienkiewicz et Taylor [Ref-35]), les corps en contact sont modélisés par un ensemble d'éléments finis et les frontières de contact sont approximées par des ensembles de polygones. Les corps en contact peuvent avoir des géométries complexes et être caractérisés par les propriétés de leurs matériaux et de plus, ils sont libres de se déformer de façon arbitraire. La résolution des problèmes de contact revient ainsi à la résolution de systèmes d'équations algébriques au lieu d'application de solutions approximatives.

Sans rentrer dans les détails relatifs à la résolution d'un problème de contact par la méthode des éléments finis, les éléments suivants constituent la base nécessaire à une telle tâche :

1. Formulation variationnelle : permet d'établir la base pour la discrétisation pour les éléments finis.
2. Formulation élémentaire : permet de calculer les contributions élémentaires à la matrice de rigidité et au vecteur résidu.
3. Modélisation du matériau : permet d'évaluer la relation contraintes-déformations et joue un rôle important dans la formulation élémentaire.
4. Loi de frottement : elle gouverne les forces de frottements (contraintes) entre interfaces de contact, lorsque les effets de frottement sont non négligeables.
5. Méthode des contraintes de contact : permet d'évaluer les forces inconnues de contact lorsqu'il y a contact.
6. Algorithme de recherche du contact : effectue la recherche de nœuds potentiels de contact et détermine les positions des nœuds de contact de façon précise et effective.
7. Méthode d'intégration temporelle : permet d'intégrer dans le domaine du temps.

Dans le cas général, les problèmes de contact avec des géométries non linéaires font appel à des méthodes spécifiques permettant de prendre en compte la non linéarité du problème, c'est ainsi qu'on distingue :

8. La méthode de linéarisation itérative : qui transforme un problème géométrique non linéaire en une série de problèmes linéaires géométriques.

De façon générale, cette méthode, qui est celle choisie dans nos travaux, est maintenant la méthode la plus populaire pour résoudre les problèmes de contact. D'autres méthodes numériques peuvent aussi être utilisées pour résoudre certains problèmes de contact. Par exemple, la méthode des éléments de frontière a été utilisée pour étudier des problèmes de contact par Andersson [Ref-36] et Batra [Ref-37].

Grâce à la méthode des éléments finis, des problèmes complexes sont résolubles sans restrictions, en principe, nécessaires sur la géométrie et les propriétés des matériaux. La section suivante présente les objectifs de la thèse qui tournent autour de l'applicabilité de cette méthode à la modélisation du contact pour des matériaux hyperélastiques.

1.3 Objectifs de la thèse

Les objectifs de nos travaux concernent la formulation théorique, l'implémentation numérique, et la mise au point informatique de procédures de calcul pour la modélisation et la simulation par la méthode des éléments finis (MEF) du contact avec frottement dans les procédés de mise en forme des métaux.

En effet, depuis plusieurs années (en 1998), le laboratoire de Conception et Fabrication Assistée par Ordinateur de l'Université Laval s'est engagé dans un programme intensif de recherche et de développement visant à améliorer les outils de simulation numérique existant afin d'être capable d'analyser le comportement de corps solides hyperélastiques subissant des grandes déformations au cours de leur mise en forme. Le travail présenté se situe dans cette continuité et a pour but la réalisation d'un prototype de

logiciel d'aide à la conception de la mise en forme capable de prédire la distribution des déformations et des contraintes sur le produit final. Les objectifs recherchés sont donc :

- Implantation du modèle de comportement hyperélastique, pour la modélisation des comportements mécaniques envisagés, dans un code d'éléments finis
- Implantation et validation des algorithmes de contact 2D
- Implantation et validation des algorithmes de contact en 3D avec la technique de résolution utilisant la méthode des pénalités
- Validation par des simulations de différents cas académiques et de cas réels en formage.

1.4 Résumé des différents chapitres

Cette thèse est organisée en cinq chapitres (outre l'introduction et la conclusion) qui présentent les différents aspects de la simulation et de la modélisation numérique de la mise en forme des corps solides hyperélastiques.

Le chapitre 2 présente des aspects associés à la cinématique des grandes déformations et fait un rappel sur les principes et les concepts qui permettent le calcul de la déformation d'un milieu continu solide sous l'action des forces extérieures. Une attention particulière sera portée au calcul des différentes mesures des contraintes et des déformations employées. Les équations du mouvement et les propriétés générales que doivent satisfaire les relations contraintes-déformations d'un matériau sont également brièvement commentées.

Le chapitre 3 est consacré à l'étude de la loi de comportement des solides hyperélastiques en grandes déformations abordés lors des travaux de recherche. Après une présentation du tenseur de l'élasticité nous abordons le cas de l'hyperélasticité isotropique à travers sa description matérielle et spatiale. Puis, nous abordons le cas des matériaux incompressibles et presqu'incompressibles. Nous développerons dans notre simulateur le cas des modèles Néo-Hookéens.

Dans le chapitre 4, nous présentons les lois de contact et de frottement les plus utilisées dans les modèles numériques de simulation de mise en forme ainsi que la

résolution des équations d'équilibre avec les inéquations de contact et de frottement. La suite de ce chapitre sera consacrée à l'adoption et au développement des algorithmes de recherche automatique des zones de contact, constituant des difficultés importantes dans la simulation des procédés.

Le chapitre 5 est consacré à l'étude de la programmation des éléments finis dans Visual C++ avec Diffpack. Les avantages de la programmation orientée objet sont présentées avec un développement des outils de programmation des éléments finis disponibles dans Diffpack à travers les classes FEM, NonLinEqSolverUDC et GridFE utiles à la compréhension de la philosophie derrière Diffpack. Nous expliquons comment développer un solveur non linéaire dans Diffpack en présentant les étapes nécessaires à la réalisation d'un tel objectif.

Le chapitre 6 a pour objectif le développement de la simulation numérique aussi bien pour un corps hyperélastique soumis à des conditions limites variées que pour des corps en contact. Ainsi, premièrement, une étude puis une comparaison est faite sur le simulateur **Hyperelasticity** développé dans Diffpack comparativement au logiciel Flagshyp développé dans [Ref-10] avec une formulation spatiale.
Deuxièmement, une étude puis une validation est faite sur le simulateur Contact développé dans Diffpack. Les résultats obtenus à travers divers exemples sont comparés à ceux donnés par des résultats théoriques de Hertz et par des références diverses afin de valider le code.

Finalement le bilan des différents travaux réalisés ainsi que les perspectives ouvertes par ce travail sont présentés en conclusion.

CHAPITRE II

THÉORIE DES GRANDES DÉFORMATIONS

2.1 Introduction

Dans ce chapitre, nous rappelons les principes de base et les concepts qui permettent de calculer la déformation d'un milieu continu solide sous l'action des forces extérieures. Pour analyser la déformation d'un corps solide, il faut définir une mesure des déformations que le corps subit et pouvoir en déduire, par une loi appropriée caractérisant le matériau, l'état de contrainte correspondant. Les principes de conservation de la physique permettent alors de relier l'évolution de ces contraintes à celles des forces appliquées.

2.2 Cinématique des grandes déformations

La théorie des déformations est en fait celle des grands déplacements et des grandes rotations. Lorsque apparaissent dans un solide de grands déplacements et de grandes déformations, il n'est pas possible de confondre les configurations déformée et non déformée comme on le fait couramment en élasticité classique.

Une attention particulière doit être portée à la cinématique, au repérage des particules et au choix des mesures des déformations et des contraintes à adopter. La description cinématique

que nous utilisons dans la suite adopte le point de vue Lagrangien : le mouvement d'une particule matérielle est repéré à partir d'une position de référence fixée dans la configuration initiale, et correspondant généralement à l'état non déformé.

Autrement dit, toutes les variables cinématiques seront rapportées à cette configuration. Ces aspects sont développés dans de nombreux ouvrages de mécanique [Ref-38], [Ref-39], [Ref-40].

2.2.1 Description du mouvement

Sous l'action de différentes sollicitations mécaniques (statique et/ou dynamique), un corps solide occupe différentes positions dans le temps et dans l'espace. La configuration $C(t)$ représente l'état du solide à l'instant courant t. Cette configuration peut être caractérisée par un ensemble de variables cinématiques et mécaniques telles que :

- Les positions géométriques, les vitesses et accélérations des points matériels du solide $\left(\vec{x}(t), \dot{\vec{x}}(t), \ddot{\vec{x}}(t)\right)$.
- La masse volumique $\rho(\vec{x},t)$ en tout point.
- Les contraintes $\sigma(\vec{x},t)$ des points matériels.

La description Lagrangienne définit ces variables en fonction des positions $\vec{x}^0$ des particules dans une configuration choisie, dite de référence C^0.

$$\vec{x}(t) = \vec{x}\left(\vec{x}^0, t\right) \tag{2.1}$$

Nous admettons que cette configuration est exempte de contraintes et de déformations et qu'elle est non chargée, mais il peut arriver que le solide soit soumis à des contraintes initiales non nulles (résultant du procédé de fabrication par exemple).

Soit donc un solide qui se déplace dans un système de coordonnées cartésiennes Fig. 2.1. Le point P^0 de ce solide dans la configuration de référence C^0 (de coordonnées x^0, y^0, z^0) devient le point P (de coordonnées (x, y, z) dans la configuration $C(t)$. Nous avons la relation où $\vec{u}$ est le vecteur déplacement de composantes $u, v,$ et w :

$$\vec{x} = \vec{x}^0 + \vec{u}(\vec{x}^0) \tag{2.2}$$

Pour chaque configuration, nous choisissons un système de coordonnées cartésiennes X, Y et Z associées à des vecteurs unitaires $\vec{i}, \vec{j}$ et $\vec{k}$ (Fig. 2.1). Ce choix de repère et de coordonnées n'est pas le plus général cependant, à cause des cas concernant les coques par exemple.

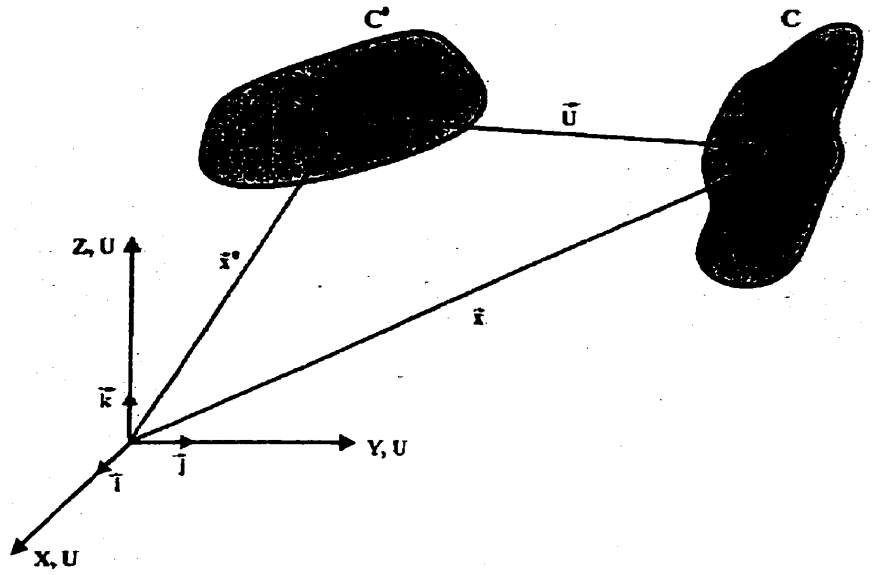


Figure 2.1 Évaluation des points matériels de la configuration initiale C^0 à la configuration actuelle $C(t) = C$

2.2.2 Gradient de déformation

Un élément différentiel $d\vec{x}^0$ dans la configuration C^0 se transforme en $d\vec{x}$ dans la configuration $C(t)$. Ces deux éléments différentiels sont reliés entre eux par la relation suivante :

$$d\vec{x} = \mathbf{F} d\vec{x}^0 \tag{2.3}$$

où $\mathbf{F}$ est le tenseur gradient de déformation, également appelé application linéaire qui permet de passer de la configuration initiale C^0 à une configuration actuelle $C(t)$. La formule (2.3) donne en effet la loi de transformation du vecteur matériel $d\vec{x}^0$ représentant le voisinage de x^0 en $d\vec{x}$ dans le voisinage du point courant x.

Les composantes cartésiennes des vecteurs $d\vec{x}$, $d\vec{x}^0$ et du tenseur gradient de déformation s'écrivent respectivement :

$$\langle dx \rangle = \langle dx, dy, dz \rangle, \qquad \langle dx^0 \rangle = \langle dx^0, dy^0, dz^0 \rangle \tag{2.4}$$

et d'après (2.2), on a :

$$[\mathbf{F}] = [\mathbf{I}] + \left[\frac{\partial \vec{u}}{\partial \vec{x}^0}\right] = \begin{pmatrix} 1+\frac{\partial u}{\partial x^0} & \frac{\partial u}{\partial y^0} & \frac{\partial u}{\partial y^0} \\ \frac{\partial v}{\partial x^0} & 1+\frac{\partial v}{\partial y^0} & \frac{\partial v}{\partial z^0} \\ \frac{\partial w}{\partial x^0} & \frac{\partial w}{\partial y^0} & 1+\frac{\partial w}{\partial z^0} \end{pmatrix} \tag{2.5}$$

L'élément de volume dv^0 défini sur la configuration C^0 se transforme au cours de la déformation et devient dv dans la configuration $C(t)$ avec :

$$dv = \det \mathbf{F} dv^0 = J dv^0 \tag{2.6}$$

Si $\vec{n}^0$ est un vecteur unitaire normal à la surface ds^0 définie sur la configuration de référence C^0, on peut trouver la relation suivante dans la configuration déformée $C(t)$:

$$\vec{n} ds = J \mathbf{F}^{-T} \vec{n}^0 ds^0 \tag{2.7}$$

où $\vec{n}$ est le vecteur unitaire normal à la surface ds dans la configuration $C(t)$.

Le tenseur gradient de déformation, s'il est défini entre deux configurations quelconques C^i et C^j est noté $\mathbf{F}_i^j$. Mais pour simplifier $\mathbf{F}_0^j$ est noté $\mathbf{F}$.

Entre plusieurs configurations C^0, C^1, C^2, le tenseur gradient de déformations vérifie la propriété de composition obtenue par dérivations en chaîne (ou successives) :

$$\mathbf{F}_i^J = \mathbf{F}_k^J \mathbf{F}_i^k \tag{2.8}$$

On admet que l'application linéaire tangente est bi-univoque, et donc que $\mathbf{F}^{-1}$ existe :

$$d\vec{x}^0 = \mathbf{F}^{-1} d\vec{x} \tag{2.9}$$

2.2.3 Description des déformations

Pour définir les déformation des solides, c'est-à-dire caractériser ses changements de forme, il faut caractériser les variations des produits scalaires qui comprennent les variations de longueurs et les variations d'angles.

En considérant les vecteurs matériels $d\vec{x}^0$ et $\delta\vec{x}^0$ et leurs homologues après déformation $d\vec{x}$ et $\delta\vec{x}$, nous pouvons écrire :

$$d\vec{x} \cdot \delta\vec{x} = d\vec{x}^0 \mathbf{F}^T \mathbf{F} \delta\vec{x}^0 = d\vec{x}^0 \mathbf{C} \delta\vec{x}^0 \quad \text{soit} \quad \mathbf{C} = \mathbf{F}^T \mathbf{F} \tag{2.10}$$

où $\mathbf{C}$ est appelé tenseur des dilatations ou tenseur de déformation de Cauchy Green droit

Inversement, il est possible de décrire la déformation dans le repère actuel. Nous avons :

$$d\vec{x}^0 \cdot \delta\vec{x}^0 = d\vec{x} \mathbf{F}^{-T} \mathbf{F}^{-1} \delta\vec{x} = d\vec{x} \mathbf{B}^{-1} \delta\vec{x} \quad \text{où} \quad \mathbf{B} = \mathbf{F}\mathbf{F}^T \tag{2.11}$$

et $\mathbf{B}$ est le tenseur de déformation de Cauchy Green gauche.

On peut également définir les deux tenseurs des déformations de Green-Lagrange $\mathbf{E}$ et d'Euler-Almansi $\mathbf{A}$ par :

$$d\vec{x} \cdot \delta\vec{x} - d\vec{x}^0 \cdot \delta\vec{x}^0 = 2 d\vec{x}^0 \mathrm{E} \delta\vec{x}^0 = 2 d\vec{x} \mathrm{A} \delta\vec{x} \tag{2.12}$$

$$\mathbf{E} = \frac{1}{2}(\mathbf{C} - \mathbf{I}) = \frac{1}{2}(\mathbf{F}^T \mathbf{F} - \mathbf{I}) = \mathbf{F}^T \mathbf{A} \mathbf{F} \tag{2.13}$$

$$\mathbf{A} = \frac{1}{2}(\mathbf{I} - \mathbf{B}^{-1}) = \mathbf{F}^{-T} \mathbf{E} \mathbf{F}^{-1} \tag{2.14}$$

où **I** désigne le tenseur identité d'ordre 2. Les deux tenseurs s'annulent pour les mouvements de corps rigides.

Les composantes cartésiennes du tenseur de déformation de Green-Lagrange, que nous utiliserons s'expriment donc en fonction des composantes du vecteur déplacement comme suit :

$$\mathbf{E}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j^0} + \frac{\partial u_j}{\partial x_i^0} + \frac{\partial u_k}{\partial x_i^0}\frac{\partial u_k}{\partial x_j^0} \right) \tag{2.15}$$

D'après le théorème de la décomposition polaire, le tenseur **F** s'écrit sous la forme du produit d'une matrice de rotation **R** et d'une matrice d'élongation **U** ou **V**. La Fig. 2.2 illustre l'orientation des configurations intermédiaires :

$$\mathbf{F} = \mathbf{RU} = \mathbf{VR} \tag{2.16}$$

avec la propriété suivante sur les tenseurs orthogonaux:

$$\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{1} \tag{2.17}$$

R : Tenseur de rotation pure correspondant au mouvement de corps rigide.

U : Tenseur de déformation pure droit.

V : Tenseur de déformation pure gauche.

Les tenseurs **C** et **U** (et respectivement **B** et **V**) décrivent les mêmes changements de forme entre la configuration de référence C^0 et la configuration $C(t)$. Ils sont liés par les relations suivantes :

$$\mathbf{C} = \mathbf{F}^T\mathbf{F} = \mathbf{U}\mathbf{R}^T\mathbf{R}\mathbf{U} = \mathbf{U}^2 \tag{2.18}$$

$$\mathbf{B} = \mathbf{F}\mathbf{F}^T = \mathbf{V}\mathbf{R}\mathbf{R}^T\mathbf{V} = \mathbf{V}^2 \tag{2.19}$$

De (2.16) on déduit également :

$$\mathbf{V} = \mathbf{R}\mathbf{U}\mathbf{R}^T \tag{2.20}$$

et

$$\mathbf{B} = \mathbf{R}\mathbf{C}\mathbf{R}^T \tag{2.21}$$

Les directions principales de **C** (resp. de **B**) coïncident avec les directions principales de **U** (resp. de **V**). Les vecteurs propres de **C** notés $\vec{\mathbf{N}}^0$ et les valeurs propres correspondantes λ vérifient la relation :

$$\mathbf{C}\vec{\mathbf{N}}^0 = \lambda^2\vec{\mathbf{N}}^0 \tag{2.22}$$

Le tenseur **C** peut ainsi se mettre sous la forme :

$$\mathbf{C} = Q^0 \lambda^2 Q^{0T} \tag{2.23}$$

où Q^0 contient les vecteurs propres de **C** dans la configuration C^0.

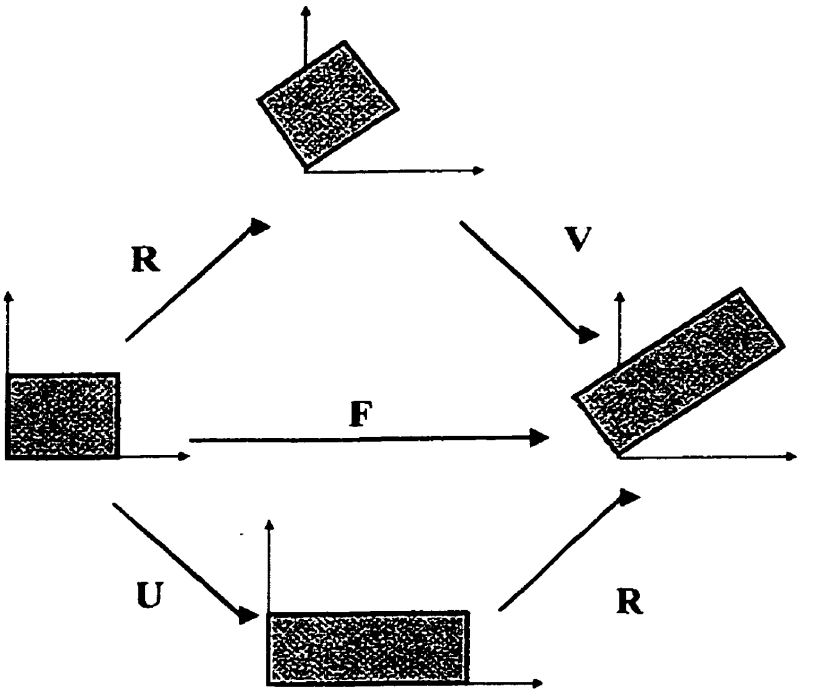


Figure 2.2 Décomposition polaire

Les directions principales (ou vecteurs propres) de **V** et de **B** sont $\vec{N}$ tels que :

$$\vec{N} = \mathbf{R}\vec{N}^0 \tag{2.24}$$

ou encore

$$Q = \mathbf{R}Q^0 \tag{2.25}$$

où Q contient les vecteurs propres de **B**.

De (2.18) et (2.20), nous pouvons déduire :

$$\mathbf{U} = \mathbf{Q}^0 \lambda \mathbf{Q}^{0T} \tag{2.26}$$

$$\mathbf{V} = \mathbf{Q} \lambda \mathbf{Q}^{T} \tag{2.27}$$

2.2.4 Taux de déformation

De nombreuses relations constitutives font intervenir des quantités incrémentales (taux) de déformation ou un gradient d'un champ représentant une vitesse. Soit $\dot{\bar{x}}$ le vecteur vitesse d'une particule matérielle, le tenseur $\dot{\mathbf{F}}$ s'exprime alors en fonction de $\dot{\bar{x}}$ par :

$$\mathbf{F} = \frac{d}{dt}\left(\frac{\partial \bar{x}(\bar{x}^0, t)}{\partial \bar{x}^0}\right) = \frac{\partial \dot{\bar{x}}(\bar{x}^0, t)}{\partial \bar{x}^0} \tag{2.28}$$

La dérivée par rapport au temps d'un vecteur $d\bar{x}$ dans la configuration $C(t)$ s'obtient en dérivant par rapport au temps l'expression (2.3) :

$$d\dot{\bar{x}} = \dot{\mathbf{F}} d\bar{x}^0 \tag{2.29}$$

en remplaçant $d\bar{x}^0$ par son expression (2.9), nous définissons le tenseur Eulérien $\mathbf{L}$ tel que :

$$d\dot{\bar{x}} = \dot{\mathbf{F}}\mathbf{F}^{-1} d\bar{x} \tag{2.30}$$

et

$$d\dot{\bar{x}} = \mathbf{L} d\bar{x} \qquad \text{avec} \qquad \mathbf{L} = \dot{\mathbf{F}}\mathbf{F}^{-1} \tag{2.31}$$

$\mathbf{L}$ est appelé tenseur gradient des vitesses de déformation dans la configuration actuelle $C(t)$, il se décompose en un tenseur de vitesse de déformation qui est symétrique et un tenseur antisymétrique lié à la rotation de la matière.

$$\mathbf{L} = \frac{\partial \dot{\bar{x}}}{\partial \bar{x}^0} \frac{\partial \bar{x}^0}{\partial \bar{x}} = \frac{\partial \dot{\bar{x}}}{\partial \bar{x}}$$

(2.32)

$$\mathbf{L} = \mathbf{D} + \mathbf{W} \tag{2.33}$$

avec :

$$\mathbf{D} = \frac{1}{2}\left(\mathbf{L} + \mathbf{L}^{T}\right) \tag{2.34}$$

$$\mathbf{W} = \frac{1}{2}\left(\hat{\mathbf{L}} - \mathbf{L}^{\mathrm{T}}\right) \tag{2.35}$$

Dans la configuration C^0, le tenseur taux de déformation Lagrangien $\dot{\mathbf{E}}$ est obtenu par simple dérivation par rapport au temps de (2.10) :

$$d\vec{\mathbf{x}} \cdot \delta\vec{\mathbf{x}} = 2d\vec{\mathbf{x}}^0 \dot{\mathbf{E}} \delta\vec{\mathbf{x}}^0 \tag{2.36}$$

L'équivalent du tenseur $\dot{\mathbf{E}}$ dans la configuration courante $C(t)$ est le tenseur taux de déformation **D** défini par :

$$d\vec{\mathbf{x}} \cdot \delta\vec{\mathbf{x}} = 2d\vec{\mathbf{x}}\mathbf{D}\delta\vec{\mathbf{x}} \tag{2.37}$$

Ces deux tenseurs étant les transportés l'un de l'autre par :

$$\dot{\mathbf{E}} = \mathbf{F}^{\mathrm{T}}\mathbf{D}\mathbf{F} \tag{2.38}$$

En utilisant (2.17) nous obtenons :

$$\dot{\mathbf{R}}\mathbf{R}^{\mathrm{T}} + \mathbf{R}\dot{\mathbf{R}}^{\mathrm{T}} = \mathbf{0} \tag{2.39}$$

En utilisant (2.16), (2.17), (2.31) et (2.34) nous obtenons :

$$\mathbf{D} = \frac{1}{2}\mathbf{R}\left(\dot{\mathbf{U}}\mathbf{U}^{-1} + \mathbf{U}^{-1}\dot{\mathbf{U}}\right)\mathbf{R}^{\mathrm{T}} \tag{2.40}$$

qui est l'expression du taux de déformation **D** en fonction de $\mathbf{R}$, $\mathbf{R}^{\mathrm{T}}$, $\mathbf{U}$ et $\dot{\mathbf{U}}$.
Le tenseur taux de rotation **W** s'exprime quant à lui comme :

$$\mathbf{W} = \dot{\mathbf{R}}\mathbf{R}^{\mathrm{T}} + \frac{1}{2}\mathbf{R}\left(\dot{\mathbf{U}}\mathbf{U}^{-1} - \mathbf{U}^{-1}\dot{\mathbf{U}}\right)\mathbf{R}^{\mathrm{T}} \tag{2.41}$$

Le tenseur **W** se compose d'une partie due à la rotation du corps rigide $\dot{\mathbf{R}}\mathbf{R}^{\mathrm{T}}$ et d'une partie due au cisaillement (partie symétrique de $\dot{\mathbf{U}}\mathbf{U}^{-1}$). Si nous faisons l'hypothèse que $\mathbf{U}$ garde ses directions propres fixes alors :

$$\mathbf{U} = \mathrm{Q}^0 \lambda \mathrm{Q}^{0\mathrm{T}}$$

et

$$\dot{\mathbf{U}} = \mathbf{Q}^0 \dot{\lambda} \mathbf{Q}^{0T} \tag{2.42}$$

d'où l'on déduit :

$$\dot{\mathbf{U}}\mathbf{U}^{-1} - \mathbf{U}^{-1}\dot{\mathbf{U}} = \mathbf{0} \tag{2.43}$$

et

$$\mathbf{W} = \dot{\mathbf{R}}\mathbf{R}^{\mathbf{T}} \tag{2.44}$$

Ceci implique que la partie $\mathbf{R}\left(\dot{\mathbf{U}}\mathbf{U}^{-1} - \mathbf{U}^{-1}\dot{\mathbf{U}}\right)\mathbf{R}^{\mathbf{T}}$ est nulle (pas de cisaillement) et par conséquent la rotation Q^0 entre la configuration C^0 initiale et actuelle $C(t)$ calculée à partir de taux de rotation correspondant à $\mathbf{W}$ coïncide avec la rotation de corps rigide $\mathbf{R}$ obtenue par décomposition polaire de $\mathbf{F}$.

Nous pouvons aussi définir une mesure Eulérienne du taux de déformation, en dérivant par rapport au temps le tenseur des déformations d'Almansi-Euler $\mathbf{A}$ (2.14) (resp. $\mathbf{B}$ (2.19)). Tout calcul fait, il viendrait :

$$\dot{\mathbf{A}} = \mathbf{D} - \mathbf{A}\mathbf{L} - \mathbf{L}^{\mathbf{T}}\mathbf{A} \tag{2.45}$$

et

$$\dot{\mathbf{B}} = \mathbf{L}\mathbf{B} + \mathbf{B}\mathbf{L}^{\mathbf{T}} \tag{2.46}$$

$\dot{\mathbf{A}}$ et $\dot{\mathbf{B}}$ ne s'annulent pas dans un mouvement rigide précédé d'une prédéformation, ainsi ils ne contiennent pas de bonnes mesures du taux de déformation.

2.2.5 Taux des contraintes

En petites et en grandes déformation, le vecteur contrainte $\vec{\tau} = \frac{d\vec{f}}{ds}$ est défini dans la configuration actuelle. Son rôle est de caractériser les efforts intérieurs de cohésion exercés sur une partie du solide à travers un élément de surface ds de normale $\vec{n}$. Nous définissons ainsi un tenseur Eulérien des contraintes, le tenseur (symétrique) des contraintes de Cauchy σ (dit aussi contraintes vraies) tel que :

$$d\vec{f} = \boldsymbol{\sigma}\vec{n}ds \tag{2.47}$$

Il est possible de transporter ce tenseur pour définir d'autres tenseurs des contraintes. C'est ainsi que si l'on choisit de décrire $\vec{df}$ en fonction de $\vec{n}^0$ et ds^0 dans la configuration C^0 qui se transforment en $\vec{n}$ et ds dans la configuration $C(t)$, nous obtiendrons :

$$\vec{df} = \pi \vec{n}^0 ds^0 \tag{2.48}$$

où π désigne le premier tenseur (non symétrique) de Piola-Kichhoff (PK1) ou tenseur de Boussinesq, qui n'est ni Lagrangien ni Eulérien.

Une autre possibilité consiste à définir un tenseur Lagrangien et symétrique, en transportant $\vec{df}$ dans C^0 :

$$\vec{df} = \mathbf{F}^{-1}\vec{df} = \mathbf{S}\vec{n}^0 ds^0 \tag{2.49}$$

où **S** désigne le second tenseur de Piola-Kirchhoff (PK2) ou tenseur de Piola-Lagrange.

La relation entre ces trois tenseurs est donnée par :

$$J\sigma = \pi \mathbf{F}^T = \mathbf{F}\mathbf{S}\mathbf{F}^T \tag{2.50}$$

Notons enfin que le tenseur **S** n'a pas de signification physique directe (dans le cas de grandes déformations). Son intérêt réside dans le fait qu'il est la variable duale conjuguée (au sens de l'énergie) du tenseur de déformation de Green-Lagrange **E**.

Quant au tenseurs σ et π, ils caractérisent directement les efforts appliqués. Ils interviennent donc directement dans l'écriture des conditions aux limites.

Par ailleurs, ces tenseurs sont tous confondus dans le cas de la théorie linéaire. Il est important de signaler que dans le cas des petites déformations (i.e. $J \approx 1$ et $\lambda \approx 1$) mais des grands déplacements, les composantes de **S** s'identifient aux composantes de contraintes de Cauchy dans un repère défini par la rotation **R**.

$$\mathbf{S} \approx \mathbf{R}^T \sigma \mathbf{R} \tag{2.51}$$

2.3 Équation du mouvement et relation contraintes-déformations

La déformation d'un corps solide est entièrement caractérisée du point de vue mécanique si on connaît les champs de déplacements donc les champs de déformations et les champs de contraintes. Ces champs ne sont pas quelconques mais doivent satisfaire les principes de

conservation de la physique caractérisant tout processus dynamiquement et thermodynamiquement admissible. En outre, les déformations et les contraintes doivent satisfaire les relations constitutives caractérisant le matériau considéré.

2.3.1 Principe de conservation

Les principes de conservation qui interviendront explicitement dans le cadre de ce travail sont la conservation de la masse, de la quantité de mouvement, du moment cinétique, et de l'énergie. Il faudra, en outre, satisfaire l'inégalité de CLAUSIUS-DUHEM assurant une production totale d'entropie non négative.

La loi de conservation de la masse implique localement :

$$\rho^0 dv^0 = \rho dv \tag{2.52}$$

Les lois de conservation de la quantité de mouvement sont satisfaites si on vérifie localement les équations d'équilibre de Cauchy.

Le solide dans la configuration $C(t)$ est soumis à l'action de forces $\vec{f}_v(\vec{x},t)$ et de forces d'accélération $\rho\ddot{\vec{x}}(\vec{x},t)$, définies par unité de volume, ces équations s'écrivent :

$$\begin{cases} div\boldsymbol{\sigma} + \vec{f}_v = \rho\ddot{\vec{x}}(\vec{x},t), & \text{sur } V \\ \boldsymbol{\sigma} = \boldsymbol{\sigma}^T & \\ \boldsymbol{\sigma}\vec{n} = \vec{f}_s, & sur\ S_f \\ \vec{u} = \bar{\vec{u}}, & sur\ S_u \end{cases} \tag{2.53}$$

$\vec{f}_s(\vec{n})$ représentent soit les forces imposées, soit les réactions associées aux liaisons où les déplacements sont imposés. S_f est la partie du contour où les forces sont imposées et S_u celle où les déplacements sont imposés.

$$\partial V = S = S_u \cup S_f \tag{2.54}$$

Naturellement les équations (2.54) peuvent être écrites sur la configuration non déformée C^0 :

$$\begin{cases} \mathrm{div}^0 \boldsymbol{\pi} + \vec{f}_v^0 = \rho \ddot{\vec{x}}^0(\vec{x}^0, t), & \text{sur } V^0 \\ \boldsymbol{\pi}\vec{n}^0 = \vec{f}_s^0 & \text{sur } S_f^0 \\ \vec{u} = \vec{\bar{u}}^0, & \text{sur } S_u^0 \\ \vec{f}_v^0 = J\vec{f}_v \\ \vec{f}_s^0 = \dfrac{ds}{ds_0}\vec{f}_s \end{cases} \tag{2.55}$$

en coordonnées cartésiennes :

$$\mathrm{div}^0 = \left\langle \frac{\partial}{\partial x^0} \quad \frac{\partial}{\partial y^0} \quad \frac{\partial}{\partial z^0} \right\rangle \tag{2.56}$$

Pour être thermodynamiquement admissible, la déformation doit aussi satisfaire les deux principes fondamentaux de la thermodynamique. La conservation de l'énergie qui se traduit par :

$$\rho \dot{\mathbf{e}} = \boldsymbol{\sigma} : \mathbf{D} + \mathbf{r} - div\vec{\mathbf{q}} \tag{2.57}$$

où : indique un produit intérieur de deux tenseurs

et $\dot{\mathbf{e}}$ est la densité de l'énergie interne spécifique, $\mathbf{r}$ est une densité volumique de production interne de chaleur et $\mathrm{div}\vec{\mathbf{q}}$ est le flux de chaleur orienté positif vers l'extérieur du corps.

La relation (2.57), peut aussi s'écrire en fonction de la densité d'énergie libre spécifique :

$$\boldsymbol{\psi} = \mathbf{e} - \mathbf{Ts} \tag{2.58}$$

où $\mathbf{s}$ est l'entropie spécifique et $\mathbf{T}$ la température. Le second principe de la thermodynamique stipule que la production totale d'entropie doit être non négative ce qui implique :

$$\boldsymbol{\sigma} : \mathbf{D} - \rho\left(\dot{\psi} + \mathbf{s}\dot{\mathbf{T}}\right) - \vec{\mathbf{q}}\frac{grad\mathbf{T}}{\mathbf{T}} \geq 0 \tag{2.59}$$

Dans notre travail, nous considérons que les équations de conservation de l'énergie (2.58) et de la quantité de mouvement (2.53), (2.55) sont découplées.

2.3.2 Relations constitutives

Les définitions des déformations et des contraintes associées aux lois de conservation ne suffisent pas à résoudre un problème d'équilibre. Leurs définitions étant "universelles", elles sont valables indépendamment du matériau envisagé et ne sont donc pas suffisantes pour déterminer de façon univoque le comportement d'un corps quelconque à partir des conditions initiales et celles aux limites. Pour cela, il faut tenir compte de la nature des matériaux impliqués et avoir recours à des équations supplémentaires caractérisant le comportement du matériau. Ces équations constituent la loi de comportement du matériau. La littérature regorge d'un nombre impressionnant de formes diverses de lois de comportement. L'une des difficultés réside dans le choix du modèle adéquat et dans la vérification expérimentale de la validité de celle-ci. Ces modèles comportent généralement un certain nombre de coefficients propres à chaque matériau d'une même classe. Ces coefficients sont identifiés à partir des tests expérimentaux assez simples. Ces modèles permettent de prédire le comportement du matériau donné soumis à des sollicitations complexes.

Dans ce chapitre on a présenté les grandes lignes de la théorie des grandes déformations. Ceci nous permet d'aborder dans le prochain chapitre la loi de comportement des solides hyperélastiques en grandes déformations.

CHAPITRE III

LOI DE COMPORTEMENT : HYPERÉLASTICITÉ

3.1 Introduction

Les équations d'équilibre d'un corps sont écrites en fonction des contraintes internes du corps. Ces contraintes résultent de la déformation du matériau et les relations entre les contraintes et les mesures de déformations sont appelées équations constitutives qui dépendent du matériau et dans certains cas du temps.

Ces équations constitutives doivent satisfaire certains principes physiques, comme l'objectivité ou indifférence matérielle. (Elles constituent la loi de comportement du matériau). Une approche moderne de formulations d'une loi de comportement utilise la thermodynamique des milieux continus pour écrire des bilans de conservation d'énergie et de l'entropie qui permettent de dériver de façon cohérente des relations valides pour une large gamme de matériau. Afin de satisfaire aux besoins d'avoir un modèle qui soit proche du matériau transformé en entreprise (impliquant des grandes transformations) la loi de comportement choisie est de type hyperélastique. C'est à dire que nous considérons des matériaux dont le comportement élastique ne dépend que de l'état actuel des déformations et pour lesquels le travail fait par les contraintes induites par le mouvement de déformation dépendent de l'état initial et de la configuration actuelle. Comme ce

comportement est indépendant de la trajectoire suivie dans l'espace des configurations, on postule alors qu'il existe une fonction énergie libre de Helmholtz ou potentiel thermodynamique pouvant représenter l'énergie emmagasinée par le matériau et d'où dérive le tenseur des contraintes conjuguée à la mesure de déformation utilisée pour décrire le mouvement. Cette loi est caractérisée par sa relative simplicité et constitue une base pour des lois plus complexes comme la viscoélasticité, l'élasto-plasticité et la viscoplasticité .

3.2 Comportement hyperélastique

Le comportement hyperélastique est un cas particulier de l'élasticité qui caractérise les matériaux pour lesquels le comportement dépend seulement de l'état courant de déformation et peut être décrit en terme d'énergie de déformation volumique Ψ. Alors dans ces conditions, toute mesure de contrainte en un point matériel donné $\mathbf{X}$ est une fonction du gradient de déformation courant $\mathbf{F}$ associé à cette particule. En d'autre termes l'élasticité peut être exprimée comme suit :

$$\mathbf{P} = \mathbf{P}(\mathbf{F}(\mathbf{X}), \mathbf{X}) \tag{3.1}$$

$\mathbf{F}$ étant le gradient de déformation et $\mathbf{P}$, son conjugué, la première mesure de contrainte de Piola-Kirchhoff.

Et lorsque le travail dû aux contraintes durant un processus de déformation est seulement fonction de l'état initial au temps t_0 et de la configuration finale au temps t, le comportement du matériau est alors indépendant du chemin suivi et le matériau est dit hyperélastique. On en déduit alors une fonction d'énergie de déformation emmagasinée ou potentiel élastique Ψ par unité de volume non déformé qui peut-être assimilé au travail effectué par les contraintes de l'état initial à la position actuelle, soit :

$$\Psi(\mathbf{F}(\mathbf{X}), \mathbf{X}) = \int_{t_0}^{t} \mathbf{P}(\mathbf{F}(\mathbf{X}), \mathbf{X}) : \dot{\mathbf{F}} dt \qquad \dot{\Psi} = \mathbf{P} : \dot{\mathbf{F}} \tag{3.2 a,b}$$

et où $\dot{F}$ représente le taux de gradient de déformation. En supposant qu'à partir de mesures expérimentales il est possible de retrouver la fonction $\Psi(\mathbf{F},\mathbf{X})$, qui définit un matériau donné, alors le taux de variation du potentiel peut être exprimé comme suit :

$$\dot{\Psi} = \sum_{i,J=1}^{3} \frac{\partial \Psi}{\partial F_{iJ}} \dot{F}_{iJ} \tag{3.3}$$

En comparant cette équation avec (3.2b) on déduit un tenseur bipoint :

$$P_{iJ} = \frac{\partial \Psi}{\partial F_{iJ}} \quad \text{ou} \quad \mathbf{P}(\mathbf{F}(\mathbf{X}),\mathbf{X}) = \frac{\partial \Psi(\mathbf{F}(\mathbf{X}),\mathbf{X})}{\partial \mathbf{F}} \tag{3.4}$$

où l'indice J correspond au point initial et l'indice i correspond au point courant. L'équation (3.4) est souvent utilisée pour définir un comportement hyperélastique. Pour exprimer le potentiel élastique en fonction du second tenseur de Piola-Kirchhoff , on doit respecter le principe d'objectivité qui se traduit en disant que Ψ doit rester invariant dans tout mouvement rigide de la configuration actuelle. Ceci implique que :

- Le potentiel dépend de $\mathbf{F}$ seulement via le tenseur de déformation pure droit $\mathbf{U}$ et est indépendant de la rotation $\mathbf{R}$; cependant il est souvent plus convenable de l'exprimer comme une fonction du tenseur de Cauchy-Green droit $\mathbf{C} = \mathbf{U}^2 = \mathbf{F}^T\mathbf{F}$, i.e. :

$$\Psi(\mathbf{F}(\mathbf{X}),\mathbf{X}) = \Psi(\mathbf{C}(\mathbf{X}),\mathbf{X}) \tag{3.5}$$

En remarquant que le taux du tenseur de déformation de Green-Lagrange $\frac{1}{2}\dot{\mathbf{C}} = \dot{\mathbf{E}}$ est énergétiquement conjugué au second tenseur des contraintes de Piola-Kirchhoff $\mathbf{S}$, on peut déduire, à partir de 3.4), une loi de comportement totalement lagrangienne (ou matérielle) sous la forme:

$$\dot{\Psi} = \frac{\partial \Psi}{\partial \mathbf{C}} : \dot{\mathbf{C}} = \frac{1}{2}\mathbf{S} : \dot{\mathbf{C}}; \qquad \mathbf{S}(\mathbf{C}(\mathbf{X}),\mathbf{X}) = 2\frac{\partial \Psi}{\partial \mathbf{C}} = \frac{\partial \Psi}{\partial \mathbf{E}} \tag{3.6a,b}$$

3.3 Tenseur d'élasticité

3.3.1 Tenseur d'élasticité matériel ou Lagrangien

La relation entre $\mathbf{S}$ et $\mathbf{C}$ ou $\mathbf{E}=\frac{1}{2}(\mathbf{C}-\mathbf{I})$, donnée par l'équation (3.6b) sera invariablement non linéaire. Dans le cadre d'un schéma de résolution de Newton-Raphson, cette relation devra être linéarisée par rapport à un incrément de déplacement $\mathbf{u}$ dans la configuration courante. En utilisant la règle de dérivation en chaîne, une relation linéaire entre la dérivée directionnelle de $\mathbf{S}$ et la déformation linéarisée $\mathbf{DE}[\mathbf{u}]$ peut-être obtenue, initialement sous forme indicielle, comme suit :

$$\begin{aligned} DS_{IJ}[\mathbf{u}] &= \frac{d}{d\varepsilon}\bigg|_{\varepsilon=0} S_{IJ}(E_{KL}[\phi+\varepsilon\mathbf{u}]) \\ &= \sum_{K,L=1}^{3} \frac{\partial S_{IJ}}{\partial E_{KL}} \frac{d}{d\varepsilon}\bigg|_{\varepsilon=0} E_{KL}[\phi+\varepsilon\mathbf{u}] \\ &= \sum_{K,L=1}^{3} \frac{\partial S_{IJ}}{\partial E_{KL}} DE_{KL}[\mathbf{u}] \end{aligned} \tag{3.7}$$

Cette relation entre les dérivées directionnelles de $\mathbf{S}$ et de $\mathbf{E}$ est exprimée de façon plus concise comme suit :

$$\mathbf{DS}[\mathbf{u}] = \mathbf{C} : \mathbf{DE}[\mathbf{u}] \tag{3.8}$$

où le tenseur symétrique de quatrième-ordre $\mathbf{C}$, aussi appelé *tenseur d'élasticité matériel* ou *Lagrangien,* est défini à l'aide de dérivées partielles comme suit :

$$\begin{aligned} \mathbf{C} &= \sum_{I,J,K,L=1}^{3} \mathsf{C}_{IJKL}\mathbf{E}_I \otimes \mathbf{E}_J \otimes \mathbf{E}_K \otimes \mathbf{E}_L ; \\ \mathsf{C}_{IJKL} &= \frac{\partial \mathbf{S}_{IJ}}{\partial \mathbf{E}_{KL}} = \frac{4\partial^2 \Psi}{\partial \mathbf{C}_{IJ}\partial \mathbf{C}_{KL}} = \mathsf{C}_{KLIJ} \end{aligned} \tag{3.9}$$

ou sous forme abrégée :

$$\mathbf{C} = \frac{\partial \mathbf{S}}{\partial \mathbf{E}} = 2\frac{\partial \mathbf{S}}{\partial \mathbf{C}} = \frac{4\partial^2 \Psi}{\partial \mathbf{C}\partial \mathbf{C}} \tag{3.10}$$

3.3.2 Tenseur d'élasticité spatial ou Eulérien

En interprétant l'équation (3.8) comme un taux de variation et en lui appliquant le transport convectif avant (ou "push forward transformation" défini plus loin) on peut retrouver le tenseur d'élasticité spatial. Ceci est réalisé en linéarisant $\mathbf{S}$ et $\mathbf{E}$ en direction de $\mathbf{v}$ plutôt que de $\mathbf{u}$ de telle sorte que :

$$\mathbf{DS}[\mathbf{v}] = \dot{\mathbf{S}} \text{ et } \mathbf{DE}[\mathbf{v}] = \dot{\mathbf{E}}$$

et on obtient :

$$\dot{\mathbf{S}} = \mathbf{C} : \dot{\mathbf{E}} \tag{3.11}$$

Parce que le transport convectif avant de $\dot{\mathbf{S}}$ donne la dérivée de Truesdell [Ref-10] de la contrainte de Kirchhoff $\tau^{\circ} = \mathbf{J}\sigma^{\circ}$ et que celui de $\dot{\mathbf{E}}$ donne le tenseur taux de déformation $\mathbf{d}$, il est alors possible d'obtenir l'équivalent spatial de l'équation constitutive linéarisée du matériau (3.11) comme étant :

$$\sigma^{\circ} = \varsigma : \mathbf{d} \tag{3.12}$$

où ς est le tenseur d'élasticité spatial ou Eulérien, défini comme étant le transport convectif avant de Piola du tenseur matériel $\mathbf{C}$. Il s'écrit sous forme indicielle comme suit :

$$\varsigma = \mathbf{J}^{-1}\phi_*[\mathbf{C}]; \qquad \varsigma = \sum_{\substack{i,j,k,l=1 \\ I,J,K,L=1}}^{3} \mathbf{J}^{-1}\mathbf{F}_{iI}\mathbf{F}_{jJ}\mathbf{F}_{kK}\mathbf{F}_{lL}\mathbf{C}_{IJKL}\mathbf{e}_i \otimes \mathbf{e}_j \otimes \mathbf{e}_k \otimes \mathbf{e}_l \tag{3.13}$$

3.4 Hyperélasticité isotrope

3.4.1 Description matérielle

Les équations constitutives de l'hyperélasticité décrites jusqu'à maintenant sont générales dans leur application. Dans cette section on s'intéresse au cas le plus courant à savoir le cas de matériaux isotropes. On dit qu'un matériau est isotrope lorsque son comportement est le même indépendamment de la direction matérielle suivie. Ceci

implique que la relation entre Ψ et $\mathbf{C}$ doit être indépendante des axes du matériau choisis et, par conséquent, Ψ doit être seulement une fonction des invariants de $\mathbf{C}$, soit :

$$\Psi(\mathbf{C}(\mathbf{X}),\mathbf{X}) = \Psi(\mathbf{I}_\mathbf{C},\mathbf{II}_\mathbf{C},\mathbf{III}_\mathbf{C},\mathbf{X}) \tag{3.14}$$

où les invariants de $\mathbf{C}$ sont définies comme suit :

$$\begin{aligned} \mathbf{I}_\mathbf{C} &= tr\mathbf{C} = \mathbf{C}:\mathbf{I} \\ \mathbf{II}_\mathbf{C} &= tr\mathbf{C}\mathbf{C} = \mathbf{C}:\mathbf{C} \\ \mathbf{III}_\mathbf{C} &= \det\mathbf{C} = J^2 \end{aligned} \tag{3.15 a,b,c}$$

En partant de l'équation (3.6b) le second tenseur de Piola-Kirchhoff , dans le cas isotrope, s'écrit alors comme suit :

$$\mathbf{S} = 2\frac{\partial\Psi}{\partial\mathbf{C}} = 2\frac{\partial\Psi}{\partial\mathbf{I}_\mathbf{C}}\frac{\partial\mathbf{I}_\mathbf{C}}{\partial\mathbf{C}} + 2\frac{\partial\Psi}{\partial\mathbf{II}_\mathbf{C}}\frac{\partial\mathbf{II}_\mathbf{C}}{\partial\mathbf{C}} + 2\frac{\partial\Psi}{\partial\mathbf{III}_\mathbf{C}}\frac{\partial\mathbf{III}_\mathbf{C}}{\partial\mathbf{C}} \tag{3.16}$$

Les tenseurs de second ordre formés par les dérivées des deux premiers invariants de $\mathbf{C}$ peuvent être exprimés sous forme indicielle comme suit :

$$\frac{\partial}{\partial C_{IJ}}\sum_{K=1}^{3} C_{KK} = \delta_{IJ}\,; \qquad \frac{\partial\mathbf{I}_\mathbf{C}}{\partial\mathbf{C}} = \mathbf{I} \tag{3.17a}$$

$$\frac{\partial}{\partial C_{IJ}}\sum_{K,L=1}^{3} C_{KL}C_{KL} = 2C_{IJ}\,; \qquad \frac{\partial\mathbf{II}_\mathbf{C}}{\partial\mathbf{C}} = 2\mathbf{C} \tag{3.17b}$$

La dérivée du troisième invariant est évaluée en utilisant l'expression relative à la linéarisation du déterminant d'un tenseur. Avec $\Delta\mathbf{C}$ représentant un incrément tensoriel arbitraire, on obtient la dérivée directionnelle:

$$\mathbf{DIII}_\mathbf{C}[\Delta\mathbf{C}] = \sum_{I,J=1}^{3}\frac{\partial\mathbf{III}_\mathbf{C}}{\partial C_{IJ}}\Delta C_{IJ} = \frac{\partial\mathbf{III}_\mathbf{C}}{\partial\mathbf{C}}:\Delta\mathbf{C} \tag{3.18}$$

Soit encore :

$$\mathbf{DIII}_\mathbf{C}[\Delta\mathbf{C}] = \det\mathbf{C}\left(\mathbf{C}^{-1}:\Delta\mathbf{C}\right) \tag{3.19}$$

En égalisant les deux équations précédentes, on déduit que :

$$\frac{\partial \mathbf{III}_C}{\partial \mathbf{C}} = J^2 \mathbf{C}^{-1} \tag{3.20}$$

En introduisant les équations (3.17a,b) et (3.20) dans l'équation (3.16) on obtient que le second tenseur de Piola-Kirchhoff s'écrit comme suit :

$$\mathbf{S} = 2\Psi_{\mathrm{I}}\mathbf{I} + 4\Psi_{\mathrm{II}}\mathbf{C} + 2J^2\Psi_{\mathrm{III}}\mathbf{C}^{-1} \tag{3.21}$$

avec $\Psi_{\mathrm{I}} = \partial\Psi/\partial\mathbf{I}_C$, $\Psi_{\mathrm{II}} = \partial\Psi/\partial\mathbf{II}_C$, *et* $\Psi_{\mathrm{III}} = \partial\Psi/\partial\mathbf{III}_C$.

3.4.2 Description spatiale

Dans le contexte du design, il est clair que les contraintes de Cauchy sont d'intérêt important pour l'ingénieur. Ceci peut être obtenu indirectement du second tenseur de Piola-Kirchhoff en appliquant le transport convectif avant, soit :

$$\sigma = J^{-1}\mathbf{F}\mathbf{S}\mathbf{F}^T \tag{3.22}$$

En remplaçant $\mathbf{S}$ à partir de l'équation (3.21) et en remarquant que le tenseur de Cauchy-Green gauche est donné par $\mathbf{b} = \mathbf{F}\mathbf{F}^{\mathrm{T}}$, on obtient alors:

$$\sigma = 2J^{-1}\Psi_{\mathrm{I}}\mathbf{b} + 4J^{-1}\Psi_{\mathrm{II}}\mathbf{b}^2 + 2J\Psi_{\mathrm{III}}\mathbf{I} \tag{3.23}$$

En notant que les invariants de $\mathbf{b}$ sont identiques à ceux de $\mathbf{C}$, comme le montre les expressions suivantes :

$$\begin{aligned}
\mathbf{I}_{\mathbf{b}} &= tr[\mathbf{b}] = tr[\mathbf{F}\mathbf{F}^{\mathrm{T}}] = tr[\mathbf{F}^{\mathrm{T}}\mathbf{F}] = tr[\mathbf{C}] = \mathbf{I}_{\mathbf{C}} \\
\mathbf{II}_{\mathbf{b}} &= tr[\mathbf{b}\mathbf{b}] = tr[\mathbf{F}\mathbf{F}^{\mathrm{T}}\mathbf{F}\mathbf{F}^{\mathrm{T}}] = tr[\mathbf{F}^{\mathrm{T}}\mathbf{F}\mathbf{F}^{\mathrm{T}}\mathbf{F}] = tr[\mathbf{C}\mathbf{C}] = \mathbf{II}_{\mathbf{C}} \\
\mathbf{I}_{\mathbf{b}} &= \det[\mathbf{b}] = \det[\mathbf{F}\mathbf{F}^{\mathrm{T}}] = \det[\mathbf{F}^{\mathrm{T}}\mathbf{F}] = \det[\mathbf{C}] = \mathbf{III}_{\mathbf{C}}
\end{aligned} \tag{3.24a,b,c}$$

il s'en suit que les termes $\Psi_{\mathrm{I}}, \Psi_{\mathrm{II}}$, *et* Ψ_{III} dans l'équation (3.23) désignent aussi les dérivées de Ψ par rapport à $\mathbf{b}$.

3.4.3 Matériau compressible Néo-Hookéen

Parmi les matériaux isotropes, on distingue le cas simple de matériau compressible néo-Hookéen. Ce matériau présente des caractéristiques qui peuvent être identifiées avec les paramètres familiers dans l'étude des matériaux rencontrés dans l'analyse élastique linéaire. L'expression de la fonction énergie de déformation volumique pour un tel matériau est définie comme suit :

$$\Psi = \frac{\mu}{2}(I_C - 3) - \mu \ln J + \frac{\lambda}{2}(\ln J)^2 \tag{3.25}$$

où les constantes λ et μ sont des coefficients de Lamé du matériau et $J^2 = III_C$. On peut remarquer qu'en absence de déformations, c'est-à-dire, quand $\mathbf{C} = \mathbf{I}$, la fonction de l'énergie emmagasinée devient nulle.

Le second tenseur des contraintes de Piola-Kirchhoff peut maintenant être obtenu de l'équation (3.21) comme suit :

$$\mathbf{S} = \mu\left(\mathbf{I} - \mathbf{C}^{-1}\right) + \lambda(\ln J)\mathbf{C}^{-1} \tag{3.26}$$

De même, les contraintes de Cauchy peuvent être dérivées à partir de l'équation (3.23) en fonction du tenseur de Cauchy-Green gauche $\mathbf{b}$, soit :

$$\sigma = \frac{\mu}{J}(\mathbf{b} - \mathbf{I}) + \frac{\lambda}{J}(\ln J)\mathbf{I} \tag{3.27}$$

Le tenseur d'élasticité Lagrangien correspondant à ce matériau néo-Hookéen peut-être obtenu par différentiation de l'équation (3.26) par rapport aux composantes de $\mathbf{C}$. On obtient après quelques manipulations algébriques en utilisant l'équation (3.19) que $\mathbf{C}$ s'écrit :

$$\mathbf{C} = \lambda \mathbf{C}^{-1} \otimes \mathbf{C}^{-1} + 2(\mu - \lambda \ln J)\mathbf{I} \tag{3.28}$$

où $\mathbf{C}^{-1} \otimes \mathbf{C}^{-1} = \sum \left(\mathbf{C}^{-1}\right)_{IJ} \left(\mathbf{C}^{-1}\right)_{KL} \mathbf{E}_I \otimes \mathbf{E}_J \otimes \mathbf{E}_K \otimes \mathbf{E}_L$ et le tenseur identité de quatrième ordre $\mathbf{I}$ est défini comme suit :

$$\mathbf{I} = -\frac{\partial \mathbf{C}^{-1}}{\partial \mathbf{C}}; \qquad I_{IJKL} = -\frac{\partial (\mathbf{C}^{-1})_{IJ}}{\partial \mathbf{C}_{KL}} = (\mathbf{C}^{-1})_{IK}(\mathbf{C}^{-1})_{JL} \tag{3.29}$$

Le tenseur d'élasticité spatial ou Eulérien peut maintenant s'obtenir en appliquant la transformation convective avant au tenseur Lagrangien, en utilisant l'équation (3.13) , comme suit :

$$\varsigma = \frac{\lambda}{J}\mathbf{I} \otimes \mathbf{I} + \frac{2}{J}(\mu - \lambda \ln J)\mathbf{i} \tag{3.30}$$

où $\mathbf{i}$ est le tenseur identité de quatrième ordre obtenu en appliquant la transformation convective avant à $\mathbf{I}$. Il est donné sous forme indicielle en terme du symbole delta de Kroneker, soit :

$$\mathbf{i} = \phi_*[\mathbf{I}]; \qquad i_{ijkl} = \sum_{I,J,K,L} F_{iI}F_{jJ}F_{kK}F_{lL}I_{IJKL} = \delta_{ik}\delta_{jl} \tag{3.31}$$

En remarquant que l'équation (3.30) ci-dessus définit un tenseur isotrope de quatrième ordre, similaire à celui utilisé dans l'élasticité linéaire, qui peut s'exprimer en termes des modules effectifs de Lamé λ' et μ', on obtient que :

$$\varsigma_{ijkl} = \lambda'\delta_{ij}\delta_{kl} + 2\mu'\delta_{ik}\delta_{jl} \tag{3.32}$$

où les coefficients effectifs λ' et μ' sont :

$$\lambda' = \frac{\lambda}{J}; \quad \mu' = \frac{\mu - \lambda \ln J}{J} \tag{3.33}$$

On notera que dans le cas des petites déformations $J \approx 1$, alors $\lambda \approx \lambda'$, $\mu \approx \mu'$, et on retrouve le tenseur standard de quatrième ordre de l'élasticité linéaire.

3.5 Matériaux incompressibles et presqu'incompressibles

La plupart des cas pratiques de transformations en grandes déformations ont lieu sous des conditions incompressibles ou presqu'incompressibles. Il en est de même dans notre cas. Il est alors pertinent de discuter des implications constitutives d'une telle limitation sur la déformation du matériau. La terminologie "presqu'incompressible" est utilisée ici pour distinguer les matériaux qui sont réellement incompressibles, mais dont l'analyse numérique fait appel à une petite mesure de déformation volumétrique. Autrement, dans le cas de grandes déformations élastoplastiques ou inélastiques, la déformation plastique est souvent entièrement incompressible et la déformation volumétrique élastique est comparativement petite.

3.5.1 Élasticité incompressible

En réarrangeant l'équation (3.6a) comme suit :

$$\left(\frac{1}{2}\mathbf{S}-\frac{\partial\Psi}{\partial\mathbf{C}}\right):\dot{\mathbf{C}}=0 \tag{3.34}$$

et $\dot{\mathbf{C}}$ étant supposé arbitraire dans le cas général, on déduit que $\mathbf{S}=2\dfrac{\partial\Psi}{\partial\mathbf{C}}$. Dans le cas incompressible, il n'est pas garanti que le terme entre parenthèse soit nul parce que $\dot{\mathbf{C}}$ n'est plus arbitraire. En effet, comme $\det \mathbf{F} = J = 1$ à travers la déformation, alors $\dot{J}=0$. Soit :

$$\dot{J}=\tfrac{1}{2}J\mathbf{C}^{-1}:\mathbf{C}=0 \tag{3.35}$$

Ainsi :

$$\frac{1}{2}\mathbf{S}-\frac{\partial\Psi}{\partial\mathbf{C}}=\gamma\frac{J}{2}\mathbf{C}^{-1} \tag{3.36}$$

où γ est une inconnue scalaire qui, dans certains cas, coïncidera avec la pression hydrostatique et sera déterminée en utilisant l'équation additionnelle donnée par la contrainte d'incompressibilité $J=1$. De l'équation (3.36), l'équation générale constitutive de l'hyperélasticité incompressible s'écrit :

$$\mathbf{S} = 2\frac{\partial \Psi(\mathbf{C})}{\partial \mathbf{C}} + \gamma J \mathbf{C}^{-1} \tag{3.37}$$

Le déterminant J dans l'équation ci-dessus peut sembler non nécessaire dans le cas de l'incompressibilité, $J = 1$, cependant retenir J a l'avantage que l'équation (3.37) est aussi applicable dans le cas presqu'incompressible. De plus, comme en pratique, une analyse par éléments finis implémente rarement $J = 1$ de façon stricte, ainsi son maintien peut-être important dans l'évaluation des contraintes.

Pour traiter efficacement la condition d'incompressibilité, on introduit la décomposition déviatorique-hydrostatique du second tenseur de Piola-Kirchhoff, donnée par $\mathbf{S} = \mathbf{S}' + pJ\mathbf{C}^{-1}$, où $p = \frac{1}{3} tr\sigma$. Il est alors utile d'exprimer le paramètre γ en fonction de la pression p. Ainsi on obtient:

$$\begin{aligned} p &= \frac{1}{3} J^{-1} \mathbf{S} : \mathbf{C} \\ &= \frac{1}{3} J^{-1} \left[2\frac{\partial \Psi}{\partial \mathbf{C}} + \gamma J \mathbf{C}^{-1} \right] : \mathbf{C} \\ &= \gamma + \frac{2}{3} J^{-1} \frac{\partial \Psi}{\partial \mathbf{C}} : \mathbf{C} \end{aligned} \tag{3.38}$$

d'où il ressort que γ et p ne sont égaux que lorsque :

$$\frac{\partial \Psi}{\partial \mathbf{C}} : \mathbf{C} = 0 \tag{3.39}$$

Ceci implique que $\Psi(\mathbf{C})$ doit être une fonction homogène d'ordre 0, c'est à dire que, $\Psi(\alpha \mathbf{C}) = \Psi(\mathbf{C})$ pour tout α. Ceci peut-être accompli en reconnaissant que pour les matériaux incompressibles $\mathbf{III}_{\mathbf{C}} = \det \mathbf{C} = J^2 = 1$. On peut alors exprimer la fonction Ψ en fonction de la composante distorsionnelle du tenseur de Cauchy-Green droit $\hat{\mathbf{C}} = \mathbf{III}_{\mathbf{C}}^{-1/3} \mathbf{C}$ pour donner une fonction d'énergie formellement modifiée $\bar{\Psi}(\mathbf{C}) = \Psi(\hat{\mathbf{C}})$.

Les propriétés homogènes de la fonction résultante $\hat{\Psi}(\mathbf{C})$ sont alors démontrées comme suit :

$$\begin{aligned}\hat{\Psi}(\alpha\mathbf{C}) &= \Psi\left[(\det\alpha\mathbf{C})^{-1/3}(\alpha\mathbf{C})\right] \\ &= \Psi\left[\left(\alpha^3\det\mathbf{C}\right)^{-1/3}\alpha\mathbf{C}\right] \\ &= \Psi\left[(\det\mathbf{C})^{-1/3}\mathbf{C}\right] \\ &= \Psi\left[\left(\alpha^3\det\mathbf{C}\right)^{-1/3}\alpha\mathbf{C}\right] \\ &= \hat{\Psi}(\mathbf{C})\end{aligned} \tag{3.40}$$

En acceptant que dans le cas de matériaux incompressibles Ψ peut-être remplacé par $\hat{\Psi}$, la condition (3.39) est satisfaite et l'équation (3.37) devient :

$$\mathbf{S} = 2\frac{\partial\hat{\Psi}(\mathbf{C})}{\partial\mathbf{C}} + \gamma J\mathbf{C}^{-1} \tag{3.41}$$

Il est désormais trivial d'identifier la composante déviatorique du second tenseur des contraintes de Piola-Kirchhoff, soit :

$$\mathbf{S}' = 2\frac{\partial\hat{\Psi}}{\partial\mathbf{C}} \tag{3.42}$$

Il est bon de noter que la dérivée $\partial\hat{\Psi}(\mathbf{C})/\partial\mathbf{C}$ n'est pas égale à la dérivée $\partial\Psi(\mathbf{C})/\partial\mathbf{C}$, malgré que $\hat{\mathbf{C}} = \mathbf{C}$ pour l'incompressibilité. Ceci est lié au fait que $\mathbf{III}_\mathbf{C}$ reste une fonction de $\mathbf{C}$ pendant que la dérivée de $\hat{\mathbf{C}}$ est calculée. C'est seulement après que cette dérivée est complétée que la substitution $\mathbf{III}_\mathbf{C} = 1$ peut être appliquée.

3.5.2 Matériaux incompressibles Néo-Hookéen

Un matériau incompressible néo-Hookéen est défini par un potentiel hyperélastique $\Psi(\mathbf{C})$ donné comme suit :

$$\Psi(\mathbf{C}) = \frac{1}{2}\mu(\mathbf{tr}\mathbf{C} - 3) \tag{3.43}$$

Le potentiel homogène équivalent $\hat{\Psi}$ est obtenu en remplaçant $\mathbf{C}$ par $\hat{\mathbf{C}}$, soit :

$$\Psi(\hat{\mathbf{C}}) = \frac{1}{2}\mu(\mathbf{tr}\hat{\mathbf{C}} - 3) \tag{3.44}$$

Le tenseur $\mathbf{S}$ est obtenu à partir des équations (3.41), (3.17a) et (3.18), soit :

$$\begin{aligned}
\mathbf{S} &= 2\frac{\partial \hat{\Psi}(\mathbf{C})}{\partial \mathbf{C}} + \gamma J\mathbf{C}^{-1} \\
&= \mu\frac{\partial tr\hat{\mathbf{C}}}{\partial \mathbf{C}} + pJ\mathbf{C}^{-1} \qquad (3.45) \\
&= \mu\frac{\partial}{\partial \mathbf{C}}\left(\mathbf{III}_{\mathbf{C}}^{-1/3}\mathbf{C}:\mathbf{I}\right) + pJ\mathbf{C}^{-1} \\
&= \mu\left[\mathbf{III}_{\mathbf{C}}^{-1/3}\mathbf{I} - \frac{1}{3}\mathbf{III}_{\mathbf{C}}^{-1/3-1}\mathbf{III}_{\mathbf{C}}\mathbf{C}^{-1}(\mathbf{C}:\mathbf{I})\right] + pJ\mathbf{C}^{-1} \\
&= \mu\mathbf{III}_{\mathbf{C}}^{-1/3}\left(\mathbf{I} - \frac{1}{3}\mathbf{I}_{\mathbf{C}}\mathbf{C}^{-1}\right) + pJ\mathbf{C}^{-1}
\end{aligned}$$

Le tenseur des contraintes de Cauchy correspondant peut maintenant être obtenu comme suit :

$$\begin{aligned}
\sigma &= J^{-1}\mathbf{F}\mathbf{S}\mathbf{F}^{T} \\
&= \mu J^{-5/3}\mathbf{F}\left(\mathbf{I} - \frac{1}{3}\mathbf{I}_{\mathbf{C}}\mathbf{C}^{-1}\right)\mathbf{F}^{T} + p\mathbf{F}\mathbf{C}^{-1}\mathbf{F}^{T} \qquad (3.46) \\
&= \sigma' + p\mathbf{I}; \qquad\qquad \sigma' = \mu J^{-5/3}\left(\mathbf{b} - \frac{1}{3}\mathbf{I}_{\mathbf{b}}\mathbf{I}\right)
\end{aligned}$$

où le fait que $\mathbf{I}_{\mathbf{b}} = \mathbf{I}_{\mathbf{C}}$ a été de nouveau utilisé.

On peut maintenant évaluer le tenseur d'élasticité Lagrangien avec l'aide de l'équation (3.9) ou (3.10). Le résultat peut être divisé en composantes déviatorique et de pression, $\hat{\mathbf{C}}$ et $\mathbf{C}_{p}$ respectivement, soit :

$$\mathbf{C} = 2\frac{\partial \mathbf{S}}{\partial \mathbf{C}} = \hat{\mathbf{C}} + \mathbf{C}_p; \qquad \hat{\mathbf{C}} = 2\frac{\partial \mathbf{S}'}{\partial \mathbf{C}} = 4\frac{\partial^2 \hat{\Psi}}{\partial \mathbf{C}\partial \mathbf{C}}; \qquad \mathbf{C}_p = 2p\frac{\partial (J\mathbf{C}^{-1})}{\partial \mathbf{C}}; \tag{3.47}$$

Avec l'aide des équations (3.20) et (3.29), ces deux composantes peuvent être évaluées pour le cas néo-Hookéen, défini par l'équation (3.44), comme suit :

$$\hat{\mathbf{C}} = 2\mu \mathrm{III}_{\mathbf{C}}^{-1/3}\left[\frac{1}{3}\mathrm{I}_{\mathbf{C}}\mathrm{III}-\frac{1}{3}\mathbf{I}\otimes\mathbf{C}^{-1}-\frac{1}{3}\mathbf{C}^{-1}\otimes\mathbf{I}+\frac{1}{9}\mathrm{I}_{\mathbf{C}}\mathbf{C}^{-1}\otimes\mathbf{C}^{-1}\right] \tag{3.48a}$$

$$\mathbf{C}_p = pJ\left[\mathbf{C}^{-1}\otimes\mathbf{C}^{-1}-2\mathbf{I}\right] \tag{3.48b}$$

Notons que la composante de pression $\mathbf{C}_p$ ne dépend pas de la définition du matériau spécifique utilisé.

Le tenseur d'élasticité spatial est obtenu par le transport convectif avant illustré à l'équation (3.13) soit :

$$\mathbf{c} = \hat{\mathbf{c}} + \mathbf{c}_p; \qquad \hat{\mathbf{c}} = J^{-1}\phi_*\left[\hat{\mathbf{C}}\right] \qquad \mathbf{c}_p = J^{-1}\phi_*\left[\mathbf{C}_p\right] \tag{3.49}$$

En appliquant la transformation convective avant aux équations (3.48a,b) on obtient :

$$\hat{\mathbf{c}} = 2\mu \mathbf{J}^{-5/3}\left[\frac{1}{3}\mathrm{I}_{\mathbf{C}}\mathbf{i}-\frac{1}{3}\mathbf{b}\otimes\mathbf{I}-\frac{1}{3}\mathbf{I}\otimes\mathbf{b}+\frac{1}{9}\mathrm{I}_{\mathbf{b}}\mathbf{I}\otimes\mathbf{I}\right] \tag{3.50a}$$

$$\mathbf{c}_p = \mathbf{p}[\mathbf{I}\otimes\mathbf{I}-2\mathbf{i}] \tag{3.50b}$$

3.5.3 Matériaux hyperélastiques presqu'incompressibles

Ce type de matériau est souvent utilisé pour l'implémentation de l'incompressibilité dans une formulation par éléments finis. Ceci est facilité par l'ajout d'une composante d'énergie volumétrique $U(J)$ à la composante de distorsion $\bar{\Psi}$ afin d'obtenir la fonction d'énergie de déformation totale soit :

$$\Psi(\mathbf{C}) = \bar{\Psi}(\mathbf{C}) + U(J) \tag{3.51}$$

où l'exemple le plus simple d'une fonction volumétrique est donnée par :

$$U(J)=\frac{1}{2}\kappa(J-1)^2 \tag{3.52}$$

avec κ un paramètre de pénalité qui, lorsque il avoisine 10^3 - 10^4 μ, permet d'obtenir un comportement presqu'incompressible. Cependant, κ peut représenter une quantité physique, appelée module de compressibilité volumétrique (bulk modulus), pour un matériau compressible qui possède une fonction d'énergie de déformation hyperélastique de la forme donnée aux équations (3.51) et (3.52).

Le second tenseur des contraintes de Piola-Kirchhoff pour un matériau défini par l'équation (3.51) est obtenu de façon standard à l'aide de l'équation (3.20), en notant que $\mathbf{III}_{\mathbf{C}} = J^2$, et s'écrit comme suit :

$$\begin{aligned}\mathbf{S} &= 2\frac{\partial \Psi}{\partial \mathbf{C}} \\ &= 2\frac{\partial \bar{\Psi}}{\partial \mathbf{C}} + 2\frac{\partial U}{\partial J}\frac{\partial J}{\partial \mathbf{C}} \\ &= 2\frac{\partial \bar{\Psi}}{\partial \mathbf{C}} + pJ\mathbf{C}^{-1}\end{aligned} \tag{3.53}$$

où, en comparant avec (3.41), nous identifions la pressions comme :

$$p=\frac{\partial U}{\partial J} \tag{3.54}$$

En remplaçant $U(J)$ par l'expression dans (3.52) on obtient :

$$p=\kappa(J-1) \tag{3.55}$$

Cette valeur de la pression peut être remplacée dans l'équation générale (3.53) ou dans l'équation (3.45) pour le cas néo-Hookéen pour l'obtention du second tenseur de Piola-Kirchhoff. De même, dans le cas néo-Hookéen, p peut être remplacée dans l'équation (3.46) pour obtenir la contrainte de Cauchy.

Afin de conclure la description de ce type de matériau, il nous reste à dériver les expressions des tenseurs d'élasticité spatial et Lagrangien.

Le tenseur d'élasticité Lagrangien peut être subdivisé en trois composantes. On a :

$$\mathbf{C} = 4\frac{\partial \hat{\Psi}}{\partial \mathbf{C}\partial \mathbf{C}} + 2p\frac{\partial\left(J\mathbf{C}^{-1}\right)}{\partial \mathbf{C}} + 2J\mathbf{C}^{-1} \otimes \frac{\partial p}{\partial \mathbf{C}} \tag{3.56}$$

Les deux premières composantes dans cette expression sont $\bar{\mathbf{C}}$ et $\mathbf{C}_p$. Le dernier terme, à savoir $\mathbf{C}_\kappa$, représente une composante de la matrice tangente volumétrique. Avec l'équation (3.20) $\mathbf{C}_\kappa$ devient :

$$\begin{aligned} \mathbf{C}_\kappa &= 2J\mathbf{C}^{-1} \otimes \frac{\partial p}{\partial \mathbf{C}} \\ &= 2J\frac{d^2U}{dJ^2}\mathbf{C}^{-1} \otimes \frac{\partial J}{\partial \mathbf{C}} \\ &= J^2\frac{d^2U}{dJ^2}\mathbf{C}^{-1} \otimes \mathbf{C}^{-1} \end{aligned} \tag{3.57}$$

De plus, avec $U(J) = \frac{1}{2}\kappa(J-1)^2$ on a :

$$\mathbf{C}_\kappa = \kappa J^2\mathbf{C}^{-1} \otimes \mathbf{C}^{-1} \tag{3.58}$$

Enfin, le tenseur d'élasticité spatial est obtenu par la transformation convective avant qui donne :

$$\mathbf{c} = J^{-1}\phi_*[\mathbf{C}] = \bar{\mathbf{c}} + \mathbf{c}_p + \mathbf{c}_\kappa$$

(3.59)

où les composantes déviatorique et de pression, $\bar{\mathbf{c}}$ et $\mathbf{c}_p$ respectivement, sont identiques à celles dérivées à la section précédente et la composante volumétrique $\mathbf{c}_\kappa$ est donnée par :

$$\mathbf{c}_\kappa = J^{-1}\phi_*[\mathbf{C}_\kappa] = J\frac{d^2U}{dJ^2}\mathbf{I} \otimes \mathbf{I} \tag{3.60}$$

et avec (3.52) on obtient :

$$\mathbf{c}_{\kappa} = \kappa \mathbf{I} \otimes \mathbf{I} \tag{3.61}$$

3.6 Principe des travaux virtuels

En vue de l'application de la méthode des éléments finis basée sur la formulation faible ou variationnelle, nous développons ci-après les principales expressions des principes variationnels en mécanique des solides en grandes déformations. La forme variationnelle associée aux équations d'équilibre (2.53) obtenue à partir d'une formulation de type Galerkin avec comme fonction de pondération un champ de déplacement virtuel δv, s'exprime dans la configuration actuelle C_t (Fig. 3.1), (configuration courante) comme suit :

$$\delta W := \delta W_{int} - \delta W_{ext} = 0 \tag{3.62}$$

avec :

$$\delta W = \int_V (div\sigma + f)\delta v dV = 0 \tag{3.63}$$

Cette expression peut être développée suivant différentes formulations.

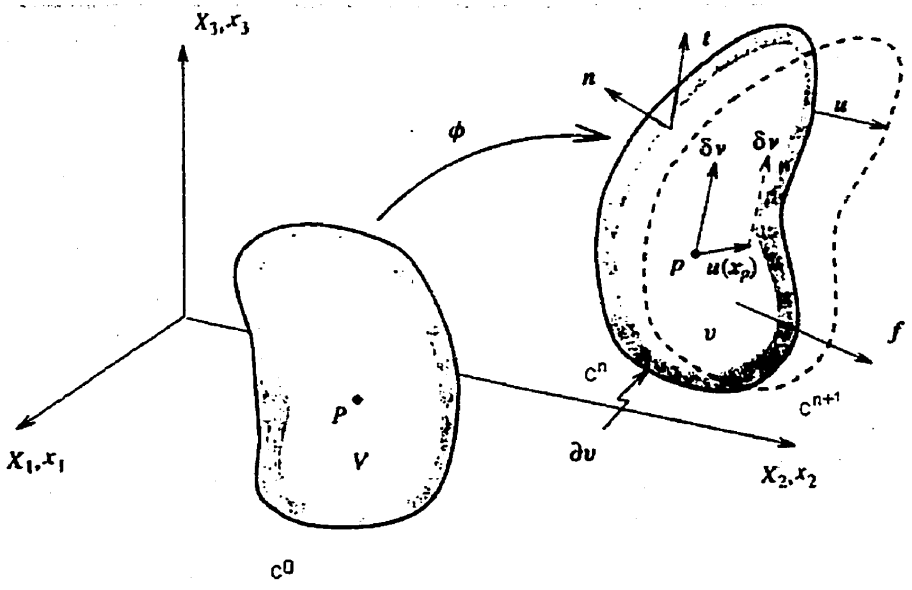


Figure 3.1 Configuration C^0, C^n et C^{n+1}

3.6.1 Formulation spatiale

Dans la formulation spatiale, la configuration courante C_t sert de référence.

Une intégration par partie de l'équation précédente donne :

$$\delta W := -\int_V \sigma : \nabla \delta v dV + \int_{\partial V} \bar{n}.\sigma \delta v dA + \int_V \bar{f} \delta v dV = 0$$

(3.64)

Du fait de la symétrie de σ on a :

$$\sigma : \nabla_x \delta v = \sigma : \delta d \tag{3.65}$$

avec $\delta d = \frac{1}{2}\left(\delta l + \delta l^T\right)$ on obtient donc:

$$\delta W := -\int_{V_t} \sigma.\delta d dV + \int_{V_t} \bar{f} \delta v dV + \int_{\partial V_t} t.\delta.\delta v dA = 0$$

(3.66)

soit :

$$\delta w_{int} = \int_{V_t} \sigma.\delta d dV \tag{3.67}$$

et :

$$\delta w_{ext} = \int_{V_t} \bar{f} \delta v dV + \int_{\partial V_t} t.\delta.\delta v dA$$

(3.68)

3.6.2 Formulation Lagrangienne Totale

Dans la formulation Lagrangienne Totale, la configuration initiale C^0 sert de référence.

On a : $dV_t = J.dV$, $f_0 = J.f$, $t_0 = t.\left(\frac{da}{dA}\right)$ avec $\frac{da}{dA} = \frac{J}{\sqrt{n.b.n}}$.

On tire que

$$\delta W := \delta W_{int} - \delta W_{ext} = \int_V J\sigma : \delta dV - \int_V \vec{f}_0 \delta v dV - \int_{\partial V} t_0 . \delta v dA = 0 \qquad (3.69)$$

soit :

$$\delta w_{int} = \int_V J\sigma : \delta d dV \text{ , avec } \tau = J.\sigma \qquad (3.70)$$

avec : $\frac{1}{\rho}\sigma : d = \frac{1}{\rho_0}\tau : d$

on tire que :

$$\delta w_{int} = \int_V \tau : \delta d dV \qquad (3.71)$$

où τ est la contrainte de Kirchhoff.

et :

$$\delta w_{ext} = \int_V \vec{f}_0 \delta v dV + \int_{\partial V} t_0 . \delta v dA \qquad (3.72)$$

3.7 Méthode de résolution: linéarisation des travaux virtuels

Pour modéliser la mise en forme, on assimile le corps soumis aux conditions limites à un milieu déformable dans une configuration C^i qui occupe un volume V^i de contour S^i. Les conditions aux limites du problème peuvent être définies par des déplacements imposés sur une partie du contour S_u et des efforts imposés sur la partie complémentaire S_f. Ces efforts représentent la pression de mise en forme.

Le problème à résoudre consiste donc à déterminer les différentes configurations de la structure au cours de la déformation. Dans le domaine des procédés de mise en forme, l'approche la plus utilisée est l'approche incrémentale qui consiste à discrétiser le chargement en plusieurs incréments et à déterminer les configurations C^1, C^2, ..., C^n correspondant aux différents pas de chargement. Après une discrétisation spatiale du principe des travaux virtuels par la méthodes des éléments finis, on obtient l'équation d'équilibre sous la forme suivante :

$$\mathbf{Res(u)} = \mathbf{F_{int}(u)} - \mathbf{F_{ext}(u)} \qquad (3.73)$$

avec **Res** le résidu d'équilibre, $\mathbf{F}_{ext}$ le vecteur des efforts externes et $\mathbf{F}_{int}$ le vecteur des forces internes. L'équation (3.73) est en général fortement non linéaire.

Parmi les algorithmes développés pour la résolution de ces équations non linéaires, nous utilisons le schéma de Newton-Raphson.

3.7.1 Schéma de Newton-Raphson

Le schéma de Newton-Raphson utilise une technique incrémentale qui consiste à discrétiser le chargement total en plusieurs incréments et à chercher les configurations $C^1, C^2, \ldots\ldots, C^n$ correspondant aux différents pas de chargement. La détermination de la configuration C^{n+1} à partir de la configuration C^n se fait de manière itérative. En effectuant un développement limité de premier ordre de $\mathbf{R}(\mathbf{u})$ au voisinage d'une solution approchée $\mathbf{u}^i$, on obtient :

$$L\left[Res\left(u^i + \Delta u^i\right)\right] = L\left(f_{int}\left(u^i\right) - f_{ext}\left(u^i\right)\right) = 0,$$

$$L\left[Res\left(\mathbf{u}^i + \Delta\mathbf{u}^i\right)\right] = Res\,\big|_{\mathbf{u}^i} + \frac{\partial Res\left(\mathbf{u}^i\right)}{\partial \mathbf{u}^i}\Delta\mathbf{u}^i = 0 \qquad (3.74)$$

L'incrément de déplacement $\Delta\mathbf{u}^i$ qui annule le résidu d'équilibre est obtenu par la résolution du système d'équations suivant :

$$K_T\left(\mathbf{u}^i\right)\Delta\mathbf{u}^i = Res\left(\mathbf{u}^i\right)$$

$$K_T\left(\mathbf{u}^i\right) = -\frac{\partial Res\left(\mathbf{u}^i\right)}{\partial \mathbf{u}^i} \qquad (3.75)$$

avec K_T la matrice tangente.

On obtient ainsi une estimation du déplacement de la configuration C^{n+1},

$$\mathbf{u}^{i+1} = \mathbf{u}^i + \Lambda\mathbf{u}^i \qquad (3.76)$$

La méthode de Newton-Raphson est associée à différentes techniques de pilotage. Celle implémentée comme on le verra au chapitre 6 est celle en déplacement. Ces techniques sont développées plus en détails par [Ref-1], [Ref-2], [Ref-3] et [Ref-4].

Comme nous l'avons dit plus haut, l'utilisation de la méthode de Newton-Raphson requiert une linéarisation de l'équation du système autour d'une position d'équilibre.

3.7.2 Linéarisation de l'équation d'équilibre

La linéarisation de l'équation d'équilibre est nécessaire à l'implémentation de notre méthode de résolution. Le principe des travaux virtuels discuté plus haut nous permet d'écrire :

$$\delta W(\phi, \delta v) := \int_{V_t} \sigma . \delta d dV - \int_{V_t} \bar{f} \delta v dV - \int_{\partial V_t} t . \delta v dA = 0 = \delta W_{\text{int}} + \delta W_{ext} \tag{3.77}$$

où ϕ_k est le champ de déplacement et δv est le déplacement virtuel admissible i.e. compatible avec les conditions aux limites essentielles.

La linéarisation dans la direction de l'incrément de déplacement u dans ϕ_k donne :

$$L\delta W(\phi, \delta v)[u] = \delta W(\phi, \delta v)|_{\phi_k} + D\delta W(\phi, \delta v)[u] = 0 \tag{3.78}$$

où D()[u] est la dérivée directionnelle en direction de u. Le processus de linéarisation requiert une configuration de référence fixe afin d'obtenir de bon résultats.

- **Linéarisation des efforts internes**
  - **Formulation matérielle**

On a :

$$\delta W_{\text{int}} = \int_V S : \delta \dot{E} dV \tag{3.79}$$

sa dérivée est donnée par :

$$D\delta W_{\text{int}}(\phi, \delta v)[u] = \oint D(\delta \dot{E} : S)[u] dV$$

$$= \oint_V \delta \dot{E} : DS[u] dV + \oint_V S : D\delta \dot{E}[u] dV , \tag{3.80}$$

où,

$$DE[u]=\frac{1}{2}F(\nabla_X u+(\nabla_X u)^T)F\,, \tag{3.81}$$

Puisque :

$$\delta\dot{E}=\frac{1}{2}\left(\delta\dot{F}^T F+F^T\delta\dot{F}\right),\ \text{avec}\ \delta\dot{F}=\frac{\partial\delta v}{\delta X}\quad=\nabla_X\delta v \tag{3.82}$$

alors :

$$D\delta\dot{E}[u]=\frac{1}{2}\left[(\nabla_X\delta v)^T\nabla_X u+(\nabla_X u)^T\nabla_X\delta v\right] \tag{3.83}$$

Cependant les vitesses virtuelles δv ne sont pas fonction de la configuration, ainsi $\nabla_X\delta v$ est constante et en utilisant la symétrie de S on obtient :

$$D\delta W_{\text{int}}(\phi,\delta v)[u]=\oint_V \delta\dot{E}:H:DE[u]dV+\oint_V S:\left[(\nabla_X u)^T\nabla_X\delta v\right]\,dV \tag{3.84}$$

Ou encore :

$$D\delta W_{\text{int}}(\phi,\delta v)[u]=\oint_V DE[\delta v]:H:DE[u]dV+\oint_V S:\left[(\nabla_X u)^T\nabla_X\delta v\right]\,dV \tag{3.85}$$

avec une partie matérielle :

$$D\delta W_{\text{int}}(\phi,\delta v)[u]=\oint_V DE[\delta v]:H:DE[u]dV \tag{3.86}$$

et une partie géométrique

$$D\delta W_{\text{int}}(\phi,\delta v)[u]=\oint_V S:\left[(\nabla_X u)^T\nabla_X\delta v\right]\,dV \tag{3.87}$$

- **Formulation spatiale**

Soit les relations de la transformation convective arrière / avant suivantes :

$DE[u]=\phi_*^{-1}[\varepsilon]=F^T\varepsilon F\,,$ avec $2\varepsilon=\nabla_X u+(\nabla_X u)^T$

$DE[\delta v]=\phi_*^{-1}[\delta d]=F^T\delta dF\,,$ avec $2\delta d=\nabla_X\delta v+(\nabla_X\delta v)^T$

$J\sigma=\phi_*[S]=FSF^T$

$Jh=\phi_*[H]$, avec H = **C** et h = ς

$$JdV = dv$$

alors :

$$DE[\delta v] : H : DE[u] dV = \delta d : h : \varepsilon \; dv \tag{3.88}$$

De plus, puisque le gradient matériel est relié au gradient spatial en utilisant la relation suivante :

$$\nabla_X u = (\nabla_x u) F$$

$$\nabla_X \delta v = (\nabla_x \delta v) F$$

alors le terme géométrique devient:

$$S : \left[(\nabla_X u)^T \nabla_X \delta v \right] = \sigma : \left[(\nabla_x u)^T \nabla_x \delta v \right] \tag{3.89}$$

Ainsi finalement la linéarisation des efforts internes en configuration spatiale donne :

$$D\delta W_{\text{int}}(\phi, \delta v)[u] = \int_{V_t} \delta d : h : \varepsilon \; dV_t + \int_{V_t} \sigma : \left[(\nabla_x u)^T \nabla_x \delta v \right] dV_t \tag{3.90}$$

Note: Puisque les relations fonctionnelles de, δd et ε sont identiques alors l'expression suivante est symétrique:

$$DW_{\text{int}}(\phi, \delta v)[u] = DW_{\text{int}}(\phi, u)[\delta v] \tag{3.91}$$

Ce qui permet d'avoir une matrice tangente symétrique.

- **Linéarisation des efforts externes**
  - Forces de volume:

On distingue parmi cette catégorie de chargements externes la force gravitationnelle donnée par :

$$f = \rho \bar{g} .$$

Ainsi
$$D\delta w_{ext}{}^{f}(\phi, \delta v)[u] = 0 \tag{3.92}$$

puisque :

$$\delta w_{ext}{}^{f}(\phi, \delta v) = \int_v \frac{\rho_0}{J} \bar{g} . \delta v \; dv = \int_V \rho_0 \bar{g} . \delta v \; dV \tag{3.93}$$

- Forces de surfaces :

Elles correspondent à différentes situations comme les forces de contact. Comme notre pilotage se fera en pression, supposons une pression de surface uniforme, alors on a :

$$w_{ext}{}^{p}(\phi,\delta v)=\int_{A} p\vec{n}.\delta v \; da \tag{3.94}$$

avec :

une paramétrisation surfacique comme suit : $\vec{x}(\xi,\eta)$, on a :

$$\vec{n}=\frac{\dfrac{\partial\vec{x}}{\partial\xi}\times\dfrac{\partial\vec{x}}{\partial\eta}}{\left\|\partial_{\xi}x\times\partial_{\eta}x\right\|} \tag{3.95}$$

$$da=\left\|\partial_{\xi}x\times\partial_{\eta}x\right\|d\xi \; d\eta \tag{3.96}$$

D'où :

$$\delta W_{ext}{}^{p}(\phi,\delta v)=\int_{A} p\vec{n}.\delta v \; da=\int_{A} p\delta v\frac{\partial\vec{x}}{\partial\xi}\times\frac{\partial\vec{x}}{\partial\eta}d\xi d\eta \tag{3.97}$$

Dans l'équation ci-dessus les seuls termes qui sont dépendants du déplacement sont : $\frac{\partial\vec{x}}{\partial\xi}$ and $\frac{\partial\vec{x}}{\partial\eta}$ dont la linéarisation donne :

$$D\frac{\partial\vec{x}}{\partial\xi}[u]=\frac{\partial\vec{u}}{\partial\xi} \text{ et } D\frac{\partial\vec{x}}{\partial\eta}[u]=\frac{\partial\vec{u}}{\partial\eta}. \tag{3.98}$$

Alors la dérivée directionnelle devient :

$$D\delta w_{ext}{}^{p}(\phi,\delta v)[u]=\int_{A_\xi} p\left[\frac{\partial\vec{x}}{\partial\xi}\cdot\left(\frac{\partial\vec{u}}{\partial\eta}\times\delta\vec{v}\right)-\frac{\partial\vec{x}}{\partial\eta}\cdot\left(\frac{\partial\vec{u}}{\partial\xi}\times\delta\vec{v}\right)\right]d\xi d\eta \tag{3.99}$$

En général, δv et u ne commutent pas.. Sinon, on obtiendraient une matrice tangente asymétrique.

L'utilisation de la permutation de produits triples ainsi que du théorème d'intégration permet d'écrire :

$$\begin{aligned} D\delta w_{ext}{}^{P}(\phi,\delta v)[u] &= \int_{A_\xi} p\left[x_{,\xi}\cdot(\delta v_{,\eta}\times u)-x_{,\eta}\cdot(\delta v_{,\xi}\times u)\right]d\xi d\eta \\ &+ \oint_{\partial A_\xi} p(u\times\delta v)\cdot(\upsilon_\eta x_{,\xi}-\upsilon_\xi x_{,\eta})dl \end{aligned} \tag{3.100}$$

où : $(\upsilon_\eta,\upsilon_\xi)=\bar{\upsilon}$: sont les valeurs d'un plan paramétrique normal à ∂A_ξ.

La loi de comportement, la formulation et sa linéarisation cohérente seront appliquées dans le contexte de problèmes de contact. Le chapitre suivant nous permet de définir comment modéliser et structurer de tels problèmes.

CHAPITRE IV

MODÉLISATION DU PROBLÈME DE CONTACT

4.1 Introduction

Les problèmes de contact forment un très vaste domaine d'investigation en mécanique des solides, l'étude du comportement des matériaux en contact constitue une science à part entière, la tribologie. Ces problèmes sont fréquemment rencontrés dans le domaine industriel et ceci a toujours fait l'objet d'un investissement scientifique important et surtout au cours des dix dernières années. Aujourd'hui, il existe plusieurs codes éléments finis [DYNA-3D], [ABAQUS] qui prennent en compte les phénomènes de contact et de frottement, ce qui a permis d'apporter plusieurs éléments de réponse aux problèmes industriels.

Les difficultés principales du problème de contact proviennent du fait que les conditions aux limites liées au contact ne sont pas connues à l'avance et qu'elles dépendent de la solution du problème (conditions aux limites évolutives). Ainsi la surface réelle de contact et les réactions de contact font partie des inconnues du problème. En présence des grandes transformations, les non linéarités provenant du contact s'ajoutent aux non linéarités géométrique et matérielle, rendant ainsi la modélisation du problème très complexe.

L'étude et la résolution mathématique du problème de contact date de 1882 avec les travaux de HERTZ. Ces derniers permettent de traiter des cas de charges linéiques ou ponctuelles

à partir d'une démarche analytique qui s'appuie sur l'élasticité linéaire pour évaluer l'écrasement, la surface de contact et les contraintes générées lors du contact de deux corps. Dans ce contexte, la plupart des solutions analytiques proposées supposent un contact sans frottement, des zones de contact connues à priori et des formes géométriques simples. Ces travaux sont d'autant plus célèbres que les solutions analytiques sont rares, et de fait ils sont très fréquemment employés pour la validation de méthodes numériques.

En 1933, SIGNORINI pose le problème général de l'équilibre d'un solide élastique linéaire en contact sans frottement avec une fondation rigide, mais il faut attendre les années soixante, avec les travaux de Duvaut et Lions [Ref-5] sur les inéquations variationnelles pour que les résultats d'existence et d'unicité puissent être définitivement établis. Dès lors, dans les années 70, de nombreuses méthodes numériques voient le jour.

Le calcul variationnnel et le développement de techniques avancées de résolution numérique ont permis de traiter des problèmes de contact plus complexes. La méthode des éléments finis, en permettant la discrétisation des surfaces de formes quelconques et la prise en compte aisée de conditions aux limites diverses, offre un outil puissant de calcul pour étudier les problèmes de contact.

Ces techniques s'appuient sur une modélisation simplifiée des phénomènes de contact et s'inspirent bien souvent des méthodes employées en dynamique des structures. Cependant, d'une façon générale, elles se heurtent à des difficultés qui sont, d'une part de traiter des conditions de contact de façon simple et, d'autre part de détecter les zones de contact avec efficacité. Les positions de ces dernières font partie des inconnues du problème et il est nécessaire de les surveiller à chaque pas de temps.

Dans la suite de ce chapitre, nous présenterons une formulation générale du problème de contact, la méthode utilisée pour le calcul des réactions de contact, et ensuite, nous dresserons un panorama des algorithmes de recherche de contact. La formulation du contact choisie correspond à celle développée principalement à travers les travaux des références suivantes: [Ref-12], [Ref-13], [Ref-41] et [Ref-42].

4.2 Problème de contact

4.2.1 Cas général

Dans ce paragraphe, nous allons présenter le problème de contact tel qu'il se pose d'un point de vue mécanique. Nous considérons ici le problème du contact entre un obstacle rigide ou flexible (moule, ou die) et un ou plusieurs corps déformables (ex : feuille d'acier). Nous ferons ensuite un descriptif des contraintes de contact et une présentation de la formulation variationnelle correspondante.

Pour simplifier l'exposé, nous considérons le cas de deux solides B_1 et B_2 de contours $\partial\Omega_1$ et $\partial\Omega_2$ (Fig. 4.1). B_1 et B_2 sont en contact si une partie de leur contour est commune :

$$\partial\Omega_1 \cap \partial\Omega_2 \neq \phi \tag{4.1}$$

où ϕ représente l'ensemble vide.

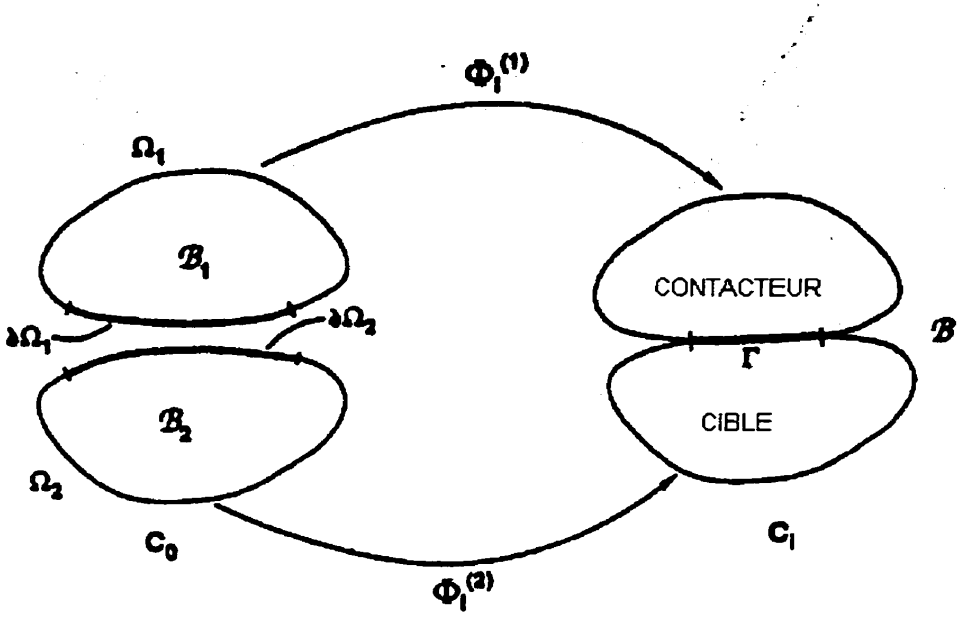


Figure 4.1 : Corps déformables en contact

Le contact crée un nouveau corps $B = B_1 \cup B_2$. Le processus de contact se traduit en général par un glissement relatif des surfaces en contact, ce qui cause des contraintes en cisaillement dues à la friction. La surface de contact du nouveau corps sera dénotée par $\Gamma = \partial\Omega_1 \cup \partial\Omega_2$. Sur les surfaces($\partial\Omega_1$ et $\partial\Omega_2$), la statique et la cinématique ont totalement changé. Avant le contact, les surfaces sont considérées comme des surfaces libres, où la cinématique n'est soumise à aucune contrainte et, comme conditions aux limites, les tractions y sont nulles. Après qu'il y ait eu contact, la cinématique est soumise à des contraintes de déformation et, par conséquent, il y a présence de forces de contact sur la surface de contact. Ces tractions de contact peuvent être assimilées à des chargements sur les corps B_1 et B_2 afin de déterminer la cinématique complète de contact.

Nous utiliserons le concept de contacteur et de cible (slave-master) afin de modéliser les paires de corps en contact. C'est l'un des concepts les plus largement utilisés dans la littérature. Ainsi le solide B_1 est le contacteur et les entités matérielles sur $\partial\Omega_1$ serviront à définir les nœuds contacteurs tandis que le solide B_2 est la cible et les entités matérielles sur $\partial\Omega_2$ serviront à définir les segments cibles. Ainsi, une paire de contact constituée d'un nœud sur le contacteur et du segment sur la cible en contact avec lui, servira à définir un élément de contact. Dans les sections suivantes, tel est le type d'élément qui nous permettra de discrétiser le domaine en contact.

4.2.2 Notion de fonction de séparation

Pour une configuration donnée C^i, $\gamma^{(2)} = \Phi_i^{(2)}(\partial\Omega_2)$ définit une surface que toutes les entités matérielles de $\partial\Omega_1$ ne peuvent pas pénétrer. Une telle restriction peut être également imposée sur tout point de $\partial\Omega_2$ par rapport à $\partial\Omega_1$, la distinction des surfaces est désuète dans une analyse tenant compte du continuum de la surface de contact.

Considérons maintenant la Fig. 4.2 qui décrit le mouvement d'une paire de points en contact, composée du nœud contacteur P_s et du point matériel associé P_c, dans un incrément de temps Δt.

Les points sont marqués avec les notations $^{(0)}$ et $^{(i)}$ pour désigner respectivement la position des points au début et à la fin de l'incrément de temps. On réalise que durant l'incrément de temps le nœud contacteur a pénétré la cible et ainsi a violé la cinématique du contact. Le point physique (ou réel) de contact est cependant défini par l'algorithme de projection vers le point le plus proche ou "closest point projection" sur la cible en direction du vecteur unitaire normal $\mathbf{n}$.

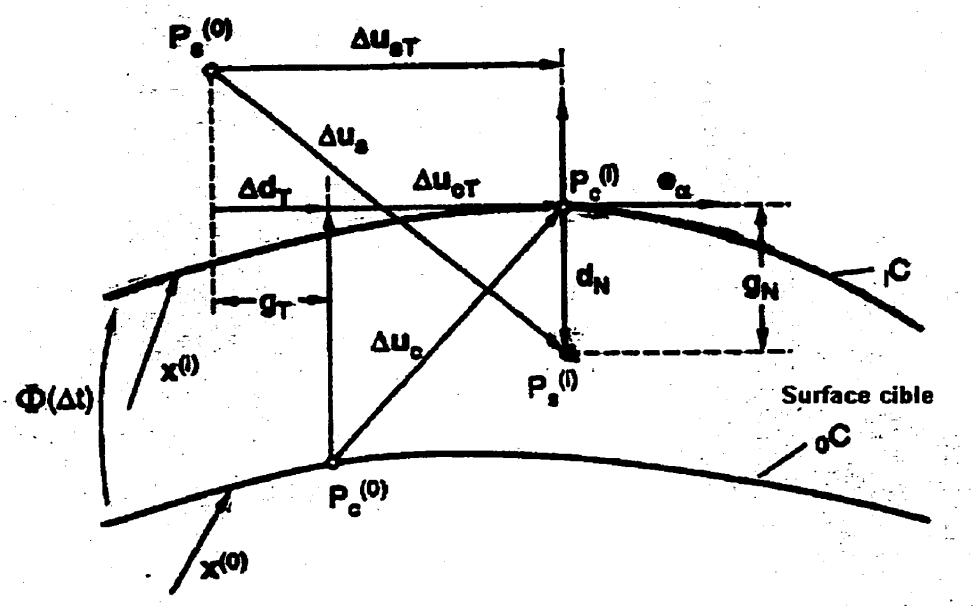


Figure 4.2 : Mouvement d'une paire de nœuds en contact P_s et P_c

De cette illustration, on voit clairement que le point P_c ne sera pas de façon générale un nœud du maillage d'éléments finis, ce qui rend d'autant plus complexe le terme de la matrice de rigidité élémentaire comme on le verra plus loin.

Le mouvement du corps en contact est séparé en un mouvement par rapport à la direction normale et en un autre par rapport à la direction tangentielle. Ainsi les fonctions de séparation associées sont,

pour la fonction de séparation normale :

$$g_N = \mathbf{n} \cdot \mathbf{d}_N = \mathbf{n} \cdot (\mathbf{x}_s^{(i)} - \mathbf{x}_c^{(i)}) \tag{4.2}$$

où $\mathbf{x}_s^{(i)}$ et $\mathbf{x}_c^{(i)}$ sont les vecteurs positions spatiales des points associés. On en déduit la variation de la fonction de séparation normale, donnée par :.

$$\delta g_N = \mathbf{n} \cdot (\delta \mathbf{u}_s - \delta \mathbf{u}_c) + \delta \mathbf{n} \cdot \mathbf{d}_N = \mathbf{n} \cdot \delta \mathbf{u} \tag{4.3}$$

où on a utilisé le fait que le terme $\delta \mathbf{n} \cdot \mathbf{d}_N$ s'annule à cause de l'orthogonalité et que $\delta \mathbf{u}$ représente un mouvement relatif.

Analogiquement on obtient, pour la fonction de séparation tangentielle :

$$g_T = \mathbf{s} \cdot \Delta \mathbf{u}_r = \mathbf{s} \cdot (\Delta \mathbf{u}_s - \Delta \mathbf{u}_c) \tag{4.4}$$

avec $\Delta \mathbf{u}_r$ représentant le vecteur de déplacement relatif incrémental et :

$$\mathbf{s} = \frac{\Delta \mathbf{d}_T}{\|\Delta \mathbf{d}_T\|} = \frac{1}{g_T} \Delta \mathbf{d}_T \tag{4.5}$$

le déplacement tangentiel normalisé du vecteur $\Delta \mathbf{d}_T$.

Bien que leurs définitions semblent similaires, les deux fonctions de séparations sont cependant différentes lorsqu'on considère leurs sens physiques à l'intérieur d'une application numérique. La fonction de séparation normale g_N représente la grandeur de la pénétration et par conséquent c'est une mesure de la violation de la cinématique de contact. Elle servira pour définir les limitations sur la cinématique, en particulier la condition de non inter-pénétrabilité des corps. La séparation normale dépend donc exclusivement de la position spatiale actuelle des nœuds/points en contact et non de leur histoire de déplacement. Ceci sera exploité à bon escient dans le traitement du contact normal sans frottement. La fonction de séparation tangentielle g_T, quant à elle, représente la grandeur du glissement tangentiel relatif de deux corps en contact. En fonction de la loi de frottement choisie, elle servira à définir une limitation sur la cinématique seulement s'il y a adhésion (sticking contact) et est donc reliée à la présence d'une force. La séparation tangentielle g_T est exprimée en termes du vecteur déplacement des points en contact obtenu dans un calcul incrémental. Et ceci est une des causes de la non symétrie inhérente à l'opérateur matériel tangent en frottement. Les autres sources de non symétrie pouvent être le changement de plan tangent pendant le glissement ou une loi de frottement non associée lorsqu'on fait l'analogie avec la plasticité.

La condition d'impénétrabilité est formulée via la fonction de la fonction de séparation g.

4.2.3 Les limitations imposées par le contact

D'un point de vue mathématique les problèmes de contact tombent dans la classe de problèmes avec conditions aux limites soumis à des limitations cinématique et/ou statique supplémentaires . Aussi on distingue les conditions de contraintes suivantes :

- Efforts de contact : Avant qu'il y ait contact, les deux surfaces sont considérées libres et après le contact, la cinématique est alors soumise à des contraintes. Ainsi, des forces de traction se développent sur la surface de contact. Ces tractions sont considérées comme des charges appliquées aux corps associés aux surfaces de contact en vue de satisfaire la cinématique du contact ou soit pour empêcher la pénétration du nœud contacteur. Par analogie avec la définition des fonctions de séparations, on définit les forces de contact normale et tangentielle conjugués à ces variables cinématiques :

$$\mathbf{t}_N = t_N \mathbf{n} \tag{4.6}$$

et

$$\mathbf{t}_T = t_T \mathbf{f} \tag{4.7}$$

où $\mathbf{f}$ est le vecteur unitaire en direction des forces de friction. Comme ici l'orientation des forces est donnée par celle des vecteurs $\mathbf{n}$ et $\mathbf{f}$, la force normale sera interprétée comme une force extérieure appliquée au corps pour empêcher la pénétration du nœud esclave.

- Conditions d'impénétrabilité: Les conditions d'orthogonalité ou encore de Kuhn-Tucker pour le contact normal peuvent se résumer comme suit:

$$\begin{aligned} g_N &\geq 0 \\ t_N &\leq 0 \\ t_N g_N &= 0 \end{aligned} \tag{4.8}$$

La première équation définit la condition d'impénétrabilité et signifie que les corps peuvent se séparer mais ne peuvent pas se pénétrer.
La seconde équation définit la pression de contact qui, selon la troisième équation, est non-nulle seulement si la fonction de séparation devient nulle, i.e. s'il y a contact.

- Contribution du contact à l'équation d'équilibre :

  L'équation d'équilibre d'un corps est défini par l'équation du travail virtuel, où seules les forces introduites via le contact sont considérées dans le développement qui suit. Le travail virtuel de contact est donné par :

$$G_c(\Phi, \delta\mathbf{u}) = \mathbf{t}_N \cdot \delta\mathbf{u} + \mathbf{t}_T \cdot \delta\mathbf{u} \tag{4.9}$$

  où $\delta\mathbf{u} = (\delta\mathbf{u}_s - \delta\mathbf{u}_c)$ sont des variations admissibles des déplacements de la paire de nœuds en contact pour la configuration fixée. En introduisant les expressions des forces de contact le travail virtuel devient :

$$G_c(\Phi, \delta\mathbf{u}) = t_N(\mathbf{n} \cdot \delta\mathbf{u}) + t_T(\mathbf{f} \cdot \delta\mathbf{u}) \tag{4.10}$$

4.2.4 Lois de comportement du contact

La relation de dépendance des forces de contact par rapport à la cinématique du contact est définie par la loi de comportement du contact. Les conditions de contact sont introduites comme des contraintes dans la minimisation de l'énergie potentielle du système. La solution est obtenue par la minimisation de la fonctionnelle à l'aide de techniques d'optimisation issues de la programmation mathématique [Ref-6]. L'efficacité de ces méthodes est encore controversée, et elles sont peu utilisées dans les logiciels classiques de calcul de structures. Cependant les nouveaux développements ont tendance à adopter les algorithmes tirés de la programmation mathématique.

- *Réactions normales sur les corps en contact* :

  Dans une implémentation par éléments finis, la *méthode des pénalités* est la procédure la plus simple et la plus facile d'utilisation pour étudier l'analyse du contact. Elle consiste à introduire les réactions de contact dans la résolution du problème global, comme des fonctions explicites de déplacements nodaux inconnus. Malgré les problèmes inhérents dans la sélection de coefficients de pénalité adéquats, cette méthode est perçue comme une des plus convenables dans la résolution des problèmes d'optimisation avec contraintes.

On définit alors les coefficients de pénalité $\varepsilon_N > 0$ et $\varepsilon_T > 0$ pour pénaliser les fonctions de séparation dans les directions normale et tangentielle. Ces coefficients représentent physiquement la rigidité de ressorts fictifs orientés dans la direction normale et tangentielle du repère local de contact. *Les valeurs de ces coefficients résultent d'un compromis : elles doivent être à la fois suffisamment grandes pour assurer la non interpénétration mais modérer pour éviter un mauvais conditionnement de la matrice tangente.* Une méthode basée sur l'algorithme du lagrangien augmenté permet de limiter le degré d'arbitraire dans le choix des coefficients de pénalité et d'obtenir la convergence même avec de faibles valeurs des coefficients de pénalité.

Une estimation optimale de ε_N est donnée par [Ref-7]. Elle est calculée à partir de la précision de l'ordinateur α (α est le plus petit chiffre machine qui vérifie $1+\alpha > 1$), du nombre total d'inconnues n et de la plus grande rigidité k_1 des éléments voisins des particules en contact :

$$\varepsilon_N = \frac{k_1}{\sqrt{n\alpha}} \tag{4.11}$$

La prise en compte des lois de contact dans les équations d'équilibre global s'effectue selon le schéma de résolution de ces équations. En général, l'état de contact est vérifié pour tous les nœuds candidats au contact, et les réactions de contact et les termes de pénalité sont estimés uniquement pour les nœuds qui ont violé l'état de contact.

On obtient ainsi deux relations forces - déplacements linéaires :

$$t_N = \varepsilon_N |g_N| = \varepsilon_N \text{sign}(g_N) g_N = -\varepsilon_N g_N \tag{4.12}$$

et

$$t_T = \varepsilon_T g_T \tag{4.13}$$

Selon la valeur des coefficients de pénalité, les fonctions de séparations associées seront plus ou moins contraintes.

- *Calcul des réactions dues aux frottements* :

La friction est un phénomène qui apparaît lorsqu'il y a déplacement relatif tangentiel entre une paire de surfaces. Ainsi, des réactions ou forces de cisaillement sont produites dans une direction opposée au mouvement relatif des corps en contact.

Expérimentalement, on distingue deux phases principales. Premièrement, on observe une phase d'adhésion, où il n'y a pas de glissement relatif des corps. Après que cette phase soit complétée, on observe du glissement. Le comportement simplifié est illustré à la Fig. 4.3(a), où la zone d'adhésion est marquée par 1 et la zone de glissement par 2. La part de friction caractérisant la phase de glissement est généralement décrite par la loi de Coulomb. Cette Loi est définie par la fonction de glissement où les forces sont représentées par leur intensité :

$$\Phi_s = t_T - \mu t_N \leq 0 \qquad (4.14)$$

et où μ est le coefficient de frottement en glissement. La fonction de glissement impose une limite à la grandeur de la force de frottement par les lignes horizontales à la Fig. 4.3. Ainsi , pour $\Phi_s < 0$ on a contact collant (ou par adhésion), $\Phi_s = 0$ caractérise le glissement et $\Phi_s > 0$ n'est pas permis.

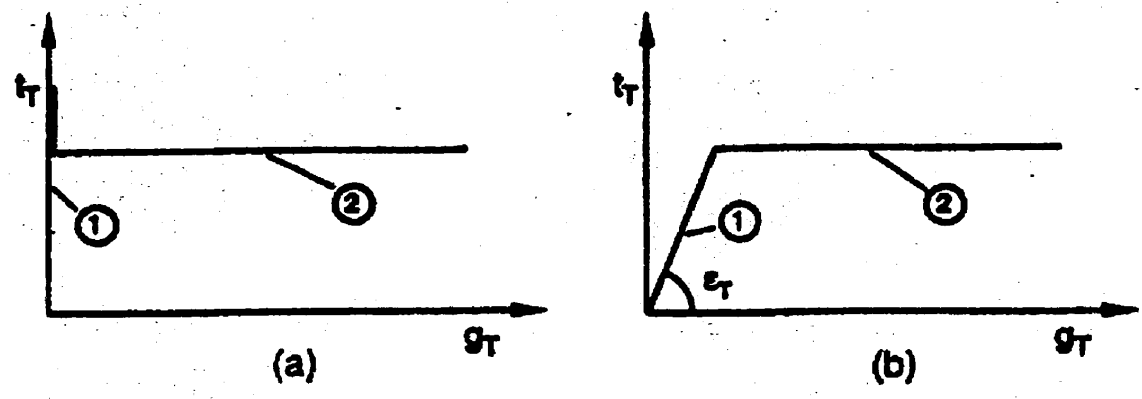


Figure 4.3 : Force de frottement en fonction de la séparation tangentielle :
(a) expérimental; (b) simulation numérique

La dépendance de la force de frottement par rapport à la fonction de séparation tangentielle g_T est illustrée à la Fig. 4.3(b). Dans ce cas, la méthode de pénalité est appliquée dans la zone

d'adhésion, permettant un petit mouvement de glissement qui est fonction du coefficient de pénalité choisi.

Dans la zone d'adhésion, la force de frottement est donc donnée par l'équation (4.13), soit $t_T = \varepsilon_T g_T$.
Dans la zone de glissement, l'intégration de la loi de comportement du frottement est nécessaire.

L'intégration de la loi de comportement du contact avec frottement utilisée dans nos travaux se fait comme en plasticité en décomposant la vitesse tangentielle en partie réversible (adhésion) et en partie dissipative (glissement) :

- pour la partie en adhésion :

$$\mathbf{t}_T = -\varepsilon_T \dot{\mathbf{d}}_T \tag{4.15}$$

- pour la partie en glissement :

$$\mathbf{t}_T = -\mu |t_N| \dot{\mathbf{d}}_T^s / \left\| \dot{\mathbf{d}}_T^s \right\| \tag{4.15}$$

où $\dot{\mathbf{d}}_T^s$ décrit la composante de glissement de la vitesse tangentielle.

Ces relations seront discrétisées dans le temps en utilisant le schéma d'Euler implicite qui permet, à travers un incrément, de trouver la configuration d'équilibre C^{n+1} à partir d'une configuration d'équilibre connue C^n.
On suppose que la force de frottement $\mathbf{t}_{Tn}$ au début de l'incrément est connue et notre but est de calculer la force de frottement $\mathbf{t}_{Tn+1}$, correspondant à la configuration C^{n+1}. Ce calcul doit respecter la condition que la fonction de glissement Φ_s doir rester admissible. i.e. $\Phi \leq 0$

L'algorithme d'intégration de la loi de frottement peut alors s'écrire comme suit :

(1) Calculer une valeur d'essai de la force de frottement en supposant qu'il y adhésion pendant l'incrément de temps.

$$\mathbf{t}_{Tn+1}^{tria} = \mathbf{t}_{Tn} + \mathbf{t}_{stick} \tag{4.16}$$

où la force d'adhésion est donnée par la loi de pénalité :

$$\mathbf{t}_{stick} = -\varepsilon_T \Delta \mathbf{d}_T \tag{4.17}$$

L'état défini par une telle supposition permet de définir la nouvelle direction de la force de frottement :

$$\mathbf{f}^{tria} = \frac{\mathbf{t}_{Tn+1}^{tria}}{\left\| \mathbf{t}_{Tn+1}^{tria} \right\|} \tag{4.18}$$

(2) Calculer le module de la force normale de contact

$$t_N = \varepsilon_N |g_N|$$

(3) Test du comportement en friction en calculant le seuil de glissement Φ_s^{tria} :

$$\Phi_s^{tria} = \left\| \mathbf{t}_{Tn+1}^{tria} \right\| - \mu t_N$$

Si $\Phi^{tria}_s < 0$ on a contact par adhésion et :

$$\mathbf{t}_{Tn+1} = \mathbf{t}_{Tn+1}^{tria} \tag{4.19}$$

Sinon $\Phi^{tria}_s \geq 0$, on a glissement et on doit corriger la force de frottement :

$$\mathbf{t}_{Tn+1} = \mathbf{t}_{slide} = \mu t_N \mathbf{f}^{tria} \tag{4.20}$$

Des algorithmes d'intégration de la loi de frottement plus sophistiquées existent et peuvent être retrouvées par exemple dans [Ref-14] et[Ref-16].

4.3 Linéarisation du travail virtuel

Le traitement numérique du travail virtuel donné à la section précédente par l'équation (4.10) doit résoudre une cinématique de contact hautement non linéaire. Aussi, en général, une procédure incrémentale sera établie pour la résolution du problème, à l'intérieur de laquelle une

prédiction linéaire est approchée par une méthode itérative de Newton-Raphson. Afin d'obtenir des solutions convergentes à partir d'une telle procédure, une linéarisation cohérente du travail virtuel par rapport aux inconnues est nécessaire.

Dans ce qui suit, le symbole d est utilisé pour représenter la différentielle totale d'une quantité. Ainsi la linéarisation du travail virtuel donné à l'équation (4.9) est donnée par :

$$D_{\mathbf{x}}(G) = \{d\mathbf{t}_N \cdot \delta\mathbf{u} + \mathbf{t}_N \cdot d(\delta\mathbf{u})\} + \{d\mathbf{t}_T \cdot \delta\mathbf{u} + \mathbf{t}_T \cdot d(\delta\mathbf{u})\} \tag{4.21}$$

Les forces totales de contact étant données par :

$$\mathbf{t} = \mathbf{t}_N + \mathbf{t}_T = t_N\mathbf{n} + t_T\mathbf{f} \tag{4.22}$$

(4.22) dans (4.21) donne :

$$D_{\mathbf{x}}(G) = d\mathbf{t}_N \cdot \delta\mathbf{u} + d\mathbf{t}_T \cdot \delta\mathbf{u} + \mathbf{t} \cdot d(\delta\mathbf{u}) \tag{4.23}$$

Ainsi on peut identifier les ingrédients de la matrice de rigidité de contact élémentaire qui comprend une composante dépendante de la déformation obtenue par dérivation des contraintes, $d\mathbf{t}_N \cdot \delta\mathbf{u} + d\mathbf{t}_T \cdot \delta\mathbf{u}$ (composante matérielle), et une composante géométrique due au changement de la géométrie du problème, $\mathbf{t} \cdot d(\delta\mathbf{u})$, qui est liée, comme on le verra plus tard, au fait que le nœud physique de contact n'est pas un point nodal du maillage, mais un point qui se déplace sur la surface cible selon les changements dans la configuration.

Afin de pouvoir détailler l'expression ci-dessus, il est nécessaire de déterminer d'abord les expressions des dérivées des forces normales et tangentielles.

4.3.1 Dérivées des forces normales et tangentielles

En utilisant la définition donnée à la section (4.2.4) ci-dessus pour les forces de contact, on est maintenant capable de déterminer les dérivées de ces forces. En omettant la notation ^{tria} pour simplifier, on obtient le développement suivant :

- *dérivée de la partie normale* :

On obtient de l'équation (4.6)

$$\mathbf{d}t_N = \mathbf{d}t_N \mathbf{n} + t_N \mathbf{d}\mathbf{n} \tag{4.24}$$

Des équations (4.3) et (4.14), on obtient :

$$\mathbf{d}t_N = -\varepsilon_N \mathbf{n} \cdot \mathbf{d}\mathbf{u} \tag{4.25}$$

ce qui une fois insérée dans (4.24) permet d'écrire :

$$\mathbf{d}t_N = -\varepsilon_N (\mathbf{n} \otimes \mathbf{n}\mathbf{d}\mathbf{u} + g_N \mathbf{d}\mathbf{n}) \tag{4.26}$$

- *dérivée de la force de frottement* :

On doit distinguer ici le cas de l'adhésion du cas du glissement.

De l'équation (4.17), on obtient pour le cas d'adhésion :

$$\mathbf{d}t_{stick} = -\varepsilon_T \mathbf{d}\Delta\mathbf{d}_T \tag{4.27}$$

et pour le cas de glissement, de l'équation (4.20) on a :

$$\mathbf{d}t_{slide} = \mu(dt_N \mathbf{f} + t_N \mathbf{d}\mathbf{f}) \tag{4.28}$$

La dérivée du vecteur unitaire $\mathbf{f}$ est calculée à partir de l'équation (4.18) et devient :

$$\mathbf{d}\mathbf{f} = \frac{1}{\|\mathbf{t}_T\|}(\mathbf{1} - \mathbf{f} \otimes \mathbf{f})\mathbf{d}t_{stick} \tag{4.29}$$

avec $\mathbf{1}$ représentant la matrice identité 3 x 3.

En utilisant les résulats obtenus aux équations (4.24), (4.25) et (4.27) dans (4.28) on obtient :

$$\mathbf{d}t_{slide} = -\mu\left[\varepsilon_N(\mathbf{f} \otimes \mathbf{n})\mathbf{d}\mathbf{u} + \varepsilon_T \frac{t_N}{\|\mathbf{t}_T\|}(\mathbf{1} - \mathbf{f} \otimes \mathbf{f})\mathbf{d}\Delta\mathbf{d}_T\right] \tag{4.30}$$

4.3.2 Le travail virtuel de contact linéarisé

L'introduction des résultats précédents dans les expressions du travail virtuel linéarisé (4.23) nous permet d'obtenir une linéarisation plus détaillée. En se référant à leur signification physique on peut distinguer deux contributions différentes à partir de l'équation 4.23 :

$$D_x(G) = d\mathbf{t}_N \cdot \delta\mathbf{u} + d\mathbf{t}_T \cdot \delta\mathbf{u} + \mathbf{t} \cdot d(\delta\mathbf{u})$$

et en séparant les contributions normale et tangentielles :

- Partie matérielle:
  - travail virtuel normal du contact : ou contact sans frottement (le terme $d\mathbf{t}_N \cdot \delta\mathbf{u}$) :

$$D_x(G)_n = -\varepsilon_N \delta\mathbf{u}(\mathbf{n} \otimes \mathbf{n}\, d\mathbf{u} + g_N\, d\mathbf{n}) \tag{4.31}$$

  - travail virtuel tangentiel du au frottement du contact (le terme $d\mathbf{t}_T \cdot \delta\mathbf{u}$)

    Pour le contact avec adhésion,

$$D_x(G)_{stick} = -\varepsilon_T \delta\mathbf{u} \cdot d\Delta\mathbf{d}_T \tag{4.32a}$$

    Pour le contact avec glissement,

$$D_x(G)_{slide} = -\mu\delta\mathbf{u}\left[\varepsilon_N(\mathbf{f} \otimes \mathbf{n})d\mathbf{u} + \varepsilon_T \frac{t_N}{\|\mathbf{t}_T\|}(\mathbf{1} - \mathbf{f} \otimes \mathbf{f})d\Delta\mathbf{d}_T\right] \tag{4.32b}$$

- et la Partie géométrique

$$D_x(G)_g = \mathbf{t} \cdot d(\delta\mathbf{u}) \tag{4.33}$$

Le développement précédent nous servira de base dans la formulation par éléments finis de notre élément de contact, en particulier dans l'écriture des matrices de rigidité tangentes dues au contact.

4.4 Formulation par éléments finis

Cette section est consacrée à l'évaluation des matrices et vecteurs élémentaires. Le calcul de la force de chargement pour les éléments finis est basé sur l'expression du travail virtuel

donnée à l'équation (4.10) et les matrices de rigidité associées sont données par les équations(4.31)-(4.32). Cependant, on peut remarquer que parmi ces expressions, les informations relatives aux quantités géométriques sont manquantes. Ces termes dépendent exclusivement de la définition de la surface cible et du mouvement de frottement associé. Ainsi, dans la suite de cette section, nous ferons tout d'abord une description mathématique de la surface du corps cible et ensuite, nous développerons la formulation du mouvement tangentiel. Le développement des matrices élémentaires est effectué aussi bien pour le contact 2D avec les éléments ElmB4n2D, en se basant sur l'article [Ref-12], que pour le contact 3D avec les éléments ElmB8n3D, en se basant sur l'article [Ref-13]. Dans le premier article des corrections ont été apportées aux équations (2.24, 2.32, 2.34) tandis que dans le second elles ont été apportées aux équations ((63), (66), (70), (72), (73)).

4.4.1 Description de la surface cible

La description qui suit est présentée de façon générale pour le cas 3D. On suppose que la surface cible est lisse, de forme arbitraire et donnée par la description paramétrique suivante :

$$\mathbf{x} = \mathbf{x}\left(\zeta^{\alpha}\right) \tag{4.34}$$

où $\zeta^{\alpha} = \{\xi, \eta\}$ sont les cordonnées paramétriques de la surface. Suivant cette description, les vecteurs tangents selon les directions paramétriques sont donnés par :

$$\mathbf{e}_{\alpha} = \partial \mathbf{x} / \partial \zeta^{\alpha} \tag{4.35}$$

et le vecteur normal unitaire associé est :

$$\mathbf{n} = \frac{\mathbf{e}_1 \times \mathbf{e}_2}{\Omega} \tag{4.36}$$

avec $\Omega = \|\mathbf{e}_1 \times \mathbf{e}_2\|$ étant l'aire de l'élément de surface.

Les tenseurs métriques covariant et contravariant $m_{\alpha\beta}$ et $m^{\alpha\beta}$ sont définis par :

$$m_{\alpha\beta} = e_{\alpha} \cdot e_{\beta}, \quad m^{\alpha\beta} = \left(m_{\alpha\beta}\right)^{-1} \tag{4.37}$$

et ils permettent de faire une transition entre les repères de base complémentaires tangentiels covariant et contravariant via la relation :

$$\mathbf{e}^{\alpha} = m^{\alpha\beta}\mathbf{e}_{\beta} \tag{4.38}$$

La description mathématique d'une surface est complétée par le calcul du tenseur de courbure :

$$b_{\alpha\beta} = \frac{\partial \mathbf{e}_{\alpha}}{\partial \zeta^{\beta}} \cdot \mathbf{n} = \mathbf{e}_{\alpha,\beta} \cdot \mathbf{n} \tag{4.39}$$

On peut compléter notre description de la surface cible en définissant de nouveaux coefficients métriques qui sont modifiés par la courbure comme suit [Réf-13] et [Réf-14]:

$$\overline{m}_{\alpha\beta} = m_{\alpha\beta} - \mathbf{d}_{\mathrm{N}} \cdot \mathbf{e}_{\alpha,\beta} \approx m_{\alpha\beta} - \mathrm{g}_{\mathrm{N}} b_{\alpha\beta} \tag{4.40}$$

Du fait que la surface cible est l'emplacement du point physique de contact avec le nœud contacteur, on doit donc définir toutes les variables de contact de la surface cible par rapport à ce point de contact. La première tache est alors de trouver les coordonnées intrinsèques ξ_c^{α} du point physique de contact sur la surface cible. Ces coordonnées sont déterminées par deux équations ($\alpha = 1,2$.):

$$\mathrm{F}_{\alpha} := \mathbf{e}_{\alpha} \cdot (\mathbf{x}_s - \mathbf{x}_c) = \mathbf{e}_{\alpha} \cdot \mathbf{d}_{\mathrm{N}} = 0 \tag{4.41}$$

qui expriment simplement la projection normale du nœud contacteur sur le plan tangent. Ce système est non linéaire et doit donc être résolu de façon itérative, par exemple en utilisant la méthode de Newton-Raphson. Sa linéarisation par rapport aux inconnues s'écrit formellement :

$$\mathrm{D}(\mathrm{F}_{\alpha}) := \mathrm{d}\mathbf{e}_{\alpha} \cdot (\mathbf{x}_s - \mathbf{x}_c) + \mathbf{e}_{\alpha} \cdot (\mathrm{d}\mathbf{x}_s - \mathrm{d}\mathbf{x}_c) = 0 \tag{4.42}$$

Le calcul des différentiels ou des variations impliquées doit se faire en distinguant deux cas. Le cas le plus général se réfère à la solution de ces équations durant les itérations d'équilibre. Dans ce cas, les coordonnées spatiales du point physique de contact changeront suite aux modifications dans les coordonnées de la surface et aux déplacements qui ont lieu. Ainsi on aura en général que $\mathbf{x}_c = \mathbf{x}_c(\xi^{\alpha}, \mathbf{u}_c)$ et par conséquent, les différentielles totales s'écriront :

$$\begin{aligned} \mathbf{dx}_s &= \mathbf{du}_s \\ \mathbf{dx}_c &= \mathbf{du}_c + \mathbf{e}_\beta \, d\zeta^\beta \\ \mathbf{de}_\alpha &= \mathbf{e}_{\alpha,\beta} \, d\zeta^\beta + \mathbf{du}_{c,\alpha} \end{aligned} \tag{4.43}$$

Dans l'autre cas, évidemment plus simple, on peut supposer que les configurations sont maintenues fixes, et alors toutes les différentielles de déplacements s'annulent. Ceci réfère au calcul des coordonnées dans la configuration où l'équilibre est évalué.

Dans ce qui suit on se placera dans la situation la plus générale. Alors lorsqu'on insère l'équation (4.42) dans l'équation linéarisée (4.43) et en utilisant les notations introduites précédemment, on obtient :

$$\overline{m}_{\alpha\beta} \, d\zeta^\beta = \left(\mathbf{du}_{c,\alpha} \cdot \mathbf{d}_N\right) + \mathbf{e}_\alpha \cdot \left(\mathbf{du}_s - \mathbf{du}_c\right) \tag{4.44}$$

La résolution du système (4.44) donne les deux composantes intrinsèques de la coordonnée ξ_c^α :

$$\begin{aligned} d\xi &= h^{-1}\left[\left(\overline{m}_{22} \, \mathbf{du}_{c,\xi} - \overline{m}_{12} \, \mathbf{du}_{c,\eta}\right) \cdot \mathbf{d}_N + \left(\overline{m}_{22}\mathbf{e}_1 - \overline{m}_{12}\mathbf{e}_2\right) \cdot \left(\mathbf{du}_s - \mathbf{du}_c\right)\right] \\ d\eta &= h^{-1}\left[\left(\overline{m}_{11} \, \mathbf{du}_{c,\eta} - \overline{m}_{12} \, \mathbf{du}_{c,\xi}\right) \cdot \mathbf{d}_N + \left(\overline{m}_{11}\mathbf{e}_2 - \overline{m}_{12}\mathbf{e}_1\right) \cdot \left(\mathbf{du}_s - \mathbf{du}_c\right)\right] \end{aligned} \tag{4.45}$$

où :

$$h = \det\left(\overline{m}_{\alpha\beta}\right) = \overline{m}_{11}\overline{m}_{22} - \overline{m}_{12}\overline{m}_{21} \tag{4.46}$$

représente le déterminant de $\overline{m}_{\alpha\beta}$. Les variables calculées en (4.45) sont fondamentales dans la linéarisation cohérente des équations d'équilibre.

Connaissant les coordonnées et la position du point de contact, on peut maintenant calculer toutes les dérivées des quantités relatives à la géométrie de la surface cible:

- **linéarisation du vecteur normal:**

  Formellement la dérivée du vecteur normal est donnée par :

$$\mathbf{dn} = \frac{1}{\Omega}(\mathbf{1} - \mathbf{n} \otimes \mathbf{n}) d(\mathbf{e}_1 \times \mathbf{e}_2) \tag{4.47}$$

où :

$$\mathbf{e}^1 = \frac{\mathbf{e}_2 \times \mathbf{n}}{\Omega}; \quad \mathbf{e}^2 = \frac{\mathbf{e}_1 \times \mathbf{n}}{\Omega}; \quad \mathbf{e}^3 = \mathbf{n} \tag{4.48}$$

en utilisant les équations (4.43), (4.45) et la relation $\mathbf{e}_\alpha \cdot \mathbf{e}^\beta = \delta_\alpha^\beta$, l'équation (4.47) devient :

$$\begin{aligned} d\mathbf{n} = &-\varpi_{11}(\mathbf{e}^1 \otimes \mathbf{n})d\mathbf{u}_{c,\xi} - \varpi_{22}(\mathbf{e}^2 \otimes \mathbf{n})d\mathbf{u}_{c,\eta} - \varpi_{21}(\mathbf{e}^2 \otimes \mathbf{n})d\mathbf{u}_{c,\xi} - \varpi_{12}(\mathbf{e}^1 \otimes \mathbf{n})d\mathbf{u}_{c,\eta} \\ &-\left[\kappa_{11}(\mathbf{e}^1 \otimes \mathbf{e}^1) + \kappa_{22}(\mathbf{e}^2 \otimes \mathbf{e}^2) + \kappa_{12}(\mathbf{e}^1 \otimes \mathbf{e}^2 + \mathbf{e}^2 \otimes \mathbf{e}^1)\right](d\mathbf{u}_s - d\mathbf{u}_c) \end{aligned} \tag{4.49}$$

où on a introduit les notations suivantes :

$$\begin{aligned} c_{\alpha\beta} &= \frac{g_N}{h}\overline{m}_{\alpha\beta} \\ \omega_{11} &= 1 + (b_{11}c_{22} - b_{12}c_{12}) \\ \omega_{22} &= 1 + (b_{22}c_{11} - b_{12}c_{12}) \\ \omega_{12} &= (b_{12}c_{11} - b_{11}c_{12}) \\ \omega_{21} &= (b_{12}c_{22} - b_{22}c_{12}) \\ \kappa_{11} &= (b_{11} + b_{11}{}^2c_{22} + b_{12}{}^2c_{11} - 2b_{11}b_{12}c_{12}) \\ \kappa_{22} &= (b_{22} + b_{22}{}^2c_{11} + b_{12}{}^2c_{22} - 2b_{22}b_{12}c_{12}) \\ \kappa_{12} &= \left[b_{12} + b_{22}b_{12}c_{11} + b_{11}b_{12}c_{22} - c_{12}(b_{11}b_{12} + b_{12}{}^2)\right] \end{aligned} \tag{4.50}$$

- **terme géométrique de l'équation (4.23):**

  Avec les données précédentes, on peut maintenant développer la partie géométrique dans l'équation (4.23). En effet, on peut écrire :

$$\mathbf{t} \cdot d(\delta\mathbf{u}) = -\mathbf{t} \cdot (\delta\mathbf{u}_{c,\xi}\, d\xi + \delta\mathbf{u}_{c,\eta}\, d\eta) \tag{4.51}$$

et on voit que ce terme géométrique, dépend entièrement du point de contact et sa présence est due au fait que ce dernier se déplace sur la surface cible durant les itérations jusqu'à ce qu'il y ait équilibre. En substituant l'équation (4.45) dans (4.51) on obtient finalement que :

$$\begin{aligned} \mathbf{t} \cdot d(\delta\mathbf{u}) = &-\left[\varpi_{11}\delta\mathbf{u}_{c,\xi}(\mathbf{t} \otimes \mathbf{e}^1) + \varpi_{22}\delta\mathbf{u}_{c,\eta}(\mathbf{t} \otimes \mathbf{e}^2) + \varpi_{21}\delta\mathbf{u}_{c,\xi}(\mathbf{t} \otimes \mathbf{e}^2) + \varpi_{12}\delta\mathbf{u}_{c,\eta}(\mathbf{t} \otimes \mathbf{e}^1)\right](d\mathbf{u}_s - d\mathbf{u}_c) \\ &- c_{22}\delta\mathbf{u}_{c,\xi}(\mathbf{t} \otimes \mathbf{n})d\mathbf{u}_{c,\xi} - c_{11}\delta\mathbf{u}_{c,\eta}(\mathbf{t} \otimes \mathbf{n})d\mathbf{u}_{c,\eta} + c_{12}\delta\mathbf{u}_{c,\xi}(\mathbf{t} \otimes \mathbf{n})d\mathbf{u}_{c,\eta} + c_{12}\delta\mathbf{u}_{c,\eta}(\mathbf{t} \otimes \mathbf{n})d\mathbf{u}_{c,\xi} \end{aligned}$$

(4.52)

4.4.2 Description du mouvement tangentiel

Si on examine les expressions des équations définissant les matrices de rigidité de contact (Éqn. 4.32a et 4.32b), on remarque qu'il reste à calculer la variation de l'incrément de

déplacement tangentiel $d\Delta\mathbf{d}_T$. En se référant à la cinématique illustrée par la Fig. (4.2), on peut exprimer les composantes tangentielles de l'incrément de déplacement comme suit :

$$\Delta\mathbf{d}_T = \Delta\mathbf{u}_{sT} - \Delta\mathbf{u}_{cT} = (\mathbf{1} - \mathbf{n}\otimes\mathbf{n})(\Delta\mathbf{u}_s - \Delta\mathbf{u}_c) \tag{4.53}$$

et sa dérivée donne :

$$d\Delta\mathbf{d}_T = (\mathbf{1} - \mathbf{n}\otimes\mathbf{n})(d\Delta\mathbf{u}_s - d\Delta\mathbf{u}_c) - [\mathbf{1}(\Delta\mathbf{u}_r\cdot\mathbf{n}) + \mathbf{n}\otimes\Delta\mathbf{u}_r]d\mathbf{n} \tag{4.54}$$

Pour les différentielles des incréments de déplacement, on doit à nouveau distinguer le nœud contacteur supposé fixe et le point physique de contact qui se déplace sur la surface cible.

$$\begin{aligned} d\Delta\mathbf{u}_s &= d\mathbf{u}_s \\ d\Delta\mathbf{u}_c &= \Delta\mathbf{u}_{c,\xi}\, d\xi + \Delta\mathbf{u}_{c,\eta}\, d\eta + d\mathbf{u}_c \end{aligned} \tag{4.55}$$

En utilisant le résultat obtenu ci-dessus et l'équation (4.45) et en remplaçant les vecteurs covariants par les tangentes contravariantes correspondantes, selon l'équation (4.38) on obtient :

$$d(\Delta\mathbf{u}_s) - d(\Delta\mathbf{u}_c) = \mathbf{E}(d\mathbf{u}_s - d\mathbf{u}_c) + \mathbf{U}_1\, d\mathbf{u}_{c,\xi} + \mathbf{U}_2\, d\mathbf{u}_{c,\eta} \tag{4.56}$$

où on a introduit les matrices 3 x 3 suivantes :

$$\begin{aligned} \mathbf{E} &= \mathbf{1} - \omega_{11}(\Delta\mathbf{u}_{c,\xi}\otimes\mathbf{e}^1) - \omega_{22}(\Delta\mathbf{u}_{c,\eta}\otimes\mathbf{e}^2) - \omega_{21}(\Delta\mathbf{u}_{c,\xi}\otimes\mathbf{e}^2) - \omega_{12}(\Delta\mathbf{u}_{c,\eta}\otimes\mathbf{e}^1) \\ \mathbf{U}_1 &= c_{12}(\Delta\mathbf{u}_{c,\eta}\otimes\mathbf{n}) - c_{22}(\Delta\mathbf{u}_{c,\xi}\otimes\mathbf{n}) \\ \mathbf{U}_2 &= c_{12}(\Delta\mathbf{u}_{c,\xi}\otimes\mathbf{n}) - c_{11}(\Delta\mathbf{u}_{c,\eta}\otimes\mathbf{n}) \end{aligned} \tag{4.57}$$

Avec ces dernières expressions, on finit la linéarisation complète du déplacement incrémental tangentiel . Ces expressions nous serviront à développer les matrices et vecteurs élémentaires.

4.4.3 Formulation matricielle des quantités élémentaires

Cette section est consacrée à l'évaluation des matrices élémentaires résultant du travail virtuel de contact discrédité par éléments finis. Dépendamment de l'application choisie, deux situations de contact peuvent se produire. Le premier cas concerne le contact entre un corps déformable, modélisé par éléments finis, et une surface rigide cible. Le second cas, qui est plus

complexe, concerne le contact entre deux corps déformables tous les deux modélisés par éléments finis. Nous allons tout d'abord nous intéresser à ce cas.

Nous supposons que la surface cible est discrétisée en éléments finis de surface compatible avec les éléments du maillage du solide considéré pour le contact. De plus, on suppose que l'élément fini de surface est de type isoparamétrique défini par N points appelés nœuds auxquels sont associées des fonctions de forme $\psi^N(\xi,\zeta)$. Ainsi, suivant la pratique standard en éléments finis[Réf- 10,13,14,22], nous pouvons écrire les relations d'interpolations suivantes pour les variables, déplacement et position, et leur dérivées associées au point de contact :

$$\begin{aligned} \mathbf{u}_c &= \psi^M \hat{\mathbf{u}}_M \\ \mathbf{u}_{c,\alpha} &= \psi^M_{,\alpha} \hat{\mathbf{u}}_M \\ \mathbf{x}_c &= \psi^M \hat{\mathbf{x}}_M \\ \mathbf{x}_{c,\alpha} &= \psi^M_{,\alpha} \hat{\mathbf{x}}_M \end{aligned} \tag{4.58}$$

où les quantités avec un chapeau désignent les points nodaux et la convention de sommation est appliquée pour $M = 1...N$. A l'aide de ces interpolations, la variation du vecteur déplacement relatif entre l'élément de la surface cible et le nœud contacteur, écrite au point de contact devient:

$$\delta\mathbf{u} = \frac{\partial}{\partial \hat{\mathbf{u}}}(\mathbf{u}_s - \mathbf{u}_c)\delta\hat{\mathbf{u}} = \begin{bmatrix} \mathbf{1} & -\psi^1 & \dots & -\psi^N \end{bmatrix} \begin{bmatrix} \delta\hat{\mathbf{u}}_s \\ \delta\hat{\mathbf{u}}^1 \\ \dots \\ \delta\hat{\mathbf{u}}^N \end{bmatrix} = \Lambda \begin{bmatrix} \delta\hat{\mathbf{u}}_s \\ \delta\hat{\mathbf{u}}^1 \\ \dots \\ \delta\hat{\mathbf{u}}^N \end{bmatrix} \tag{4.59}$$

où ψ^N est une matrice diagonale 3x3 construite à partir des fonctions de forme. Par analogie, à partir de l'équation (4.58), on peut alors écrire pour la dérivée de δu :

$$\delta\mathbf{u}_{,\alpha} = \frac{\partial}{\partial \hat{\mathbf{u}}}(-\mathbf{u}_{c_\alpha})\delta\hat{\mathbf{u}} = \begin{bmatrix} \mathbf{0} & -\psi^1_{,\alpha} & \dots & -\psi^N_{,\alpha} \end{bmatrix} \begin{bmatrix} \delta\hat{\mathbf{u}}_s \\ \delta\hat{\mathbf{u}}^1 \\ \dots \\ \delta\hat{\mathbf{u}}^N \end{bmatrix} = \Lambda_{,\alpha} \begin{bmatrix} \delta\hat{\mathbf{u}}_s \\ \delta\hat{\mathbf{u}}^1 \\ \dots \\ \delta\hat{\mathbf{u}}^N \end{bmatrix} \tag{4.60}$$

où $\mathbf{0}$ la matrice nulle 3x3.

Avec ces équations, les termes tels que $(\mathbf{n}\cdot\delta\mathbf{u})$ peuvent maintenant être écrites convenablement sous forme matricielle. Les identités matricielles suivantes seront utilisées:

$$\begin{aligned}
&\mathbf{n}\cdot\delta\mathbf{u} = \mathbf{n}^{T}\Lambda\delta\tilde{\mathbf{u}} = \mathbf{N}\delta\tilde{\mathbf{u}}\\
&\mathbf{e}^{\alpha}\cdot\delta\mathbf{u} = \mathbf{e}^{\alpha T}\Lambda\delta\tilde{\mathbf{u}} = \mathbf{V}_{,\alpha}\delta\tilde{\mathbf{u}}\\
&\mathbf{f}\cdot\delta\mathbf{u} = \mathbf{f}^{T}\Lambda\delta\tilde{\mathbf{u}} = \mathbf{S}\delta\tilde{\mathbf{u}}\\
&\mathbf{n}\cdot\delta\mathbf{u}_{,\alpha} = \mathbf{n}^{T}\Lambda_{,\alpha}\delta\tilde{\mathbf{u}} = \mathbf{D}_{,\alpha}\delta\tilde{\mathbf{u}}\\
&\mathbf{f}\cdot\delta\mathbf{u}_{,\alpha} = \mathbf{f}^{T}\Lambda_{,\alpha}\delta\tilde{\mathbf{u}} = \mathbf{G}_{,\alpha}\delta\tilde{\mathbf{u}}\\
&\mathbf{t}\cdot\delta\mathbf{u}_{,\alpha} = \mathbf{t}^{T}\Lambda_{,\alpha}\delta\tilde{\mathbf{u}} = \mathbf{F}_{,\alpha}\delta\tilde{\mathbf{u}}
\end{aligned} \tag{4.61}$$

où ^{T} signifie transposé.

Pour la mise en œuvre de la formulation des matrices élémentaires, on doit se référer à l'expression du travail virtuel donnée à l'équation (4.10). En appliquant le principe d'invariance du travail virtuel, on peut aussi écrire :

$$G(\Phi,\delta\mathbf{u}) = t_{N}(\mathbf{n}\cdot\delta\mathbf{u}) + t_{T}(\mathbf{f}\cdot\delta\mathbf{u}) = \delta\tilde{\mathbf{u}}^{T}\mathbf{R} \tag{4.62}$$

où $\mathbf{R}$ est le vecteur des forces de contact. L'introduction des matrices d'interpolation donne :

$$\mathbf{R}_{c} = t_{N}\mathbf{N}^{T} + t_{T}\mathbf{S}^{T} = \Lambda^{T}\mathbf{t} \tag{4.63}$$

L'évaluation des matrices de rigidité suit la même procédure que celle pour le vecteur résidu. En introduisant les matrices d'interpolation dans le travail virtuel linéarisé, on obtient une expression de la forme :

$$D_{s}(G) = \delta\tilde{\mathbf{u}}^{T}\mathbf{K}_{c}d\tilde{\mathbf{u}} \tag{4.64}$$

où $\mathbf{K}_{c}$ est la matrice de rigidité de l'élément de contact. En appliquant la séparation utilisée dans le travail virtuel linéarisé, la matrice tangente de l'élément de contact est alors constituée de trois parties :

$$\mathbf{K}_{c} = \mathbf{K}_{n} + \mathbf{K}_{g} + \mathbf{K}_{f} \tag{4.65}$$

où $\mathbf{K}_{n}$ réfère à la contribution du contact sans friction, $\mathbf{K}_{f}$ représente la contribution due au frottement et $\mathbf{K}_{g}$ est la partie géométrique.

Le réarrangement de l'équation (4.49) sous forme matricielle, en appliquant les équations (4.61) donne :

$$d\mathbf{n} = -\Lambda\begin{bmatrix} \omega_{11}\mathbf{V}_1^T\mathbf{D}_1 + \omega_{22}\mathbf{V}_2^T\mathbf{D}_2 + \omega_{21}\mathbf{V}_2^T\mathbf{D}_1 + \omega_{12}\mathbf{V}_1^T\mathbf{D}_2 + \\ \kappa_{11}\mathbf{V}_1^T\mathbf{V}_1 + \kappa_{22}\mathbf{V}_2^T\mathbf{V}_2 + \kappa_{12}\left(\mathbf{V}_1^T\mathbf{V}_2 + \mathbf{V}_2^T\mathbf{V}_1\right) \end{bmatrix} \tag{4.66}$$

Pour l'évaluation de la quantité $d\Delta\mathbf{d}_T$, on définit les matrices 3x3 suivantes :

$$\mathbf{A} = \mathbf{1} - \mathbf{n}\otimes\mathbf{n} \tag{4.67}$$

$$\mathbf{B} = \mathbf{1}\left(\Delta\mathbf{u}_T \cdot \mathbf{n}\right) + \mathbf{n}\otimes\Delta\mathbf{u}_T \tag{4.68}$$

$$\mathbf{C} = \mathbf{1} - \mathbf{f}\otimes\mathbf{f} \tag{4.69}$$

En combinant ces définitions aux équations (4.54), (4.56) et (4.66) on obtient :

$$\begin{aligned} d\Delta\mathbf{d}_T &= \left\{\mathbf{A}\left[\mathbf{E}\Lambda + \mathbf{U}_1\Lambda_{,1} + \mathbf{U}_2\Lambda_{,2}\right] + \mathbf{B}\Lambda\begin{bmatrix} \omega_{11}\mathbf{V}_1^T\mathbf{D}_1 + \omega_{22}\mathbf{V}_2^T\mathbf{D}_2 + \omega_{21}\mathbf{V}_2^T\mathbf{D}_1 + \omega_{12}\mathbf{V}_1^T\mathbf{D}_2 + \\ \kappa_{11}\mathbf{V}_1^T\mathbf{V}_1 + \kappa_{22}\mathbf{V}_2^T\mathbf{V}_2 + \kappa_{12}\left(\mathbf{V}_1^T\mathbf{V}_2 + \mathbf{V}_2^T\mathbf{V}_1\right) \end{bmatrix}\right\} d\tilde{\mathbf{u}} \\ &= \mathbf{Q}\,d\tilde{\mathbf{u}} \end{aligned} \tag{4.70}$$

Les composantes de la matrice de rigidité en contact s'expriment alors comme suit :

- La rigidité du contact normal à partir de l'équation (4.31) s'écrit :

$$\mathbf{K}_n = -\varepsilon_N\left\{\mathbf{N}^T\mathbf{N} - g_N\begin{bmatrix} \omega_{11}\mathbf{V}_1^T\mathbf{D}_1 + \omega_{22}\mathbf{V}_2^T\mathbf{D}_2 + \omega_{21}\mathbf{V}_2^T\mathbf{D}_1 + \omega_{12}\mathbf{V}_1^T\mathbf{D}_2 + \\ \kappa_{11}\mathbf{V}_1^T\mathbf{V}_1 + \kappa_{22}\mathbf{V}_2^T\mathbf{V}_2 + \kappa_{12}\left(\mathbf{V}_1^T\mathbf{V}_2 + \mathbf{V}_2^T\mathbf{V}_1\right) \end{bmatrix}\right\} \tag{4.71}$$

- La rigidité pour la friction avec adhérence à partir de l'équation (4.32) s'écrit :

$$\mathbf{K}_t = -\varepsilon_T\Lambda^T\mathbf{Q} \tag{4.72}$$

- La rigidité pour la friction avec glissement à partir de l'équation (4.33) s'écrit :

$$\mathbf{K}_t = -\mu\left\{\varepsilon_N\mathbf{S}^T\mathbf{N} + \varepsilon_T\frac{t_N}{\|\mathbf{t}_T\|}\Lambda^T\mathbf{C}\mathbf{Q}\right\} \tag{4.73}$$

- La rigidité géométrique s'écrit quant à elle comme suit :

$$\mathbf{K}_g = -\begin{Bmatrix} \omega_{11}\mathbf{F}_1{}^T\mathbf{V}_1 + \omega_{22}\mathbf{F}_2{}^T\mathbf{V}_2 + \omega_{21}\mathbf{F}_2{}^T\mathbf{V}_1 + \omega_{12}\mathbf{F}_1{}^T\mathbf{V}_2 + \\ c_{11}\mathbf{F}_1{}^T\mathbf{D}_1 + c_{22}\mathbf{F}_2{}^T\mathbf{D}_2 - c_{12}\left(\mathbf{F}_1{}^T\mathbf{D}_2 + \mathbf{F}_2{}^T\mathbf{D}_1\right) \end{Bmatrix} \qquad (4.74)$$

Il est à noter que dans le cas du contact normal sans friction la matrice de rigidité élémentaire est symétrique et elle s'écrit comme suit :

$$\mathbf{K}_c = \mathbf{K}_n + \mathbf{K}_g \qquad (4.76)$$

Ces matrices de rigidité seront assemblées avec les matrices globales par modification des coefficients des degrés de liberté des éléments actifs impliqués dans la zone de contact.

4.5 Algorithmes de recherche automatique des zones de contact

Dans la section précédente, nous avons décrit l'essentiel des ingrédients permettant le calcul des forces de contact. Cependant nous n'avons pas évoqué les algorithmes de détection des zones de contact. Ce sont ces derniers qui permettent de trouver les zones de contact (interférence entre les maillages) et s'il y a lieu d'introduire ou de supprimer les forces de contact s'exerçant entre les deux corps.

Le problème du contact est surtout de nature géométrique et s'il nous paraît extrêmement facile de déterminer si deux maillages interfèrent, la modélisation de cette interférence et surtout de sa rapidité de traitement s'avère particulièrement complexe.

Les algorithmes de recherche sont d'une importance capitale dans le succès de la modélisation d'un problème de contact. L'expérience a montré que dans certaines simulations, ils peuvent accaparer plus de la moitié du temps CPU total.

L'intérêt pour ce sujet est donc grand et plusieurs chercheurs se sont penchés sur lui. Ainsi, on peut citer les travaux de la [Ref-1] qui a développé un algorithme de recherche de contact, sous le nom de "Pinball algorithm", et qui consiste à insérer des sphères dans des éléments qui sont connectés aux nœuds comme le montre la Fig. 4.4. Ces sphères sont centrées

aux barycentres des mailles et ont le volume de ces dernières. La détection des pénétrations devient alors très simple : il suffit de comparer la somme des rayons à la distance entre les centres des deux sphères pour savoir si les deux éléments interfèrent. Les forces de contact sont alors calculées de façon approchée grâce à une technique de pénalisation.

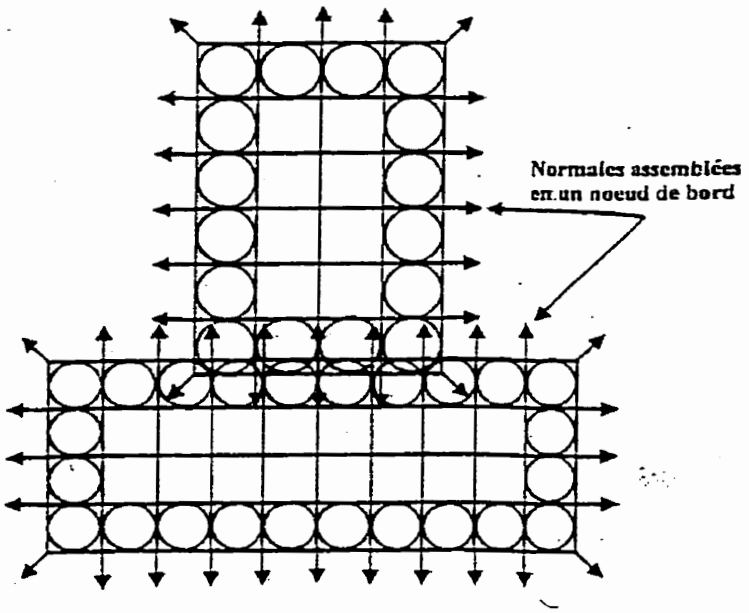


Figure 4.4 :Techniques de Pinball

Il existe d'autres techniques de recherche des zones de contact [Ref-8], qui permettent d'éviter un balayage systématique de l'ensemble des nœuds à chaque temps. Le principe repose sur un découpage de l'espace à l'aide d'un maillage virtuel en sous zones (cases, octants).

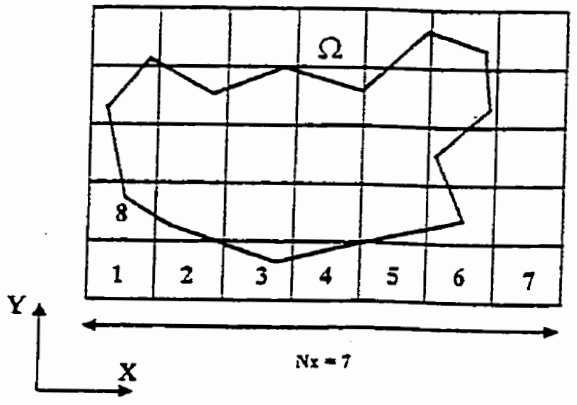


Figure 4.5 Maillage virtuel d'un domaine plan

Des listes d'objets contenant des nœuds dans chaque zone sont mises à jour à chaque introduction d'objet. La structure la plus simple est la grille régulière recouvrant le domaine. Chaque case de la grille est indexée par un entier. Cette structure est décrite dans le plan (Fig. 4.5).

Le pas de la grille est calculé en fonction des longueurs d'arête minimale D_{min} et maximale D_{max} du contour (2D) ou de surface (3D).

$$D = \sqrt[\alpha+\beta]{D_{min}^{\alpha} \cdot D_{max}^{\beta}} \tag{4.77}$$

Par défaut, $\alpha = 2$ et $\beta = 1$ (le poids le plus important est affecté à la plus petite taille).

Connaissant les coordonnées (x,y) d'un nœud, il est immédiat d'en déduire le numéro de la case à laquelle il appartient :

$$\text{clé}(x, y) = \text{int}\left(\frac{x - x_0}{D}\right) + N_x \cdot \text{int}\left(\frac{y - y_0}{D}\right) \tag{4.78}$$

Ensuite dans la boucle sur les nœuds déformables, pour chacun d'eux, seuls les segments rigides faisant partie de la même cellule sont testés, ce qui permet une grande économie de tests.

Pour faciliter la description des algorithmes de recherche de contact que nous avons développé et celui que nous avons adopté [Ref-15], [Ref-21], [Ref-22], pour les problèmes de contact multi-corps et dans lesquels nous utiliserons la notion d'élément cible appartenant à un corps rigide, et d'un nœud contacteur appartenant à un corps déformable, les hypothèses suivantes sont adoptées :

- le contact apparaît entre plusieurs corps rigides et un corps déformable,
- le contact peut être adhérent ou glissant, avec ou sans frottement,
- les solides en contact peuvent avoir des surfaces à petits rayons de courbure et même des surfaces qui présentent des discontinuités de pente,
- le contact est bi-dimensionnel et tridimensionnel.

4.5.1 Algorithme de recherche basé sur la hiérarchie du contact

Cet algorithme a été développé à l'université de Linköping en Suède par [Ref-23]. Il est basé sur la hiérarchie du contact et permet une recherche automatique des zones potentielles de contact.

4.5.1.1 Hiérarchies de contact

Un système de contact est composé de plusieurs corps de contact, un corps de contact consiste lui même en plusieurs surfaces de contact qui sont définies comme étant l'assemblage de plusieurs facettes de contact. Afin de simplifier nos descriptions, notre développement est axé sur le cas en 2D, cependant le cas 3D en découle par analogie. Une facette est une collection de plusieurs arêtes en 2D (ou éléments de surface en 3D) de contact auxquelles nous associons des nœuds de contact (Fig.4.6).

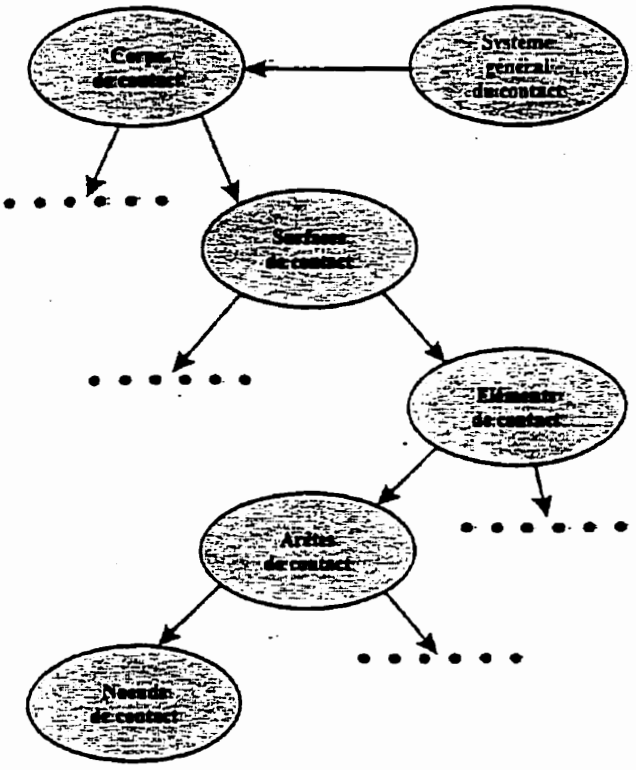


Figure 4.6 Hiérarchies de contact

Dans le cas bidimensionnel, nous pouvons définir trois hiérarchies de contact dans la procédure de recherche :

- surfaces de contact,
- arêtes de contact,
- nœuds de contact.

4.5.1.2 Territoire hiérarchique de contact

Pour une recherche efficace des zones de contact, nous introduisons la notion de territoire pour une hiérarchie du corps de contact. Ce territoire est défini comme une boîte englobant un maillon de la hiérarchie du solide. Ainsi, nous associons un territoire pour chaque hiérarchie (Fig. 4.7).

Dans le cas d'une surface de contact, en 2D le territoire hiérarchique de contact est déterminé par 4 points.

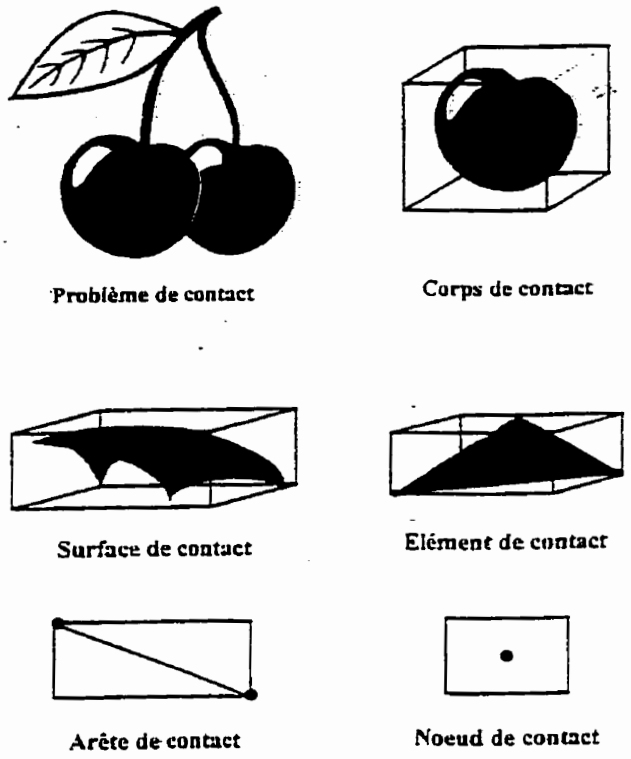


Figure 4.7 : Territoire hiérarchique de contact

Soit T le territoire hiérarchique d'une surface de contact et $x_i (i = 1, N)$ le vecteur position d'un nœud de contact i dans une hiérarchie de contact. N représente le nombre total des nœuds de contact dans cette hiérarchie. Le territoire hiérarchique est défini comme suit :

$$T = \left\{ (x_1, x_2) \mid x_i^{min} \le x_i \le x_i^{max}, i = 1,2 \right\} \tag{4.79}$$

x_1, x_2 sont les composantes cartésiennes du vecteur position **x** et :

$$\begin{aligned} x_i^{min} &= \min\left(x_i^1, x_i^2, \ldots\ldots\ldots\ldots, x_i^N\right) \\ x_i^{max} &= \max\left(x_i^1, x_i^2, \ldots\ldots\ldots\ldots, x_i^N\right) \end{aligned} \tag{4.80}$$

Pour une recherche efficace des nœuds de contact, nous introduisons la notion de territoire hiérarchique élargi dans l'équation (4.30) qui s'exprime par :

$$T = \left\{ (x_1, x_2) \mid x_i^{min} - E_{ep} \le x_i \le x_i^{max} + E_{ep}, i = 1,2 \right\} \tag{4.81}$$

où E_{ep} est un paramètre définissant le territoire élargi.

4.5.1.3 Territoires de contact

Nous associons à chaque segment cible un territoire de contact, fonction de la tolérance de contact h_t (sa valeurs est fixée par l'utilisateur) et qui s'exprime, dans le cas bidimensionnel, sous la forme suivante (4.8) :

$$T_c = \left\{ -h_t \le g_N \le h_t, 0 \le \xi \le 1 \right\} \tag{4.82}$$

g_N et ξ représentent la distance de pénétration et la coordonnée normalisée définies dans (4.2) et dans [Ref-9].

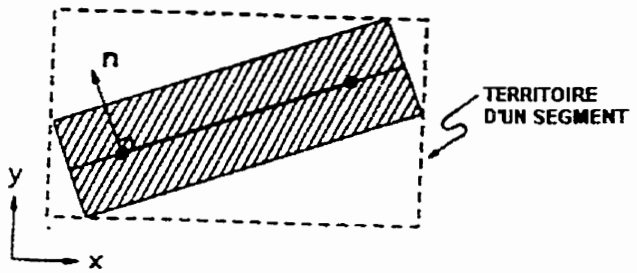


Figure 4.8 : Territoire de contact

4.5.1.4 Procédure de recherche de contact

Les zones potentielles initiales de contact sont initialisées par l'utilisateur dans le fichier d'initialisation. La procédure de recherche de contact consiste à trouver une zone commune entre deux territoires hiérarchiques élargis T_1 et T_2 (Fig. 4.9) qui se traduit mathématiquement par la relation suivante :

$$T = \left\{ (x_1, x_2) \mid x_i^{*min} \le x_i \le x_i^{*max}, i = 1,2 \right\} \tag{4.83}$$

avec :

$$\begin{aligned} x_i^{*min} &= \max(x_i^{1min}, x_i^{2min}) \\ x_i^{*max} &= \max(x_i^{1max}, x_i^{2max}) \end{aligned} \tag{4.84}$$

Une variable logique L est définie par :

$$L = L_1 \cup L_2 \tag{4.85}$$

avec :

$$L_1 = x_1^{*min} \le x_1^{*max} ; \; L_2 = x_2^{*min} \le x_2^{*max} \tag{4.86}$$

Si l'opérateur logique L est vrai, ceci implique l'existence d'une zone potentielle de contact, alors nous retenons uniquement les nœuds contacteurs et les éléments cibles appartenant à cette zone commune, dite zone potentielle de contact. Cette tâche est aussi appelée *recherche globale de contact.* Elle est réalisée en exploitant la notion de ***code de positionnement*** dont l'algorithme est développé dans [Ref-21].

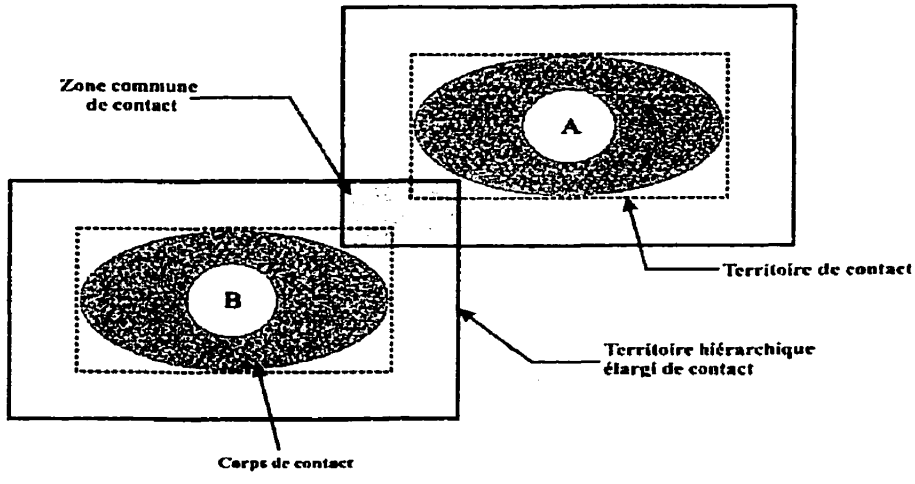


Figure 4.9 : Zone commune de contact

Ainsi la détection des nœuds de contact à l'intérieur des segments de contact est réalisée avec un algorithme basé sur la recherche suivant chaque dimension. L'espace tridimensionnel contenant les surfaces de contact du modèle est divisé en boîtes cubiques (Fig. 4.10) et à chaque boîte est associée un nombre relatif à sa position dans le système de coordonnées global. À chaque nœud de contact, on associe un code de position correspondant à la position de la boîte où il est situé.

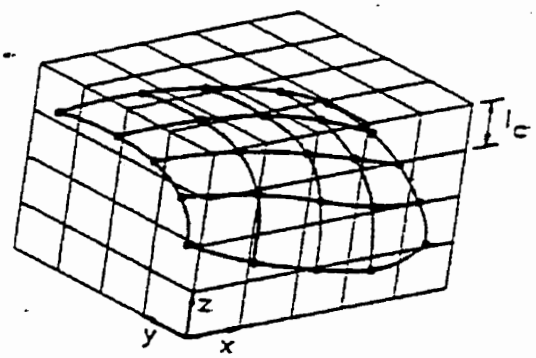


Figure 4.10 Cube des positions des boîtes englobant une surface de contact

L'expression pour le code de position est donnée par :

$$p_c = 1024^2 b_x + 1024 b_y + b_z \tag{4.87}$$

où p_c est le code de position et b_x, b_y, b_z sont les dimensions de la boîte dans chaque dimension.

Le vecteur $\mathbf{b} = (b_x, b_y, b_z)$, associé à un nœud est évalué par l'expression suivante :

$$\mathbf{x}_0 = \mathbf{x}_m - \frac{n_c l_c}{2} \tag{4.88}$$

$$\mathbf{b} = \text{int}\left(\frac{{}^t\mathbf{x} - \mathbf{x}_0}{l_c} \right) \tag{4.89}$$

où $\mathbf{x}_m$ est le point milieu initial de la structure basé sur l'extension dans chaque dimension, n_c est le domaine des dimensions maximum de la boîte, présentement égal à 1024, l_c est la largeur d'une cellule et ${}^t\mathbf{x}$ est la position courante du nœud. L'équation (4.88) dans notre algorithme est

évaluée une seule fois et les équations (4.87) et (4.89) sont utilisées chaque fois que les codes de positions sont recalculées.

Connaissant les codes de position de tous les nœuds de la zone potentielle de contact, on évalue le territoire des arêtes ou des surfaces de frontière sur le corps cible. Ce qui nous permet maintenant de déterminer les nœuds qui sont dans le territoire de chaque segment potentiel de contact. Pour chaque segment cible appartenant à la zone potentielle de contact, nous *recherchons son plus proche voisin* parmi les nœuds contacteurs (Fig. 4.11).

L'algorithme se lit comme suit :

- Pour chaque segment cible
  - Pour chaque nœud de la zone potentielle de contact
    - Tester si le nœud est à l'intérieur du territoire du segment de contact dans chaque dimension
      - Si cela est le cas vérifier la tolérance de contact $C_d = h_t$
        - Le nœud se trouve au-delà de la tolérance $C_d = h_t$ et il est dit libre;
        - Le nœud se trouve à $C_d = h_t$ près du segment cible et il est dit en contact
        - Le nœud se situe en dessous de C_d et il est dit pénétré. Un algorithme de résolution de type pénalité est alors mis en œuvre.
  - Si tous les tests sont positifs alors on peut former un élément potentiel de contact
  - Fin de boucle sur les nœuds
- Fin de boucle sur les segments

Après avoir trouvé les éléments potentiels de contact qui sont formés d'un nœud contacteur et d'un segment cible on procède à la recherche locale de contact qui consiste à établir lesquels des éléments potentiels trouvés répondent au critère du segment le plus proche. Ceci est réalisé grâce à l'algorithme du "closest point projection".

Les conditions de contact seront testées uniquement pour les nœuds contacteurs qui, à l'étape considérée, se trouvent à l'intérieur des territoires hiérarchiques des segments cibles potentiels,

ces nœuds sont déclarés comme "actifs". Cette procédure permet d'éliminer les nœuds potentiels de contact "inactifs" qui seront temporairement ignorés dans les calculs.

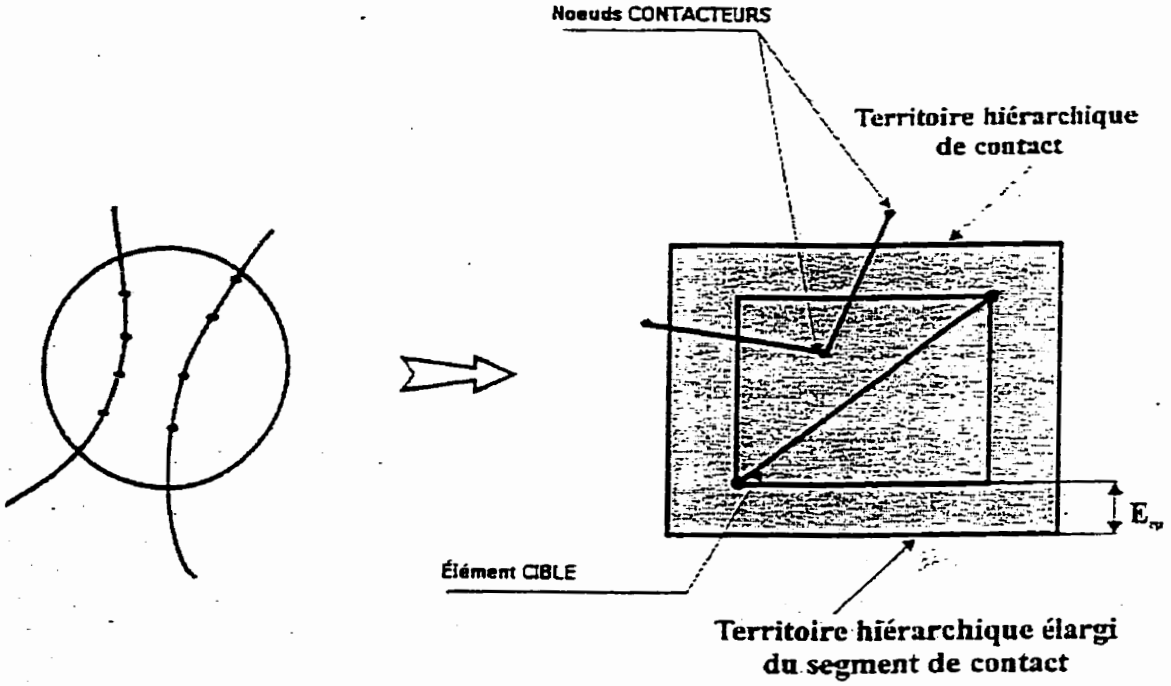


Figure 4.11 Nœud contacteur à l'intérieur du territoire hiérarchique d'un élément cible

L'algorithme de recherche de contact doit être capable de gérer les situations ambiguës qui peuvent se produire lors de la procédure de recherche de contact :

- le nœud contacteur candidat se trouve en face d'un angle mort, et ne possède aucune projection sur les éléments maîtres proches, dans ce cas-là deux situations se présentent :

  - le nœud est libre ou en contact (Fig. 4.12(a)). Ainsi, nous lui associons le nœud cible le plus proche, et un des segments cibles connectés au nœud cible le plus proche.
  - Le nœud est en pénétration (Fig. 4.12(c)), et dans ce cas nous lui associons le segment cible le plus proche dont la distance du point candidat et de sa projection sur cet élément est la plus grande :

$$x_n = \max(x_{n_1}, x_{n_2}) \tag{4.90}$$

- Le nœud contacteur "actif" libre, en contact ou pénétrant est en face d'une vallée et possède donc plusieurs projetés sur des éléments cibles (Fig. 4.12(b)), (Fig. 4.12(d)). Dans ce cas, nous lui associons un des éléments cibles le plus proche dont la distance du point candidat et de sa projection sur cet élément est la plus petite :

$$x_n = \min(x_{n_1}, x_{n_2}) \tag{4.91}$$

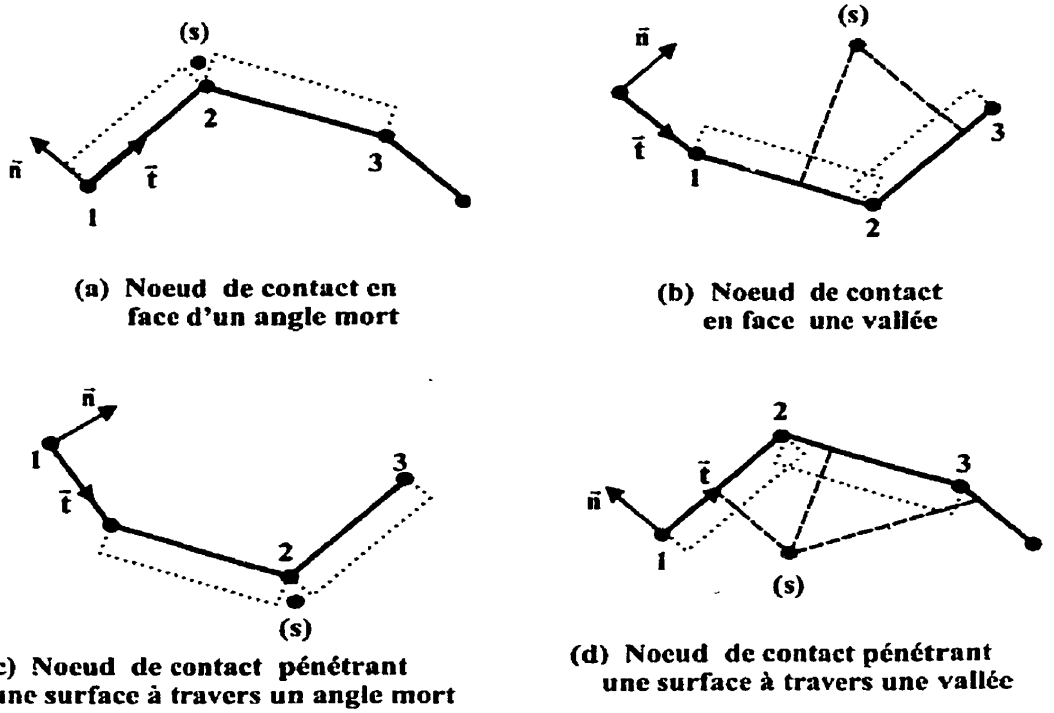


Figure 4.12 Cas ambiguës de contact

4.5.1.5 Fréquence de recherche des nœuds de contact

Afin d'accroître l'efficacité de l'algorithme de recherche, la recherche globale n'est normalement pas effectuée à chaque pas de chargement. Avec une tolérance de contact donnée,

h_t, et une vitesse relative maximum au nœud connue, le nombre de pas de chargement entre les recherches globales peut être ajusté afin d'assurer que tous les nœuds approchant le segment de surface soient détectés à l'intérieur du territoire de contact. La fréquence de recherche globale est évaluée avec l'expression suivante [Ref-21]:

$$n_s = \text{int}\left(\frac{h_t}{v_M \Delta t_M}\right) \tag{4.92}$$

où n_s est le nombre de pas de chargement entre les recherches globales, h_t est la tolérance de fonction de séparation normale, Δt_M est la durée maximale d'un pas de chargement et v_M est la vitesse relative maximale des nœuds de contact.

Dans ce chapitre, on a présenté les lois de contact et de frottement les plus utilisées dans les modèles numériques de simulation de mise en forme ainsi que la résolution des équations d'équilibre avec les inéquations de contact et de frottement. On a également présenté les algorithmes de recherche automatique des zones de contact utilisés dans la simulation des procédés de mise en forme.

Le chapitre suivant sera consacré à l'étude de la programmation des éléments finis dans Visual C++ avec Diffpack.

CHAPITRE V

PPROGRAMMATION ET MODÈLES ÉLÉMENTS FINIS ORIENTÉS OBJETS DANS L'ENVIRONNEMENT DIFFPACK

5.1 Introduction

On a connu au cours des dernières décennies un avancement remarquable dans le domaine du calcul numérique et en mécanique. Cette avancée est surtout due au développement dans la conception des ordinateurs conjugué au développement parallèle en génie logiciel. C'est ainsi que les logiciels de calculs scientifiques empruntant de plus en plus la voie des langages naturels ont connu un essor remarquable. Des outils numériques de plus en plus conviviaux et performants permettent de résoudre des problèmes réels de plus en plus complexes dans divers domaines comme le génie mécanique, le génie civil, la physique, les mathématiques appliquées, la géologie, l'astrophysique, la géophysique, etc...

Les outils mathématiques permettant de modéliser ces problèmes sont les équations aux dérivées partielles (E.D.P.s) et les méthodes numériques associées. C'est

dans le but de mettre au point une boîte à outils pour résoudre ces derniers que Diffpack a été conçu dans les années 90. [Ref-11].

5.2 Qu'est-ce que Diffpack ?

Diffpack est une boîte à outils complexe qui permet de développer des simulateurs numériques. Pour un programmeur, Diffpack est un outil de modélisation et simulation numérique qui est composé de bibliothèques de fonctions organisées en classes en C++. Diffpack utilise en grande partie des techniques de programmation orientée objet. Cette particularité permet à Diffpack de bénéficier d'avantages importants.

En effet, la programmation en classes est plus facile et plus flexible que celle faite en manipulant les données par des variables d'entrées et de sorties qui est utilisée dans les routines du Fortran ou du C. Les classes en langage C++ peuvent être associées afin de former des codes d'applications résolvant des problèmes dans des domaines variés, notamment dans l'ingénierie, les sciences naturelles, l'économie et la médecine.
La programmation en classes permet aussi de sauver considérablement du temps à écrire et débugguer les codes, en ce sens que le programmeur bénéficie non seulement de bibliothèques de fonctions qui ont été largement testées et d'abstractions très avancées, comparativement à la programmation en Fortran ou en C, mais aussi de la possibilité de réutiliser plus facilement son code.

Les outils disponibles dans Diffpack vont de simples abstractions, comme les vecteurs, à d'autres plus complexes comme des collections d'éléments finis. Parmi ces abstractions on distingue :

- vecteurs, matrices, tableaux à indices multiples, chaînes de caractères, ...
- outils E/S simples et complexes, un système de menu pour initialiser des programmes, outils de gérance des fichiers résultats, un système couplé d'outils de visualisation

- représentation de systèmes et solveurs linéaires, représentation de systèmes et solveurs de systèmes non linéaires
- maillage d'éléments finis et de différences finies, champs scalaires et vectoriels sur des maillages d'éléments
- une collection d'éléments finis, plusieurs algorithmes d'éléments finis et structures de données utiles en éléments finis
- outils de calculs des probabilités, générateur de nombres aléatoires, méthodes de solution pour des équations différentielles ordinaires stochastiques, outils pour des champs aléatoires, maillages adaptatifs, estimation d'erreur, méthodes de manipulation de problèmes à plusieurs maillages, méthodes de décomposition de domaines
- méthodes éléments finis généralisées (formulations mixtes)
- support pour le calcul parallèle
- et plusieurs exemples de simulateurs de problèmes comme le transfert de chaleur, l'élasticité, et la mécanique des fluides.

Dans nos travaux on s'est intéressé aux outils de programmation relatifs à la résolution de problèmes de mécanique des solides non linéaires par la méthode des éléments finis.

5.3 Outils de programmation des éléments finis (EF) dans Diffpack

L'utilisation de Diffpack ne requiert pas une connaissance avancée en C++, cependant il est impératif de connaître quelques principes de base du C++ afin de pouvoir se servir de Diffpack. De plus, il est utile de développer des aptitudes à programmer pour pouvoir tirer le maximum d'avantages à programmer avec Diffpack.

Ainsi, on doit distinguer et bien comprendre quelques propriétés fondamentales au langage C++ et dont nous donnons quelques exemples :

- La programmation orientée objet est basée sur l'utilisation de classes et d'objets.
- Une classe est formée de deux parties : des variables membres, des fonctions membres. Les variables membres étant les attributs de la classe tandis que les fonctions membres sont des méthodes de transformation et d'échanges des données.
- Un objet d'une classe est une entité ayant les propriétés(attributs et fonctions) développées dans sa classe.
- Une classe peut hériter d'une autre classe. L'héritage peut-être de trois formes : un héritage public, un héritage protégé et un héritage privé. L'héritage est dit public lorsque l'héritier a accès à toutes les propriétés de la classe dont elle hérite, l'héritage est dit protégé lorsque l'héritier a accès à toutes les propriétés de la classe dont elle hérite qui sont publiques ou protégées, l'héritage est dit privé lorsque l'héritier a accès à toutes les propriétés de la classe dont elle hérite qui sont juste publiques.
- Les relations entre classes peuvent être aussi : une association ou une composition. Une classe A est associée à une classe B lorsqu'un objet de B est une variable membre dans la classe A. Une classe A est composée d'une classe B lorsqu'une fonction de A utilise un objet de la classe B.

Ce bref aperçu démontre la capacité du C++ à créer des abstractions assez diversifiées. Cette section est consacrée à l'introduction au développement et à la programmation des éléments finis dans l'environnement Diffpack. Afin d'atteindre cet objectif, le paragraphe suivant est consacré à la présentation d'un simulateur de résolution par éléments finis non linéaire dans Diffpack.

5.3.1 Implantation d'un simulateur EF non linéaire dans Diffpack

L'apprentissage du développement et de la programmation éléments finis dans Diffpack peut être divisé en trois sections. La première concerne les notions de base dans Diffpack, la seconde est relative à l'implantation d'un solveur élément fini linéaire et la dernière à l'implantation d'un solveur élément fini non linéaire.

5.3.1.1 Concepts de base

Les concepts de base dans Diffpack peuvent être regroupés suivant les sujets suivants :

- *Initialisation de l'environnement de Visual C++ 6.0* : Le travail dans l'environnement de Diffpack requiert des initialisations relatives à la configuration du langage Visual C++ 6.0. Ainsi l'option "active configuration" du C++ doit être mis à "Release", ceci est relié au fait que la version 3.0 de Diffpack avec laquelle nos travaux ont été effectués est fonctionnelle seulement en mode "Release". Le programmeur doit également initialiser les répertoires utiles du C++ en mettant à jour les adresses disponibles à l'option TOOLS > OPTIONS > DIRECTORIES en fonction du répertoire d'installation de Diffpack

- *Objets et tableaux dans Diffpack* : Le développement de codes numériques dans Diffpack est rendu plus facile grâce à l'existence de classes permettant une représentation numérique proche du modèle mathématique (Fig 5.1). Par exemple au niveau algèbre linéaire on distingue l'organisation des classes de tableaux, de matrices et de vecteurs dans Diffpack. Comme il est illustré dans le schéma(Fig 5.1) ci-dessous, le design des hiérarchies de la classe **array** peut sembler compliqué à première vue, mais la raison pour un tel design est la flexibilité, la généralité et la robustesse.

La subdivision du concept de vecteur en plusieurs types **VecSimplest**, **VecSimple**, **VecSort** et **Vec** est motivée par les diverses fonctionnalités de la classe **Type**. Le principe de l'héritage est utilisé ici de façon effective afin que les procédures d'indexation, d'allocation, et de désallocation soient programmés et testées seulement une seule fois. Les classes VecSimplest, VecSimple et Vec sont définies pour des vecteurs dont l'indexation commence par 1 contrairement à la convention du C/C++ où l'indice des tableaux commence à 0.

La hiérarchie du concept vecteur possède une classe de base **VecSimplest(Type)** qui est aussi simple que possible. Il n'y a aucune condition nécessaire sur le type des données d'entrée du tableau (**Type**), autre que la classe **Type** doit avoir un constructeur sans argument; c'est tout simplement un tableau en C/C++ de données d'entrées **Type** avec

une classe comme interface. Cette classe nous sera d'une grande utilité dans nos travaux comme il est démontré à la section 6.2.2.
Une classe un petit peu plus avancée, **VecSimple**, est dérivée de **VecSimplest**, héritant de toutes les variables et fonctions de **VecSimplest.**

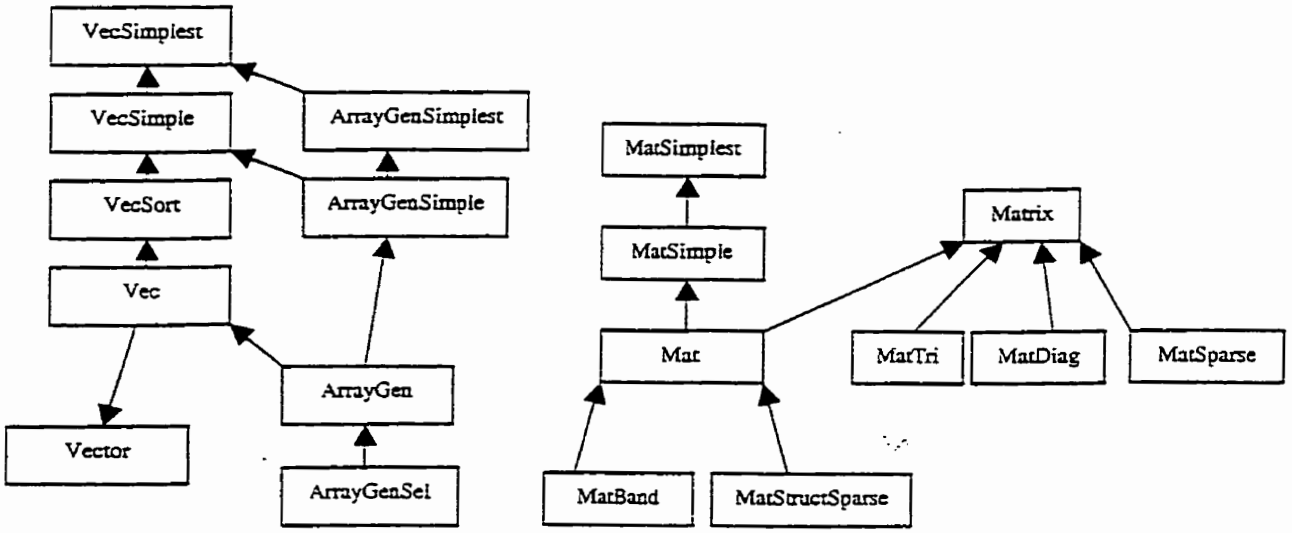


Figure 5.1 Hiérarchie des classes pour vecteurs(a) et matrices(b)

La classe **VecSimple** peut initialiser un vecteur, elle peut aussi égaliser deux vecteurs. Elle permet de créer des tableaux, où chaque entrée est un objet d'une classe (**Type**). Cette classe (**Type**) doit contenir une copie du constructeur, ainsi que les opérateurs suivant : **operator=**, **operator<<**, et **operator>>**.

La classe **VecSort** est une sous classe de **VecSimple** et elle ajoute des fonctionnalités d'impression des données d'entrée du vecteur. Les procédures d'impression requièrent que la classe d'entrée (**Type**) ait défini les opérateurs <, >, <= et >=. Si la classe **Type** peut aussi supporter les opérateurs +, -, *, et /, on peut alors réaliser des opérations numériques avec les données d'entrée.

Une classe avec plusieurs opérations numériques (comme la norme, le produit scalaire, etc...) est la classe **Vec**.

Afin de généraliser les fonctionnalités de la classe **VecSimplest**, la classe **ArrayGenSimplest** permet d'ajouter aux fonctionnalités de **VecSimplest** la possibilité d'avoir des tableaux à indices multiples comme des tenseurs. Ainsi **ArrayGenSimplest** dérive de **VecSimplest**. Cette même approche permet de définir les classes **ArrayGen**, et **ArrayGenSimple**. **ArrayGenSimple** peut imprimer son contenu et égaliser les tableaux entre eux. La classe **ArrayGen** peut, quant à elle, réaliser toutes les opérations numériques disponibles dans **Vec** par une dérivation de la classe **Vec**.

Il est à noter que les fonctionnalités numériques de **Vec(Type)** n'ont pas de sens quand **Type** est **int**, de même il n'y a pas de **ArrayGen(int)**.

Une approche similaire est utilisée pour le design de la hiérarchie pour les matrices. La classe pour une matrice la plus primitive et dense est appelée **MatSimplest**, la classe **MatSimple** est une extension par fonctionnalité, tandis que **Mat** est utilisée pour des opérations numériques.

- *Abstractions usuelles : Maillages et Champs* : Parmi les abstractions développées dans Diffpack on distingue celles relatives au maillage et aux divers champs de données afin de bien refléter les entités mathématiques et physiques d'un simulateur. Ainsi dans la formulation mathématique d'une méthode discrète pour des équations aux dirivées partielles à variable scalaire, on suppose la fonction inconnue u comme un champ scalaire(*field*) défini sur un maillage(*grid*).

Les champs sont divisés en deux ensembles : les champs scalaires et les champs vectoriels (Fig. 5.2). Leurs classes de base respectives sont **Field** et **Fields**. Les classes pour le maillage ont pour classe de base la classe **Grid**. On développera dans la section 5.3.3 de façon plus détaillée une sous classe de la classe **Grid**, **GridFE**.

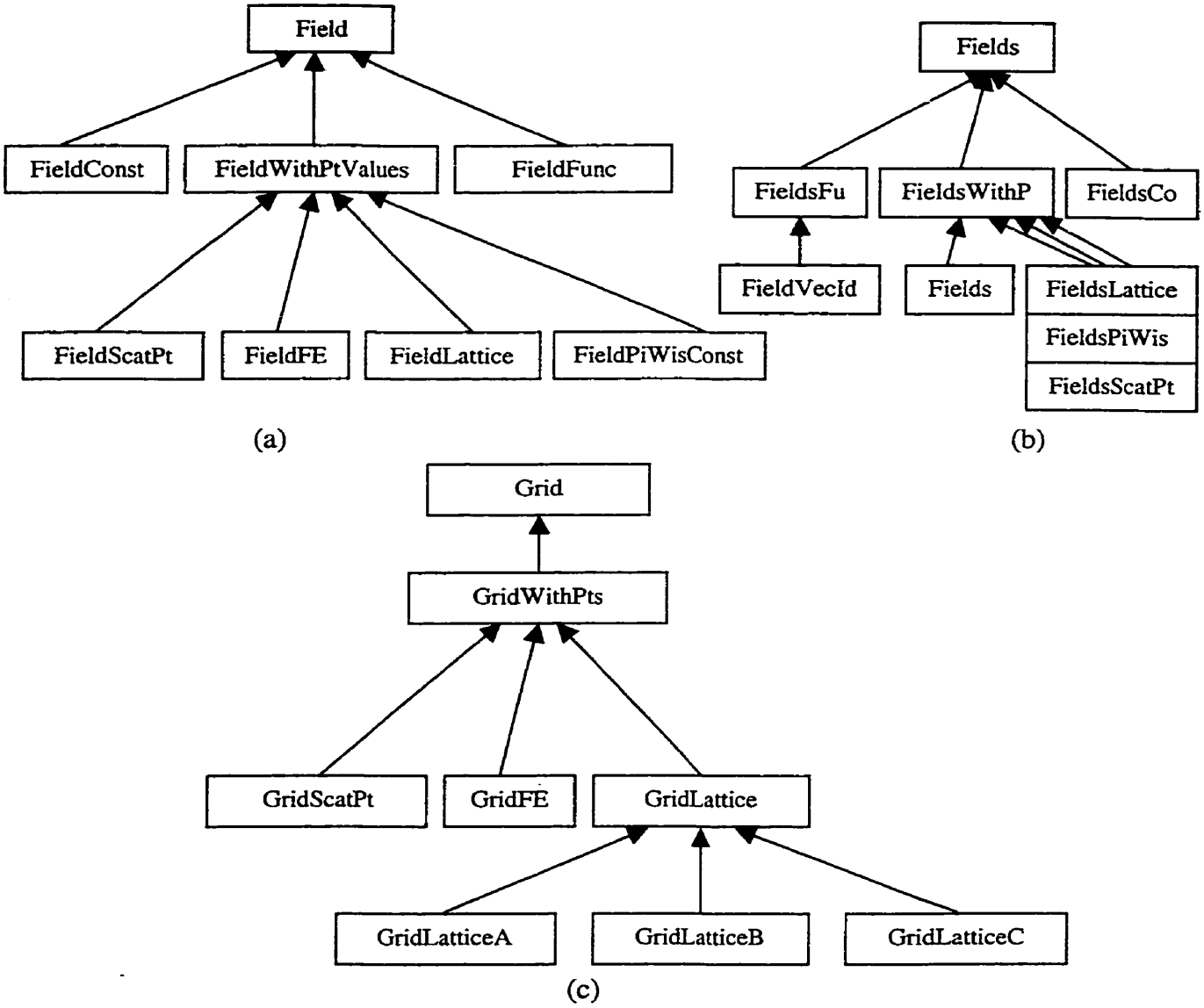


Figure 5.2 Hiérarchie des classes pour les champs scalaires(a) et les champs vectoriels(b) et pour le maillage (c)

- *Codage d'un simulateur d'E.D.P. comme une classe* : Un simulateur sous Diffpack comprend des éléments de base à partir desquels l'on peut développer son code. Ces éléments sont :

- Main.cpp : fichier principal
- Simulator_name.cpp : Le code source de l'application
- Simulator_name.h : Le code header (déclaration) de l'application

D'autres classes peuvent s'ajouter à la classe du simulateur.

5.3.1.2 Solveur E.F. linéaire

Un solveur EF linéaire dans Diffpack doit répondre à certaines normes intrinsèques à Diffpack. L'implantation d'un tel solveur se fait à travers une classe qui est dénommée d'après la classe du solveur ou du simulateur. Cette étude essayera donc d'être la plus générale possible afin de faire ressortir les particularités d'un tel simulateur dans Diffpack. La structure "UML" (Unified Modeling Language) d'un tel simulateur a la forme suivante :

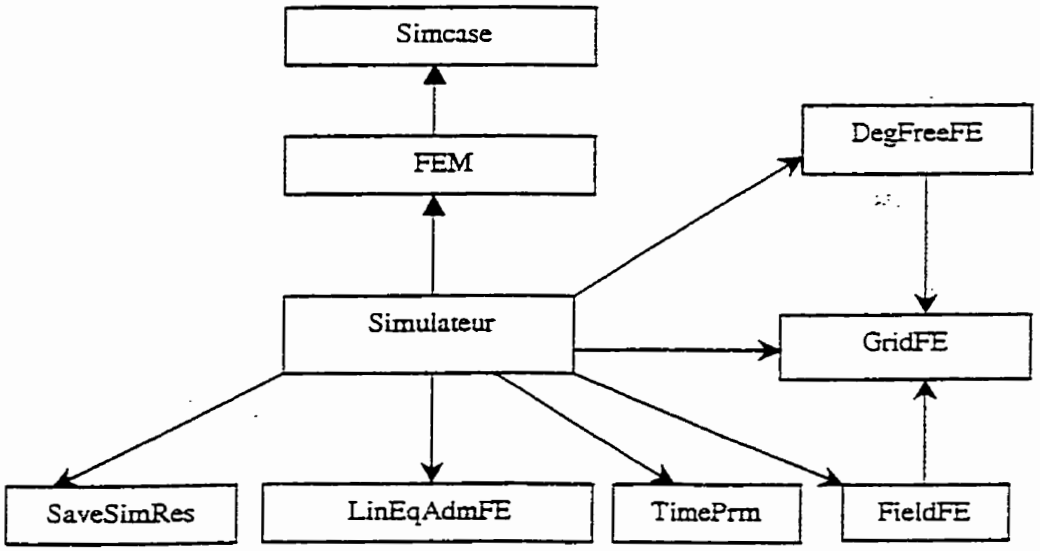


Figure 5.3 Structure UML d'un simulateur EF linéaire (flèche droite : héritage, flèche ronde : composition)

Le simulateur doit être une sous-classe de la classe **FEM** étant donné que la méthode des éléments finis est choisie pour la discrétisation spatiale. La classe **FEM** est disponible dans la librairie de classes prédéfinies de Diffpack et elle contient des algorithmes par défaut qui sont souvent utilisés dans des problèmes d'éléments finis. Une étude plus détaillée de la dite classe est faite à la section 5.3.2.

La classe du simulateur utilise donc des variables pointeurs, objets des classes **LinEqAdmFE**(Interface aux classes **LinEq*** utiles pour les systèmes linéaires), **DegFreeFE**(Administrateur des degrés de liberté pour les systèmes linéaires d'éléments finis), **GridFE**(Maillage d'éléments finis), **FieldFE**(Champ scalaire d'éléments finis).

Les principales fonctions utiles à la résolution d'un problème sont déclarées dans le fichier header, elles sont définies et codées dans le fichier source et sont appelées dans le main. Les fonctions standard d'un simulateur de base sont donc :

Les fonctions membres standards	
adm	Administre le menu
define	Définit les items du menu
scan	Lit les données à partir du menu et initialise les objets internes(maillages, champs, conditions limites, etc....)
integrands (ElmMatVec&,FiniteElement&)	Définit l'intégrande pour la matrice de rigidité et le résidu élémentaire à partir de la forme variationnelle
fillEssBC	Établit les conditions limites essentielles
solveProblem	Appelle la résolution du problème
resultreport	Enregistre les résultats obtenus.
Routines optionnelles	
calcElmMatVec(int,ElmMatVec&,Finite Element&)	Calcule les matrices et les vecteurs élémentaires
integrands4side	Définit l'intégrande sur la frontière à partir de la formulation faible
saveResults	Enregistre les résultats obtenus dans des fichiers

Tableau 5.1 : Les fonctions membres standards d'un simulateur EF linéaire

Le programmeur peut accroître la flexibilité de son programme en :

- utilisant le système de menus disponible dans Diffpack :

Le système de menu disponible dans Diffpack est défini dans la classe **MenuSystem**. Le système peut opérer sous différents modes.

- Mode graphique
- Mode question/réponse

- Et Mode terminal interactif

L'utilisation du menu requiert :

- L'initialisation du menu dans le main : Le main doit initialiser l'objet du menu global juste après avoir appelé la fonction **initDiffpack** :

global_menu.init ("Description du simulateur","nom_du_simulateur");

L'extension du solver peut être réalisé dans le main par :
global_menu.multipleLoop (Simulator); où simulator est un objet de la classe du simulateur.

La fonction **multipleLoop** dans le système de menu administre l'appel de **adm** pour construire le menu et initialiser le simulateur, l'appel de **solveProblem** pour calculer une solution numérique, et l'appel de **resultReport** pour reporter des résultats relatifs à la simulation.

- Définition de l'arbre du menu : Le menu est constitué d'items et de sous-menus.

Chaque menu ne peut comporté plus de 3 sous-menus. La définition des items est réalisée grâce à la fonction **addItem**, fonction membre de **MenuSystem** tandis que la définition des sous-menus est réalisée grâce à la fonction **addSubmenu**, fonction membre de **MenuSystem.**

Déclaration de la fonction addItem :

```
addItem
(
  int     level,                  // menulevel to add item (1 2 3)
  const String&  command,         // stream and graphic command for item
  const String&  cl_option_string,// commandline command "+argument"
  const String&  description,     // header and description
  const String&  default_answer,  // default answer - last read answer
  const String&  valid_answer,    // validity string eg. R[-1 1]
  char    hot_key,                // short key stream and graphic
  char    cl_option_char,         // short key commandline "-argument"
```

```
MenuCallBack* func = NULL    // callback function
);
```

Déclaration de la fonction **addSubMenu:**

```
bool addSubMenu
(
  int    level,                 // level of menu hiearchy to add submenu
  const String& name_label,     // prompttext for submenu: "prompt>"
  const String& command,        // stream and graphic command
  const String& description,    // header and description
  char   hot_key                // hot key for streams and graphics
  );
```

Ainsi dans la fonction **define** de la source du simulateur le menu est souvent défini avec les commandes suivantes : **menu.addItem(..,...,..); menu.addSubMenu(..,...,...);**

- créant de façon dynamique des maillages à l'aide de préprocesseurs :

Il y a plusieurs préprocesseurs simples d'éléments finis dans Diffpack. On peut citer : PreproBox, ProproSupElset, Geompack

- prenant avantage des différents outils de visualisations disponibles dans Diffpack :

La classe **SaveSimRes** permet de sauvegarder des champs de données dans des fichiers. Le format de sauvegarde est propre à Diffpack et est appelé le *simres* format. Afin d'imprimer on exporte le champ en simres au fichier de format requis par le programme d'impression. Ceci est accompli en utilisant un *filtre*. Des filtres pour plusieurs systèmes d'impression existent dans Diffpack, par exemple , Vtk, Matlab, AVS, IRIS Explorer, Plotmtv, et GumPlot.

5.3.1.3 Solveur E.F. non linéaire

Un solveur EF non linéaire est une extension du solveur E.F. linéaire. En plus de dériver de la classe **FEM,** le simulateur doit dériver de la classe **NonLinEqSolverUDC** qui est une classe pour les systèmes non linéaires. Une étude plus détaillée de la dite

classe est faite à la section 5.3.2. La structure UML d'un tel simulateur a la forme suivante :

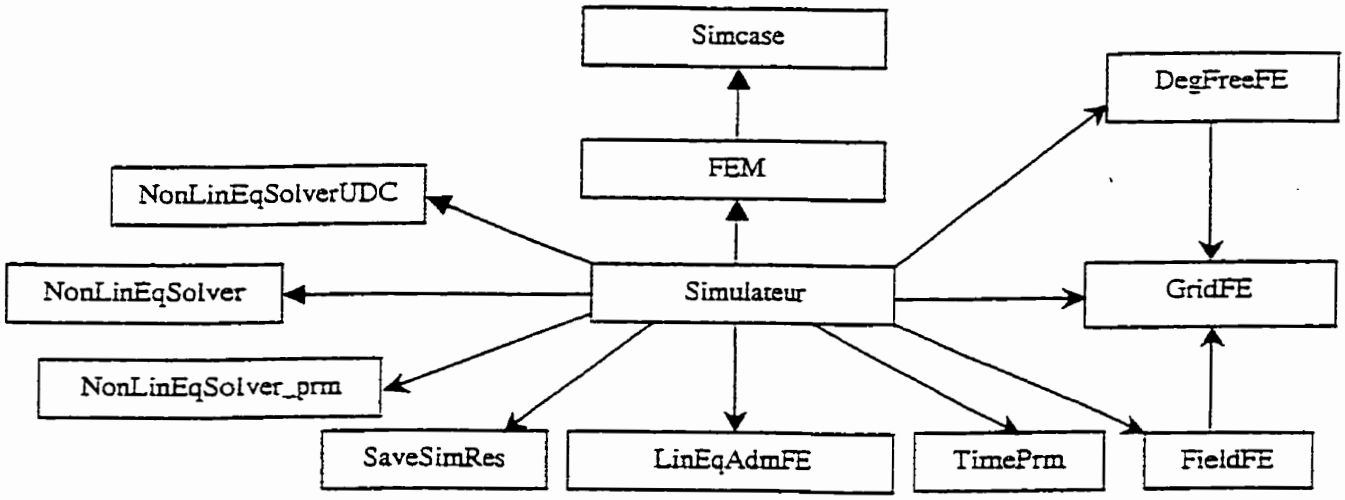


Figure 5.4 Structure UML d'un simulateur EF non linéaire (flèche droite : héritage, flèche ronde : composition)

On ajoute donc des variables pointeurs permettant de gérer le problème non linéaire. Parmi ces variables on peut citer :

- le système non linéaire : objet de **NonLinEqSolver**
- les paramètres d'initialisation du système non linéaire : objet de **NonLinEqSolver_prm**

En plus de nouvelles variables, des nouvelles fonctions s'ajoutent. La fonction majeure est **MakeAndSolveLinearSystem** qui est une fonction membre de **FEM**. Elle est généralement appelée par le code source du simulateur pour résoudre le système linéaire à chaque itération.

5.3.2 Introduction aux classes **FEM** et **NonLinEqSolverUDC**

Ces classes sont disponibles dans la librairie de classes de Diffpack et sont utiles à la définition d'un simulateur d'éléments finis non linéaire. Ce sont les classes de base d'un simulateur E.F. non linéaire. Dans cette section nous parcourrons la définition de chaque classe en décrivant les variables et fonctions utiles.

La classe **FEM** comprend une partie protégée et une partie publique. La partie protégée est consacrée aux variables membres de la classe et la partie publique aux fonctions utiles de la classe. Ces fonctions, pour la plupart virtuelles, permettent au programmeur d'avoir accès aux algorithmes éléments finis standard tels que l'assemblage d'éléments finis, l'intégration numérique sur les éléments, et diverses autres fonctions dont la liste est donnée ci-après :

makeSystem - résout le système linéaire issue d'une méthode d'éléments finis. La fonction est entièrement générale et appelle des fonctions virtuelles pour la définition de la matrice et du vecteur élémentaire. Une version par défaut de cette fonction virtuelle est donnée. Elle appelle simplement une fonction pour l'intégration numérique sur l'élément qui à son tour appelle une fonction virtuelle integrands que le programmeur doit implémenter dans la classe dérivée du simulateur. Il est donné plusieurs implémentations de cette fonction dépendamment des besoins du programmeur.

calcElmMatVec – Calcule la matrice et le vecteur élémentaire. Quand le programmeur dispose d'expressions analytiques pour le vecteur et la matrice élémentaires, ces expressions peuvent simplement être implémentées dans une sous-classe(classe de simulation).
Si une intégration numérique est requise, la classe **FEM** dispose d'une implémentation par défaut de *calcElmMatVec* qui effectue les initialisations nécessaires et un appel à *numItgOverElement*. Ce dernier effectue l'intégration numérique et donne l'intégrande en appelant la fonction *integrands* qui doit être implémentée dans la classe du simulateur.
Dépendamment des besoins du programmeur, cette fonction peut être modifiée afin de satisfaire à l'algorithme de résolution du problème,(intégration sur l'intérieur d'un élément, intégration pour différentes formulations faibles, etc...).

numItgOverElm - Intégration numérique sur l'élément. L'objet de la classe **FiniteElement** doit être initialisé en appelant sa fonction *refill* avant l'appel de *numItgOverElm*. La fonction évalue l'intégrande de l'intégrale en appelant la fonction virtuelle *integrands*. Si l'intégrande est défini par une fonction (dérivée de **IntegrandCalc**) plutôt qu'une fonction virtuelle de la classe **FEM**, une version supplémentaire de *numItgOverElm* peut supporter ce cas de figure.

numItgOverSide - Intégration numérique sur le côté (surface) d'un élément. Contrairement à *numItgOverElm* l'objet de **FiniteElement** est automatiquement initialisé pour le côté courant (refill4side) dans *numItgOverSide*. (Cependant, *refill* doit être appelé, mais cela est habituellement fait avant *numItgOverElm* qui précède normalement le calcul de l'intégrale sur le côté). Une version supplémentaire permet de tenir compte du cas où l'intégrande sur le côté (surface) est défini en terme d'une fonction de **IntegrandCalc**.

integrands4energyErrorNorm - une fonction similaire à *integrands*, mais dont le but est d'évaluer la contribution (d'un point d'intégration dans un élément) à la norme de l'énergie de l'erreur. Ainsi, l'on doit dans cette fonction avoir l'inconnu principal du problème et une solution de référence (ex : une solution exacte).

attachMassMatrix – Cette fonction permet à l'utilisateur d'attacher de façon explicite sa propre matrice masse. Si la fonction n'est pas appelée, diverses autres fonctions pour calculer des dérivées et des flux lissés utiliseront une matrice masse interne, qui est automatiquement réutilisée dans des calculs ultérieurs (à moins que le maillage soit modifié durant une simulation). En pratique, plusieurs simulateurs n'auront ou n'attacheront jamais une matrice masse, mais plutôt reposent sur **FEM** et ses calculs internes de matrices masses.. Si la matrice masse doit être utilisée directement dans le solveur, elle est disponible comme variable membre, mass_mat.

En plus, on distingue des fonctions calculant des dérivées de champs définis sur des éléments finis. Comme ces dérivées sont discontinues, plusieurs algorithmes de lissage sont offerts. Ce sont :

smoothField, *makeGradient*, *makeFlux*, *smoothDerivative*, *smoothGradient* – qui permettent le calcul de dérivées lissées de champs sur des éléments finis.

smoothMultipleFields – Elle réalise l'algorithme de lissage des moindres-carrés pour plusieurs champs simultanément.

modifyField4Constraints - Cette fonction colle les contraintes de façon explicite sur le vecteur dans le champ de données. La fonction *modifyField4Constraint* est automatiquement appelée à partir de toutes les fonctions dans la classe **FEM** qui effectue du lissage sur des champs de données. Ainsi, cette fonction n'est utilisée par les programmeurs que pour effectuer des lissages locaux dans le simulateur.

La classe **NonLinEqSolverUDC** contient l'information nécessaire à l'implémentation d'un solveur non linéaire . Elle comprend une partie publique composée de fonctions dont voici la liste.

makeAndSolveLinearSystem – résoud le sous-problème linéaire à travers un algorithme non linéaire itératif.

makeResidual - calcule le résidu des équations non linéaires en se basant sur l'approximation disponible la plus récente de la solution non linéaire.

normOfResidual - retourne la norme du vecteur résidu. Pour la méthode de Newton, cela est habituellement très facile du fait que le résidu est la composante de droite du système linéaire(dans **FEM** en utilisant la classe **LinEqAdmFE**, on retourne simplement lineq.b().norm(l2)). Cependant, dans d'autres méthodes l'on a généralement besoin de calculer le résidu.

startSolveInContinuationMethod – La fonction *continuationSolve* de la classe **NonLinEqSolver** est utilisée pour remettre le paramètre Lambda à l'échelle de paramètres pratiques , ex : Nombre de Reynolds dans la mécanique des fluides.

comment - peut être utilisée pour donner un commentaire dans le résultat de solveurs non linéaires.

beforeSolveInContinuationMethod – cette fonction est appelée avant chaque appel de la fonction *solve* dans la méthode de continuation dans la classe **NonLinEqSolver** (la fonction *continuationSolve*). L'utilisateur doit implémenter une version de cette fonction dans sa classe de simulation. L'argument Lambda est la valeur courante du paramètre dans la méthode de continuation. Ce paramètre est toujours dans l'intervalle [0,1], où 0 correspond à un problème facile et 1 correspond au problème que l'on désire résoudre.

5.3.3 Introduction à la classe GridFE

La classe **GridFE** a pour rôle de définir le maillage d'éléments finis dans Diffpack. Son rôle dans une analyse éléments finis est d'autant plus important qu'il est l'outil par lequel l'utilisateur définit la discrétisation de son problème. La définition d'un objet **GridFE** peut se faire à l'aide de préprocesseurs ou mailleurs, allant du simple au compliqué. Par exemple, on a :

- PreproBox : maillage de domaine topologiquement équivalent à un hexaèdre.
- PreproSupElset : maillage en utilisant le concept de super-élement
- Geompack : maillage automatique en éléments triangulaires et tétraédriques dans le cas de géométrie quelconque.

Dans notre étude, seul le mailleur PreproBox est utilisé. C'est un simple préprocesseur à base rectangulaire. Les éléments générés sont des types suivants :

- famille d'éléments multi-linéaires (éléments linéaire en 1D, éléments quadrilatéraux bilinéaires en 2D, éléments trilinéaires en 3D)

- éléments de type multi-quadratique

La Fig. 5.5 présente les relations hiérarchiques entre les classes **Prepro**, **Geometry** et **Partition** qui sont à la base de la classe **PreproBox**.

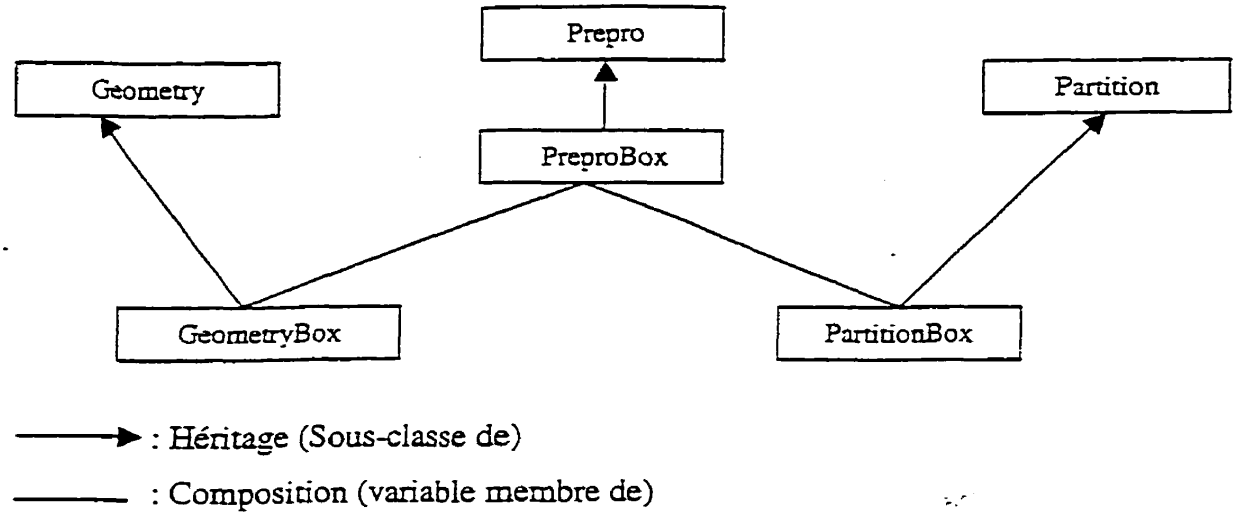


Figure 5.5 Structure UML relatif à la classe PreproBox

La création d'un objet **Grid** suit donc une syntaxe propre à sa classe. Ainsi à l'aide de la fonction *readOrMakeGrid* l'on peut à la fois :

- lire un fichier contenant des données d'un maillage dont le format correspond au format du fichier produit par la fonction *print* de **GridFE**.
  Comme exemple, si le fichier formaté est Tmp.txt alors la commande est : *readOrMakeGrid (*grid, Tmp.txt);*

- Initialiser un objet Grid à partir d'un fichier d'entrée contenant un appel d'initialisation à l'aide d'un préprocesseur ou simplement d'une chaîne de caractères contenant un appel d'initialisation à l'aide d'un préprocesseur.
  Par exemple, le fichier peut s'appeler **"inputfile.i"** et contenir le texte suivant :
  set time parameters = dt=0.04 t in [0,0.16]
  set gridfile = PREPROCESSOR=PreproBox | d=2 [0,1]x[0,1] | d=2 elm_tp=ElmB4n2D div=[8,8], grading=[1,1]

ok !mark the end of the file.

ou la chaîne de caractère peut être :

String s = PREPROCESSOR=PreproBox | d=2 [0,1]x[0,1] | d=2 elm_tp=ElmB4n2D div=[8,8], grading=[1,1]

La commande est donc *readOrMakeGrid (g," inputfile.i ");* dans le premier cas et *readOrMakeGrid (g,s);* dans le second.

La classe **GridFE** a pour objectifs de donner :

- La liste et les coordonnées des points aux nœuds
- La liste et connectivité de chaque élément
- un numéro pour le matériau associé à chaque élément
- le type de chaque élément (linéaire, quadratique, trilinéaire, etc...)
- les nœuds sujets à certaines conditions limites.

L'utilisateur de Diffpack a très peu d'interaction avec la classe **GridFE** dans la mesure où l'objet Grid est intimement lié au préprocesseur qui le génère. Afin de spécifier les conditions limites, les nœuds et les éléments peuvent être marqués avec des conditions limites, ceci est fait automatiquement avec le préprocesseur PreproBox. L'utilisateur peut cependant modifier les conditions limites du problème en utilisant la fonction *redefineBoInds*. Les différents types d'éléments disponibles dans Diffpack sont :

- ElmTensorProd1
- ElmTensorProd2
- ElmTensorProd

Ces éléments sont obtenus par produit tensoriel d'éléments unidimensionnels, ils sont coûteux et on leur préfère des éléments plus spécialisés comme :

- ElmB2n1D, où Elmxnyd désigne un élément fini à x nœuds et de dimension y
- ElmB4n2D

- ElmB9n1D
- ElmT3n2D
- ....

Aux sections 5.6 et 5.7 nous passerons en revue les éléments qui nous ont été d'intérêt dans nos travaux.

La classe **GridFE** est donc une classe nécessaire à l'analyse E.F.. Elle possède en dehors des fonctions citées ci-haut d'autres fonctions dont voici un aperçu :

redim, getNoNodesInElm, setNoNodesInElm, getMaterialType, setMaterialType, getElmType, setElmType, fillCoor, getElmCoor, getCoor, putCoor, getMinMaxCoord, getLocalCentroid, getGlobalCentroid, loc2glob, putLoc2Glob, setBoInd, clearBoInd, putBoIndName, getBoIndName, boSide, boNode, boNodesInElm, BoInd, redefineBoInds, addBoIndNodes, addMaterial, polygon, notEqual, scanLattice, isLattice, getLattice, isReallyLattice, refineIfBox, setUniformMesh, setNonUniformMesh, hasUniformMesh, move, startIterator, nextPt, getNoPoints, isNode, nearestPoint, isNodeInElm, boundingBox, findElmAndLocPt, print, scan, newNodalNumbering, nodeCouplings, constraintCouplings, calcBandwidth, calcTolerance

Comme on le voit, la classe **GridFE** en dehors de relations intrinsèques(Fig 5.2) dont elle bénéficie, dispose d'une panoplie de fonctions qui facilitent de beaucoup la tâche de l'utilisateur.

5.4 Éléments finis 2D à 4 nœuds ElmB4n2D

L'analyse 2D par éléments finis est réalisé avec l'élément bilinéaire 2D défini dans Diffpack. Dans cette section notre objectif est de décrire cet élément.
Il est caractérisé par quatre nœuds aux sommets du quadrilatère, qui sont numérotés autour de l'élément selon la convention suivante(voir aussi page 106) :

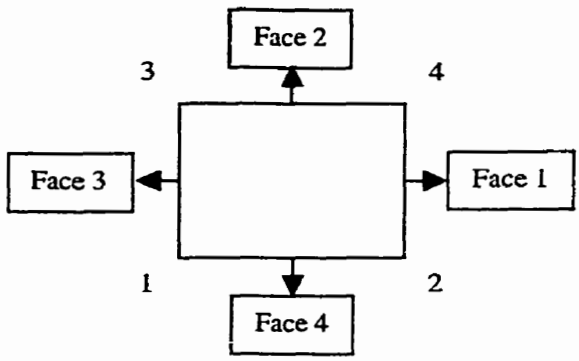


Figure 5.6 Numérotage de l'élément ElmB4n2D(ou Q4)

En mécanique des solides chaque nœud possède deux degrés de liberté (i.e. peut se déplacer en x et y). Les déplacements u et v sont supposés être des fonctions linéaires de l'élément, c'est-à-dire :

$$u = b_1 + b_2 x + b_3 y + b_4 xy, \qquad v = b_5 + b_6 x + b_7 y + b_8 xy \tag{5.1}$$

où b_i (i = 1, 2, ..., 8) sont des constantes. Le champ de déplacements défini par (5.1) doit satisfaire les 8 équations suivantes écrites aux nœuds :

$$\begin{aligned} u_1 &= b_1 + b_2 x_1 + b_3 y_1 + b_4 x_1 y_1 \\ u_2 &= b_1 + b_2 x_2 + b_3 y_2 + b_4 x_2 y_2 \\ &\cdots \\ v_4 &= b_5 + b_6 x_4 + b_7 y_4 + b_8 x_4 y_4 \end{aligned} \tag{5.2}$$

En résolvant ces équations, on peut exprimer les coefficients b_i (i = 1, 2, ..., 8) en termes des déplacements et coordonnées des nœuds de l'élément. Alors en substituant ces coefficients dans (5.1) et en réarrangeant les termes, on obtient :

$$\begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{bmatrix} N_1 & 0 & \cdots & N_4 & 0 \\ 0 & N_1 & \cdots & 0 & N_4 \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ \cdots \\ u_4 \\ v_4 \end{Bmatrix} \tag{5.3}$$

Nous avons deux systèmes de coordonnées sur l'élément : les coordonnées globales (x,y) et les coordonnées naturelles (ξ,η). Les fonctions de forme en fonction des coordonnées naturelles sont: [Ref-44]

$$N_1 = \frac{1}{4}(1-\xi)(1-\eta), \qquad N_2 = \frac{1}{4}(1+\xi)(1-\eta)$$
$$N_3 = \frac{1}{4}(1-\xi)(1+\eta), \qquad N_4 = \frac{1}{4}(1+\xi)(1+\eta) \qquad \text{avec} \quad \sum_{i=1}^{4} N_i = 1 \qquad (5.4)$$

Le champ de déplacements en fonction des déplacements aux nœuds est alors donné par :

$$u = \sum_{i=1}^{4} N_i u_i, \qquad v = \sum_{i=1}^{4} N_i v_i \qquad (5.5)$$

de sorte que u et v sont des fonctions bilinéaires sur l'élément.

5.5 Éléments finis 3D à 8 nœuds ElmB8n3D

L'analyse 3D par éléments finis est réalisé avec l'élément trilinéaire 3D défini dans Diffpack. Dans cette section on veut décrire brièvement cet élément. Il est caractérisé par huit nœuds aux sommets de la boîte hexaédrique, qui sont numérotés autour de l'élément selon le couverture présentée dans la Fig. 5.7 ci-dessous.

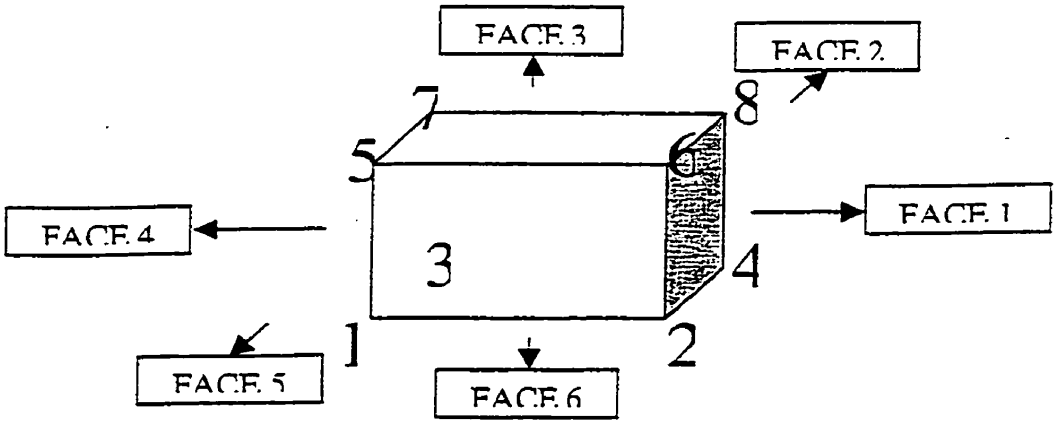


Figure 5.7 Numérotage de l'élément ElmB8n3D(H8)

En mécanique des solides, chaque nœud possède trois degrés de liberté (i.e. peut se déplacer en x, y et z). Les déplacements u, v et w sont supposés être des fonctions trilinéaires de l'élément, c'est-à-dire :

$$\begin{aligned} u &= b_1 + b_2 x + b_3 y + b_4 z + b_5 xy + b_6 xz + b_7 yz + b_8 xyz \\ v &= b_9 + b_{10} x + b_{11} y + b_{12} z + b_{13} xy + b_{14} xz + b_{15} yz + b_{16} xyz \\ w &= b_{17} + b_{18} x + b_{19} y + b_{20} z + b_{21} xy + b_{22} xz + b_{23} yz + b_{24} xyz \end{aligned} \tag{5.6}$$

où b_i (i = 1, 2, ..., 24) sont des constantes.

Le champ de déplacements défini par 5.1 doit satisfaire les 24 équations suivantes écrites aux nœuds :

$$\begin{aligned} u_1 &= b_1 + b_2 x_1 + b_3 y_1 + b_4 z_1 + b_5 x_1 y_1 + b_6 x_1 z_1 + b_7 y_1 z_1 + b_8 x_1 y_1 z_1 \\ u_2 &= b_1 + b_2 x_2 + b_3 y_2 + b_4 z_2 + b_5 x_2 y_2 + b_6 x_2 z_2 + b_7 y_2 z_2 + b_8 x_2 y_2 z_2 \\ &\cdots \\ w_8 &= b_{17} + b_{18} x_8 + b_{19} y_8 + b_{20} z_8 + b_{21} x_8 y_8 + b_{22} x_8 z_8 + b_{23} y_8 z_8 + b_{24} x_8 y_8 z_8 \end{aligned} \tag{5.7}$$

En résolvant ces équations, on peut exprimer les coefficients b_i (i = 1, 2, ..., 24) en termes des déplacements et coordonnées des nœuds de l'élément. Alors en substituant ces coefficients dans (5.6) et en réarrangeant les termes, on obtient :

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = \begin{bmatrix} N_1 & 0 & 0 & \cdots & N_8 & 0 & 0 \\ 0 & N_1 & 0 & \cdots & 0 & N_8 & 0 \\ 0 & 0 & N_1 & \cdots & 0 & 0 & N_8 \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ w_1 \\ \cdots \\ u_8 \\ v_8 \\ w_8 \end{Bmatrix} \tag{5.8}$$

Nous avons deux systèmes de coordonnées sur l'élément : les coordonnées globales (x,y,z) et les coordonnées naturelles (ξ_1, ξ_2, ξ_3). Les fonctions de forme en fonction des coordonnées naturelles sont :

$$N_1 = \frac{1}{8}(1-\xi_1)(1-\xi_2)(1-\xi_3), \qquad N_2 = \frac{1}{8}(1+\xi_1)(1-\xi_2)(1-\xi_3)$$
$$N_3 = \frac{1}{8}(1-\xi_1)(1+\xi_2)(1-\xi_3), \qquad N_4 = \frac{1}{8}(1+\xi_1)(1+\xi_2)(1-\xi_3)$$
$$N_5 = \frac{1}{8}(1-\xi_1)(1-\xi_2)(1+\xi_3), \qquad N_6 = \frac{1}{8}(1+\xi_1)(1-\xi_2)(1+\xi_3) \quad \text{avec } \sum_{i=1}^{8} N_i = 1 \quad (5.9)$$
$$N_7 = \frac{1}{8}(1-\xi_1)(1+\xi_2)(1+\xi_3), \qquad N_8 = \frac{1}{8}(1+\xi_1)(1+\xi_2)(1+\xi_3)$$

Le champ de déplacements en fonction des déplacements aux nœuds est donné par :

$$u = \sum_{i=1}^{8} N_i u_i, \qquad v = \sum_{i=1}^{8} N_i v_i, \qquad w = \sum_{i=1}^{8} N_i w_i \qquad (5.10)$$

de sorte que u, v et w sont des fonctions trilinéaires sur l'élément.

Dans ce chapitre, on a présenté les grandes lignes de la programmation de simulateurs non linéaires dans l'environnement Diffpack en mettant l'accent sur la modélisation orientée objet et les différents modèles éléments finis disponibles dans Diffpack. Le chapitre suivant sera consacré à la présentation des comparaisons et à la validation des simulateurs programmés ainsi que des résultats obtenus.

CHAPITRE VI

SIMULATION NUMÉRIQUE : COMPARAISON ET VALIDATION

6.1 Hyperélasticité

6.1.1 Introduction

L'étude de la formulation de l'hyperélasticité nous a permis de développer un programme permettant la résolution de problèmes pour des matériaux hyperélastiques compressibles et incompressibles. Le programme est orienté objet et est basé sur l'environnement offert par le logiciel Diffpack.

Nous avons implanté une formulation spatiale donnée dans [Ref-10]. Les auteurs ont développé un code en Fortran, permettant de résoudre l'hyperélasticité, nommé Flagshyp.

Notre approche a été de se servir de Flagshyp comme point de départ et de modéliser l'hyperélasticité, non avec une notation indicielle comme dans Flagshyp mais plutôt avec une notation matricielle, tout en profitant de la flexibilité offerte par la programmation orientée objet de Diffpack. Dans un premier temps nous présenterons la

structure de Flagshyp et ensuite, nous montrerons comment une implantation orientée objet fût réalisée. Les deux algorithmes sont simples et l'on peut passer de l'un à l'autre sans une compréhension approfondie de la notion orienté objet.

6.1.2 Hyperélasticté dans Flagshyp

Flagshyp est un programme en Fortran qui résoud pour 7 types différents de matériaux le problème hyperélastique. La formulation développée dans Flaghyp utilise une notation indicielle pour représenter les tenseurs.
Une présentation du développement de l'hyperélasticité fait dans Flagshyp pour le matériau 5 est présenté à l'annexe A. L'algorithme procédural présenté est celui utilisé pour résoudre le problème avec les éléments quad4db et hexa8db correspondant à notre développement dans Diffpack. Chaque fonction est accompagnée d'un commentaire, lorsque possible, décrivant les formules utilisées dans le programme.

6.1.3 Hyperélasticité dans Diffpack

Le programme Hyperlasticité développé dans Diffpack a une structure orienté objet. Il est donc caractérisé par l'utilisation de classes. Ces classes comportent des attributs et fonctions permettant de résoudre le problème. Comme l'hyperélasticité est un problème non linéaire, la structure de notre simulateur est semblable à celle présentée à la section 5.3.1.3. pour les applications non linéaires. Cette structure est présentée à la Fig. 6.1.

La classe **Hyperelasticity** utilise donc comme variables pointeurs des objets de classes diverses, ce qui permet de profiter d'abstractions multiples afin de réduire le temps de programmation et de débuggage. Les principales fonctions de la classe ainsi que leurs descriptions sont décrites à l'annexe B.

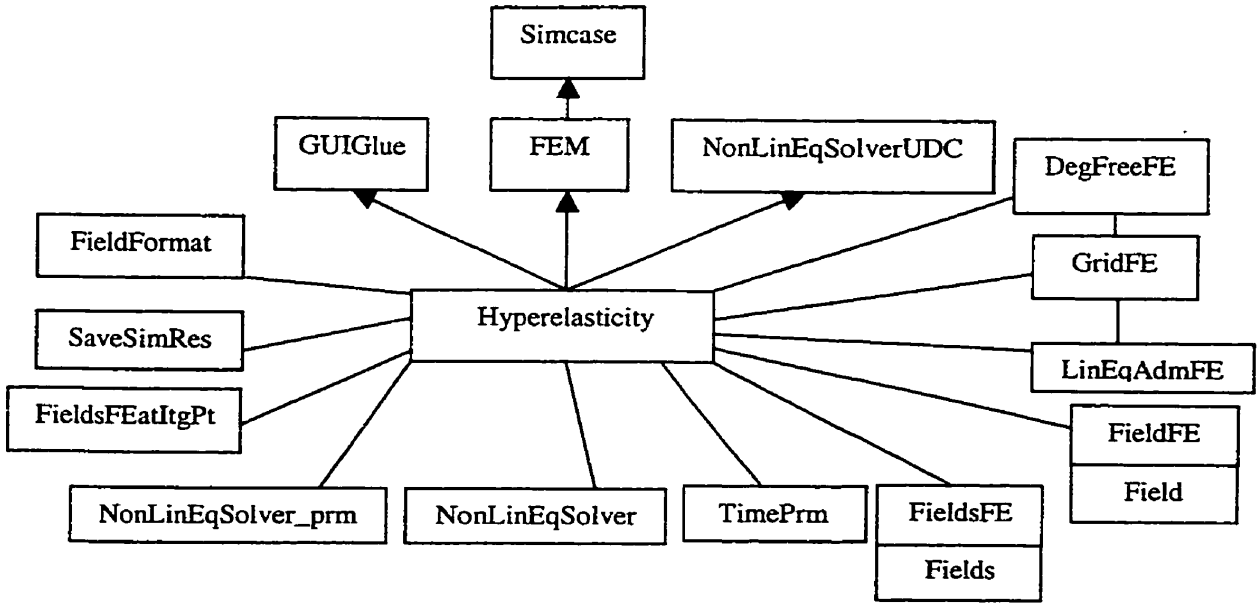


Figure 6.1 Structure UML du simulateur Hyperelasticity
(flèche droite : héritage, ligne simple : composition)

6.1.4 Comparaison et validation

Afin de valider le programme **Hyperelasticity** développé dans Diffpack un exemple simple est utilisé (Fig. 6.2). Considérons en effet, un bloc solide de forme rectangulaire. Le bas du solide est fixé, son dessus est soumis à un chargement de pression constante et ses bords sont libres.

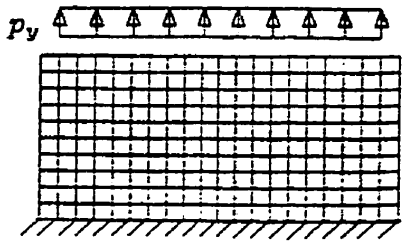


Figure 6.2 Cas simple d'étirage

Le même exemple est tourné sous Flagshyp afin d'effectuer une comparaison. Flagshyp s'est avéré un code fiable à la suite de comparaisons faites entre Flagshyp et

Abaqus. On compare le déplacement maximum aux nœuds du solide. L'exemple est présenté au tableau 6.1 en 2D.

- On applique en 2 pas de chargement une pression de 50 avec E=250 et nu=0.3 sur un corps ayant pour dimensions 1x1.

Type	Nbre d'éléments	Déplacement maximum	
		Flagshyp	Diffpack
Déformations planes	Matériau Compressible Neo Hookéen		
	1 élément	1.176	1.17644
	16 éléments	1.188	1.18773
	Matériau Incompressible Neo Hookéen		
	1 élément	1.180	1.18026
	16 éléments	1.194	1.19414

Tableau 6.1 : Comparaison Flagshyp et HyperElasticity dans Diffpack

Comme on le voit les résultats donnés pour les matériaux compressible et incompressible Neo Hookéen par Diffpack et Flagshyp en déformations planes sont semblables. La déformation est de l'ordre de 20% et la seule différence dans les résultats obtenus se trouve au niveau de la précision des résultats soit à 10^{e-4} près.
Le cas 3D et le cas en contrainte plane donne des résultats concluants sous Diffpack, cependant on remarque que le code Flagshyp comprend des erreurs de programmation pour ces cas. Afin de valider nos résultats nous ferons des exemples de contact ayant des résultats analytiques.

6.2 Contact

6.2.1 Introduction

L'étude de la formulation du contact nous a permis de développer un programme permettant la résolution du problème du contact pour des matériaux hyperélastiques compressibles et incompressibles. Le programme est orienté objet et est basé sur l'environnement offert par le logiciel Diffpack.

Le programme Contact découle de l'implantation de l'hyperélasticité. Les corps en interaction sont déformables. La structure UML du programme se présente comme suit :

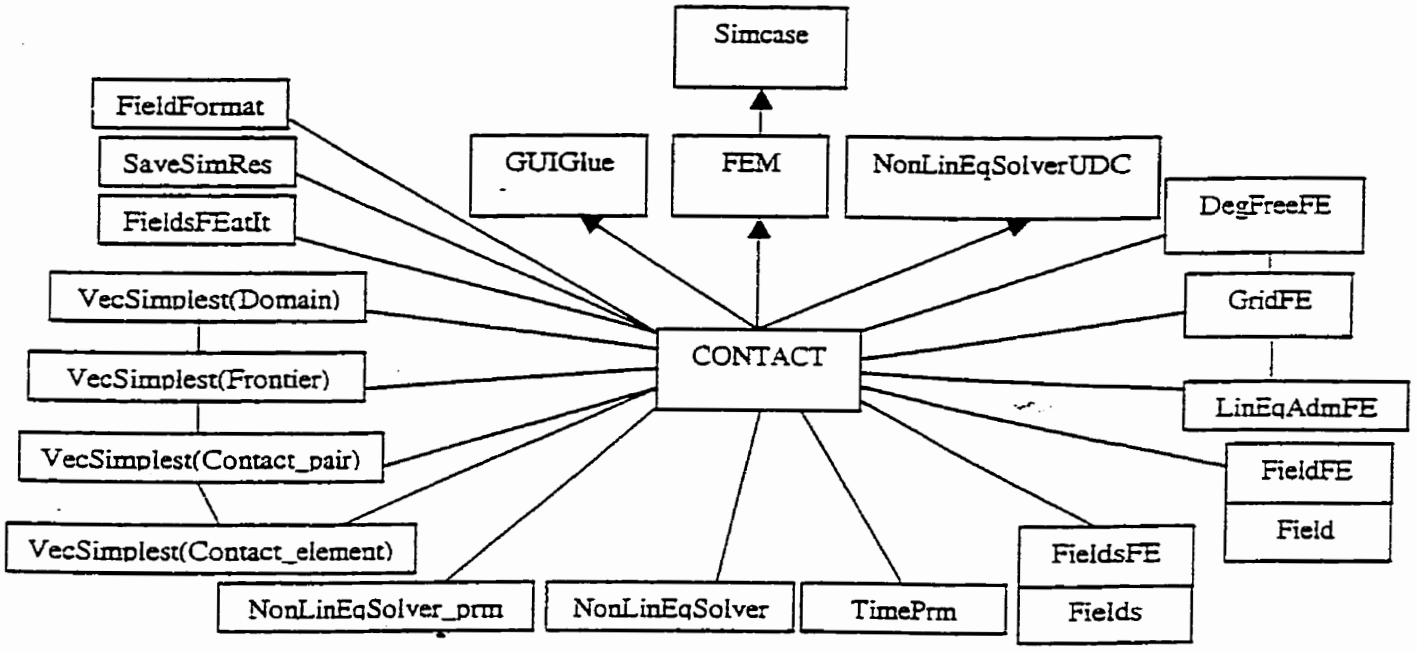
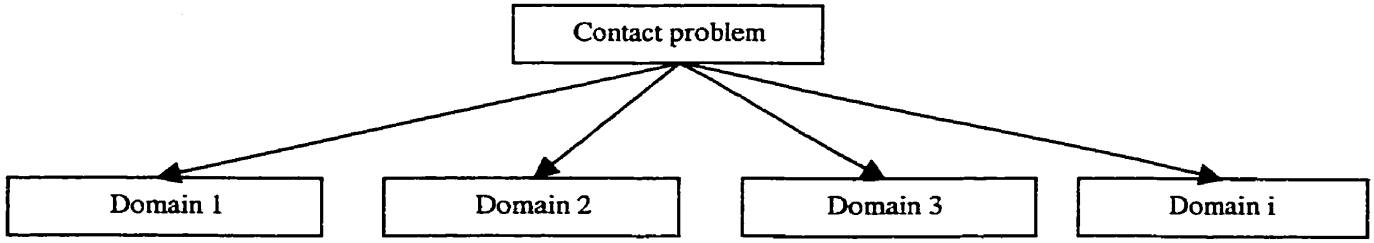


Figure 6.3 Structure UML du simulateur Contact
(flèche droite : héritage, ligne simple : composition)

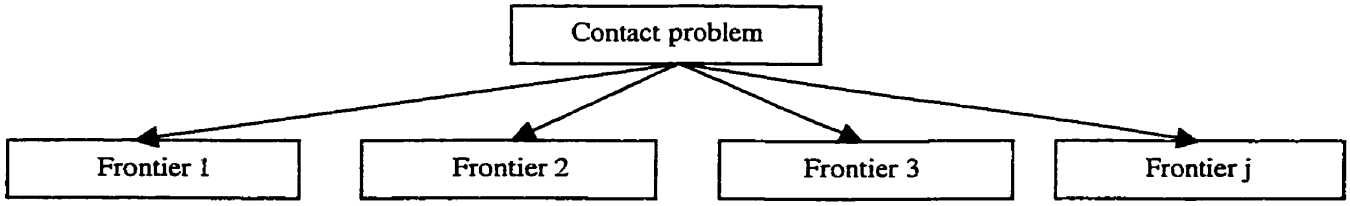
Comparativement au programme Hyperelasticity le programme Contact dispose de classes additionnelles qui permettent de gérer les diverses structures algorithmiques reliées à la résolution du contact. La classe **Contact** gère comme la classe **Hyperelasticity** les lois de comportement des solides en contact, cependant elle dispose de variables supplémentaires qui sont les objets des classes additionnelles lui permettant de réaliser la recherche des éléments de contact, le calcul des matrices et vecteurs élémentaires de contact et leur assemblage dans le système global afin de résoudre le problème de conatct.

6.2.2 Indexation de classes complexes dans le simulateur de Contact

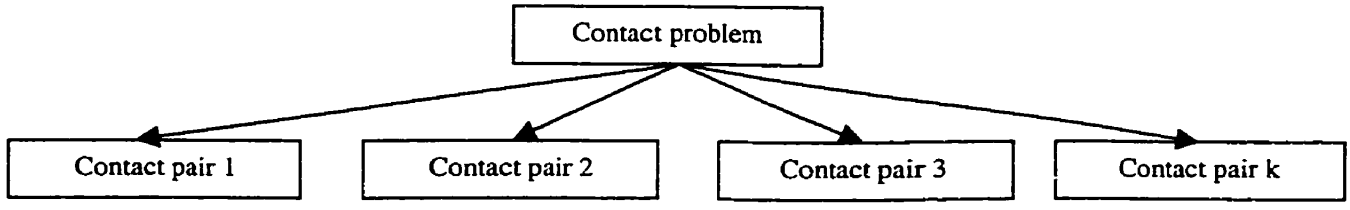
Afin de profiter des abstractions disponibles dans Diffpack et compte tenu de la nécessité de devoir gérer plusieurs corps dans le cas du contact nous utiliserons un moyen d'indexation disponible dans Diffpack.
Une bonne indexation permettra de sauver du temps à écrire et à débugger le code. En effet, un problème de contact est composé de plusieurs corps appelés ici *Domain*. Soit donc le cas montré à l'illustration suivante :



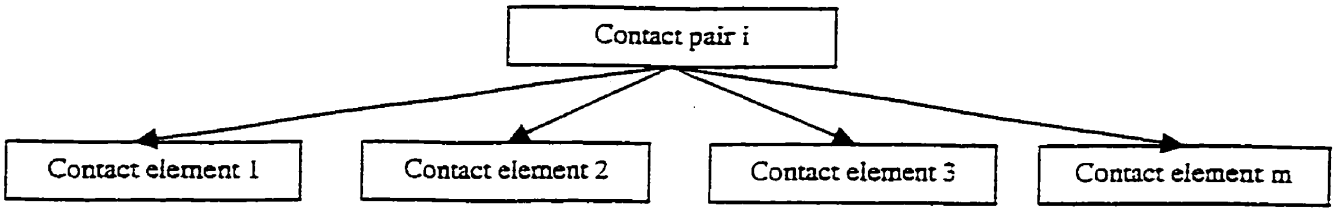
Sur chaque domaine on peut avoir plusieurs surfaces ici nommés *frontier* intervenant dans l'analyse du contact.



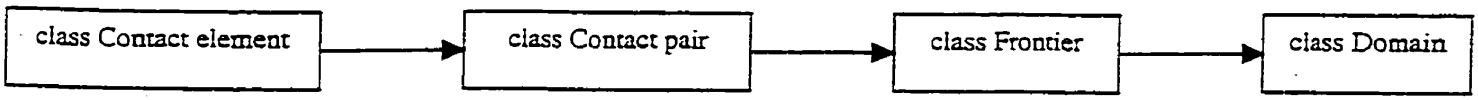
Le problème de contact est résolu par pair d'objets en contact. On distingue le contacteur et la cible qui peuvent être *Domain 1* et *Domain 2* dans le cas présenté plus haut. Pour chaque corps on peut avoir des surfaces de contact qui peuvent être *Frontier 1 et Frontier 2*. Un problème de contact est donc modélisable par un ou plusieurs paires de corps en contact.



Sur chaque paire de corps en contact il est défini des éléments de contact qui doivent respecter des contraintes de contact et qui aident à la résolution des équations d'équilibre.



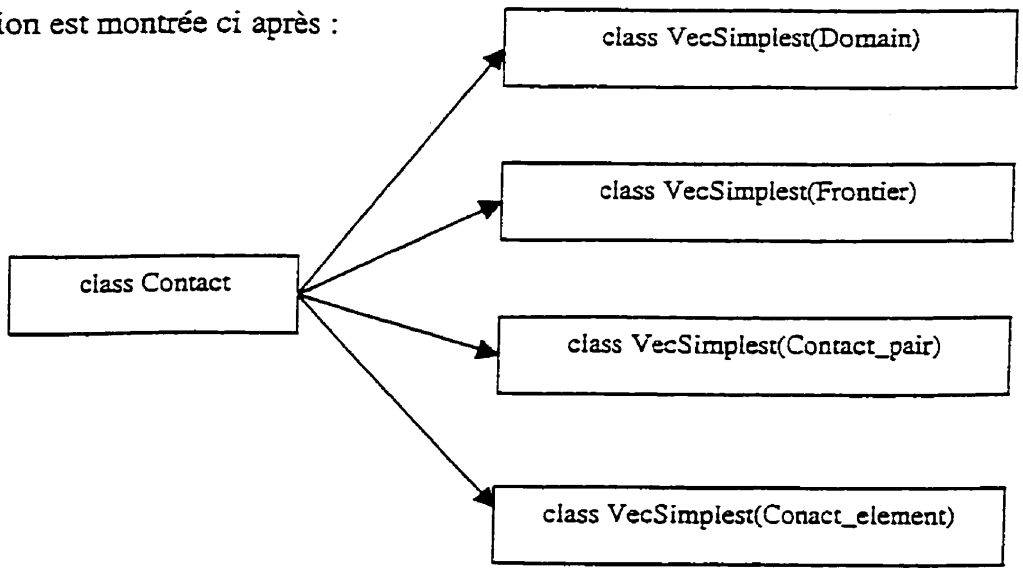
Une illustration des relations décrites ci-dessus en utilisant les relations de programmation en C++ (notation UML) est la suivante :



Avec la flèche indiquant une relation de possession. Nous pouvons donc définir quatre classes. Avec :

- class Contact_element ayant un objet de class Contact_pair.
- class Contact_pair ayant deux objets de class Frontier nommés target et contactor.
- Class Frontier ayant un objet de class Domain

Toutes ces classes sont indexées avec **VecSimplest_Type**, qui est une fonctionnalité de Diffpack. La classe qui gère toutes les données relatives à ces classes est bien évidemment class Contact qui possède des objets indexés de ces quatre classes.
Une illustration est montrée ci après :



6.2.3 Comparaison et validation

Notre but est de modéliser tout d'abord des exemples simples et fréquents dans la littérature afin de valider le code du contact. Nous considérerons donc des cas-tests standard permettant de valider aussi le modèle hyperélastique presqu'incompressible. Les tâches de traitement des données sont présentées à l'annexe C. Les calculs effectués respectent les règles de correspondance des unités en éléments finis.

6.2.3.1 Exemple 1 : Un poinçon comprimant un bloc en 2D

Notre but à travers cet exemple est de vérifier la performance numérique de notre simulateur dans le cas 2D et de comparer le comportement des forces de contact normales par rapport à celles trouvées suivant la théorie de Euler [Ref-26]. Cet exemple est présenté à la Fig. 6.4. Il s'agit d'un poinçon rigide comprimant un bloc sous une condition de contact normal. Le bloc est fixé à sa base. La symétrie du problème nous indique que nos résultats devraient également être symétriques.

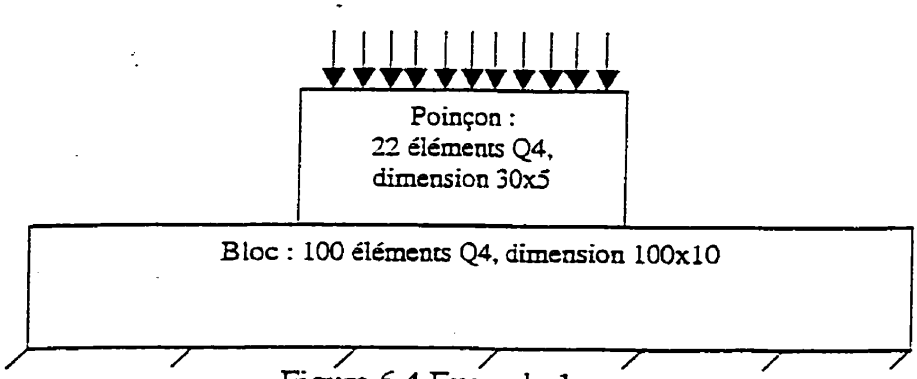


Figure 6.4 Exemple 1

Les propriétés du matériau hyperélastique incompressible néo-Hookéen en déformation plane employées pour notre simulation sont : pour le poinçon $E_{poinçon} = 10^8$, $\nu_{poinçon} = 0$, et pour le bloc $E_{bloc} = 10^3$, $\nu_{bloc} = 0.3$. Le coefficient de pénalité a été choisi $\varepsilon_N = 2.10^5$. Sous une pression constante de p = 500 notre algorithme converge de façon satisfaisante en 12 itérations. La performance numérique est présentée dans le tableau ci-après.

Itération	1	2	3	4	5	6
Erreur	2.0883e+001	9.3958e-002	3.4113e-002	9.2914e-003	2.9361e-003	9.8297e-004
Itération	7	8	9	10	11	12
Erreur	4.3298e-004	1.8085e-004	7.0828e-005	3.9179e-005	1.7405e-005	9.4737e-006

Tableau 6.2 : Performance numérique de l'exemple 1

L'environnement graphique offert par Diffpack nous permet de visualiser des champs scalaires et vectoriels sur le maillage d'éléments finis traité. Cependant on ne peut observer malheureusement qu'un seul maillage à la fois. C'est ainsi qu'on peut tirer les figures suivantes.

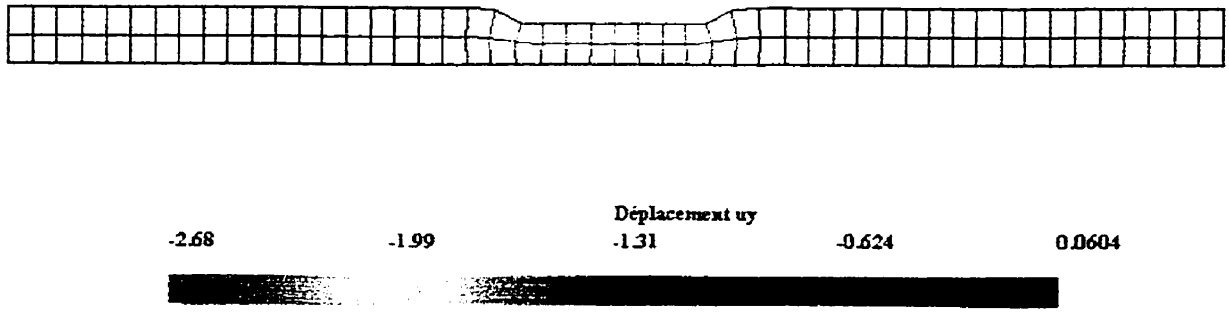


Figure 6.5 Composante y du déplacement (dir. normale) du bloc.

Le bloc connaît une déformation dans la zone de contact. La zone déformée est symétrique, ce qui respecte nos hypothèses.

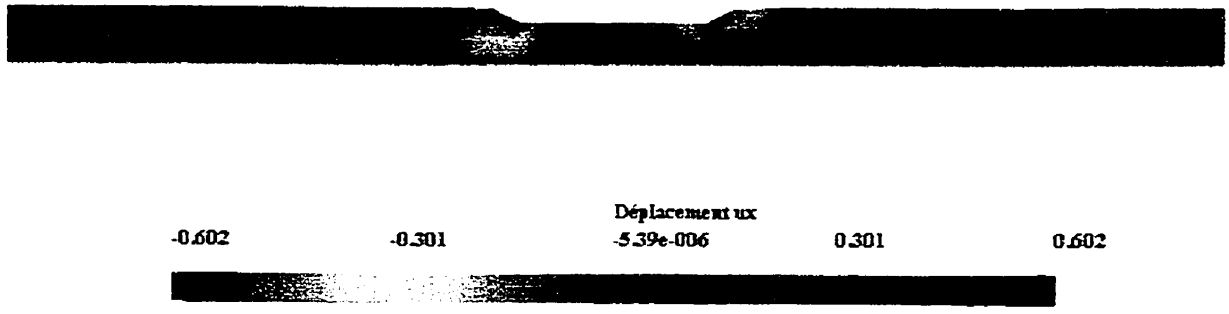


Figure 6.6 Composante x du déplacement (dir. tangentielle) du bloc

La déformation est importante et est de l'ordre de 26.8% de l'épaisseur du bloc, on observe donc de grandes déformations. Afin de valider le cas présenté, il est intéressant de représenter la courbe des forces de contact normale en fonction de la zone de contact.

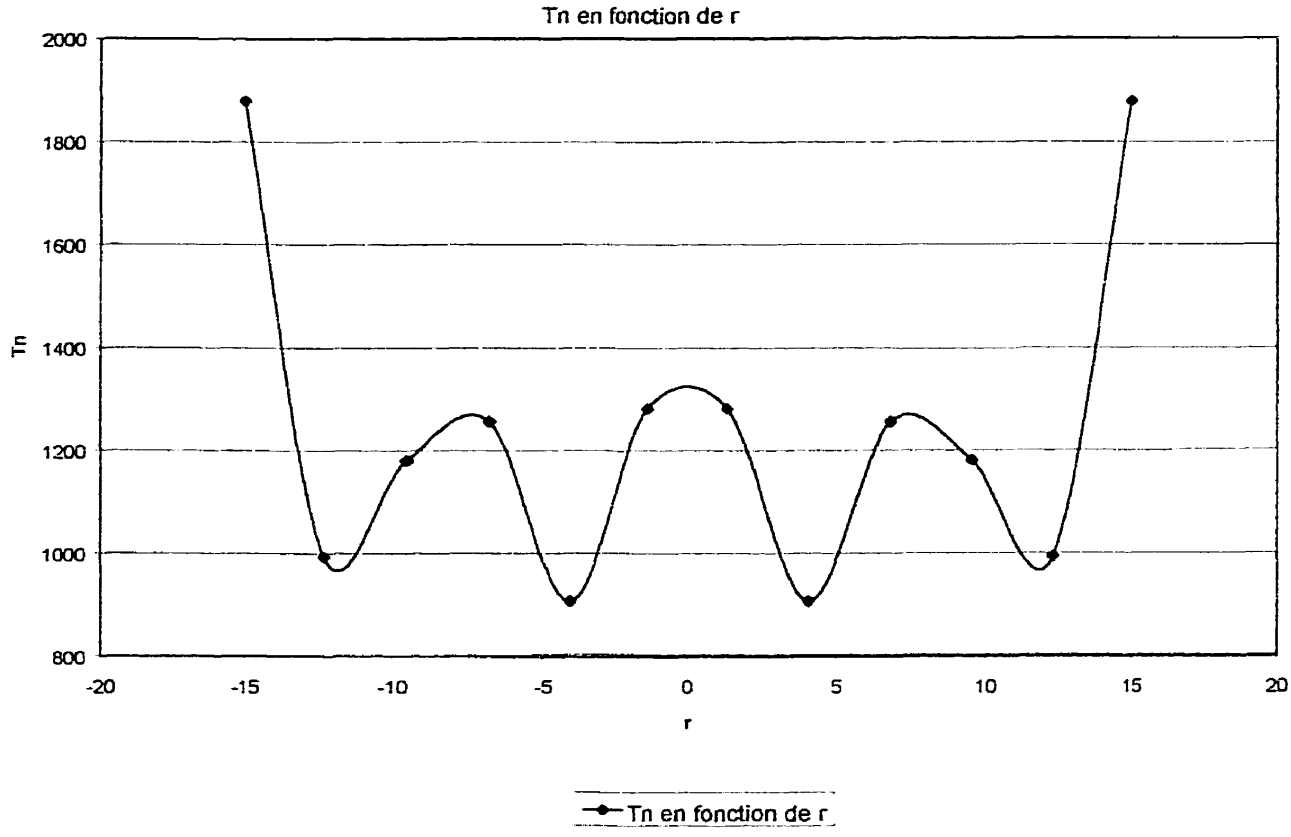


Figure 6.7 : Courbe de la force de contact normale en fonction de r

La force de contact normale suivant la théorie de Euler [Ref-26] doit croître du centre de la zone de contact vers l'extérieur de celle-ci et c'est ce qu'on observe ici. La Fig. 6.6 nous permet en plus, d'observer que les nœuds qui se déplacent le plus tangentiellement sont ceux qui se retrouvent aux extrémités de la zone de contact.

6.2.3.2 Exemple 2 : Un poinçon comprimant bloc en 3D

Notre but à travers cet exemple est de vérifier la performance numérique de notre simulateur dans le cas 3D et de comparer le comportement des forces de contact normales

et tangentielles par rapport à celles trouvées suivant la théorie de Euler [Ref-26]. Cet exemple est présenté à la Fig. 6.8. Il s'agit d'un poinçon rigide comprimant un bloc sous une condition de glissement. L'analyse du frottement effectué dans cet exemple consiste à comparer la grandeur de la force de contact tangentielle à la force de contact normale.

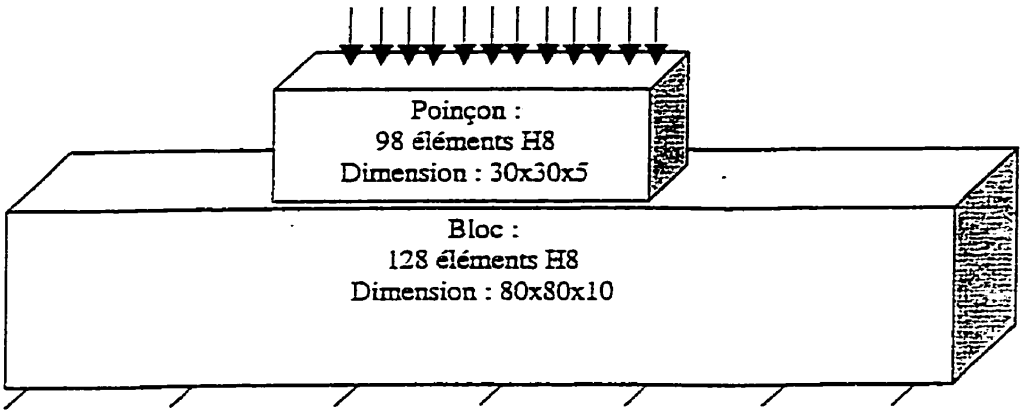


Figure 6.8 Exemple 2

Les propriétés du matériau hyperélastique incompressible néo-Hookéen employé pour notre simulation sont : au niveau du poinçon déformable $E_{poinçon} = 10^8$, $\nu_{poinçon} = 0$, et au niveau du bloc déformable $E_{bloc} = 10^3$, $\nu_{bloc} = 0.3$. Les coefficients de pénalité ont été choisis $\varepsilon_N = 10^{19}$, $\varepsilon_T = 10^{14}$. Ils sont beaucoup plus élevés que dans le cas 2D. Ceci est lié aux nouvelles dimensions des corps et au chargement qui est ici plus élevé. Le coefficient de frottement est $\mu = 0.6$.

Sous une pression constante de $p = 6x10^5$ notre algorithme converge de façon quadratique en 3 itérations. La performance numérique est présentée dans le tableau ci-après.

Itération	1	2	3
Erreur	2.9304e+001	3.0503e-003	1.4683e-007

Tableau 6.3 : Performance numérique de l'exemple 2

La Fig. 6.9 nous permet de visualiser la déformation du bloc. Comme on le voit le bloc connaît une déformation symétrique.

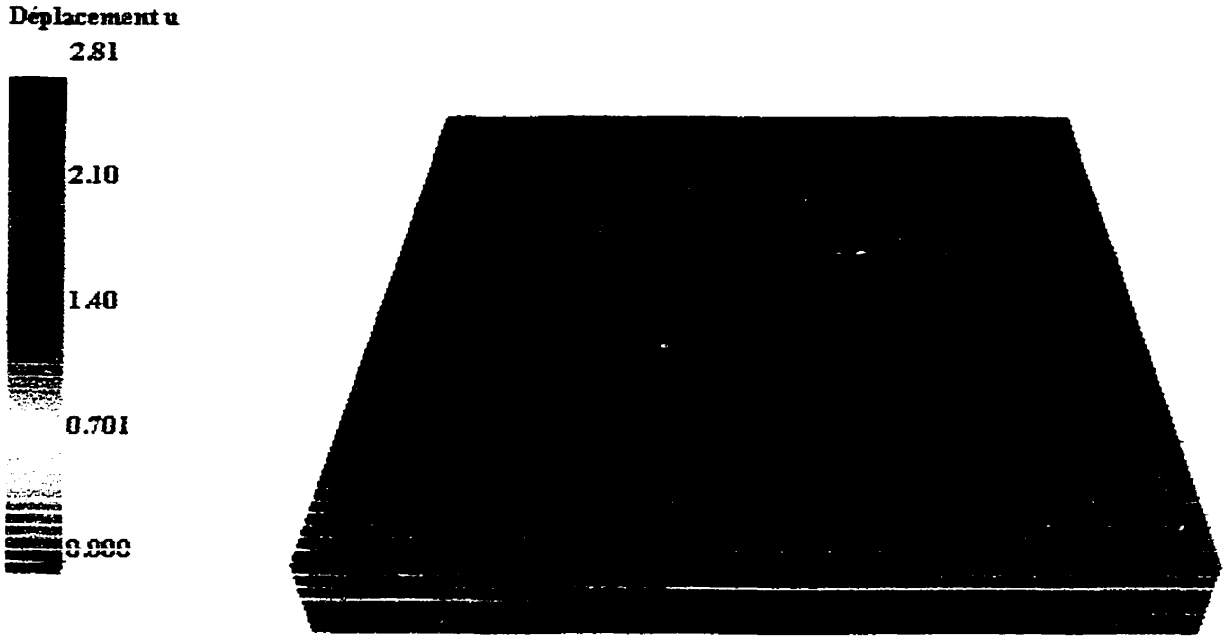


Figure 6.9 Déplacement dans la direction z du bloc.

La déformation est importante et est de l'ordre de 28.1% de l'épaisseur du bloc. Une coupe médiane permet d'obtenir la Fig. 6.10 qui représente la contrainte de déformation de Von Mises lissée sur le champ d'éléments finis du bloc.

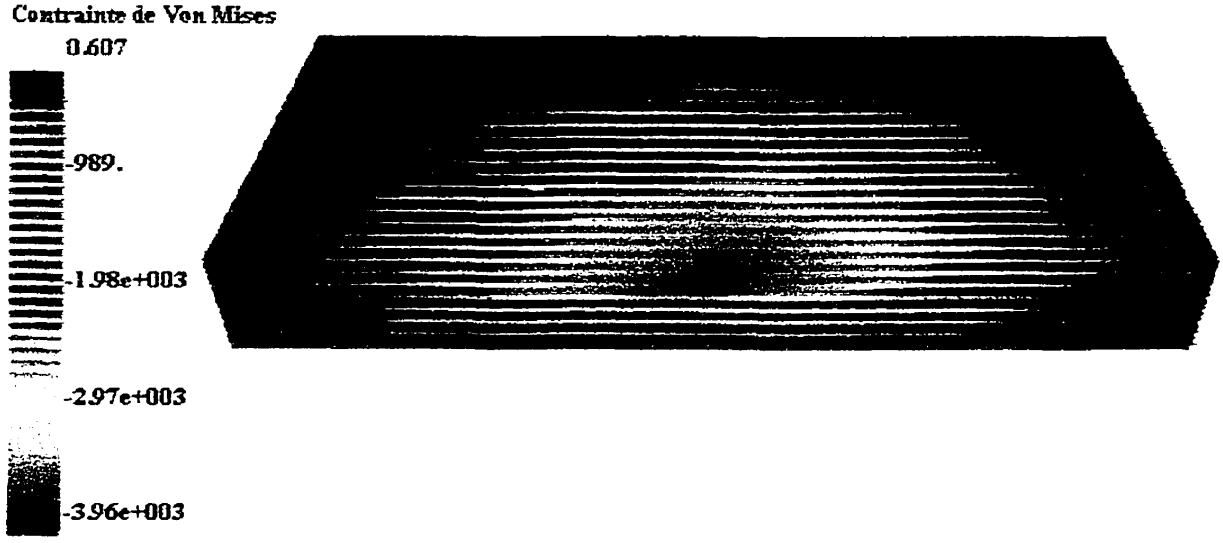


Figure 6.10 Contrainte équivalente de Von Mises lissée sur le bloc déformé.

Tout comme dans le cas précédent, afin de valider le cas présenté il est intéressant de représenter la courbe des forces de contact normale et tangentielle en fonction de la zone de contact.

La figure illustrant la variation des forces de contact en fonction de la position des points de contact par rapport au centre de la zone de contact est présentée ci-après.

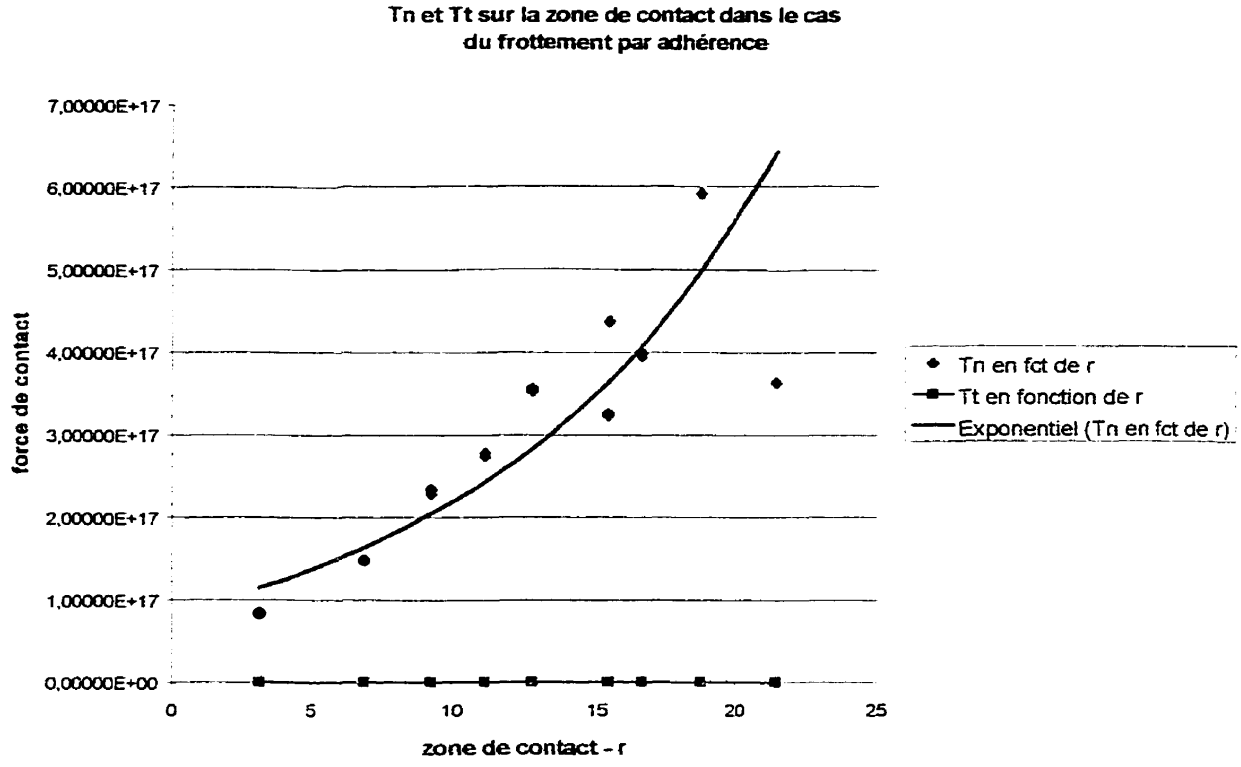


Figure 6.11 : Courbe de la force de contact en fonction de r en cas d'adhésion

Le fait que la composante de la force de contact tangentielle est plus faible par rapport à la force de contact normale s'observe très bien sur la courbe. La force de contact normale suivant la théorie de Euler [Ref-26] dans le cas d'un poinçon à base rectangulaire sur une surface plane supposée infinie doit croître du centre de la zone de contact vers l'extérieur de celle-ci et c'est ce qu'on observe ici. Des résultats obtenus, la force de contact tangentielle varie de 0 à 4.22×10^{06} et la zone de contact connaît un frottement par adhésion, ce qui est logique du fait que le chargement est normal à la zone de contact.

6.2.3.3 Exemple 3 : Un bloc rectangulaire pressé contre un cylindre creux en 2D

Notre but à travers cet exemple est de vérifier la performance de l'algorithme de recherche de la zone de contact de notre simulateur. Contrairement aux exemples précédents la zone de contact évolue à travers les itérations. Cet exemple est présenté à la Fig. 6.12. Il s'agit d'un corps en forme d'un cylindre creux sur lequel est pressé un bloc rectangulaire, plus déformable que le corps cylindrique, sous une condition de contact normal dans le cas 2D. L'analyse du contact comprend l'évaluation des algorithmes de recherche de contact.

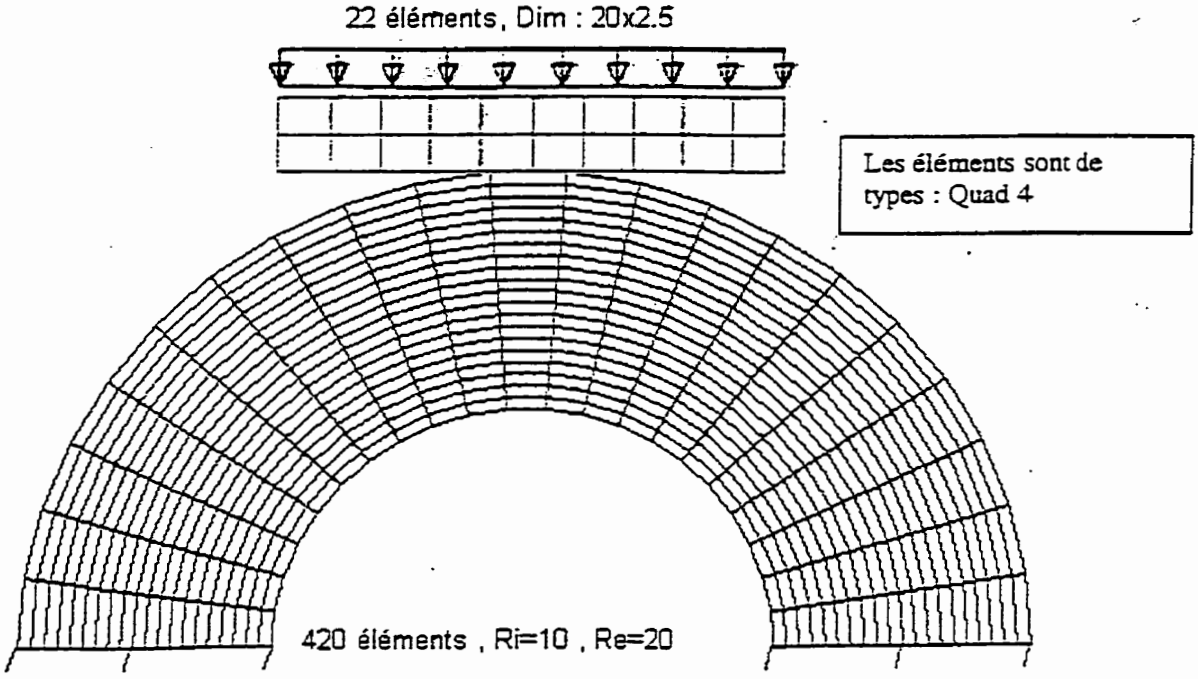


Figure 6.12 Exemple 3

Les propriétés du matériau hyperélastique incompressible néo-Hookéen en déformation plane employé pour notre simulation sont : au niveau du poinçon déformable $E_{poinçon} = 10^5$, $\nu_{poinçon} = 0$, et au niveau du cylindre creux plus rigide $E_{cyl} = 10^{10}$, $\nu_{bloc} = 0.3$. Le coefficient de pénalité a été choisi $\varepsilon_N = 2.10^6$.

Sous une pression constante de p = 100 appliquée sur les faces supérieur du bloc rectangulaire notre algorithme converge en 8 itérations . La performance numérique est présentée dans le tableau ci-après.

Itération	1	2	3	4
Erreur	7.996e-01	4.227e-01	7.800e-02	3.904e-01
Itération	5	6	7	8
Erreur	4.527e-01	1.204e-02	7.00e-03	2.991e-03

Tableau 6.4 : Performance numérique de l'exemple 3

Cet exemple a pour but de montrer l'habileté de notre algorithme de recherche du contact à repérer de façon efficace les éléments de contact à travers les itérations. Au cours de la simulation, le poinçon tend à s'enfoncer dans le bloc ce qui crée de nouveaux éléments de contact. La zone potentielle de contact comprend 11 nœuds contacteurs(1, 2 , ...11) centré à 6. Le tableau suivant présente la performance de l'algorithme de recherche du contact :

Itérations	Nœud en contact – état de contact
0	1-5 libres 6 en contact (au milieu du bloc rectangulaire) 7-11 libres
1	1-3 libres 4-8 pénétrés 9-11 libres

Tableau 6.5 : Performance de l'algorithme de recherche du contact

Ce tableau nous permet d'observer et de confirmer le fait que la zone de contact s'élargit lors de la déformation du bloc. Les résultats relatifs aux déplacements des corps en contact se présentent comme suit :

Pour le bloc rectangulaire :

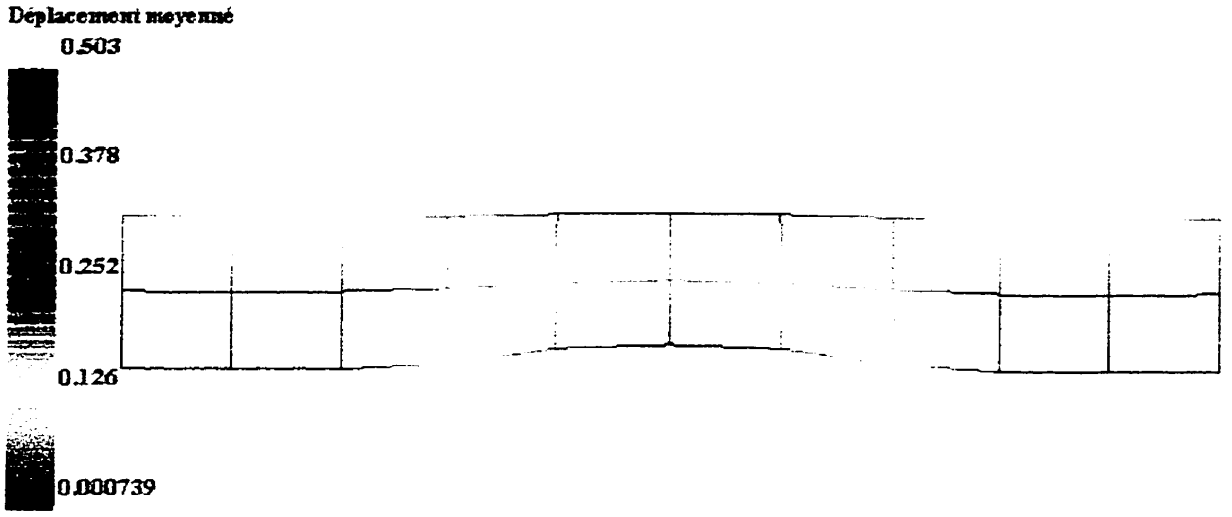


Figure 6.13 Déplacements moyennés sur le bloc rectangulaire

Pour le corps cylindrique :

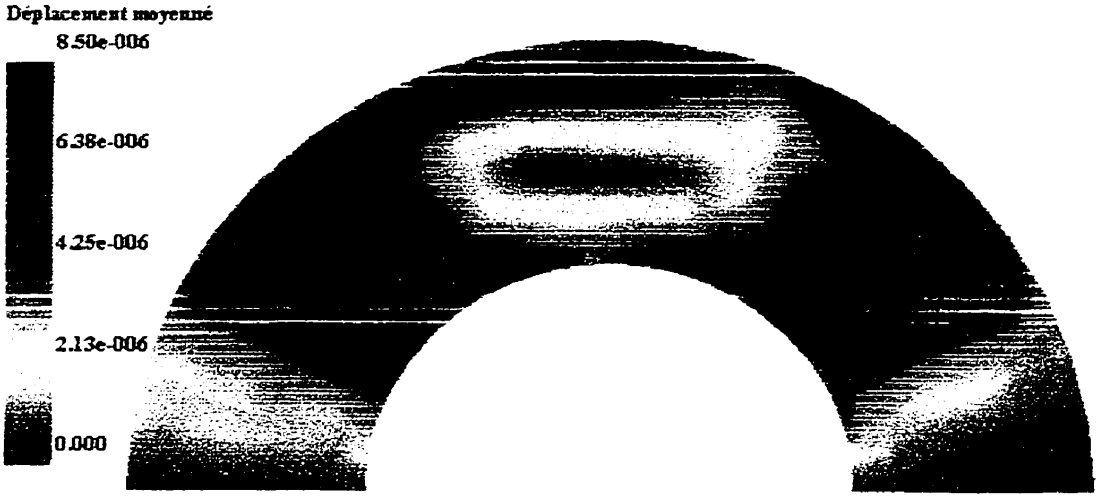


Figure 6.14 Déplacements moyennés sur le corps cylindrique

On observe une représentation correcte du déplacement dans la direction du chargement sur les deux corps et de façon symétrique. Le corps rectangulaire tend a épousé la forme du corps cylindrique. Le corps cylindrique étant plus rigide, il connaît un déplacement qui tend vers zéro à 10^{e-6} près. Les résultats relatifs aux contraintes de Von Mises sur le corps rectangulaire se présentent comme suit :

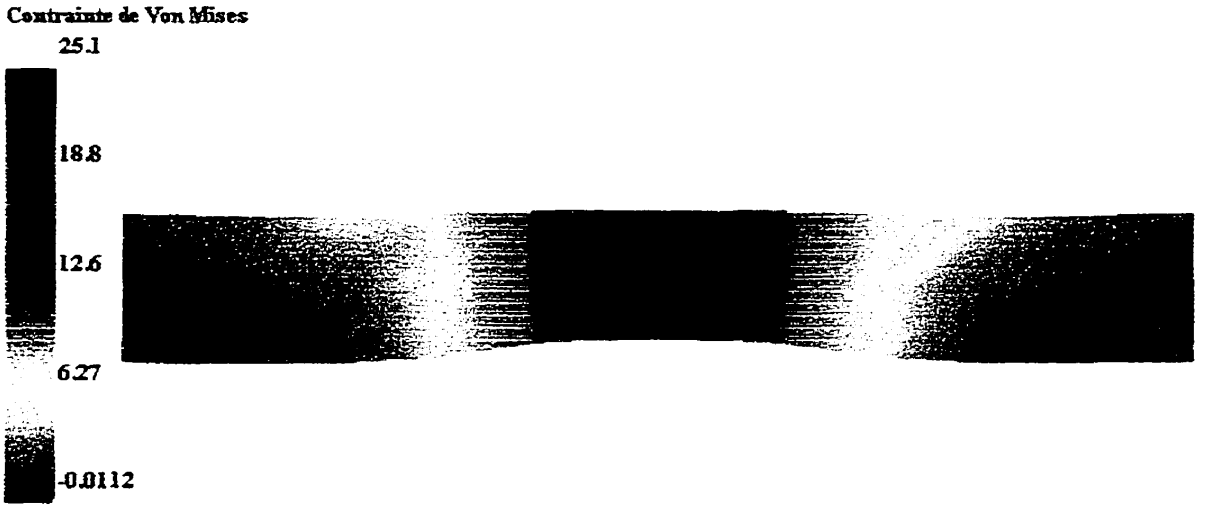


Figure 6.15 Contraintes de Von Mises

La partie en contact connaît une contrainte plus élevée du fait du contact et la distribution de la contrainte de déformation de Von Mises obtenue est semblable à celle obtenue par Taylor et al. dans [Réf-12]. L'épaisseur du corps rectangulaire grandit de l'axe de symétrie vers le bord.

Dans ce chapitre on a présenté les différentes classes qui ont été nécessaires pour l'implémentation de l'hyperélasticité et du contact dans l'environnement Diffpack, ainsi que les relations qui existent entre elles. Une comparaison du programme d'hyperélasticité dans Diffpack par rapport à Flagshyp nous a permis de vérifier la fiabilité de la loi de comportement implantée. Le simulateur de contact a été mis à l'épreuve à travers trois exemples permettant de vérifier les lois de contact à travers la théorie d'Euler, la convergence numérique ainsi que la performance des algorithmes de recherche de contact.
Le prochain chapitre sera consacré au bilan des différents travaux réalisés ainsi qu'à la discussion des perspectives ouvertes par ce travail.

CHAPITRE VII

CONCLUSION GÉNÉRALE ET PERSPECTIVES

7.1 Conclusion

L'objectif principal de ce travail a été de mettre au point une méthodologie pratique pour la simulation et la modélisation par la méthode des éléments finis du contact avec frottement dans les procédés de mise en forme de structures hyperélastiques . L'aspect modélisation est relatif au développement, grâce aux outils offerts par la programmation orienté objet, et à l'évaluation du modèle de comportement hyperélastique ainsi qu'au développement et à l'évaluation des algorithmes de contact 2D et 3D avec la technique de résolution utilisant la méthode des pénalités. L'aspect simulation est relatif aux applications des outils de modélisation pour l'étude du comportement des structures impliquant des non linéarités diverses (grands déplacements, grandes déformations) et pour la simulation de la mise en forme de solides hyperélastiques.

Ce document a abordé divers aspects pour la modélisation des corps solides. Nous pouvons citer la formulation et le développement de la loi de comportement hyperélastique en formulation lagrangienne et spatiale. Divers types de matériaux hyperélastiques sont traités et implémentés dans l'environnement Diffpack. On distingue les matériaux compressible néo-Hookéen et incompressible néo-Hookéen.

Ce travail a permis de développer un simulateur du contact avec des outils nouveaux offerts par le logiciel commercial Diffpack qui est implanté dans le Visual C++. C'est ainsi que le développement du code est rendu allégé par l'existence d'abstractions diverses et puissantes facilitant la programmation et la correction de programme. Les concepts de relations entre les classes sont abordées dans cette étude du contact. Ces relations ont permis de distinguer à partir de l'algorithme de recherche basé sur la hiérarchie du contact quatre classes, à savoir :

- une classe pour définir les domaines en contact
- une classe pour définir les surfaces en contact
- une classe pour définir les paires de surface en contact
- et enfin une classe pour définir l'élément de contact.

Ces classes sont incluses et utilisées dans la classe de base qui administre et résoud le problème de contact.

Plusieurs algorithmes d'intégration numérique ont été présentés et programmés. La résolution des équations d'équilibre avec un schéma de Newton-Raphson, couplé aux inéquations de contact avec frottement (contact collant ou glissant) dans le cadre des lois de comportement hyperélastique permet d'obtenir des solutions satisfaisantes selon les coefficients de pénalité choisis. On a implanté dans le simulateur les lois de comportement hyperélastique compressible, applicable dans le code en 3D et en déformation plane, incompressible, applicable en 3D, en déformation plane et en contrainte plane,

Au niveau du traitement du contact, nous proposons des aspects intéressants dans la formulation d'algorithmes de recherche de contact basés sur une localisation rapide des nœuds de contact, sur lesquels nous construisons des boîtes, qui permettent d'économiser du temps en CPU. Ces algorithmes constituent une pièce maîtresse dans un code de simulation, et permettent une localisation précise de tous les nœuds de contact à chaque instant, ainsi que la description de leur état de contact.

Cette étude du contact ouvre des portes pour un développement plus intelligent et moins coûteux en temps de calcul.

7.2 Perspectives

Les perspectives du présent travail sont intéressantes et concernent donc différents aspects. On distingue :

- Un développement orienté objet de la loi de comportement afin de faciliter l'implantation d'autres lois de comportement

- Réaliser une comparaison numérique/expérimental lorsque les presses utilisées en entreprise seront disponibles afin de corriger le modèle de comportement choisi.

- Implantation d'un interface de visualisation adapté au problème de contact sous Diffpack

- Un développement orienté objet de nouvelles méthodes de résolution du contact, permettant ainsi de pouvoir implémenter autre chose que la méthode des pénalités d'autres méthodes comme la méthode du Lagrangien augmenté afin d'éviter la perte de temps reliée à la recherche des bons coefficients de pénalité.

- Rendre plus flexible le calcul des tolérances de contact en appliquant un algorithme itératif au calcul de la longueur moyenne d'une cellule et de la tolérance du territoire de contact.

- Extension de cette étude à la résolution de problèmes axisymétriques.

- Extension du mode de pilote par pression en développant celui du pilotagepar déplacement.

RÉFÉRENCES BIBLIOGRAPHIQUES

[1] J.L. Batoz anf D. Gouri. Incremental displacement algorithms for non-linear problems. *International Journal for Numerical Methods in Engineering*, 14 :1262-1266, 1979.

[2] M.A. Crisfield. Non-linear finite element analysis of solids and structures, volume1. *Wifey,* England, 1991.

[3] P. Pol. Simulation du comportement elasto-plastique de coques minces par éléments finis. PhD thesis, *Université de Technologie de Compiègne*, France, 1992.

[4] M. Fafard. Calcul automatique des configurations pré et post-flambement en calcul non linéaire des structures. PhD thesis, *Univesité Laval*, Canada, 1987.

[5] G. Duvaut and J.L. Lions. Inequalities in mechanics and physics. *Springer* Verlag, Berlin, 1976.

[6] Z.Q. Feng. Contribution à la modélisation des problèmes non linéaires : contact, plasticité et endommagement. PhD thesis, *Université de Technologie*, France, 1991.

[7] P. Wriggers and B. Nour-Omid. Solution methods for contact problems. *Technical Report* UCB/SESM 84-09, University of California, CA. 1984.

[8] A. Rassineux. Maillage automatique tridimensionnel par une technique frontale pour la méthode des éléments finis. Ph D thesis, *Université de Nancy*, France, 1995.

[9] M. Ziane. Simulation numérique de la mise en forme des corps creux plastiques soufflage-thermoformage. Ph D thesis, *Université de Technologie de Compiègne*, France, 1999.

[10] J. Bonet and R. D. Wood. Nonlinear continuum mechanics for finite element analysis. *Cambridge University Press* 1997.

[11] H. P. Langtangen. Computational Partial Differential Equations : Numerical Methods and Diffpack Programming. *Sprnger* 1999.

[12] J.-W. JU and R. L. Taylor. A pertubed Lagrangian formulation for the finite element solution of nonlinear frictional problems. *Journal of theoretical and apllied mechanics* 0750-7240/1988

[13] H. Parisch and Ch. Lubbing. A formulation of arbitrarily shaped surface elements for three-dimensional large deformation contact with friction. *International Journal for Numerical Methods in Engineering*, vol. 40, 3359-3383, 1997.

[14] T. A. Laursen and J. C. Simo. A continuum-based finite element formulation for the implicit solution of multibody, large deformation frictinal contact problems friction. *International Journal for Numerical Methods in Engineering*, vol. 36, 3451-3485, 1993.

[15] S. P. Wang and E. Nakamachi. The inside-outside contact search algorithm for the finite element analysis friction. *International Journal for Numerical Methods in Engineering*, vol. 40, 3665-3685, 1997.

[16] C. Agelet de Saracibar. A new frictional time integration algorithm for large slip multi-body frictional contact problems friction. *Computer Methods Applied Mechanics and Engineering*, 142, 303-334, 1997.

[17] Z.-H. Zhong, L. Nilsson. A unified contact algorithm based on the territory concept. *Computer Methods Applied Mechanics and Engineering*, 130, 1-16, 1996.

[18] J. C. Simo, R. L. Taylor and Wriggers. A note on finite element implementation of pressure boundary loading. *Communications in applied numerical methods*, vol. 7, 513-525, 1991

[19] T. A. Laursen and V. G. Oancea. Automation and assessment of augmented Lagrangian algorithms for frictional contact problems. *Journal of Applied Mechanics*, vol. 61/1994

[20] T. A. Laursen and B. N. Maker. An augmented lagrangian quasi-newton solver for constrained nonlinear finite element applications. *International Journal for Numerical Methods in Engineering*, vol. 38, 3571-3590, 1995.

[21] M. Oldenburg and L. Nilsson. The position code algorithm for contact searching. *International Journal for Numerical Methods in Engineering*, vol. 37, 359-386, 1994.

[22] Z. H. Zhong. Finite element procedures for contact-impact problems. *Oxford Science Publications*, 1993

[23] Z. H. Zhong and L. Nilsson. A contact searching algorithm for general contact problems. *Computers and structures*, 33(1) :197-209, 1989

[24] Hertz, H. Über die Beruhrung fester elastischer Korper. *Journal für die Reine und Angewandte Mathematik*, **92**. 156-71, 1881

[25] Hertz, H. Über die Beruhrung fester elastischer Korper und Uber die Harte. *Verhandlungen des Vereins zur Beforderung des Gewerbfleifses*, Berlin., 1882

[26] Johnson, K. L. Contact mechanics. *Cambridge University Press,* Cambridge. 1985

[27] Gladwell. G. M. L. Contact problems in the classical theory of elasticity. Stijthoff & Noordhoff, Alphen ann den Rijn. 1980

[28] Goldsmith, W. Impact - the theory and physical behavior of colliding solids. Edward Arnold, London. 1960

[29] Zienkiewicz, O. C. The finite element method in structural and continuum mechanics. *European Civil Engineering Series*, 99-0388-272, London. 1967

[30] Zienkiewicz, O. C. The finite element method in engineering science. *McGraw-Hill*, London. 1971

[31] Zienkiewicz, O. C. The finite element method. *McGraw-Hill*, London. 1977

[32] Oden, J. T. Finite elements of nonlinear continua. *McGraw-Hill*, New-York. 1972

[33] Bathe, K. J. Finite element procedures in engineering analysis. *Prentice-Hall*, Englewood Cliffs. 1982

[34] Hughes, T. J. R. The finite element method – Linear static and dynamic finite element analysis. *Prentice-Hall*, Englewood Cliffs. 1987

[35] Zienkiewicz, O. C. and Taylor, R. L. The finite element method. 4th edn. *McGraw-Hill*, New-York. 1989

[36] Andersson, T. The boundary element method with application to contact mechanics, Report R-135, Linköping Institute of Technology. 1979

[37] Batra, R. C. Quasistatic indentation of a rubber-covered roll by a rigid roll. *International Journal for Numerical Methods in Engineering*, vol. 17, 1823-33, 1981.

[38] P. Germain. Cours de mécanique des milieux continus. Masson et Cie, 1986.

[39] F. Sidoroff. Cours sur les grandes déformations. Rapport GRECO 51, école d'été à Sophia-Antipolis, 8-10 Dept. 1982.

[40] J. L. Bathoz et G. Dhatt. Modélisation des structures par éléments finis, volume 1. Hermes, 1990.

[41] P. Wriggers, P. Panagiotopoulos. New developments in contact problems. CISN Courses and lectures no. 384, *International Centre for Mechanical Sciences*, SpringerWienNewYork, 1999

[42] P. Wriggers, T. Vu Van and E. Stein. Finite element formulation of large deformation impact-contact problems with feiction. *Computers & Structures* vol. 37, No 3, pp 319-331, 1990

[43] Getting Started with ABAQUS/Standard. *Hibbitt, Karlsson & Sorensen, Inc.*

[44] A. Gakwaya. Conception mécanique assistée par ordinateur(CMAO) Modélisation et analyse en CAO-GMC-17694/63855. *Université Laval.* Automne 1998

[45] M. Toukourou Moubarack, A. Gakwaya, A. A. Yazdani. An object oriented finite element implementation of large deformation frictional contact problems and applications. *Article soumis à "First M.I.T. Conference on Computational Fluid and Solid Mechanics"*, Massachusetts Institute of Technology, Cambridge, June 12-14 2001

ANNEXES

ANNEXE A - Algorithme procédural de Flagshyp en anglais

Structure de Flagshyp :

Flagshyp .....master routine

|__welcome.....types welcome message and reads files names

|__elinfo.....reads element type

| |__quad4db.....evaluates the 4-noded quadrilateral data

| |__hexa8db.....evaluates the 8-noded hexahedron data

|__innodes.....reads nodal coordinates and boundary codes

|__inelems.....reads element connectivities and material types

|__nodecon.....evaluates node to node connectivities

|__degfrm.....numbers degrees of freedom with profile minimization

|__profile.....determines the profile column heights and addressing

|__matprop.....reads material properties

|__inloads.....reads loads and prescribed displacements

|__incontr.....reads solution control parameters

|__initno.....initializes nodal data such as coordinates

| $\mathbf{x} = \mathbf{x}0 \qquad \mathbf{K} = \mathbf{0} \qquad \mathbf{R} = \mathbf{0}$

|__initel.....finds the initial tangent matrix

| |__gradsh.....finds the Cartesian derivatives of the shape functions

| $\frac{\partial \mathbf{N}}{\partial \mathbf{x}}, \ \det \mathbf{J}\mathbf{x}\mathbf{W}$

| |__kvolume.....finds the mean dilatation component of the stiffness matrix

| $\mathbf{K}_{\kappa} = \kappa \cdot \overline{\nabla \mathbf{N}_a} \otimes \overline{\nabla \mathbf{N}_b} / \mathbf{V}(\mathbf{e})$

| |__ update $\mathbf{F}_{ext}$ by adding the gravity

ANNEXE B - Descriptions des principales fonctions de la classe **HyperElasticity** en **anglais**

adm : administers the menu by calling :

- SimCase::**attach** (menu);
- **define** (menu); : definition of all the items and levels on the menu
- menu.**prompt**();
- **scan** (); : Initialization of all member variables related to the input file

solveProblem : Resolution of the problem

- **setIC** (); : sets the displacement to zero

  u()(k).values()(i) = 0.0; // deplacement value at the node i in direction k

  u_prev ()(k).values()(i) = 0.0; // prev. deplacement value at the node i in direction k

- **timeLoop**(); : makes the loop over the load steps
  - Initialization of the timeloop
  - Initialization of the variables xlamb (current load step), xlmax(maximum load step), dlamb(incremental load step)
  - **eldbase** : initialize the database relative to the element used (only two elements are implemented: ElmB4n2D and ElmB8n3D
    - **gradsh** : in case of incompressible material heps find vol0 for each element
  - For each load step until xlamb = xlmax
    - **solveAtThisTimeStep** ()
      - **fillEssBC** (); : set essential boundary condition
        - if (grid->boNode (noind., boind.)) dof->fillEssBC (noind.,boind., 0.0);
      - Initialize nonlin_solution at zero
      - If there are new BC at this time step, update the present guess
      - Call nonlinear solver with :

        bool converged = **nlsolver->solve** (); // which calls **makeAndSolveLinearSystem** at each iteration of the NEWTON_RAPHSON solver
      - After the NEWTON_RAPHSON method has solved the problem, load the nonlinear solution found by the solver into the u field:
      - If convergence was not attained print a file and stop the program

- If not print a file for convergence over the step and continue.

- **calcDerivedQuantities** ()
  - calculate the Von Mises stress measures at integration points

- **saveResults** ()
  - dump all fields to files for plotting and post-treatment
  - Update deformed_grid
  - Update deformed_grid_prev

- **UpdateDataStructures**
  - Update grid as deformed_grid
  - **setIC**(); : sets the displacement to zero

    u = 0

    u_prev = 0

makeAndSolveLinearSystem : This function is called during the Newton_Raphson iteration

- Copy most recent guess in nonlin_solution to u;
- For a NEWTON_RAPHSON method call **dof->fillEssBC2zero**() if not call dof->unfillEssBC2zero();
- **makeSystem (*dof, *lineq);** : the function itself is not seen in the .cpp file. It is developped in FEM and calls **calcElmMatVec**
- Initialize start vector (linear_solution) to zero for iterative linear solver:
- **lineq->solve**(); : invoke a linear system solver, the solution of the linear system is now available in the vector linear_solution.
- Update of the currentgrid
- Print all state variables of the current iteration in a debug file

calcElmMatVec : sets the integration rules for the volume integral calculated in integrands and the element surface integral calculated in integrands4side .

For incompressible material calculates:

- evol : the current volume
- elacd : the average cartesian derivatives "[Ref-10] - Eq. 7.59"
- KK : the volumetric component of the tangent matrix "[Ref-10] - Eq. 7.59"

integrands : This function is called for each element at each Gauss Point and calculates the contribution to the tangent matrix and the residue

- Initialization of local variables
- Initialization of the radius r used in axisymmetry problems
- **calcgradN_x**
- **calcgradDF**
- **calcDmat**
- **calcmat2vec (Sigvec, Sigma, elasticity_tp);**
- **calcBmats**
- **calcBgeom**
- **calcSigmap**
- calculate matBtDB and matBgtKsBg
- calculate elmat.A the tangent matrix
- calculate elmat.b the residual vector

integrands4side: Calculates the integrands over the side of the element under pressure.

- Initialization of local variables
- Initialization of the pressure variable depending on a pressure in x direction, in y or z direction
- Initialization of the pressure at this step : apres = pressure*xlamb;
- Print in a debug file of the variable used and calculated in this function for each integration point over the side.
- For a PLANE_STRESS and a PLANE_STRAIN problem : Following the article by SIMO the results are **conclusive**. [Ref-18]
- For an AXISYMMETRY problem : Following the article by SIMO (**not tested yet**) [Ref-18].
- For a THREE_DIM problem : Formulation in [Ref-10] pp 179, SIMO the results are **conclusive**.

calcgradN_x - Evaluates the spatial derivatives of the basis functions N

- Update the configuration by using the solution at each iteration in linear_solution

$$\frac{\partial N}{\partial x} = \frac{\partial N}{\partial \xi} \cdot \frac{\partial \xi}{\partial x} = \frac{\partial N}{\partial \xi} \cdot \left(\frac{\partial x}{\partial \xi} \right)^{-1} = \frac{\partial N}{\partial \xi} \cdot \left(\sum_{a=1}^{nnode} x_{a,i} \frac{\partial N_a}{\partial \xi_j} \right)^{-1}$$

calcgradDF - Evaluates the current deformation gradient

in 3D and PLANE_STRESS	in PLANE_STRAIN	in AXISYMMETRY:
$F = \frac{\partial x}{\partial X} = \left(\frac{\partial X}{\partial x}\right)^{-1} = \left(\sum_{a=1}^{nnode} X_a \right.$ where X is the matrix coordinates of the nodes in the element at the original configuration and x is the matrix coordinates of the nodes in the element at the current configuration	$F = \begin{bmatrix} \frac{\partial x_1}{\partial X_1} & \frac{\partial x_1}{\partial X_2} & 0 \\ \frac{\partial x_2}{\partial X_1} & \frac{\partial x_2}{\partial X_2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$F = \begin{bmatrix} \frac{\partial x_1}{\partial X_1} & \frac{\partial x_1}{\partial X_2} & 0 \\ \frac{\partial x_2}{\partial X_1} & \frac{\partial x_2}{\partial X_2} & 0 \\ 0 & 0 & \frac{r}{R} \end{bmatrix}$ where r is the current radius at the present Gauss point and R is the radius of the present Gauss point in the original configuration

CalcDmat: - Evaluates the behavioral matrix D

- Initialization of the local variables
- nuE2Lame : converts nu and E to Lame coefficients
- calcId : calculate the identity vector and matrix for the present problem type
- Calculate the determinant of F
- Calculate $b = F \cdot F^T$
- For the present problem type calculate the Cauchy stress and the behavioral matrix D.

Behavioral formulation :

In 3D or PLANE_STRAIN

compressible neo-Hookean in [Ref-10] pp 201:

$$\sigma_{ij} = \frac{\mu}{J}\left(b_{ij} - \delta_{ij}\right) + \frac{\lambda}{J}(\ln J)\delta_{ij}$$

$$\varsigma_{ijkl} = \lambda'\delta_{ij}\delta_{kl} + 2\mu'\delta_{ik}\delta_{jl}$$

$$\lambda' = \frac{\lambda}{J}; \qquad \mu' = \frac{\mu - \lambda \ln J}{J}$$

in3D,

$$\delta_{ij}\delta_{kl} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = 1 \otimes 1 \qquad \delta_{ik}\delta_{jl} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 \end{bmatrix} = \mathbf{I}_4$$

in PLANE_STRAIN,

$$\delta_{ij}\delta_{kl} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = 1 \otimes 1 \qquad \delta_{ik}\delta_{jl} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix} = \mathbf{I}_4$$

<u>incompressible neo_Hookean in [Ref-10] pp 203 :</u>

$$\bar{\mathbf{J}} = \frac{\mathbf{v}^{(e)}}{\mathbf{V}^{(e)}}, \bar{\kappa} = \kappa \frac{\mathbf{v}^{(e)}}{\mathbf{V}^{(e)}}, \mathbf{p} = \kappa\left(\bar{\mathbf{J}} - 1\right)$$

$$\sigma'_{ij} = \mu \mathbf{J}^{-5/3}\left(\mathbf{b}_{ij} - \frac{1}{3}\mathbf{I}_b \delta_{ij}\right)$$

$$\sigma_{ij} = \sigma'_{ij} + \mathbf{p}\delta_{ij}$$

$$\hat{\varsigma}_{ijkl} = 2\mu \mathbf{J}^{-5/3}\left[\frac{1}{3}\mathbf{I}_b \delta_{ik}\delta_{jl} - \frac{1}{3}\mathbf{b}_{ij}\delta_{kl} - \frac{1}{3}\delta_{ij}\mathbf{b}_{kl} + \frac{1}{9}\mathbf{I}_b \delta_{ij}\delta_{kl}\right]$$

$$\varsigma_{p,ijkl} = \mathbf{p}\left(\delta_{ij}\delta_{kl} - 2\delta_{ik}\delta_{jl}\right)$$

In PLANE_STRESS

<u>incompressible neo_Hookean in [Ref-10] pp 203 :</u>

J = 1; $(J = dv/dV, \quad j = da/dA)$

$$\sigma_{ij} = \mu b_{ij} - j^2 \delta_{ij} \qquad \left(j^2 = \det_{2x2} \mathrm{b} \right)$$

$$\varsigma_{ijkl} = \lambda' \delta_{ij} \delta_{kl} + 2\mu' \delta_{ik} \delta_{jl}$$

$\lambda' = \frac{2\mu}{j^2} \quad \mu' = \frac{\mu}{j^2} \quad h = \frac{H}{j}$, in my program H is supposed to be 1.

with,

$$\delta_{ij}\delta_{kl} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = 1 \otimes 1 \qquad \delta_{ik}\delta_{jl} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix} = \mathbf{I}_4$$

AXISYMMETRY (is not implemented)

calcmat2vec : converts the Cauchy stress in a matrix format to a vector format.

CalcBmats: calculated at each Gauss point

in 3D	in PLANE_STRESS	in PLANE_STRAIN	in AXISYMMETRY
$B_{mat} = \begin{bmatrix} \frac{\partial N}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N}{\partial z} \\ \frac{\partial N}{\partial y} & \frac{\partial N}{\partial x} & 0 \\ 0 & \frac{\partial N}{\partial z} & \frac{\partial N}{\partial y} \\ \frac{\partial N}{\partial z} & 0 & \frac{\partial N}{\partial x} \end{bmatrix}$	$B_{mat} = \begin{bmatrix} \frac{\partial N}{\partial x} & 0 \\ 0 & \frac{\partial N}{\partial y} \\ \frac{\partial N}{\partial y} & \frac{\partial N}{\partial x} \end{bmatrix}$	$B_{mat} = \begin{bmatrix} \frac{\partial N}{\partial x} & 0 \\ 0 & \frac{\partial N}{\partial y} \\ 0 & 0 \\ \frac{\partial N}{\partial y} & \frac{\partial N}{\partial x} \end{bmatrix}$	$B_{mat} = \begin{bmatrix} \frac{\partial N}{\partial x} & 0 \\ 0 & \frac{\partial N}{\partial y} \\ \frac{N}{r} & 0 \\ \frac{\partial N}{\partial y} & \frac{\partial N}{\partial x} \end{bmatrix}$

calcBgeom: calculated at each Gauss point

in 3D	in PLANE_STRESS	in PLANE_STRAIN	in AXISYMMETRY
$B_{geom} = \begin{bmatrix} \frac{\partial N}{\partial x} & 0 & 0 \\ \frac{\partial N}{\partial y} & 0 & 0 \\ \frac{\partial N}{\partial z} & 0 & 0 \\ 0 & \frac{\partial N}{\partial x} & 0 \\ 0 & \frac{\partial N}{\partial y} & 0 \\ 0 & \frac{\partial N}{\partial z} & 0 \\ 0 & 0 & \frac{\partial N}{\partial x} \\ 0 & 0 & \frac{\partial N}{\partial y} \\ 0 & 0 & \frac{\partial N}{\partial z} \end{bmatrix}$	$B_{geom} = \begin{bmatrix} \frac{\partial N}{\partial x} & 0 \\ \frac{\partial N}{\partial y} & 0 \\ 0 & \frac{\partial N}{\partial x} \\ 0 & \frac{\partial N}{\partial y} \end{bmatrix}$	$B_{geom} = \begin{bmatrix} \frac{\partial N}{\partial x} & 0 \\ \frac{\partial N}{\partial y} & 0 \\ 0 & 0 \\ 0 & \frac{\partial N}{\partial x} \\ 0 & \frac{\partial N}{\partial y} \end{bmatrix}$	$B_{geom} = \begin{bmatrix} \frac{\partial N}{\partial x} & 0 \\ \frac{\partial N}{\partial y} & 0 \\ \frac{N}{r} & 0 \\ 0 & \frac{\partial N}{\partial x} \\ 0 & \frac{\partial N}{\partial y} \end{bmatrix}$

CalcSigmap : Converts the Cauchy stress matrix to a new matrix format used to calculate BgtKsBg

ANNEXE C - Pré et post-traitement avec le simulateur du contact sous Diffpack

Avant de lancer une simulation il faut préparer un fichier d'entrée d'Extension **i** (Ex : Nom_de_fichier.**i**). Les chaînes de caractères à la droite du point d'exclamation (!) désignent toujours un commentaire. L'exemple 2 présenté au chapitre 6 a pour fichier d'entrée le fichier suivant :

```
set time integration parameters = dt=1 t in [0,1]
set elasticity type = THREE_DIM
set hyperelasticity type = INC_NH

! contact datas
set no_contact_obj = 2
set objgrid = P=PreproStdGeom | BOX d=3 [-15,15]x[-15,15]X[20,25]  |
d=3  e=ElmB8n3D  [7,7,2] [1,1,1] , P=PreproStdGeom | BOX d=3 [-
40,40]x[-40,40]x[10,20]  |  d=3  e=ElmB8n3D  [8,8,2] [1,1,1]
  set objname = punch , elastic_foundation
  set objtype = RIGID , NOT RIGID
  set redefine boundary indicators = n=7 names= P1 P2 P3 u1=0 u2=0 u3=0
free 1=() 2=() 3=(3) 4=() 5=() 6=() 7=(1 2 4 5 6) , n=7 names= P1 P2 P3
u1=0 u2=0 u3=0 free 1=() 2=() 3=() 4=(6) 5=(6) 6=(6) 7=(1 2 3 4 5)
  set nu value = 0.00 0.30
  set E value = 100000000 1000
  set pressure 1 = 0.0 0.0
  set pressure 3 = 60000.0 0.0
  set deformed grid magnification = 1 1

set debug = 0
set global serach per iteration = 0

set no_elset = 2
 set ELSET NAME = punch_B , elastic_foundation_T
 set elmlist = 1 49 1 65 128 1

set no_surface_definition = 2
 set SURFACE DEFINITION NAME = punch_B , elastic_foundation_T
  set ELSET number = 1 2
  set direction = 6 3
  set surface_obj = 1 2

set no_contact_pair = 1
  set CONTACT PAIR SURF= punch_B , elastic_foundation_T
    set CONTACT PAIR INTERACTION = FRICTION

set FRICTION = 0.6
set contact zone tolerance = 0.0000000001
set epsilon_N = 20000000000000000000
set epsilon_T = 100000000000000
set territory tolerance = 20
set width of one position cell = 50
set no_ref_node = 0
```

```
sub NonLinEqSolver_prm
 set nonlinear iteration method = NewtonRaphson
 set max nonlinear iterations = 200
 set max estimated nonlinear error = 1.0e-5 ! eps in termination crit.
 set nonlinear relaxation prm = { 1.0 }
ok
sub LinEqAdmFE
 sub Matrix_prm
  set matrix type = Mat
 ok
 sub LinEqSolver_prm
  set basic method = ConjGrad
 ok
 sub Precond_prm
  !set preconditioning type = { PrecRILU & PrecSSOR & PrecNone } ! 14,
44, 128 it
  set preconditioning type = PrecRILU
  set (S)SOR relaxation parameter = 1
  set RILU relaxation parameter = 0.0  ! 22 (MILU, 1.0) vs 14 (ILU,
0.0)
 ok
ok

sub LinEqSolver_prm
set basic method = ConjGrad
ok

sub Precond_prm
set preconditioning type = {PrecNone & PrecJacobi}
ok
ok
```

Les items sont précédés par le mot **set**, tandis que les sous menus sont précédés par le mot **sub**. Il est important de noter que compte tenu de la largeur du présent document le format du fichier d'entrée n'est pas tout à fait respecté parce que toute chaîne de caractères à la droite d'un set ou d'un sub doit se trouver sur la même ligne.

Comme il s'agit ici d'un problème concernant plus d'un corps modélisé par des éléments finis, les données relatives à chacun des corps sont toujours sur la même ligne et sont de façon générale séparées par la chaîne de caractères suivante (,) , soit **un espace** suivi d'**une virgule** suivi d'**un espace** lorsqu'il s'agit de données sous formes non numériques comme les items : objgrid, objname, objtype, redefine boundary indicators, ELSET NAME, SURFACE DEFINITION NAME, CONTACT PAIR SURF. Lorsqu'ils s'agit de données numériques un **espace** suffit comme pour les items : nu value, E value, pressure 1, pressure 3, elmlist, ELSET number, direction, surface_obj.

Le fichier d'entrée est assez simple et permet d'initialiser convenablement le problème à résoudre.

Pour résoudre un problème relatif à un seul corps soumis à des conditions limites données il suffit de suivre la même procédure. C'est ainsi que un fichier d'entrée permettant de résoudre le problème de l'hyperélasticité se présente comme suit :

```
set time integration parameters = dt=0.2 t in [0,1]
set elasticity type = PLANE_STRAIN
set hyperelasticity type = INC_NH

! contact datas limited
set no_contact_obj = 1
set objgrid = P=PreproStdGeom | BOX d=2 [-1.5,1.5]x[1.0,2.0]   |  d=2
e=ElmB4n2D  [6,4] [1,1]
  set objname = punch
  set objtype = NOT RIGID
  set redefine boundary indicators = n=5 names= P1 P2 u1=0 u2=0 free
1=() 2=(2) 3=(4) 4=(4) 5=(1 2 3)
  set nu value = 0.30
  set E value = 10000
  set pressure 1 = 0.0
  set pressure 2 = 1000.0

set no_elset = 0

set no_surface_definition = 0

set no_contact_pair = 0

set no_ref_node = 0

sub NonLinEqSolver_prm
 set nonlinear iteration method = NewtonRaphson
 set max nonlinear iterations = 200
 set max estimated nonlinear error = 1.0e-5 ! eps in termination crit.
 set nonlinear relaxation prm = { 1.0 }
ok
sub LinEqAdmFE
 sub Matrix_prm
  set matrix type = Mat
 ok
 sub LinEqSolver_prm
  set basic method = ConjGrad
 ok
 sub Precond_prm
  !set preconditioning type = { PrecRILU & PrecSSOR & PrecNone } ! 14,
44, 128 it
  set preconditioning type = PrecRILU
  set (S)SOR relaxation parameter = 1.6
```

```
  set RILU relaxation parameter = 0.0   ! 22 (MILU, 1.0) vs 14 (ILU,
0.0)
 ok
ok

sub LinEqSolver_prm
set basic method = ConjGrad
ok

sub Precond_prm
set preconditioning type = {PrecNone & PrecJacobi}
ok
ok
```

Le fichier est donc moins long et les items relatifs au contact sont mis à 0 afin de désactivé le module du contact. L'environnement graphique sous Diffpack se présent comme suit :

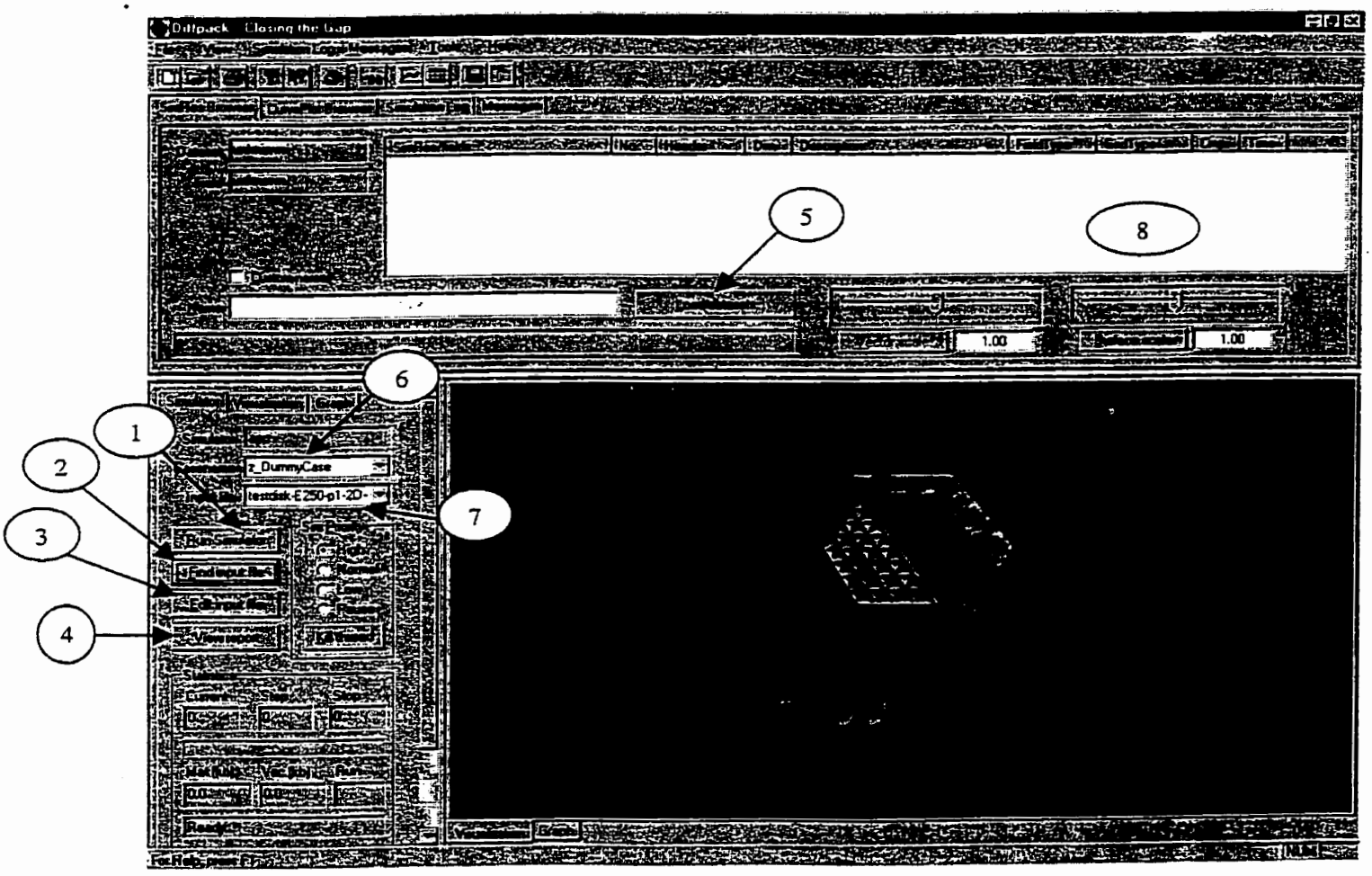


Figure C.1 Environnement de Diffpack

Afin de simuler un problème de contact il faut :

Pour le **pré-traitement** :

1- Préparer son fichier d'entrée comme décrit plus haut

2- Lancer le simulateur sous Visual C++

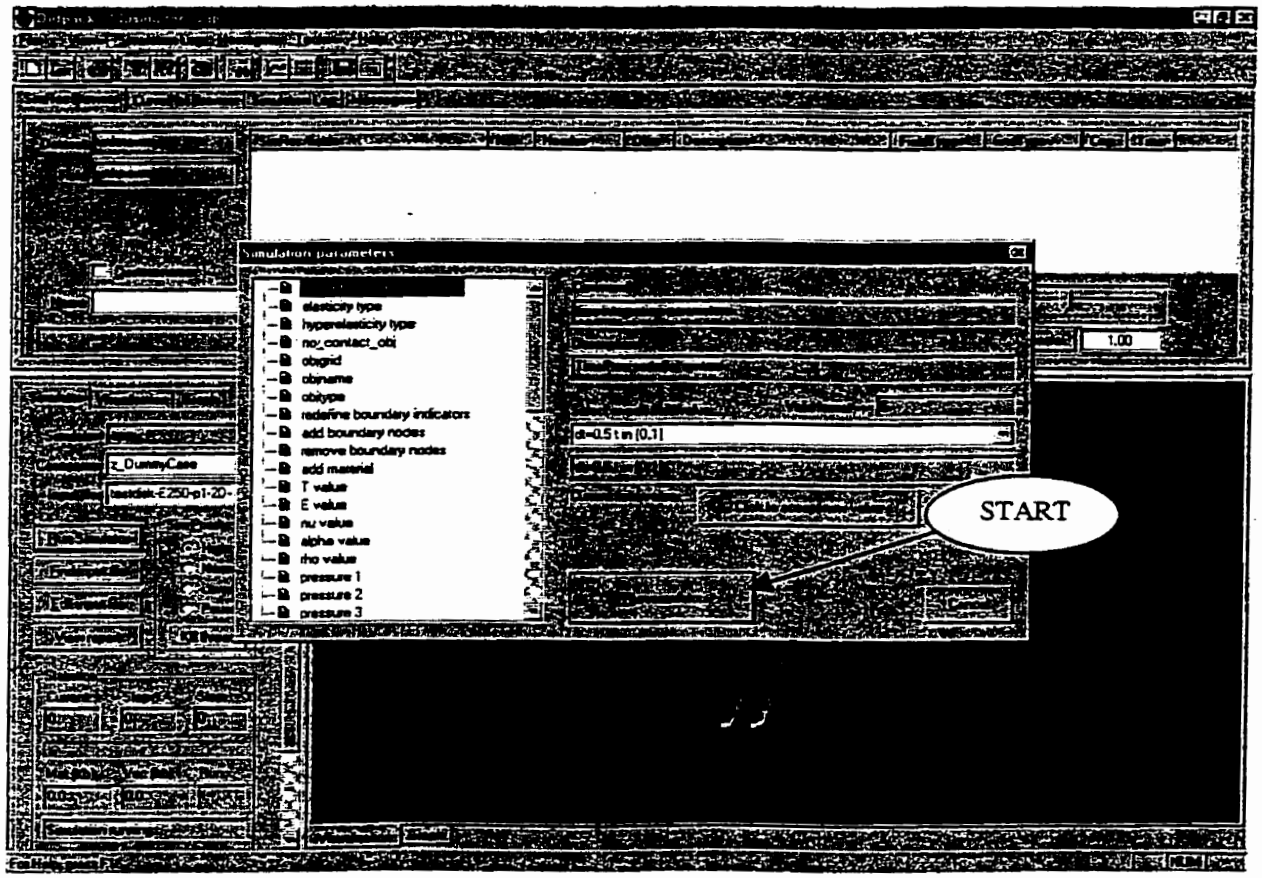
3- Appuyer sur la touche **2** pour choisir son fichier d'entrée. Son nom apparaît dans la case **7**

4- Donner un nom au fichier résultat (Ex : result_name)dans la case 6

5- On peut visualiser son fichier d'entrée avec la touche **3** avant de lancer la simulation.

6- On lance la simulation avec la touche **1**

7- Avant de résoudre le problème posé, une fenêtre s'ouvre permettant à l'utilisateur de vérifier à nouveau ses paramètres de simulation.



8- Une fois la touche START est appuyée la résolution du problème est en cours.

Pour le **post-traitement** :

9- Lancer le simulateur sous Visual C++

10- Appuyer sur la touche 5 pour choisir votre chier de post-traitement d'extension **field**.

11- Sélectionner le résultat que l'on veut visionner dans la fenêtre 8 qui présente la champ de données désiré par pas de chargement.

12- On peut également utiliser les différents fichiers géner par Diffpack pour analyser sa simulation. Parmi ces fichiers on distingue :

- le fichier .result_name_m01.**simres** qui présente les types de données disponibles pour le post-traitement
- le fichier .result_name_m01.**nonlinconv** qui présente la performance de la méthode numérique choisie pour la résolution du problème non linéaire.
- le fichier .result_name_m01.**grid** qui présente les maillages déformés

On distingue aussi différents fichiers de rapports de simulations comme

- result_name-report.html, result_name-report.txt, result_name-report.tex, result_name-summary.dat, result_name-summary.html, result_name-summary.txt, result_name-summary.tex