République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieure
et de la Recherche Scientifique
Université M'hamed Bougara Boumerdèse
Faculté des Sciences de L'ingénieur
Département : De Maintenance Industrielle
Laboratoire Dynamique des Moteurs et Vibroacoustiquede

MEMOIRE

En vue l'obtention du titre de Magister en Génie Mécanique
Option : Dynamique des Moteurs et Vibroacoustique

Présenté par :

Mansouri Lotfi Zoher

THEME
Identification des fissures dans les structures Tridimensionnelles

Soutenu devant le jury :

Mr.Nour Abdelkader	Prof	UMBB, Boumerdès	Président
Mr.Khalfi Ali	Prof	U.D.L. Sidi Bel Abbes	Rapporteur
Mr. Necib Brahim	Prof	EMC Constantine	Examinateur
Mr.Rechak Saïd	Prof	E.N.P. Alger	Encadreur
Mr.Kebir Hocine	M.C.	UT. Compiègne	Co-encadreur
Mr.Si-Chaïb Med. Ouali	M.C.	UMBB , Boumerdès	Examinateur

ANNEE 2009/2010

Remerciements

Je tiens à remercier mon encadreur Le Professeur Rechak Saïd et le Docteur Kebir Hocine de m'avoir proposé ce thème, pour leurs encouragements pendant toute la période de préparation de ce mémoire et pour leur disponibilité et le matériel qu'ils nous ont mis à notre disposition.

Je remercie vivement le Professeur Nour Abdelkader d'avoir accepté d'examiner ce travail et d'avoir accepté de présider le jury de ce mémoire.

Je remercie vivement Le Professeur Khalfi Ali de l'Université Djillali Liabbes et le Professeur Necib Brahim de l'Université Mohamed Mentrouri de Constantine d'avoir accepter de faire parti du jury d'évaluation de ce mémoire, et pour leurs déplacements et le temps qu'ils ont consacré à l'étude de ce document.

Je remercie notre cher enseignant Le Docteur M.O. Si-chaib pour sa participation au jury et pour ses critiques pertinentes.

Je remercie très vivement les membres de ma famille pour leur compréhension, encouragement et aide pendant toute la période de préparation de ce mémoire, et surtout mes chers parents.

Je remercie toutes les personnes que j'ai connu à l'UTC et les agents de la bibliothèque de l'INGM, de la bibliothèque centrale de l'UMBB.

Table des matières

Chapitre I : Théorie de l'élasticité

Résumé

Dans ce chapitre, nous décrivons les équations de bases de la théorie de l'élasticité, nécessaires aux développements mathématiques qui régissent la méthode des éléments de frontières, qui seront par la suite décrits au chapitre deux.

1.1 Introduction

Le but de ce chapitre est de rassembler les équations qui régissent la réponse linéaire des solides sous chargements donnés. Ces équations forment la base des codes d'éléments de frontières pour l'analyse linéaire des contraintes. Naturellement, il y a beaucoup d'excellents textes (Timoshenko & Goodier, 1970) ; (Fung, 1965) qui décrivent la théorie de l'élasticité dans un détail beaucoup plus ample et avec une rigueur que nous ne pouvons nous permettre. Néanmoins, nous énonçons les grandes lignes de la théorie de l'élasticité afin que le lecteur puisse suivre notre démarche dans les prochains chapitres.

1.2 Champs de déformations

Nous supposons que les solides restent continus pendant le chargement (c.-à-d., aucune dislocation ni fissure ne se produit). Concernant le repère orthogonal représenté dans la figure (1.1), le vecteur déplacement u peut être décomposé en trois composants: u_x, u_y et u_z et, qui peut être représenté par la forme tensorielle u_i .

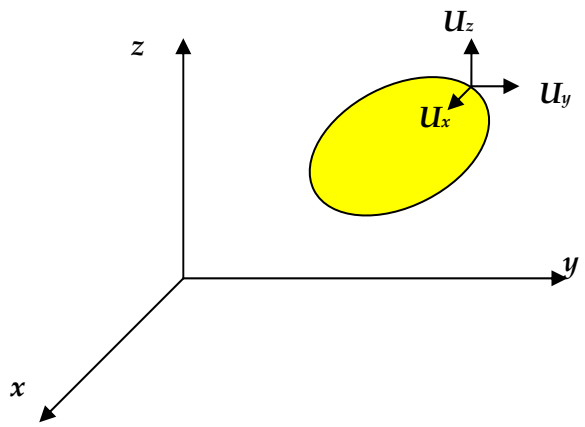


Figure 1.1

Si nous supposons plus loin que les déplacements sont suffisamment petits, alors les contraintes correspondantes sont définies par l'équation suivante :

$$\varepsilon_{ij} = \left(u_{i,j} + u_{j,i}\right)/2 \qquad (1.1)$$

Ici la virgule exprime la différentiation par rapport à la variable espace. À titre d'illustration, considérons le cas où les indices inférieurs sont égaux, par exemple, $i = j = 2 \equiv y$ L'équation (1.1) va donner le résultat suivant :

$$\varepsilon_{yy} = \frac{\partial u_y}{\partial y} \qquad (1.2)$$

1.3 Champs de contraintes

Une contrainte peut être simplement définis comme étant une " force par unité de surface." Les neuf composants du tenseur de contrainte par rapport au système d'axes orthogonal sont montrés dans la figure (1.2). Le premier indice inférieur définit la face de l'élément, le second définit la direction de l'effort. On assume que les efforts de tension, par définition, sont positifs. L'équilibre du cube élémentaire nous procure la symétrie des efforts de cisaillement, à savoir.

$$\sigma_{xy} = \sigma_{yx} \qquad \sigma_{yz} = \sigma_{zy} \qquad \sigma_{xz} = \sigma_{zx} \qquad (1.4.a, b, c)$$

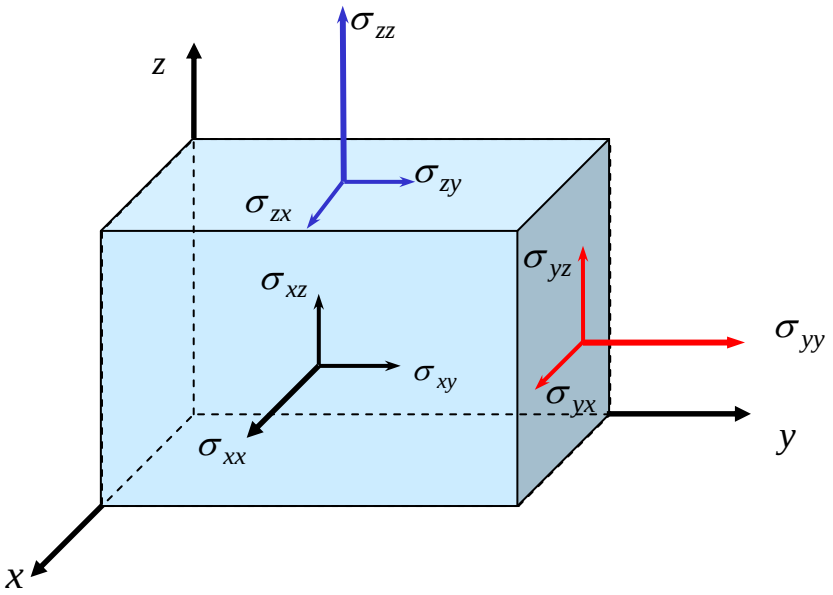


Figure 1.2 : Représentation de l'état de contraintes

Ou bien en utilisant la notation indicielle :

$$\sigma_{ij} = \sigma_{ji} \qquad (1.5)$$

1.4 Relations Contraintes-Déformations

La relation contraintes-déformations pour un matériau isotrope, linéairement élastique, est données par loi de Hooke. Sous forme algébrique, cette loi peut être exprimée de la manière suivante :

$$\varepsilon_{xx} = \frac{1}{E}\left[\sigma_{xy} - \nu\left(\sigma_{yy} + \sigma_{zz}\right)\right] \qquad \varepsilon_{xy} = \sigma_{xy} / 2G$$
$$\varepsilon_{yy} = \frac{1}{E}\left[\sigma_{yy} - \nu\left(\sigma_{xx} + \sigma_{zz}\right)\right] \qquad \varepsilon_{yz} = \sigma_{yz} / 2G \qquad (1.6)$$
$$\varepsilon_{zz} = \frac{1}{E}\left[\sigma_{zz} - \nu\left(\sigma_{xx} + \sigma_{zz}\right)\right] \qquad \varepsilon_{zx} = \sigma_{zx} / 2G$$

Dans ces équations, les constants, E , ν et G sont respectivement le module d'élasticité de Young, le coefficient de Poisson, et le module d'élasticité transversale.

$$G = \frac{E}{2(1-2\nu)} \qquad (1.7)$$

Dans quelques applications de la théorie de l'élasticité, il est commode de se servir d'autres constantes relatives aux propriétés du matériau, telles que la constante de Lamé,

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \qquad (1.9)$$

L'inversion de l'équation (1.6) nous donne :

$$\begin{aligned} \sigma_{xx} &= \lambda e + 2G\varepsilon_{xx} \\ \sigma_{yy} &= \lambda e + 2G\varepsilon_{yy} \\ \sigma_{zz} &= \lambda e + 2G\varepsilon_{zz} \end{aligned} \qquad (1.10)$$

Le paramètre de l'équation (1.10) « e » représente la déformation volumétrique, à savoir :

$$e = \varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz} \qquad (1.11)$$

Puisque le deuxième terme du côté droit de l'équation. (1.10) est semblable au terme qui relier les déformations transversales (distorsion) aux les contraintes de cisaillement, ceci induit que la loi de Hooke peut être écrite sous la forme tensorielle suivante

$$\sigma_{ij} = \lambda\delta_{ij}\varepsilon_{kk} + 2G\varepsilon_{ij} \qquad (1.12)$$

Cette équation incorpore les termes normaux et transversaux du tenseur de contraintes, en vertu des propriétés du delta de Kronecker, nous pouvons réécrire la loi de Hooke sous la forme indicielle suivante :

$$\sigma_{ij} = D^{e}_{ijkl}\varepsilon_{kl} \qquad (1.13)$$

D^{e}_{ijkl} se nomme le tenseur constitutif élastique (ou le tenseur d'élasticité) défini par l'équation (1.14) :

$$D^{e}_{ijkl} = \lambda\delta_{ij}\delta_{kl} + G\left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right) \qquad (1.14)$$

1.5 Equation de Navier-Cauchy (Equation d'équilibre)

Si nous considérons l'équilibre d'un cube infinitésimal, nous pouvons aisément dériver les équations de Navier-Cauchy pour le tenseur de contraintes:

$$\sigma_{ij,i} + b_i = 0 \qquad (1.15)$$

bj représente ici le vecteur force de volume unitaire agissant dans la direction x_j et la virgule exprime la différentiation par rapport à la variable espace x_i. En outre, sur une frontière possédant une normale extérieure unitaire ***n***, les tractions extérieures doivent équilibrer les efforts internes, ce qui produit l'état d'équilibre.

$$t_j = \sigma_{ij} n_i \qquad (1.16)$$

Les n_i, sont les composantes du vecteur de normale unitaire (cosinus directeurs). Il vaut la peine de noter qu'en raison de la symétrie du tenseur de contrainte, ces deux équations sont souvent écrites sous la forme $\sigma_{ij,i} + b_i = 0$ et $t_j = \sigma_{ij} n_i$. En substituant maintenant les relations déplacements-déformations (équation. 1.1) et la loi de Hooke (équation 1.13) dans l'équation. (1.16), nous obtenons les fameuses équations de Navier-Cauchy, en termes de déplacements:

$$G u_{i,jj} + (\lambda + G) u_{i,ji} + b_i = 0 \qquad (1.17)$$

Cette équation, ainsi que des conditions aux limites bien définies, définissent le champ de déplacement dans un solide isotrope élastique linéaire. Dans la méthode des éléments de frontières, la solution fondamentale de cette équation (le champ de déplacement dû à une force ponctuelle dans un solide infini) est superposée pour satisfaire ces conditions aux limites.

1.6 Formes réductibles en deux dimensions

Quelques problèmes en mécanique des solides peuvent êtres modélisés d'une manière satisfaisante en deux dimensions plutôt que de considérer la géométrie réelle (3D). Dans certains cas, cette simplification surgit naturellement du fait de la nature physique du problème, mais souvent elle est imposée comme approximation simplificatrice au cas réel. Trois cas distincts se présentent alors: (a) état de contraintes planes, (b) état de déformations planes, et (c) axisymétrie. Le premier cas est applicable aux géométries ayant une dimension plus petites que les deux autres sollicités dans le plan perpendiculaire à petite dimension (figure 1.4); le deuxième cas par contre est applicables aux géométries ayant une dimension

plus grande que les deux autres, en génie civil, nous trouvons les digues de barrages qui sont un exemple type (figure 1.3). Le troisième cas traite les solides de révolution autour d'un axe donné (figure 2.5).

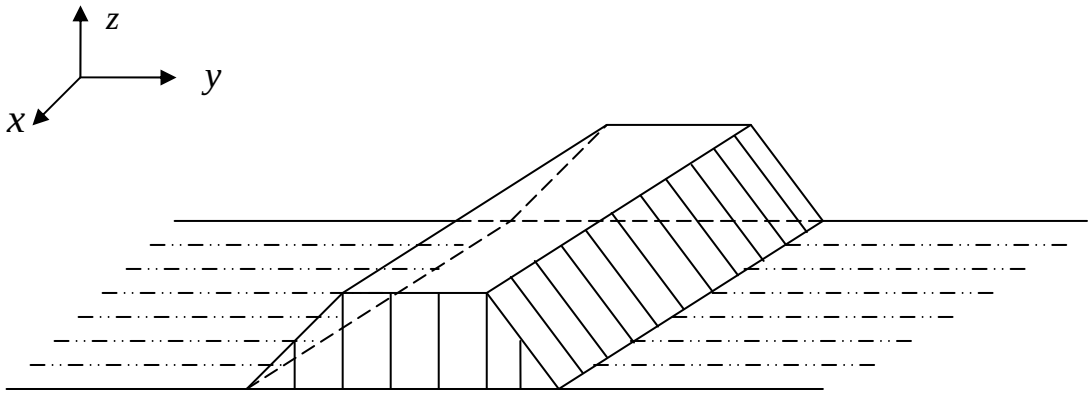


Figure 1.3 : Etat plan de déformation : exemple d'une digue de barrage

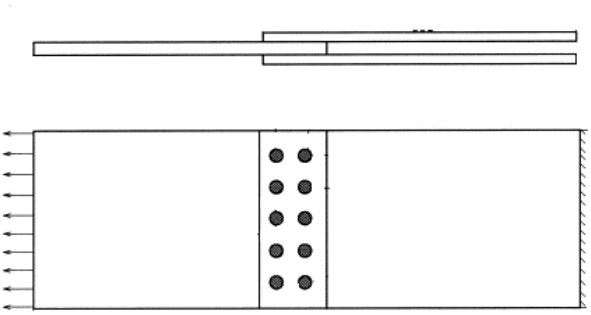


Figure 1.4: État plan de contrainte, représentation d'un assemblage boulonné.

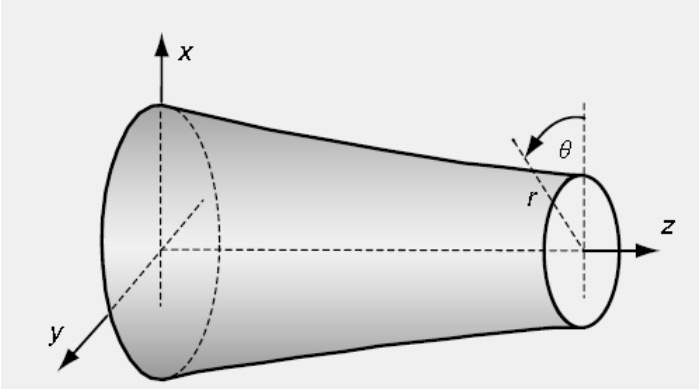


Figure 1.5: Exemple de solide axisymétrique

Beaucoup de structures industrielles (pipelines, réservoirs de stockage, etc.) sont symétriques autour d'un axe (figure 1.5). Si le domaine axisymétrique est sollicité par des efforts indépendant de la coordonnée circulaire θ, alors un tel problème peut être modélisé par une formulation axisymétrique dans les deux dimensions restantes à savoir *(r, z)*.

L'ensemble des équations réduites applicables à chacune de ces trois conditions sont brièvement présentés dans ce qui suit.

1.6.1 État plan de déformation

Nous supposons que les sections transversales normales au troisième axe *(z)* sont identiques et le chargement le long de cet axe est constant. Par conséquent, $u_3 = 0$ et les déformations $\varepsilon_{33}, \varepsilon_{32}$, et ε_{31} sont toutes nulles. Substituant ces valeurs dans l'équation. (1.6), nous obtenons immédiatement l'ensemble réduit

$$\sigma_{ij} = \lambda \delta_{ij} \varepsilon_{kk} + 2G\varepsilon_{ij} \qquad (1.18)$$

Les indices *i* et *j* varient de *1* à *2* seulement. La forme de cette équation est identique à la loi de Hooke généralisée énoncée plus haut. Sous conditions élastiques, les composantes du tenseur de contrainte suivant la direction de l'axe *(z)* sont

$$\begin{aligned} \sigma_{33} &= \nu(\sigma_{11} + \sigma_{22}) \\ \sigma_{32} &= 0 \\ \sigma_{31} &= 0 \end{aligned} \qquad (1.19)$$

Les équations qui régissent l'équilibre peuvent êtres déterminées de la même manière quant au cas tridimensionnel. Elles diffèrent des équations de Navier-Cauchy (1.18) seulement par leur nombre réduit d'indices.

1.6.2 État plan de contraintes

Sous conditions de contraintes planes, nous supposent que les contraintes dans le plan normale au troisième axe *(z)* sont nulles, c.-à-d., $\sigma_{33} = \sigma_{32} = \sigma_{31} = 0$. La substitution de ces valeurs dans l'équation. (1.6) mène aux rapports constitutifs

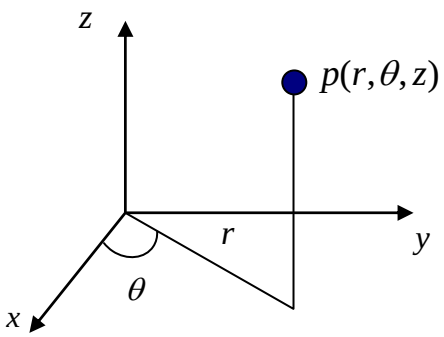
$$\sigma_{ij} = \lambda \delta_{ij} \varepsilon_{kk} + 2G\varepsilon_{ij} \qquad (1.20)$$

Le paramètre élastique *A*, est égal à $(1-2\nu)\lambda/(1-\nu)$, et les indices ***i***, ***j*** *et* ***k*** varient de *1* à *2*. La contrainte normale dans la direction x_3 est

$$\varepsilon_{33} = -\nu(\sigma_{11} + \sigma_{22})/E \qquad (1.21)$$

1.6.3 Axisymétrie

Il est avantageux d'analyser des problèmes axisymétriques dans un système de coordonnées cylindrique (figure 1.6), plutôt que dans un système cartésien orthogonal.



Le schéma 2.6: Coordonnées de cylindriques polaires**.**

Si le chargement tout comme la géométrie sont axialement symétriques, alors toutes les quantités sont constantes par rapport à l'angle θ, et les déplacements peuvent être définis en termes de composantes radiales et axiales ($u_x et u_z$) seulement. Ainsi, ces problèmes peuvent être résolus entièrement dans le plan de *r*-*z*. Ici, les avantages d'écrire des rapports de contrainte-déplacement sous la forme de tenseur ne nous contraignent pas et Nous retournons à la forme algébrique:

$$\begin{aligned}
\varepsilon_{rr} &= \frac{\partial u_r}{\partial r} \\
\varepsilon_{\theta\theta} &= \frac{u_r}{r} \\
\varepsilon_{zz} &= \frac{\partial u_z}{\partial z} \\
\varepsilon_{rz} &= \frac{1}{2}\left(\frac{\partial u_r}{\partial z} + \frac{\partial u_z}{\partial r}\right)
\end{aligned} \qquad (1.22)$$

Toutes les autres déformations sont nulles. Les efforts correspondants $\sigma_{rr}, \sigma_{\theta\theta}, \sigma_{zz}, et\, \sigma_{rz}$ peuvent être obtenus directement à partir de la forme cartésienne de la loi de Hooke (équation 1.13) en remplaçant les indices inférieurs r, θ et z par les indices inférieurs *1, 2, et 3*, respectivement. Si désiré, les équations d'équilibre peuvent également être écrites en coordonnées cylindriques, qui pour des conditions d'axisymétrie nous donne l'équation suivante :

$$G\left[u_{\alpha,rr} + \frac{1}{r}u_{\alpha,r} + u_{\alpha,zz}\right] + (\lambda + G)\left[\frac{1}{r}(ru_r)_{,r} + u_{z,z}\right]_{,\alpha} + b_\alpha = 0 \qquad (1.23)$$

L'indice inférieur α représente, successivement, les directions radiales *(r)* et axiale (z).

1.7 Conclusion

L'examen des principaux éléments de la théorie de l'élasticité décrite dans ce chapitre a été prévue afin préparer le lecteur aux développements de la formulation intégrale de frontières décrite dans le prochain chapitre. Les équations qui régissent la théorie de l'élasticité et qui doivent être satisfaites sur tout le domaine étudié, sont transformées en équations intégrales équivalentes définies au-dessus de la frontière de ce dernier. Une conséquence immédiate à cette transformation est : la dimension de l'espace du problème se réduit d'une échelle. Ainsi, l'exemple d'un problème en élasticité tridimensionnelle se réduit à une solution dans un espace bidimensionnel. Nous pouvons raisonnablement prévoir que ceci aura comme conséquence des avantages numériques significatifs, comme il deviendra apparent dans les chapitres suivants.

Chapitre II : La méthode des éléments de frontières

Résumé

Dans ce chapitre, nous présentons les équations intégrales de frontières qui fournissent la solution formelle aux équations qui régissent la théorie de l'élasticité. En plus des équations intégrales de frontière, nous dérivons également des résultats pour les efforts internes qui seront utiles pour le reste du travail.

2.1 Introduction

Dans ce chapitre, nous présentons les équations intégrales de frontières qui fournissent la solution formelle aux équations qui régissent la théorie de l'élasticité. Ces équations intégrales peuvent être dérivées de plusieurs manières, avec divers degrés de rigueur. Ici, nous commençons en développant le théorème des travaux réciproques (Betti) en utilisant la méthode d'intégration par parties, nous présentons ensuite l'identité de Somigliana. Les équations intégrales de frontière suivent alors comme cas particulier. En plus des équations intégrales de frontière, nous dérivons également des résultats pour les efforts internes qui seront utiles pour le reste du travail. Les premiers travaux qui traites les équations intégrales de frontière appliquées à l'élasticité peuvent être attribués à Rizzo (Rizzo, 1967) en 2D et à Cruse (Cruse, 1969) en 3D.

2.2 Formulation des équations intégrales

La méthode des équations intégrales est utilisable pour touts phénomènes physiques régis par des équations aux dérivées partielles linéaires à coefficients constants. Donc, dans le cadre de la mécanique des solides, le matériau doit être élastique, linéaire et homogène. Dans cette étude, on traite les problèmes avec les hypothèses supplémentaires suivantes :

- le matériau est isotrope ;
- les forces de volumes sont négligeables ;
- le contour de la structure est borné. Ainsi, on peut traiter le cas d'un trou dans un plan infini et non dans un demi-plan ;
- la formulation des équations intégrales est faite pour des problèmes en déformations planes.

2.3 Solution Elémentaire (Problème de Kelvin)

William Thomson a donné la solution fondamentale des équations d'équilibre de Navier-Cauchy pour un solide élastique tridimensionnel infiniment grand (Thomson, 1848).

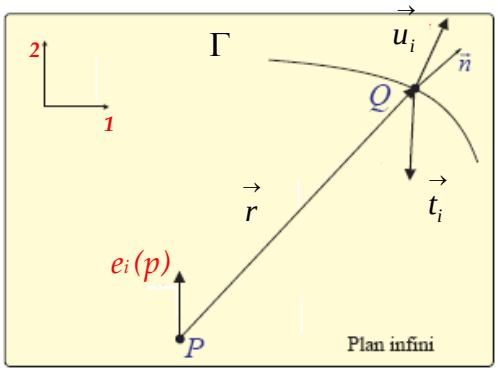


Figure 2.1 : problème de Kelvin

Ses résultats donnent les composantes cartésiennes du champ de déplacement $u_i(q)$, dus à une force ponctuelle unitaire $e_i(q)$ (voir figure 2.1):

$$u_j = U_{ij}(Q,p)e_i(p) \qquad (2.1)$$

Où

$$U_{ij}(Q,p) = \frac{A}{r}\left[B\delta_{ij} + r_{,i}r_{,j}\right] \qquad (2.2)$$

Et

$$\begin{aligned} A &= 1/16\pi G(1-\nu) \\ B &= 3-4\nu \\ r_i &= x_i(Q) - x_i(p) \\ r^2 &= r_i r_i \end{aligned} \qquad (2.3)$$

Les points Q et p se nomment point champ et point source respectivement. Il est important de distinguer soigneusement les termes r_i et $r_{,i}$ telles que r_i définit en haut représente le vecteur $\overrightarrow{pQ}$ alors que $r_{,i}$, signifie $\partial r / \partial x_i(q)$, Ici, le dernière terme est équivalent à r_i / r. Il est également essentiel d'observer soigneusement l'ordre des indices inférieurs dans l'équation (2.1), et dans toutes les équations qui vont suivre. Ainsi, dans la fonction U_{ij} de l'équation (2.2), la sommation se fait sur le premier indice inférieur *(i).* Dans la notation tensorielle, le choix est arbitraire mais, pour des raisons qui deviendront apparentes plus tard, il y a un certain avantage de définir U_{ij} de cette manière. Néanmoins, quelques auteurs adoptent la convention opposée et une grande précaution doit être prise quand à la comparaison des deux formulations. Une analyse plus approfondit de l'équation (2.1) indique clairement que la fonction $U_{ij}(Q,p)$ est singulière, c'est-à-dire, qu'elle tend vers l'infini quand les points source et champ se rapprochent (quand $r \to 0$). Etant donné le champ de déplacement, le champ de déformation ε_{jk} est aisément déterminé à partir des relations de déformations-déplacements, qui nous donne :

$$\varepsilon_{jk} = E_{ijk}(Q,p)e_i(p) \qquad (2.4)$$

Où

$$E_{ijk}(Q,p) = -\frac{A}{r^2}\left[C\left(\delta_{ik}r_{,j} + \delta_{ij}r_{,k}\right) - \delta_{jk}r_{,i} + 3r_{,i}r_{,j}r_{,k}\right] \qquad (2.5)$$

Et

$$C = 1-2\nu \qquad (2.6)$$

La démonstration de ces équations est donnée dans l'annexe A. Maintenant, En se servant de la loi de Hooke généralisée, nous pouvons obtenir la contrainte correspondante σ_{ij} au point champ, ainsi nous obtenons les relations suivantes :

$$\sigma_{jk}(Q) = \Sigma_{ijk}(Q, p) e_i(p) \qquad (2.7)$$

Avec

$$\Sigma_{ijk}(Q, p) = \frac{-2GA}{r^2}\left[C\left(\delta_{ik} r_{,j} + \delta_{ij} r_{,k} - \delta_{jk} r_{,i}\right) + 3 r_{,i} r_{,j} r_{,k}\right] \qquad (2.8)$$

Finalement, nous aurons besoin également du vecteur contraintes $t(Q)$, qui est définit par rapport à un plan par sa normale extérieure $n(Q)$ au point champ Q :

$$t_j(Q) = T_{ij}(Q, p) e_i(p) \qquad (2.9)$$

Avec

$$T_{ij}(Q, p) = \frac{-2GA}{r^2}\left[C\left(n_i r_{,j} - n_j r_{,i}\right) + \left(3 r_{,i} r_{,j} + C\delta_{ij}\right) n_m r_{,m}\right] \qquad (2.10)$$

De plus amples détails sur la dérivation des tenseurs E_{ijk}, Σ_{ijk} et T_{ij}, sont donnés dans l'annexe A.

2.4 Théorème de réciprocité de Maxwell-Betti

Pour développer les équations intégrales de frontières, nous dérivons d'abord l'identité intégrale classique, connue sous le nom du théorème des travaux réciproques de Maxwell-Betti, en utilisant la méthode d'intégration par parties.

Nous commençons en considérant deux états d'équilibre dans un domaine Ω de frontière Γ. Les déformations et les contraintes dans ces deux états sont notés par $(\sigma_{ij}, \varepsilon_{ij})$ et $(\sigma^*_{ij}, \varepsilon^*_{ij})$, respectivement.

En utilisant la loi de Hooke (équation 2.13) et en multipliant les deux côtés de cette équation par ε^*_{ij}, nous obtenons

$$\begin{aligned} \sigma_{ij}\varepsilon^* &= \lambda\delta_{ij}\varepsilon_{kk}\varepsilon^*_{ij} + 2G\varepsilon_{ij}\varepsilon_{ij}{}^* \\ &= \lambda\varepsilon_{kk}\varepsilon^*_{mm} + 2G\varepsilon_{ij}\varepsilon_{ij}{}^* \\ &= \left(\lambda\delta_{ij}\varepsilon^*_{mm} + 2G\varepsilon^*_{ij}\right)\varepsilon_{ij} \\ &= \sigma^*_{ij}\varepsilon_{ij} \end{aligned} \qquad (2.11)$$

Par conséquent, l'équation suivante est vraie:

$$\int_{\Omega} \sigma_{ij}\varepsilon^{*} d\Omega = \int_{\Omega} \sigma^{*}_{ij}\varepsilon_{ij} d\Omega \qquad (2.12)$$

Maintenant, intégrons par parties le côté droit de cette équation:

$$\begin{aligned} I_D &= \int_{\Omega} \sigma_{ij}\varepsilon^{*} d\Omega \\ &= \int_{\Omega} \sigma_{ij} u^{*}_{i,j} d\Omega \\ &= \int_{\Gamma} \sigma_{ij} u^{*}_{i} n_j d\Gamma - \int_{\Omega} u^{*}_{i}\sigma_{ij,j} d\Omega \\ &= \int_{\Gamma} t_i u^{*}_{i} d\Gamma - \int_{\Omega} b_i u^{*}_{i} d\Omega \end{aligned} \qquad (2.13)$$

Dans la dérivation ci-dessus, nous nous sommes servis des relations déformations-déplacement (équation. 2.1), de la symétrie du tenseur de contrainte, du théorème de Gauss, et des équations d'équilibre (équations 2.16-2.17). Par analogie à ce que nous avons fait au coté droit de l'équation *(2.12)* le coté gauche de celle-ci nous donne

$$I_G = \int_{\Gamma} t^{*}_{i} u_i d\Gamma - \int_{\Omega} b^{*}_{i} u_i d\Omega \qquad (2.14)$$

En vertu de l'équation (2.12), $I_D = I_G$ nous obtenons l'identité réciproque suivante:

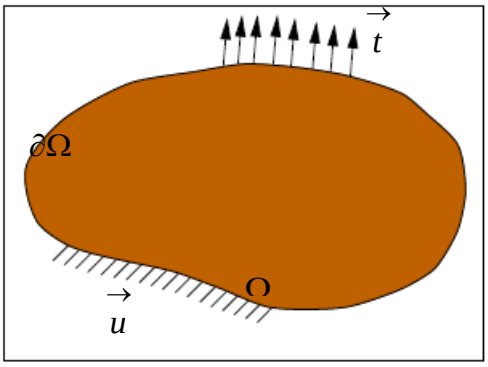
$$\int_{\Gamma} t^{*}_{i} u_i d\Gamma - \int_{\Omega} b^{*}_{i} u_i d\Omega = \int_{\Gamma} t_i u^{*}_{i} d\Gamma - \int_{\Omega} b_i u^{*}_{i} d\Omega \qquad (2.15)$$

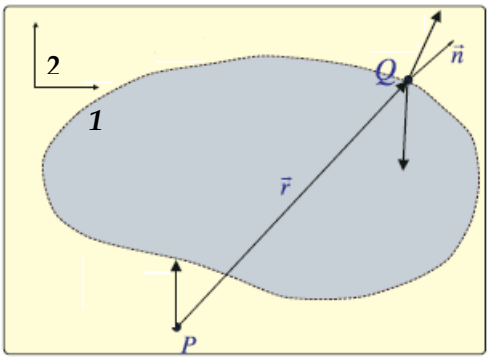
La relation précédente représente le second théorème des travaux réciproques de Maxwell-Betti, qui exprime l'égalité des travaux réciproques effectués par deux états en équilibre à travers le domaine Ω. Ce théorème forme la base de l'identité de Somigliana qui soutient la méthode des éléments de frontières.

2.5 Identité de Somigliana (1885)

Pour développer l'identité de Somigliana, nous réécrivons d'abord le théorème des travaux réciproques sous une forme plus explicite (en prenant l'opportunité de substituer l'indice inférieur *j* par l'indice *i* partout dans l'équation).

$$\begin{aligned} &\int_{\Gamma} t_j(Q) u_i(Q) d\Gamma(Q) + \int_{\Omega} b_j(q) u^{*}_{j}(q) d\Omega(q) \\ &\qquad = \int_{\Gamma} t_j(Q) u^{*}_{j}(Q) d\Gamma(Q) + \int_{\Omega} b^{*}_{j}(q) u_j(q) d\Omega(q) \end{aligned} \qquad (2.16)$$





a) Problème à résoudre (u_i, t_i, b_i) b) Problème de Kelvin

p et Q sont des points dans Ω et Γ respectivement, nous supposons maintenant que l'état caractérisé par l'ensemble (u_j, t_j, b_j) est l'état réel tandis que l'état caractérisé par (u_j^*, t_j^*, b_j^*) correspond à celui produit par une force unitaire dans un domaine infini. À partir des définitions des solutions fondamentales (équations. (2.1) et (2.9)), nous obtenons :

$$\begin{aligned} t_j^*(Q) &= T_{ij}(Q,p)e_i^*(p) \\ u_j^*(Q) &= U_{ij}(Q,p)e_i^*(p) \end{aligned} \qquad (2.17)$$

En substituant cette équation dans l'équation (2.16), aussi nous supposons pour des raisons de simplicité que les forces de volumes correspondant à l'état réel sont nulles. Ceci nous mène à écrire la relation (2.18)

$$\begin{aligned} \int_\Gamma T_{ij}(Q,q)e_i^*(p)u_j(Q)d\Gamma(Q) + \int_\Omega b_j^*(q)u_j(q)d\Omega(q) \\ = \int_\Gamma t_j(Q)U_{ij}(Q,p)e_i^*(p)d\Gamma(Q) \end{aligned} \qquad (2.18)$$

Pour finaliser le développement des équations, nous devons établir une équivalence mathématique formelle entre le vecteur force de volume $b_j^*(Q)$ et le vecteur force ponctuelle $e_i^*(p)$. Ceci peut être établi par les étapes suivantes:

$$\begin{aligned} \int_\Omega b_j^*(Q)u_j(Q)d\Omega(Q) &= \int_\Omega e_j^*(Q)u_j(Q)d\Omega(Q) \\ &= \int_\Omega e_j^*(p)\delta(Q,p)u_j(Q)d\Omega(Q) \end{aligned} \qquad (2.19)$$

Qui peut être réécrit sous la forme suivante

$$\int_{\Omega} b_j^*(Q)u_j(Q)d\Omega(Q)=\int_{\Omega}\delta_{ij}e_i^*(p)\delta(Q,p)u_j(Q)d\Omega(Q) \qquad (2.19)$$

$\delta(Q,p)$ et δ_{ij} représentent respectivement la fonction de Dirac et le delta de Kronecker. En substituant ce résultat dans l'équation. (2.18) nous observons que le vecteur force unitaire $e_i^*(p)$ est commun à toutes les intégrales. En prenant chaque composante du vecteur de force indépendamment, nous obtenons, après une certaine remise en ordre, le relation suivant :

$$\int_{\Omega}\delta_{ij}\delta(Q,p)u_j(Q)d\Omega(Q)=\int_{\Gamma}U_{ij}(Q,p)t_j(Q)d\Gamma(Q) - \int_{\Gamma}T_{ij}(Q,q)u_j(Q)d\Gamma(Q) \qquad (2.20)$$

Notez que le côté gauche de cette équation peut être simplifié d'avantage, à partir des propriétés de la fonction Dirac et du delta de Kronecker :

$$\int_{\Omega}\delta_{ij}\delta(Q,p)u_j(Q)d\Omega(Q)=\int_{\Omega}\delta(Q,p)u_i(Q)d\Omega(q) = u_i(p) \qquad (2.21)$$

Proprement dit, ce résultat est seulement vrai si *p* est à l'intérieur du domaine Ω. Evidemment, si *p* est extérieur au domaine, alors l'intégrale est nulle. Pour le moment nous reportons la considération du cas particulier lorsque *p* se trouve sur la frontière du domaine.

En substituant le résultat de l'équation (2.21), dans l'équation. (2.20), nous obtenons finalement l'équation intégrale

$$u_i(p)=\int_{\Gamma}U_{ij}(Q,p)t_j(Q)d\Gamma(Q)-\int_{\Gamma}T_{ij}(Q,p)u_j(Q)d\Gamma(Q) \qquad (2.22)$$

Trois particularités doivent êtres observées dans cette équation, connus sous le nom de l'identité de Somigliana (Somigliana, 1885):

1. Les rôles des indices inférieurs *i* et *j* sont inversés; l'addition est maintenant effectuée par rapport au deuxième indice *(j).*

2. L'intégration est effectuée par rapport au point champ Q et le point source p devient maintenant le point de collocation
3. La normale extérieure n dans le noyau T_{ij} est associée à la surface au point Q.

L'identité de Somigliana permet de calculer les déplacements à l'intérieur d'un domaine donné connaissant la distribution des contraintes et des déplacements sur la frontière. Dans un problème à valeurs limites bien-posé, la moitié des conditions au limites est connus par conséquent l'identité de Somigliana est insuffisante pour résoudre de tels problèmes. Pour poursuivre ce chemin, il est nécessaire de prendre la forme limite de l'identité car le point p peut se rapprocher de la frontière, comme démontré dans la section suivante

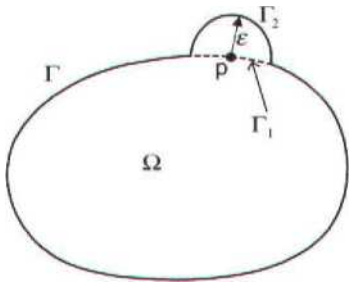


Figure 3.2: Point singulier p s'approchant de la frontière

2.6 Équations Intégrales de Frontière

Afin de trouver les équations intégrales de frontière, nous commençons par établir l'identité de Somigliana développée dans la section précédente. Nous déterminons ainsi sa forme limite quand le point p se rapproche la frontière. Pour faire la démonstration, nous supposons que le point p est entouré d'une sphère de surface Γ_2 et de rayon ε, comme représenté dans figure 2.1.

L'identité de Somigliana (équation 2.22) peut être réécrite sous la forme suivante :

$$u_i(p) = \int_{\Gamma-\Gamma_1+\Gamma_2} U_{ij}(Q,p)t_j(Q)d\Gamma(Q) - \int_{\Gamma-\Gamma_1+\Gamma_2} T_{ij}(Q,p)u_j(Q)d\Gamma(Q) \qquad (2.23)$$

Γ_1 et Γ_2 sont les parties originales et auxiliaires de la frontière p, respectivement. Nous considérons maintenant chacune de ces intégrales quand $\varepsilon \to 0$ ou, d'une manière équivalente, quand $p \to Q$. La deuxième intégrale peut être divisée comme suit :

$$\lim_{\varepsilon\to 0}\int_{\Gamma-\Gamma_1+\Gamma_2} T_{ij}(Q,p)u_j(Q)d\Gamma(Q) = \lim_{\varepsilon\to 0}\int_{\Gamma_2} T_{ij}(Q,p)u_j(Q)d\Gamma(Q) + \lim_{\varepsilon\to 0}\int_{\Gamma-\Gamma_1} T_{ij}(Q,p)u_j(Q)d\Gamma(Q) \tag{2.24}$$

Le déplacement du premier terme de l'équation (2.24) peut être isolé, ceci permet d'écrire :

$$\lim_{\varepsilon\to 0}\int_{\Gamma_2} T_{ij}(Q,p)u_j(Q)d\Gamma(Q) = \lim_{\varepsilon\to 0}\int_{\Gamma_2} T_{ij}(Q,p)\left[u_j(Q)-u_j(p)\right]d\Gamma(Q) + \lim_{\varepsilon\to 0}\left\{u_j(p)\int_{\Gamma_2} T_{ij}(Q,p)d\Gamma(Q)\right\} \tag{2.25}$$

A cause de la continuité des déplacements, le premier terme du côté gauche de cette équation va disparaitre tandis que le second devient

$$\lim_{\varepsilon\to 0}\left\{u_j(p)\int_{\Gamma_2} T_{ij}(Q,p)d\Gamma(Q)\right\} = \beta_{ij}(p)u_j(p) \tag{2.26}$$

$\beta_{ij}(p)$ est fonction de la géométrie locale de la surface au point p. Revenons maintenant à l'équation (2.24), la deuxième intégrale du côté droit de cette équation est interprétée en valeur principale de Cauchy. C'est-à-dire, bien que la fonction à intégrer soit fortement singulière au point p, l'intégrale devient finie en adoptant les nouvelles bornes d'intégration (i.e. $\Gamma-\Gamma_1+\Gamma_2$). En résumé nous obtenons

$$\lim_{\varepsilon\to 0}\int_{\Gamma-\Gamma_1+\Gamma_2} T_{ij}(Q,p)u_j(Q)d\Gamma(Q) = \beta_{ij}(p)u_j(p) + \lim_{\varepsilon\to 0}\oint_{\Gamma} T_{ij}(Q,p)u_j(Q)d\Gamma(Q) \tag{2.27}$$

La notation $\oint$ signifie que l'intégrale doit être interprétée dans le sens des valeurs principales de Cauchy (ajouter référence). Le même processus peut être suivi pour l'intégrale impliquant le noyau des déplacements U_{ij}, mais puisque le degré de singularité est moindre que celui de T_{ij} ($1/r$ pour U_{ij}, $1/r^2$ pour T_{ij}), il n'y a pas de difficulté particulière pour

effectuer cette intégrale. En d'autres termes, elle peut être intégrée par des moyens normaux (nous allons voir de plus ample détails sur l'intégration des noyaux au prochain chapitre). Nous arrivons finalement à l'équation intégrale de frontière

$$c_{ij}u_i(p)=\int_\Gamma U_{ij}(Q,p)t_j(Q)d\Gamma(Q)-\oint_\Gamma T_{ij}(Q,p)u_j(Q)d\Gamma(Q) \qquad (2.28)$$

La constante c_{ij} est donnée par :

$$c_{ij}=\delta_{ij}+\beta_{ij}(p) \qquad (2.29)$$

Cette équation, vient du processus limiteur suggéré par l'équation (2.26). Pour une frontière plane, le terme $\beta_{ij}(p)$ est égale à $-\delta_{ij}/2$, mais aux coins (ou arrêtes), son analyse devient plus complexe et le résultat dépend de l'angle que fait le coin et son orientation dans l'espace. Nous verrons au chapitre 5 l'élaboration d'un programme pour le calcul de l'angle solide dans des cas de géométries complexes.

2.7 Efforts Internes

La distribution des efforts interne dans un domaine élastique peut être retrouvée directement à partir de l'identité de Somigliana (équation 2.22), connaissant les déplacements et les contraintes à la frontière. Premièrement, les déformations au point source (point p) sont déterminées à partir des relations de déformations-déplacements (équation 2.1) comme suit

$$\varepsilon_{ij}(p)=\int_\Gamma U^{\varepsilon}_{ijk}(Q,p)t_k(Q)d\Gamma(Q)-\int_\Gamma T^{\varepsilon}_{ijk}(Q,p)u_k(Q)d\Gamma(Q) \qquad (2.30)$$

Avec

$$U^{\varepsilon}_{ijk}=\left(U_{ik,j}+U_{jk,i}\right)/2 \qquad (2.31)$$

$$T^{\varepsilon}_{ijk}=\left(T_{ik,j}+T_{jk,i}\right)/2 \qquad (2.32)$$

Puisque les déformations sont calculées au point source à partir des déplacements en ce point, les dérivées spatiales indiquées dans ces deux équations doivent être effectuées par rapport à ce point, et non par rapport au point champ (point Q) comme nous l'avons fait jusqu'ici. Ainsi, le terme $r_{,i}$, devient maintenant égal à $-r_i/r$. Les efforts correspondants peuvent être déterminés à partir de la loi de Hooke, et le résultat final peut alors être écrit sous la forme :

$$\sigma_{ij}(p)=\int_{\Gamma} U_{ijk}(Q,p)t_k(Q)d\Gamma(Q)-\int_{\Gamma} T_{ijk}(Q,p)u_k(Q)d\Gamma(Q) \qquad (2.33)$$

Avec

$$U_{ijk}=2G\left(\frac{\nu}{1-2\nu}\delta_{ij}U^{\varepsilon}_{mmk}+U^{\varepsilon}_{ijk}\right) \qquad (2.34)$$

Et

$$T_{ijk}=2G\left(\frac{\nu}{1-2\nu}\delta_{ij}T^{\varepsilon}_{mmk}+T^{\varepsilon}_{ijk}\right) \qquad (2.35)$$

En suivant ces étapes, qui sont semblables à ceux démontrés dans l'annexe A en ce qui concerne la dérivation des noyaux Σ_{ijk}, nous obtenons ainsi les expressions explicites suivantes

$$U_{ijk}=\frac{1}{8\pi(1-\nu)}\frac{1}{r^2}\left(C\left(\delta_{ki}r_{,j}+\delta_{kj}r_{,i}-\delta_{ij}r_{,k}\right)+3r_{,i}r_{,j}r_{,k}\right) \qquad (2.36)$$

Et

$$\begin{aligned}T_{ijk}=&\frac{G}{4\pi(1-\nu)}\frac{1}{r^3}\left\{3r_{,m}n_m\left[C\delta_{ij}r_{,k}+\nu\left(\delta_{ik}r_{,j}+\delta_{jk}r_{,i}\right)-5r_{,i}r_{,j}r_{,k}\right]\right.\\&\left.+3\nu\left(n_ir_{,j}r_{,k}+n_jr_{,i}r_{,k}\right)+C\left(3n_kr_{,i}r_{,j}+n_j\delta_{ik}+n_i\delta_{jk}\right)-Dn_k\delta_{ij}\right\}\end{aligned} \qquad (2.37)$$

Où

$$\begin{aligned}C&=1-2\nu\\D&=1-4\nu\end{aligned} \qquad (2.38)$$

Des expressions très semblables peuvent être obtenues (qui diffèrent seulement par rapport à certaines constantes) pour les états de déformations planes et de contraintes planes, comme montrés dans la section A.4 de l'annexe A. d'autre part, il peut être observé que le noyau T_{ijk} est hypersingulier, ceci est dû à la présence du terme $1/r^3$. Par conséquence, il est susceptible de croire que le calcul exact de cette intégrale sera complexe au voisinage de la frontière (c.-à-d., quand $r\to 0$) en utilisant uniquement l'équation (2.33), ceci implique que des techniques spéciales doivent être employées dans ce cas.

2.8 Conclusion

En ce chapitre, les équations qui régissent la théorie de l'élasticité ont étés transformées en équations intégrales. Un aspect important de cette transformation est que les équations différentielles ont été réduites en intégrales de surface, donc une réduction de la dimension du problème par un. Cet avantage qui est évident pose des difficultés si la

formulation est généralisée pour inclure la considération de facteurs complexes tels que l'anisotropie, les non-homogénéités, ainsi que les forces de volume.

CHAPITRE 3 : Aspects numériques de la méthode des éléments de frontières

Résumé

Dans ce chapitre, nous allons présenter l'aspect numérique de la méthode des éléments de frontières à savoir, la discrétisation des équations énoncées au chapitres précédent, en vu de l'obtention d'un système algébrique à résoudre pour l'évaluation des contraintes et des déformations dans le solide, aussi la partie traitement des intégrales singulières (faibles et fortes) sera étudiée en détail.

3.1 Introduction

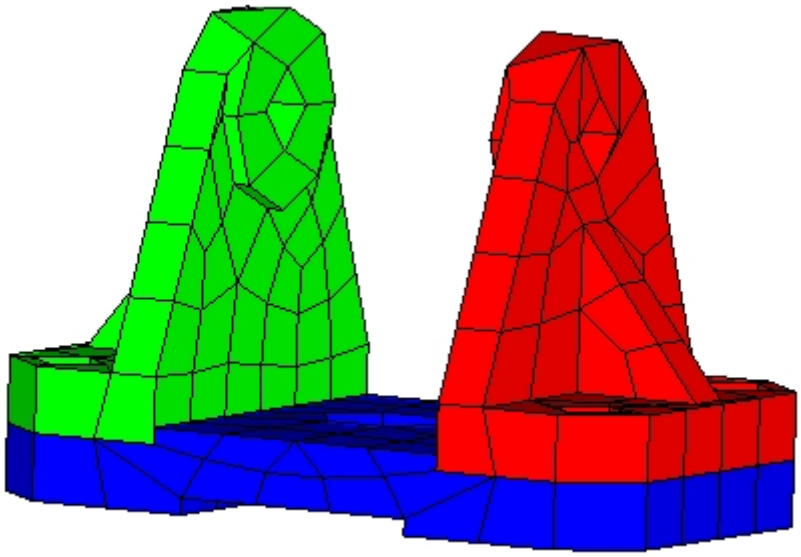
Les équations intégrales de frontières présentées au chapitre deux fournissent une description exacte d'un milieu continu élastique, et en principe la solution exacte peut être trouvée pour un ensemble de conditions aux limites bien posées. Dans la pratique, les solutions exactes ne sont pas possibles et on a recoure aux techniques numériques pour résoudre les problèmes. Ces techniques d'approximations contiennent trois principaux constituants:

(a) interpolation, (b) intégration numérique (quadrature), et (c) inversion de matrice.

Dans ce chapitre nous décrivons l'aspect numérique de la méthode des éléments de frontières en passant par la discrétisation de l'équation intégrale de frontière ainsi que les techniques d'intégration des noyaux.

3.2 Discrétisation de la Frontière

La première étape est de discrétiser la frontière Γ en un nombre suffisant d'éléments (voir figure. 3.1). Les éléments devraient former une approximation continue par morceaux de la frontière. Naturellement, plus le nombre d'éléments utilisés à cette fin est grand, plus est l'approximation meilleure, mais pour préserver des temps de calculs raisonnables il est nécessaire de mailler le moins finement possible tout en insistant sur le précision et la fiabilité des résultats.



$$x_i = \sum_{i=1}^{M} N_\alpha(\xi,\eta) x_i^\alpha \qquad (3.1)$$

Figure 3.1: Discrétisation de la peau d'une pièce en éléments de frontière.

Dans chaque élément, les coordonnées globales x_i sont interpolés à partir des coordonnées des nœuds de cet élément par des fonctions d'interpolation(ou de formes). Ces dernières sont empruntés à la de la méthode des éléments finis (Zienkiewicz, 1977).

Ainsi, dans l'équation (3.1) M représente le nombre de nœuds de l'élément, et les $N_\alpha(\xi,\eta)$ sont de fonctions d'interpolation, elles sont connues aussi sous le nom de " fonction forme". Les paramètres (ξ,η) sont les coordonnées locales (intrinsèques), définies en tout point de l'élément (figure. 3.2).

Par définition, les coordonnées intrinsèques pour un élément prennent des valeurs comprises entre **±1**. La figure (3.2) illustre des éléments quadratiques qui sont largement utilisés dans beaucoup d'applications de l'engineering. Ces éléments peuvent être de type serendip (à 8 nœuds) ou de type Lagrange (9 nœuds avec un nœud au milieu)

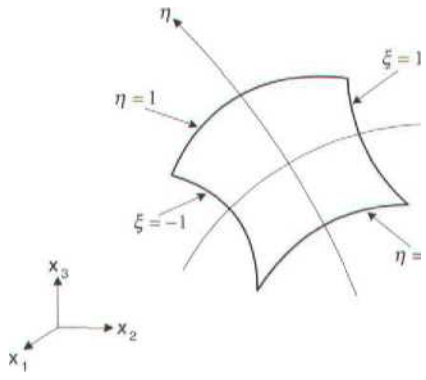


Figure 3.2 : système de coordonnées intrinsèques

Quelques propriétés spéciales des fonctions de formes peuvent être employées pour vérifier qu'elles ont été calculées correctement. D'abord, chaque fonction de forme est égale à

l'unité en son nœud associé et 0 en tous autres nœuds. Cette propriété peut être écrite sous la forme suivante :

$$N_\alpha(\xi_\beta,\eta_\beta)=\delta_{\alpha\beta} \qquad (3.2)$$

l'indice β dénote la valeur des coordonnées intrinsèques au nœud α. Cette équation implique que la valeur de la fonction interpolée en un nœud (ici, les coordonnées globales x_i) est la valeur nodale elle-même; aucune interpolation n'est nécessaire dans ces cas-ci. En second lieu, il peut être vérifié que la somme de toutes les fonctions de forme, à n'importe quel point arbitraire, est égale à 1:

$$\sum_{\alpha=1}^{M} N_\alpha(\xi,\eta)=1 \qquad (3.3)$$

3.3 Interpolation des champs de contraintes et de déplacements

La variation des déplacements et des tractions dans un élément de frontière peut être décrite en termes de valeurs nodales, plus ou moins de la même façon car la géométrie est interpolée entre les valeurs nodales. Bien que non nécessaire, il est commode de se servir des mêmes fonctions d'interpolation pour décrire les champs de tracions et de déplacements comme il a été fait pour la géométrie. Ainsi, dans de telles formulations dites " **isoparamétriques** ", les tractions et les déplacements, aux coordonnées intrinsèques (ξ,η), sont interpolés entre les valeurs nodales, identifiées par l'indice supérieur α, en utilisant les équations

$$u_i(\xi,\eta)=\sum_{\alpha=1}^{M} N_\alpha(\xi,\eta)u_i^\alpha \qquad (3.3a)$$

$$t_i(\xi,\eta)=\sum_{\alpha=1}^{M} N_\alpha(\xi,\eta)t_i^\alpha \qquad (3.3b)$$

α désigne le nœud de l'élément contenant un nombre de nœuds égale à M. Ceux qui ont déjà travaillé avec la méthode des éléments finis noterons qu'ici, les déplacements et les tractions n'ont pas besoin d'être interpolés par des fonctions d'ordre différent (ceci évitera des complications inutiles).

3.4 Discrétisation des équations intégrales de frontière

Après avoir défini les déplacements et les tractions dans un élément en termes de valeurs nodales, nous pouvons maintenant traiter ces derniers comme variables discrètes du problème. Une fois que ces quantités sont déterminées, les déplacements et les tractions sur

la frontière sont connus partout. Après avoir discrétisée la frontière en un nombre d'éléments N_e, nous pouvons maintenant réécrire les équations intégrales de frontière (3.28) en termes de ces derniers. Ainsi,

$$c_{ij}u_j(p)=\sum_{e=1}^{N_e}\left\{\sum_{\alpha=1}^{M}t_j^{\alpha}(Q)\int_{\Gamma e}U_{ij}(Q,p)N_{\alpha}(Q)d\Gamma(Q)\right\}-\sum_{e=1}^{N_e}\left\{\sum_{\alpha=1}^{M}u_j^{\alpha}(Q)\oint_{\Gamma e}T_{ij}(Q,p)N_{\alpha}(Q)d\Gamma(Q)\right\}$$

Γ_e représente le domaine de l'élément « ***e*** ». Pour produire un ensemble fermé d'équations, nous choisissons d'écrire cette équation pour chaque nœud *(p)* alternativement; c'est-à-dire, nous colloquons en chacun de ces nœuds. Pour l'instant, on suppose que les intégrales ont été calculées donnent le résultat suivant

$$\int_{\Gamma e}U_{ij}(Q,p)N_{\alpha}(Q)d\Gamma(Q)=G_{ij}^{e}(Q,p) \qquad (3.5)$$

Et

$$\oint_{\Gamma e}T_{ij}(Q,p)N_{\alpha}(Q)d\Gamma(Q)=H_{ij}^{e}(Q,p) \qquad (3.6)$$

Ainsi, à partir des équations (3.8), en notant que le déplacement au point *p* est lui-même une valeur nodale, nous obtenons :

$$c_{ij}u_j(p)=\sum_{e=1}^{Ne}\sum_{\alpha=1}^{M}t_j^{\alpha}(Q)G_{ij}^{e}(Q,p)-\sum_{e=1}^{Ne}\sum_{\alpha=1}^{M}u_j^{\alpha}H_{ij}^{e}(Q,p) \qquad (3.7)$$

Cette équation peut être mieux appréciée par son équivalent en termes de matrices, on obtiens ainsi la relation suivante :

$$[c]\{u\}=[G]\{t\}-[H']\{u\} \qquad (3.8)$$

Les vecteurs $\{u\}$ et $\{t\}$ constituent un système complet (de dimension 3*N*) des déplacements et des tractions nodales respectivement, et les matrices $[c]$, $[G]$ et $[H]$ correspondent aux coefficients définis plus haut $[c]$est, par définition, une matrice diagonale (constitué de sous- matrices 3x3, en 3D). Ces équations peuvent évidemment être condensées en une forme plus commode donnée ci-après.

$$[H]\{u\}=[G]\{t\} \qquad (3.9)$$

Avec

$$[H]=[H']+[C] \qquad (3.10)\,.$$

Néanmoins, il devrait être clair que le système ci-dessus est constitué de matrices plaines contrairement à ce que nous avons l'habitude de voir en éléments finis. Ceci est dus au fait que dans chaque élément contenant le point Q, on fait varier le point p sur tous les éléments et on calcul ainsi les matrices $[G]$ et $[H]$.

Aussi, nous notons le fait que la matrice $[G]$ soit symétrique. Ceci n'est pas vrai en ce qui concerne la matrice traction $[H]$ du fait que la normale sortante au point p diffère en générale de celle au point Q.

3.5 Traitement des Intégrales Singulières

Quand le point de départ de lecture p est situé dans le même élément que Q, les noyaux U_{ij} et T_{ij} deviennent singuliers parce qu'ils contiennent des termes d'ordre r^{-1} et r^{-2}, respectivement. Dans ce cas-ci l'application directe de la quadrature gaussienne (voir chapitre 5) est insatisfaisante et des techniques spéciales doivent être utilisées pour résoudre les singularités. Celles-ci sont décrites après.

Figure 3.6: Subdivision des éléments pour des intégrales singulières.

3.5.1 Intégrales faiblement singulières en trois dimensions

Une technique de subdivision de l'élément initiée par Lachat et Watson (Lachat & Watson, 1976) constitue un moyen efficace pour traiter des intégrales faiblement singulières. Dans cette technique (figure 3.6), les éléments contenant le point source p sont une fois de plus divisés en deux sous-éléments triangulaires (si p est situé à un nœud faisant le coin), ou trois sous-éléments triangulaires (si p est situé sur un coté de l'élément), avec p étant situé au sommet commun de ces sous-éléments. L'essence de cette technique est de tracer les sous-éléments triangulaires dans l'espace intrinsèque carré de l'élément ; (Mustoe, 1984). Ainsi, dans la figure 3.7, les nœuds 1, 8, et 4 de l'élément intrinsèque coïncident avec le point p.

En raison de cette dégénérescence, il peut être montré (annexe B) que le Jacobien de la transformation est de l'ordre r, où r est la distance entre *p et Q*.

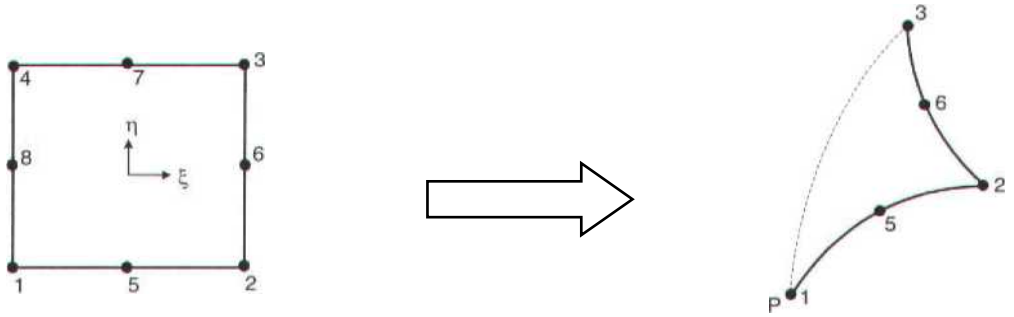


Figure 3.7: représentation d'un sous-élément

Par conséquent, cette technique nous permet de contourner singularité faible et l'intégrale peut être évaluée par une quadrature de Gauss normale.

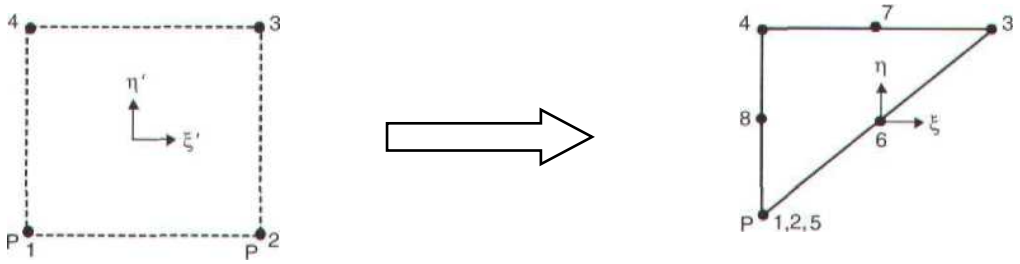


Figure 3.8: Système de coordonnée intrinsèque du sous-élément.

Puisque la transformation des coordonnées intrinsèques de l'élément aux coordonnées intrinsèques du sous-élément est linéaire, une stratégie alternative est possible (Becker, 1992). Dans ce cas-ci, un nouveau système de coordonnées intrinsèque ξ' et η' avec son origine au centre de l'élément est défini pour chaque sous-élément (figure 3.8). Les fonctions de formes linéaires (figure 3.9) sont maintenant employées pour déterminer les coordonnées intrinsèques originales pour un point dans le nouveau système de coordonnées intrinsèque comme suit:

$$\xi(\xi',\eta') = \sum_{\alpha=1}^{4} N'_{\alpha}(\xi',\eta')\xi^{\alpha} \qquad (3.11)$$

$$\eta(\xi',\eta') = \sum_{\alpha=1}^{4} N'_{\alpha}(\xi',\eta')\eta^{\alpha} \qquad (3.12)$$

Les fonctions linéaires de forme sont données par :

$$N_1^{'}(\xi',\eta')=\frac{1}{4}(1-\xi')(1-\eta') \qquad (3.13a)$$

$$N_2^{'}(\xi',\eta')=\frac{1}{4}(1+\xi')(1-\eta') \qquad (3.13b)$$

$$N_3^{'}(\xi',\eta')=\frac{1}{4}(1+\xi')(1+\eta') \qquad (3.13c)$$

$$N_4^{'}(\xi',\eta')=\frac{1}{4}(1-\xi')(1+\eta') \qquad (3.13a)$$

Et les paramètres ξ^{α} et η^{α} sont des valeurs nodales des coordonnées intrinsèques originales.

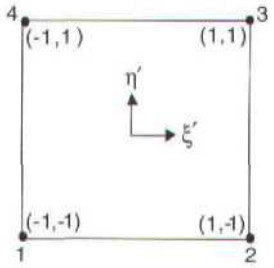


Figure 3.9: Élément linéaire a quatre nœuds.

Si l'élément est subdivisé en N_S sous-éléments, les intégrales singulières prennent la forme suivante :

$$\int_{\Gamma e} f(P,Q)d\Gamma = \int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta)J(\xi,\eta)d\xi d\eta = \sum_{s=1}^{Ns}\int_{-1}^{1}\int_{-1}^{1} f(\xi'',\eta'')J(\xi'',\eta'')J_S(\xi',\eta')d\xi' d\eta' \qquad (3.14)$$

ξ'' et η'' signifient respectivement $\xi(\xi',\eta')$, $\eta(\xi',\eta')$ et $J_S(\xi',\eta')$ représente le Jacobien de la transformation du système de coordonnées intrinsèques originale au nouveau de système de coordonnées, le Jacobien de la transformation est donné par la relation suivante :

$$J_S(\xi',\eta') = \frac{\partial(\xi,\eta)}{\partial(\xi',\eta')} = \begin{vmatrix} \dfrac{\partial\xi(\xi',\eta')}{\partial\xi'} & \dfrac{\partial\eta(\xi',\eta')}{\partial\xi'} \\ \dfrac{\partial\xi(\xi',\eta')}{\partial\eta'} & \dfrac{\partial\eta(\xi',\eta')}{\partial\eta'} \end{vmatrix} \qquad (3.15)$$

Par analogie à ce qui a été étudié précédemment, le Jacobien $J_S(\xi',\eta')$ tend vers $O(r)$ quand $r \to 0$, puisque les coordonnées intrinsèques originales des deux nœuds associés au point *p* prennent les mêmes valeurs que le sous-élément transformé. Par exemple, les nœuds 1 et 2 (voir figure.3.8) sont rendus coïncidents en posant :

$$\begin{aligned} \xi_1 &= \xi_2 = -1 \\ \eta_1 &= \eta_2 = -1 \end{aligned} \qquad (3.16)$$

3.5.2 Intégrales faiblement singulières en deux dimensions

Dans un espace bidimensionnel (intégration sur une ligne), le noyau des déplacements U_{ij} est faiblement singulier avec une singularité de l'ordre de $\log(r)$. La stratégie que nous adoptons doit dans ce cas-ci isoler la singularité logarithmique et l'intégrer en employant la règle d'intégration de Gauss pour les fonctions logarithmiques singulières. Naturellement, le résidu non singulier peut être facilement intégré en utilisant l'intégration ordinaire de Gauss.

Au niveau du développement analytique, nous considérons un élément quadratique (à trois nœuds) de manière assez détaillée, alors que seulement les résultats finals pour l'élément linéaire (à deux nœuds) sont donnés. Pour l'élément quadratique, trois cas doivent être considérés puisque le point source *p* peut être situé à un des nœuds d'extrémité ou au nœud du milieu. Au premier nœud d'extrémité (le nœud 1), la distance ***r*** entre un point arbitraire définit par la coordonnée intrinsèque ξ et le point source peut être obtenu par l'équation suivante

$$r^2 = [x(\xi) - x_1]^2 + [y(\xi) - y_1]^2 \qquad (3.17)$$

x_1 et y_1 sont les coordonnées globales du point source. En Substituant les fonctions de formes quadratiques 1D (Annexe B) dans cette équation, nous obtenons

$$r^2 = \left[\frac{1}{2}(1+\xi)\right]^2 \left[(-(2-\xi)x_1 + \xi x_2 + 2(1-\xi)x_3)^2 + (-(2-\xi)x_1 + \xi x_2 + 2(1-\xi)x_3)^2\right] \qquad (3.18)$$

De même, quand P est à l'autre extrémité de l'élément (nœud 2), nous obtenons

$$r^2 = \left[\frac{1}{2}(1-\xi)\right]^2 \left[\left(-(2+\xi)x_2 + \xi x_1 + 2(1+\xi)x_3\right)^2 + \left(-(2+\xi)y_2 + \xi y_1 + 2(1+\xi)y_3\right)\right] \qquad (3.19)$$

Ces deux équations peuvent être exprimées sous la forme unifiée

$$r^2 = \eta^2\left[f_1^2 + f_2^2\right] \qquad (3.20)$$

Où

$$\eta = \frac{1}{2}(1-\xi_p\xi) \qquad (3.21)$$

Avec $\xi_p = -1$ si le point source *p* est situé au premiers nœud et $\xi_p = 1$ si *p* est à l'autre extrémité de l'élément (nœud 2). Les fonctions f_1 et f_2 sont alors

$$\begin{aligned} f_1 &= -(2+\xi_p\xi)x_a - \xi_P\xi x_b + 2(1+\xi_P\xi)x_3 \\ f_2 &= -(2+\xi_p\xi)y_a - \xi_P\xi y_b + 2(1+\xi_P\xi)y_3 \end{aligned} \qquad (3.22)$$

Avec $a = 1\ et\ b = 2$, quand le point singulier *p* est au nœud 1, et $a = 2\ et\ b = 1$, quand le point singulier *p* est au nœud 2.

Pour le cas où *p* est situé au milieu de l'élément (nœud 3), nous obtenons

$$r^2 = \xi^2\left[g_1^2 + g_2^2\right] \qquad (3.23)$$

Avec

$$\begin{aligned} g_1 &= \frac{1}{2}\left[(\xi-1)x_1 + (\xi+1)x_2\right] - \xi x_3 \\ g_2 &= \frac{1}{2}\left[(\xi-1)y_1 + (\xi+1)y_2\right] - \xi y_3 \end{aligned} \qquad (3.24)$$

Pour l'élément linéaire, Equations (3.52 à 3.53) s'appliquent toujours mais les fonctions f_1 et f_2 réduisent aux expressions simples

$$\begin{aligned} f_1 &= x_2 - x_1 \\ f_2 &= y_2 - y_1 \end{aligned} \qquad (3.25)$$

Maintenant, prenant le logarithme de l'équation.(3.52), nous obtenons :

$$\log_e\left(\frac{1}{r}\right) = \log_e\left(\frac{1}{\eta}\right) - \frac{1}{2}\log_e\left[f_1^2 + f_2^2\right] \qquad (3.26)$$

Une expression semblable à l'équation (3.26) peut être obtenue à partir de l'équation (3.55). Dans l'équation. (3.58), le deuxième terme du côté droit est non singulier et peut être intégrée sans difficulté, alors que le premier terme peut être intégré en utilisant une quadrature de Gauss logarithmique, comme décrit ci-dessous. La règle de quadrature de Gauss pour une fonction logarithmique prend la forme (Stroud & Secrest, 1966) :

$$I = \int_0^1 \log_e(1/x) f(x) dx \approx \sum_{k=1}^{n} w_k f(x_k) \qquad (3.27)$$

L'intervalle d'intégration est compris entre zéro et un et les ordonnées x_k et les poids w_k sont donnés dans le tableau 4.2. Dans notre cas, la fonction $f(x)$ représente la fonction de forme.

Tableau 3.1 Quadrature de Gauss : coordonnées et poids pour des fonctions logarithmiques singulières d'après Stroud et Secrest (Stroud & Secrest, 1966)

Ordre	Points de Gauss	Poids de Gauss
1	0.25	1
2	0.1120088061 0.60227 69081	0.7185393190 0.2814606809
3	0,0638907931 0.3689970637 0.7668803039	0.5134045522 0.3919800412 0.0946154066
4	0.0414484802 0.2452749143 0.5561654535 0.8489823945	0.383464061 0.3868753177 0.19043 51269 0.03922 54871
5	0.0291344722 0.1739772133 0.4117025205 0.6773141745 0.89477 1361	0.2978934717 0.3497762265 0.2344882900 0.0989304595 0.0189115521

Pour appliquer la méthode d'intégration logarithmique, aux nœuds situés en milieu d'arrête, l'élément doit être divisé en deux sous-élément portés sur ce nœud. Puisque l'intervalle d'intégration logarithmique gaussienne va de zéro à un, tandis que la l'intervalle du même système de coordonnées intrinsèque est inclus entre ±1, On propose donc le changement de variables suivant :

$$x = -\xi \quad \text{pour } \xi < 0 \qquad \text{et} \qquad x = -\xi \quad \text{pour } \xi < 0$$

Ceci permettra de faire la liaison entre l'intervalle d'intégration logarithmique et l'intervalle de définition du repère intrinsèque. Dans ce cas le Jacobien est simplement égale à un. Pour les nœuds situés sur les extrémités, l'intégration peut être Calculée sur l'élément entier, en utilisant la transformation $x = (1 - \xi\xi_p)/2$ et en notant que le Jacobien dans ce cas-ci sera égale à 2.

3.5.3 Intégrales Fortement Singulières

Les intégrales fortement singulières dans l'équation (3.8), interprétées au sens des valeurs principales de Cauchy, assemblée à la constant c_{ij}, donne les termes diagonaux de la matrice $[H]$ dans l'équation. (3.13). L'évaluation précise de ces termes est d'une importance critique. Dans la littérature nous trouvons deux différentes techniques pour aborder le problème

(a) Evaluation directe en utilisant le théorème de Cauchy (Guiggiani & Gigante, A general algorithme for multidimensional cauchy principal value in the boundary element method, 1990) (Guiggiani & Krishnasamy, 1992) et d'Hadamard (1903) pour des intégrales à singularités fortes d'ordre $O(r^{-1})$ et $O(r^{-2})$ respectivement (Kebir, 1999).
(b) Une méthode indirecte, qui exploite la technique du mouvement de corps rigides. C'est la plus utilisée pour des problèmes d'élastoplasticité cependant en mécanique de la rupture cette technique n'est pas applicable. C'est cette méthode qui est explicitée dans ce chapitre.

Si une région finie est soumise à un mouvement de corps rigide u_j^n unitaire dans la nième direction cartésienne, les tractions extérieures doivent être toutes nulles (Absence de contraintes internes vu que les déformations sont nulles). En d'autres termes, au kième nœud, nous produisons ***n*** équations de forme :

$$\begin{aligned} u_j^k &= u_j^n \\ t_j^k &= 0 \end{aligned} \qquad (3.28)$$

En Substituant les équations ci-dessus dans l'équation (3.13), nous obtenons le système d'équations suivant

$$[H]\{I\}^n = \{0\} \qquad (3.29)$$

$\{I\}^n$ est un ensemble de vecteurs colonnes de nombre n, dans lesquels (pour tous les nœuds) des déplacements unitaire sont appliqués dans la nième direction et des valeurs nulles pour toutes autres directions. De l'équation (3.61), les coefficients singuliers de la sous-matrice pour le kième nœud (qui apparaissent sur la diagonale de la matrice H) peuvent êtres déterminés à partir de la somme des éléments qui sont sur la même ligne ou se trouve le terme singulier, donc :

$$[H]_{ij}^{kk} = (\delta_{km} - 1)\sum_{m=1}^{N} [H]_{ij}^{km} \qquad (3.30)$$

N est le nombre de nœuds, les indices inférieurs $\boldsymbol{i}$ et $\boldsymbol{j}$ varient de 1 à 3 (en 3D), et les indices supérieurs $\boldsymbol{k}$ et $\boldsymbol{m}$ font références aux nœuds des éléments.

3.6 Évaluation des contraintes de frontières

Les contraintes sur la frontière du domaine du problème ne peuvent pas être déterminées directement à partir des équations intégrales de frontière (équation 3.33) parce que le noyau T_{ijk} contient une singularité de l'ordre $O(r^{-3})$ (Hypersingularité). La technique la plus utilisée pour surmonter ce problème est la méthode de « recouvrement de tractions » (plus connue sous le nom de Traction Recovery Method) (Cruse T. , 1974); (Telles & Brebbia, 1979); (Banerjee & Davies, 1984); (Kane, 1994), dans cette section, nous décrivons cette approche en termes explicites. La stratégie ici est de déterminer les déformations tangentielles (à partir des déplacements) et par conséquent, en utilisant la loi de Hooke et en connaissant les tractions récupérez les contraintes.

Les contraintes tangentielles locales peuvent être exprimées en termes de dérivés particulières des déplacements comme suit

$$\varepsilon_{IJ}^{'} = \frac{1}{2}\left(\frac{\partial u_I^{'}}{\partial x_J^{'}} + \frac{\partial u_J^{'}}{\partial x_I^{'}}\right) \qquad (3.31)$$

Les indices inférieurs en majuscules I *et* J (prenant les valeurs 1 et 2) soulignent le fait ce ceux-ci se rapportent au système d'axe local. Les dérivés locales du déplacement peuvent être obtenus à partir des équations suivantes :

$$\frac{\partial u_I^{'}}{\partial x_J^{'}} = \frac{\partial u_I^{'}}{\partial \xi_k}\frac{\partial \xi_k}{\partial x_J^{'}} \qquad (3.32)$$

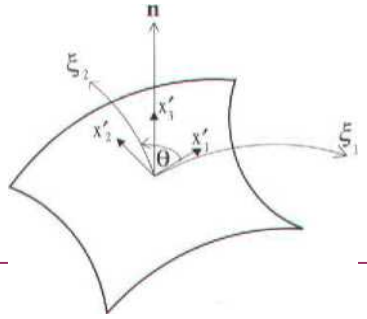


Figure 3.1: système d'axes local orthogonal sur un élément de frontière.

L'indice inférieur *k* prend les valeurs 1 et 2, les dérivés des coordonnées intrinsèques par rapport aux coordonnées locales sont données par l'équation suivante (Lachat, 1975); (Becker, 1992)

$$\begin{aligned}
&\frac{\partial \xi_1}{\partial x_1^{'}} = \frac{1}{|m_1|} \\
&\frac{\partial \xi_1}{\partial x_2^{'}} = \frac{-\cos\theta}{|m_1|\sin\theta} \\
&\frac{\partial \xi_2}{\partial x_1^{'}} = 0 \\
&\frac{\partial \xi_2}{\partial x_2^{'}} = \frac{1}{|m_1|\sin\theta}
\end{aligned} \qquad (3.33)$$

Dans les équations précédentes, θ représente l'angle définis dans figure 3.11, et

$$\begin{aligned}
&|m_k| = \sqrt{\left(\frac{\partial x_1}{\partial \xi_k}\right)^2 + \left(\frac{\partial x_2}{\partial \xi_k}\right)^2 + \left(\frac{\partial x_3}{\partial \xi_k}\right)^2} \\
&\cos\theta = \frac{1}{|m_1||m_2|}\frac{\partial x_i}{\partial \xi_1}\frac{\partial x_i}{\partial \xi_2}
\end{aligned} \qquad (3.34)$$

Les composantes locales des déplacements dans l'équation. (3.33) peuvent êtres exprimés en termes de déplacements nodaux, par rapport au système d'axe global, par l'intermédiaire de la transformation

$$u_I^{'} = L_{Ij} u_j \qquad (3.35)$$

Les déplacements et les tractions peuvent êtres obtenus par interpolation des valeurs nodales de la manière habituelle

$$\begin{aligned}
&u_i(\xi,\eta) = \sum_{\alpha=1}^{M} N_\alpha(\xi,\eta) u_i^\alpha \\
&t_i(\xi,\eta) = \sum_{\alpha=1}^{M} N_\alpha(\xi,\eta) t_i^\alpha
\end{aligned} \qquad (3.36)$$

Les L_{ij} sont les cosinus directeur du système de coordonnées local par rapport au système de coordonnées global. Pour ***j*** allant de 1 à 3, nous obtenons

$$L_{1j} = \frac{1}{|m_1|}\frac{\partial x_j}{\partial \xi_1} \qquad (3.37)$$

Les termes restants de ce tenseur sont donnés par :

$$\begin{aligned} L_{21} &= n_2 L_{13} - n_3 L_{12} \\ L_{22} &= n_3 L_{11} - n_1 L_{13} \\ L_{23} &= n_1 L_{12} - n_2 L_{11} \end{aligned} \qquad (3.38)$$

Où n_1, n_2, et n_3 représentent les trois composantes du vecteur normale. En utilisant ces équations, nous pouvons maintenant calculer les déformations tangentielles locales. Puis, en utilisant la loi des Hooke et en éliminant la déformation principale ε'_{33} nous obtenons :

$$\begin{aligned} \sigma'_{11} &= \frac{2G}{1-\nu}\left(\varepsilon'_{11} + \nu\varepsilon'_{22}\right) + \frac{\nu}{1-\nu}\sigma'_{33} \\ \sigma'_{22} &= \frac{2G}{1-\nu}\left(\varepsilon'_{22} + \nu\varepsilon'_{11}\right) + \frac{\nu}{1-\nu}\sigma'_{33} \\ \sigma'_{12} &= 2G\varepsilon'_{12} \end{aligned} \qquad (3.39)$$

L'équilibre exige d'imposer les relations suivantes :

$$\begin{aligned} \sigma'_{33} &= t'_3 = L_{3j}t_j \\ \sigma'_{23} &= t'_2 = L_{2j}t_j \\ \sigma'_{13} &= t'_1 = L_{1j}t_j \end{aligned} \qquad (3.40)$$

L'équilibre nous donne :

$$L_{3j} = n_j \qquad (3.41)$$

En Prenant ces résultats ensemble, nous obtenons les expressions suivantes pour les efforts, par rapport au système d'axes locales

$$\sigma_{11}^{'} = \frac{2G}{1-\nu}\left(\frac{\partial \xi_k}{\partial x_1^{'}} L_{1j} + \nu \frac{\partial \xi_k}{\partial x_2^{'}} L_{2j}\right)\frac{\partial u_j}{\partial \xi_k} + \frac{\nu}{1-\nu} L_{3j} t_j$$

$$\sigma_{11}^{'} = \frac{2G}{1-\nu}\left(\frac{\partial \xi_k}{\partial x_2^{'}} L_{2j} + \nu \frac{\partial \xi_k}{\partial x_1^{'}} L_{1j}\right)\frac{\partial u_j}{\partial \xi_k} + \frac{\nu}{1-\nu} L_{3j} t_j \qquad (3.42)$$

$$\sigma_{12}^{'} = G\left(\frac{\partial \xi_k}{\partial x_1^{'}} L_{2j} + \nu \frac{\partial \xi_k}{\partial x_2^{'}} L_{1j}\right)\frac{\partial u_j}{\partial \xi_k}$$

Finalement, nous utilisons les relations de transformation suivantes

$$\sigma_{mn} = L_{rm} L_{sn} \sigma_{rs}^{'} \qquad (3.43)$$

Afin d'exprimer les efforts locaux en termes de leurs équivalents cartésiens globaux, comme suit:

$$\sigma_{mn} = A_{mnj\alpha} u_j^{\alpha} + B_{mnj} t_j \qquad (3.44)$$

Après quelques transformations mathématiques, nous obtenons les coefficients $A_{mnj\alpha}$ et B_{mnj}

$$A_{mnj\alpha} = 2G\left\{\frac{1}{1-\nu}\left[L_{1m}L_{1n}\left(\frac{\partial \xi_K}{\partial x_1^{'}} L_{1j} + \nu \frac{\partial \xi_K}{\partial x_2^{'}} L_{2j}\right) + L_{2m}L_{2n}\left(\frac{\partial \xi_K}{\partial x_2^{'}} L_{2j} + \nu \frac{\partial \xi_K}{\partial x_1^{'}}\right)\right]\right.$$
$$\left. + \frac{1}{2}\left(L_{1m}L_{2n} + L_{2m}L_{1n}\right)\left(\frac{\partial \xi_K}{\partial x_1^{'}} L_{2j} + \frac{\partial \xi_K}{\partial x_2^{'}} L_{1j}\right)\right\}\frac{\partial N_\alpha}{\partial \xi_K} \qquad (3.45)$$

$$B_{mnj} = \left(L_{3m}L_{1n} + L_{1m}L_{3n}\right)L_{1j} + \left(L_{2m}L_{3n} + L_{3m}L_{2n}\right)L_{2j} + \left(\frac{\nu}{1-\nu}\delta_{mm} + \frac{1-2\nu}{1-\nu} L_{3m}L_{3n}\right)L_{3j} \qquad (3.46)$$

Nous rencontrons un problème analogue en deux dimensions et il peut être résolu d'une manière semblable. Dans ce cas, la coordonnées locale ξ est tangentiel à l'élément tandis que x_1'et x_2' représentent le système d'axes locale, comme montré dans figure 3.12.

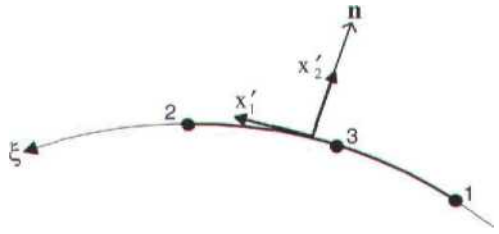


Figure 3.12: système d'axe orthogonal local sur une élément de frontière 1D

Dans le cas bidimensionnel, les indices inférieurs (m, n, j, α) varient généralement de 1 à 2. Après le procédé décrit ci-dessus, nous obtenons

$$A_{mnj\alpha} = \frac{2G}{1-\nu} L_{1m} L_{1n} L_{1j} \frac{\partial \xi}{\partial x_1^{'}} \frac{\partial N_\alpha}{\partial \xi} \qquad (3.48)$$

$$B_{mnj} = \left(L_{1m} L_{2n} + L_{2m} L_{1n}\right) L_{1j} + \left(\delta_{mn} - \frac{1-2\nu}{1-\nu} L_{1m} L_{1n}\right) L_{2j} \qquad (3.49)$$

Avec

$$\frac{\partial \xi}{\partial x_1^{'}} = \frac{1}{J(\xi)} \qquad (3.50)$$

$J(\xi)$ est le Jacobien de la transformation du système global *(x,y)* au système local ξ.

$$\begin{aligned} L_{22} &= -L_{11} = n_2 \\ L_{21} &= L_{12} = n_1 \end{aligned} \qquad (3.51)$$

n_1 et n_2 sont les composantes du vecteur normal (équation. 3.25). Pour un état plan de déformation, les coefficients ci-dessus doivent être complétés par les termes additionnels suivant :

$$A_{33j\alpha} = \frac{2G\nu}{1-\nu} L_{1j} \frac{\partial \xi}{\partial x_1^{'}} \frac{\partial N_\alpha}{\partial \xi} \qquad (5.52)$$

Et

$$B_{33j} = \frac{\nu}{1-\nu} L_{2j} \qquad (5.53)$$

Ces équations nous permettent de calculer les contraintes sur la frontière à partir des déplacement et tractions nodales. Si les efforts sont calculés en un nœud qui est partagé par plusieurs éléments, et que le champ de contrainte est continu en ce point, les résultats obtenus à partir de chacun des éléments sont moyennés. Cependant, si le champ de contrainte est discontinu, à cause d'une discontinuité des tractions, faire la moyenne de tous les composants des contraintes donne des résultats incorrects. Dans ce cas, la discontinuité

devrait être préservée en définissant des nœuds multiples à la discontinuité et en exécutant les calculs indépendamment pour chaque élément.

3.7 Conclusion

Dans ce chapitre, nous avons vu les principales techniques qui sont employées pout transformer l'équation intégrale de frontière en un système algébrique. Plusieurs techniques citées dans ce chapitre sont programmées dans le code KSP, citons par exemple la technique proposée par Lachat et Watson pour calculer les intégrales faiblement singulières en 3D, rappelons aussi que dans KSP, les intégrales fortement singulières sont calculées analytiquement.

Chapitre 4 : La programmation orientée objet et le code KSP

Résumé

Dans ce chapitre nous présentons la programmation orienté objet et l'évolution des techniques de programmation au fil du temps, nous détaillons ensuite l'architecture du code KSP qui constitue la plateforme de notre travail.

4.1 Introduction

Pour résoudre des problèmes modernes de modélisations, de dimensionnements où bien de calculs en mécanique et dans n'importe quel autre domaine technologique, il ne suffit pas de développé des méthodes numériques et de trouver des interprétations mathématiques pour les phénomènes physiques que nous souhaitons étudier. Il faut en plus trouver des algorithmes adéquats et développer des codes de simulations numériques pour rendre possible le traitement de ces problèmes où du moins diminuer leur complexité.

La simulation numérique est devenue ces dernières années un marché « juteux » en termes de rentes, au fil du temps nous avons assisté à la création de beaucoup compagnies spécialisées dans la conception de codes de calcul et de dimensionnement tel que Altaire (HYPERMESH, RADIOSS…), Dassault systèmes (CATIA, ABAQUS …), c'est dans cette philosophie que le code KSP à été conçu afin de répondre à une demande croissante en terme d'outils de simulations et de dimensionnements des structures.

KSP (voir plus bas pour la présentation du code) possède cependant un avantage que la plus part des logiciels de simulations ne possèdent pas, le fait de traiter numériquement les problèmes de mécanique par la méthode des équations intégrale. En effet, la plus part des logiciels commerciaux utilisent la méthode des éléments finis pour la résolution des équations de la mécanique des milieux continus, de ce fait KSP se révèle original est plus performant que les autres dans certains domaines de la mécanique, citons par exemple la mécanique de la rupture, car ne nécessitant pas de remaillage de tout le domaine. KSP est écrit en Visual C++, un langage de programmation orienté objet qui jouit d'une modularité que n'ont pas les autres langages de programmations plus classique, la plus part des logiciels éléments finis, nous citons à titre d'exemple le code commercial ABAQUS sont programmés en FORTRAN, cependant, ce genre de langage appelé fonctionnel se trouve limité par le fait qu'on doit gérer un nombre trop important de ligne (parfois des centaines de milliers de lignes), la nouvelle tendance est que les concepteurs de logiciels on décidé de « traduire » leur code en orienté objet, soit en optant pour le langage C++ qui est en terme de vitesse d'exécution aussi rapide que le FORTRAN, ou bien en optant pour le langage JAVA. De plus les solutions graphiques que peut offrir C++ sont impossible à obtenir avec le langage FORTRAN, car il existe une multitude d'outils pour le développement d'interfaces graphique, à titre d'exemple MFC qui est livré avec MICROSOFT VISUAL C++, on peut citer aussi Qt qui est un outil de développement d'interfaces graphique puissant et en plus distribué sous licence « Freeware ». Dans ce qui suit nous allons voir plus en détails la programmation orienté objet, nous passerons ensuite à la description de l'architecture interne de KSP.

4.2 La programmation orientée objet (P.O.O):

Présentation

On attend d'un programme informatique :

- la robustesse (réaction correcte à une utilisation « hors normes »)
- l'extensibilité (aptitude à l'évolution)
- la réutilisabilité (utilisation de modules)
- la portabilité (support d'une autre implémentation)
- l'efficacité (performance en termes de vitesse d'exécution et de consommation mémoire)

Les langages évolués de type *C* ou *PASCAL*, reposent sur le principe de la programmation structurée (algorithmes + structures de données). Le C++ et un langage orienté objet. Un langage orienté objet permet la manipulation de ***classes***. Comme on le verra dans ce chapitre, la classe généralise la notion de structure (qui est elle-même un généralisation de la sub-routine en Fortran). Une ***classe*** contient des variables (ou « attributs ») et des fonctions (ou « méthodes ») permettant de manipuler ces variables.

Les langages « orientés objet » ont été développés pour faciliter l'écriture et améliorer la qualité des logiciels en termes de modularité et surtout de réutilisation. Un langage orienté objet est livré avec une bibliothèque de classes. Le développeur utilise ces classes pour mettre au point ses logiciels.

4.3 Rappel sur la notion de prototype de fonction

En C++, comme en C, on a fréquemment besoin de déclarer des *prototypes* de fonctions. En particulier, on trouve dans les fichiers d'en-tête (reconnaissable par l'extension « ***.h*** »), des *prototypes* de fonctions appelées par le programme. Le *prototype* d'une fonction est constitué du nom de la fonction, du type de la valeur de retour, du type des arguments à passer.

Un exemple simple pour comprendre les objets :

```
#ifndef NORM_H
#define NORM_H
# include "point.h"
# include <Cmath>
# include <iostream>
using namespace std;

class vecteur
{
```

```
 private :  //la structure interne d'un vrai objet n'a pas à être publique (encapsulation)
 double x,y,z;
 point a,b,c;   // point est une classe qui possède des attributs ( x,y,z) , c'est pourquoi on
                // note la présence  de  # include "point.h"

 public :
    vecteur ()    // constructeur sert à initialiser et à appeler la classe
           {
           double x=0,y=0,z=0;
            }
     vecteur(double a,double b,double c)  // second constructeur permet de créer un
           {
            x=a;y=b;z=c;                  // vecteur avec les composantes a, b et c
           }

//les accesseurs en lecture sont publics
  double getx() {return(x);}
  double gety() {return(y);}
  double getz() {return(z);}
 void setx( double a) {x=a;}
  void sety(double a) {y=a;}
  void setz(double a) {z=a;}
  void affiche_vect()
     {cout<<"["<<x<<","<<y<<","<<z<<"]";}
  void saisir(double x,double y,double z);
  void add_vecteur (vecteur a, vecteur b)
     {x=a.x+b.x; y=a.y+b.y; z=a.z+b.z;}

  double norme(void) {return(sqrt(x*x+y*y+z*z));}  //calcule la norme
  void normer(void) {double n=norme();x/=n;y/=n;z/=n;}
  void prodvect(vecteur v, vecteur w)
    {
         x= v.gety()*w.getz()-w.gety()*v.getz();
         y= v.getz()*w.getx()-w.getz()*v.getx();
         z= v.getx()*w.gety()-w.getx()*v.gety();
    }
   double prodscal(vecteur v vecteur w) {return(v.x*w.x+v.y*w.y+v.z*w.z);}
};  // c'est la fin d'une déclaration
 # endif
```

Cette partie de code appartient au programme qui calcul l'angle solide et que nous avons élaboré pour la formulation des éléments conformes, nous allons voir plus en détail l'organigramme de ce code au chapitre 5 de ce mémoire.

Revenons sur l'exemple précédent. « vecteur » est une ***classe*** (un objet), cette classe est constituée d'attributs privées (données privées) x , y, z, a ,b et c (a,b et c étant eux-mêmes des objets), et de fonctions membres publiques (ou méthodes) déclarées par le mot clé publique « affiche_vect », « add_vecteur », « prodscal » etc.

La classe est toujours déclarée au début du programme (données et prototype des fonctions membres), puis on définit le contenu des fonctions membres. Les données *x*, *y*.. sont dites ***privées.*** Ceci signifie que l'on ne peut les manipuler qu'au travers des fonctions membres publiques. On dit que le langage C++ réalise ***l'encapsulation des données.***

a, b et c sont des ***objets*** de la classe «point», c'est-à-dire des variables de type «point». On a défini ici un nouveau type de variable, propre à cette application, comme on le fait en C avec les structures. Si nous déclarons V comme étant un vecteur, suivant le principe dit de « l'encapsulation des données », la notation **V.x (**qui permet de donner la première composante du vecteur V**)** est interdite à l'extérieur de la classe. C'est pour cette raison que nous remarquons la présence de ce qu'on appel les ***accesseurs*** ou bien les « ***setteurs*** » et les « ***getteurs*** », ceci vont nous permettre de manipuler les données d'une classe à l'extérieur de celles-ci et bien évidement l'intérêt principale de l'encapsulation est de minimiser les risques d'erreurs et de bugs au niveau du programme.

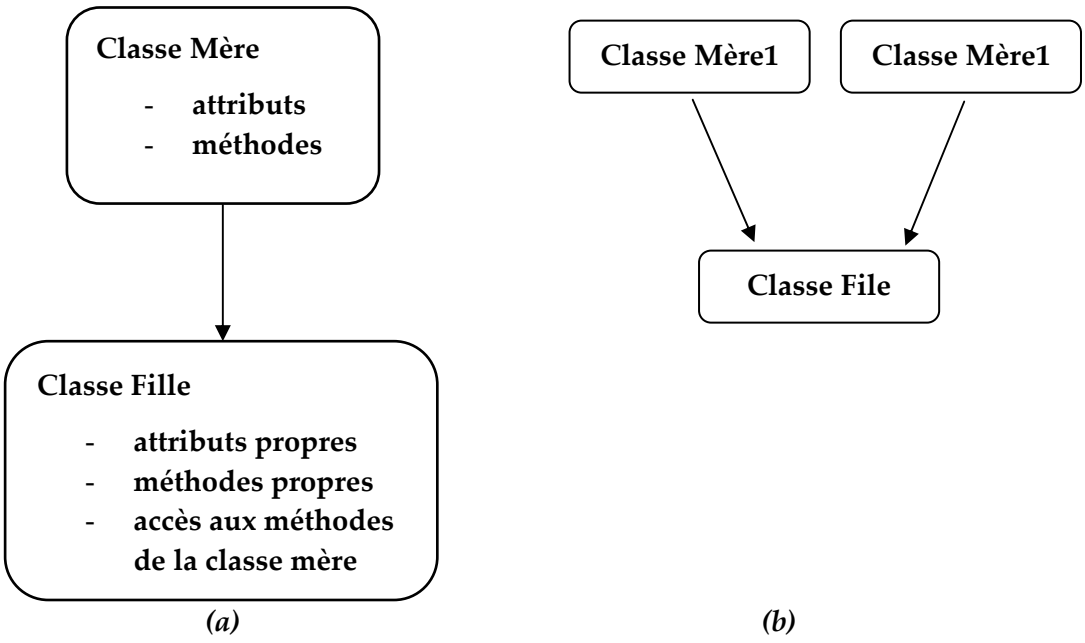


Figure 4.1 : repésentation de la notion d'héritage

(a) : héritage simple ; **(b)** : héritage multiple.

4.4 L'héritage

La programmation orientée objet permet de définir à partir de classes existantes de nouvelles classes dérivés appelés ***classes filles*** (Figure 4.1). Ces classes filles peuvent hériter des attributs et méthodes de leurs ***classes mères***. Cette notion est appelée *héritage.* L'héritage permet entre autres à l'utilisateur, à partir d'une bibliothèque de classes donnée, de développer ses propres classes munies de fonctionnalités propres à l'application. On dit alors qu'une classe fille ***dérive*** d'une ou de plusieurs classes mères.

La classe fille n'a pas accès aux attributs (privées) de la classe mère.

Afin d'illustrer la notion d'héritage, nous présentons un exemple concret utilisant cette notion. Le programme suivant représente une partie du fichier prototype (header) de la classe CAXIBEMProbleme implémentée dans le code KSP.

```
#pragma once
#include "Probleme.h"     //
#include "SystemeDEquation.h" //

class AFX_EXT_CLASS CAXIBEMProbleme : public CProbleme
{
DECLARE_SERIAL(CAXIBEMProbleme)
public:
        void CalculdesContraintesInterieursParLissage();
        void ComputeVContraintes(CObArray *ListePointsPourContraintes);
        void ComputeInternalStressRapid(CObArray *ListePointsPourContraintes);
        void EvaluerLesValeursNodales();
        void SetCondiLimiOnCollocationPoints();
        void ComputeVonMisesStress();
        void AffecterBoundaryStressSurDualMesh();
        void ComputeBoundaryStress();
        void ComputeInternalStress();
        …

BOOL ComputeDeplacementPI;
BOOL ComputeContraintePI;
BOOL ComputeContraintePC;
};
```

Dans le prototype, les méthodes et les attributs sont tous déclarés comme étant publiques, ceci implique que ces derniers seront accessibles même à partir de l'extérieur de la classe sans avoir à recourir aux accesseurs. Nous remarquons aussi que la classe CAXIBEMProbleme dérive (hérite) de la classe CProbleme.

4.6 Le code KSP (Kernel of the Simulation Platform)

Dans ce qui suit nous allons présenter l'architecture générale du programme K.S.P, développé au sein du laboratoire Roberval de l'Université de technologie de Compiègne par le maître de conférences M. Hocine KEBIR, ainsi qu'une étude approfondie des différentes classes faisant intervenir la méthode des équations intégrales.

Le logiciel KSP est un logiciel de calcul de mécanique numérique, faisant intervenir aussi bien la méthode des éléments finis que la méthode des éléments de frontières, ainsi que les simulations de découpe ou de clinchage. Il offre la possibilité de modélisation des problèmes d'élasticités 2D et 3D, des problèmes d'élastoplasticité et de mécanique de la rupture et de calcul et dimensionnement des structure soumises à la fatigue (chargement cycliques).

Décomposition des problèmes dans KSP

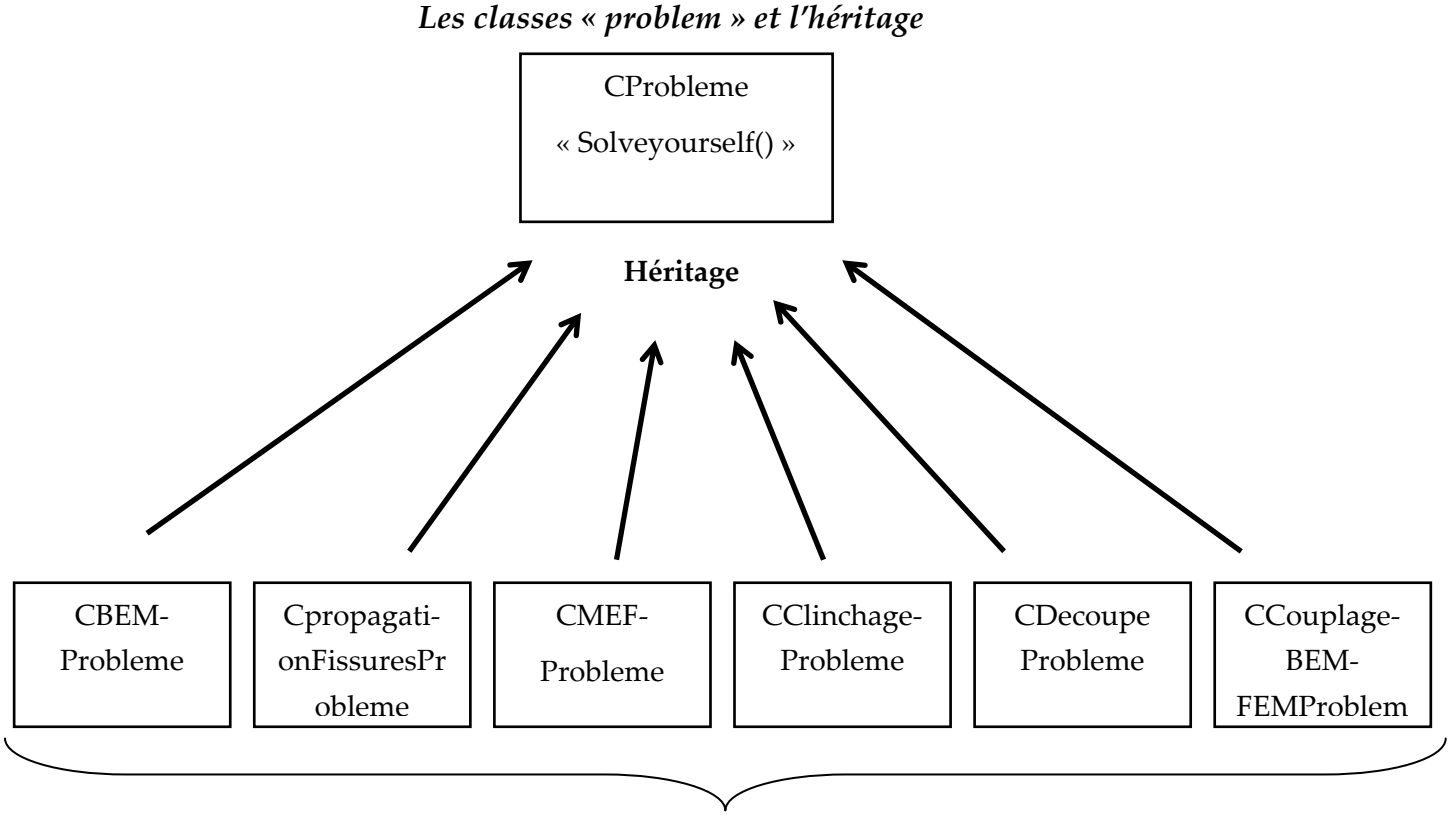


Figure 4.2 : Présentation des classes Probleme

- *CBEMProbleme : Cette classe permet le calcul de problèmes mécaniques par la méthode des éléments de frontières.*
- *CpropagationFissuresProbleme : Cette classe permet de gérer la propagation des fissures.*
- *CMEFProbleme : Cette classe permet le calcul de problèmes mécaniques par la méthode des éléments finis.*
- *CClinchageProbleme : Cette classe permet de simuler le clinchage.*
- *CDecoupeProbleme : Simulation de la découpe de tôles.*
- *CCouplageBEM-FEMProbleme : Module de couplage entre la méthode des équations intégrales et la méthode par éléments.*

KSP est un programme pluridisciplinaire de calculs mécaniques numériques. En effet, il peut traiter aussi bien les problèmes mécaniques par la méthode des éléments finis, que par la méthode des équations intégrales (ou par éléments de frontières). Des modules de clinchage, de découpe ou même de couplage entre la méthode des éléments finis et la méthode des équations intégrale (Bigot & Kebir 2009), ou encore la propagation des fissures sont également présents (Figure 4.2).

Ainsi, chaque méthode de calcul numérique (élément finis, équations intégrales) est appelée dans la programmation de KSP un problème. Cette partie aura pour but de présenter les différentes catégories de problèmes de KSP et l'héritage entre les différentes classes problèmes.

4.7 Étude approfondie de la classe CBEMProbleme

Dans KSP, la classe de base est la classe CProblem. La classe pour la méthode des équations intégrales est la classe CBEMProblem. CBEMProblem est une classe dérivée de la classe CProblem, qui elle même est une classe dérivée de la classe CObject, classe de base pour tout le programme KSP.

4.7.1 Fonction Solveyourself

But : Initialise les résultats si il y avait des résultats auparavant et met la variable de type bool m_isrunning=true.

4.7.2 Fonction Dopreparation()

Cette fonction permet

- La création de la liste des nœuds de collocation

```
void CBEMProbleme::CreateListNoeudCollocation()
{CElement *ptE;
int n;
        m_ListNoeudCollocation.RemoveAll();
        for(int i=0; i<GetCurrentPiece()->GetMesh()->GetNbrEle();i++)
{ ptE=GetCurrentPiece()->GetMesh()->GetEleAt(i);
n=ptE->GetNbrPoint();
        for(int j=0;j<n;j++)
        {
m_ListNoeudCollocation.Add(ptE->GetPointAt(j));
        }
}
```

Cette fonction boucle sur les éléments et les nœuds de collocation de chacun des éléments.

- La création de la numérotation des nœuds de collocation

```
void CBEMProbleme::CreateNumeroOfNoeudCollocation()
{CElement *ptE;
int n,num=0;
        for(int i=0; i<GetCurrentPiece()->GetMesh()->GetNbrEle();i++)
{ ptE=GetCurrentPiece()->GetMesh()->GetEleAt(i);
n=ptE->GetNbrPoint();
        for(int j=0;j<n;j++)
        {ptE->GetPointAt(j)->SetNumero(num++);
        }
```

- La création des conditions aux limites des noeuds

```
void CBEMProbleme::SetCondiLimiOnNoeuds()
{
GetCurrentPiece()->GetEtat()->CreateNodalElemCondiLimi();
        for(int  j=0;j<GetCurrentPiece()->GetEtat()->GetNbrNodalCondiLimi();j++)
        {
        CNoeud*ptN=(CNoeud*)->GetCurrentPiece()->GetEtat()->GetNodalCondiLimiAt(j)-
>GetSupport();
                switch(GetCurrentPiece()->GetEtat()->GetNodalCondiLimiAt(j)->GetType())
```

```
{
case TBC_UXUYZERO :
        ptN->SetDDLValue(0,0.);
        ptN->GetDDL(0)->SetIndex(-1);
        ptN->SetDDLValue(1,0.);
        ptN->GetDDL(1)->SetIndex(-1);
        break;
case TBC_UXUYUZZERO :
        ptN->SetDDLValue(0,0.);
        ptN->GetDDL(0)->SetIndex(-1);
        ptN->SetDDLValue(1,0.);
        ptN->GetDDL(1)->SetIndex(-1);
        ptN->SetDDLValue(2,0.);
        ptN->GetDDL(1)->SetIndex(-1);
        break;
case TBC_UXZERO :
        ptN->SetDDLValue(0,0.);
        ptN->GetDDL(0)->SetIndex(-1);
        break;
case  TBC_UYZERO:
        ptN->SetDDLValue(1,0.);
        ptN->GetDDL(1)->SetIndex(-1);
        break;
case  TBC_FX100:
        ptN->SetDDLDualValue(0,100);
        ptN->GetDDL(0)->SetIndex(1);
        break;
case  TBC_FY100:
        ptN->SetDDLDualValue(1,100);
        ptN->GetDDL(1)->SetIndex(1);
        break;
case  TBC_FYFS:
```

```
ptN->SetDDLDualValue(1,GetCurrentPiece()->GetEtat()->GetNodalCondiLimiAt(j)-
>m_Fs.GetFxAt(0));
                    ptN->GetDDL(1)->SetIndex(1);
                    break;
            default : break;
            }
            }
```

- La création des conditions aux limites sur les nœuds de collocation

Cette fonction est identique à la fonction précédente, sauf que dans chacun des cas de l'instruction Switch, il est nécessaire d'effectuer une boucle sur les noeuds de collocation.

La fonction CreateSystemEquation

- La fonction SetTailleDuProbleme

Tout d'abord, il est nécessaire de connaître la taille du problème à résoudre. Cette opération est gérée à l'aide de cette fonction :

```
void CBEMProbleme::SetTailleDuProbleme()
{int taille=0;
CElement *ptE;
for(int i=0; i<GetCurrentPiece()->GetMesh()->GetNbrEle();i++)
{
ptE=GetCurrentPiece()->GetMesh()->GetEleAt(i);
for(int j=0; j<ptE->GetNbrPoint();j++)
{
taille+= ptE->GetPointAt(j)->GetNbrDDL();
}
```

- La fonction AssemblerElement(ptE)

Ensuite, on assemble le système d'équation pour chacun des éléments, notamment pour faire l'assemblage des matrices K et F. On fait donc un bouclage sur tous les éléments.

La fonction Solve ()

Cette fonction permet le lancement de la triangularisation du système d'équation précédemment formé.

La fonction DoGetResult()

Cette fonction permet d'obtenir la résolution du système.

Comme touts logiciels de calculs en mécanique, KSP est constitué essentiellement de trois parties : l'introduction des données, le traitement et la sortie des résultats. Nous présentons dans ce qui va suivre, ces différentes parties.

4.8 Le Pré-traitement dans KSP

L'introduction des données est la première étape dans le déroulement d'une manipulation sur KSP. Elle se décompose elle même à deux étapes : l'introduction des données géométriques et l'introduction des conditions aux limites.

Les données géométriques sont introduites dans le cas de problèmes 2D par un « sketchboard » de dessin offrant la possibilité de faire les différentes opérations principales d'un dessin 2D. Le maillage est réalisé par un mailleur propre à KSP et qui offre un maillage bien adapté pour l'utilisation par la méthode des équations intégrales.

En ce qui concerne les problèmes 3D les données géométriques sont introduites à l'aide d'une fonction qui permet de les importer depuis d'autre logiciel telle que ABAQUS où IDEAS.

Le KSP offre un menu de fonction pour l'introduction des différentes conditions aux limites (déplacements imposés, charges imposés ou les différents blocages) directement sur son interface d'utilisation pour les problèmes 2D et 3D.

4.9 Le traitement

Cette partie se révèle la plus importante car c'est dans cette partie que le remplissage et la résolution du système est faite. Différentes méthodes de résolution sont implémentées dans KSP parmi les quelles on trouve un algorithme de résolution appelé GMRES (Leung C.Y & Walker S.P, 1997) qui permet des gains de temps pour les systèmes à très grand nombre de degrés de liberté.

4.10 Le post-traitement des résultats dans KSP

Il est assuré dans le KSP par une interface graphique qui permet de voire la distribution des contraintes clairement à l'aide d'un jeux de couleur qui offre une très bonne visibilité aux résultats. Le KSP donne aussi la possibilité de voir les déformés ainsi que des animations de déformation à l'échelle normale et exagéré. Beaucoup d'autres résultats peuvent être tiré du KSP tel que les contraintes maximales et minimales ainsi que des graphiques pour une lecture et une interprétation plus fiable.

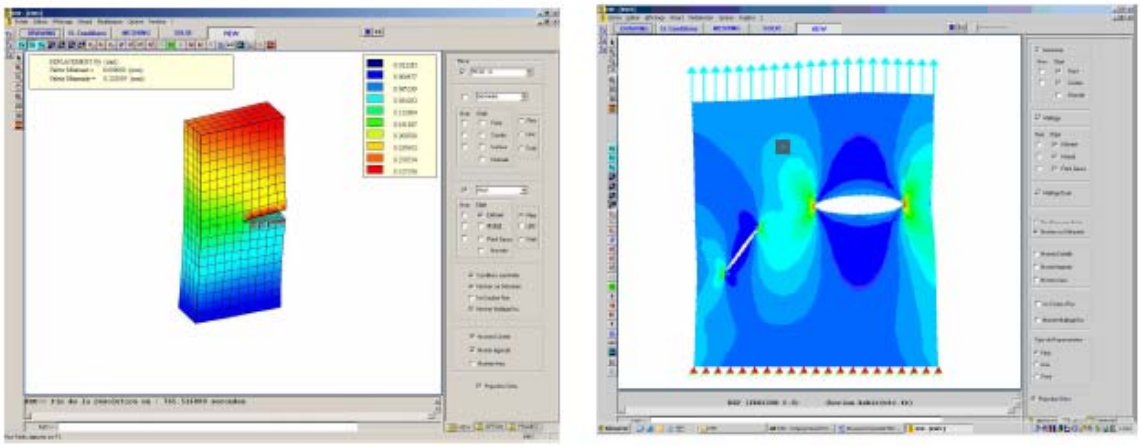


FIG.2.2. exemple de sortie de résultats sur le KSP

4.11 Conclusion

Dans ce chapitre nous avons fait une bref présentation de la programmation orientée objet, nous avons illustré par des exemples les avantages que procure cette dernière, parmi les quelles on trouve la notion d'héritage. Cette fonctionnalité est probablement l'un des points les plus attirants de la P.O.O, ensuite nous avons présenté l'architecture du code KSP, ainsi que ses principales fonctions et modules, dans le chapitre suivant nous présentons le cœur de notre travail dans ce mémoire, nous parlerons des interventions que nous avons opéré pour réduire et optimiser les temps de calcul lors de simulation via KSP.

Chapitre V

Première partie : Intégration Adaptative

Résumé

Dans cette première partie du chapitre cinq, nous proposons une nouvelle technique de calcul des intégrale des noyaux réguliers. Cette technique est basée sur le choix du nombre de point de gauss nécessaire au calcul de ces noyaux. Nous proposons ici une formule heuristique qui permet de donner un nombre de point de gauss minimal afin de réduire les temps de calculs, à la fin de cette première partie du chapitre, nous validerons notre approche par des cas test sur des pièces mécaniques variées.

5.1 Introduction

L'intégration de manière précise et efficace des déplacements et des contraintes au niveau de la frontière du domaine est extrêmement importante. Le défi ici est de proposer des techniques qui fournissent une précision acceptable à un coût de calcul minimal. Puisque les noyaux doivent être évalués jusqu'à des millions de fois (nécessité de raffiner le maillage au niveau des zones de concentration de contraintes dans le cas de calculs en mécanique de la rupture ou en plasticité), des gains significatifs en temps de calcul peuvent être réalisées en programmant l'intégration de manière efficace.

Plus que tout, il est nécessaire d'examiner la nature des noyaux pour s'assurer que des évaluations inutiles ne soient faites. En général, il est nécessaire d'effectuer les intégrations en utilisant des méthodes numériques de quadrature, dans ce qui suit, nous allons passer en revue quelques aspects des méthodes d'intégrations numériques. Ensuite nous proposons une technique qui permettra de réduire les couts d'intégration des noyaux réguliers, le but étant de réduire au maximum les temps de calcul.

5.2 Intégration de Gauss-Legendre

Pour approcher numériquement la valeur d'une intégrale définie, la quadrature de Gauss-Legendre est celle qui permet d'avoir une formule exacte pour les polynômes de degré élevé en choisissant au mieux à la fois les points d'évaluation de la fonction et les coefficients correspondants. On peut ainsi obtenir des valeurs très proches de la valeur exacte avec un nombre de points assez réduit.

Pour appliquer les règles de quadrature de Gauss-Legendre à un intervalle arbitraire, il est seulement nécessaire de tracer cet intervalle dans l'espace de quadrature de Gauss, dénoté par le symbole ξ sans oublier d'introduire le Jacobien :

$$I = \int_a^b f(x)dx = \int_{-1}^{1} f(x(\xi))J(x,\xi)d\xi \qquad (5.1)$$

Le principe de l'intégration de Gauss-Legendre, est de transformer une intégrale propre en une somme de produits de plusieurs fonctions. Par conséquent, l'équation.(5.1), devient

$$I \approx \sum_{k=1}^{n} w_k f(x_k)J(x,\zeta) \qquad (5.2)$$

x_k et w_k représentent respectivement les points et poids de Gauss et peuvent être calculés par les formule suivante :

x_k : racines du polynôme de Legendre donné par

$$P_n = \prod_{\substack{j=1 \\ j \neq i}}^{n} \left(\frac{x - x_j}{x_i - x_j} \right) \qquad (5.3)$$

Et w_k est retrouvé par la relation suivante :

$$w_k = \int_{-1}^{1} \prod_{\substack{j=1 \\ j \neq i}}^{n} \left(\frac{x - x_j}{x_i - x_j} \right) dx \qquad (5.4)$$

Dans les équations précédentes, $J(x,\xi) = dx/d\xi$ représente le Jacobien de la transformation. Notons que pour une transformation géométrique linéaire le Jacobien peut être considéré comme constant, L'intégration en deux et trois dimensions peut être traitée plus ou moins de la même façon.

En deux dimensions, nous aurons la relation suivante

$$I = \int_{a_1}^{b_1} \int_{a_2}^{b_2} f(x_1, x_2) dx_1 dx_2 \;\; = \sum_{k_1=1}^{n_1} \sum_{k_2=2}^{n_2} w_{k_1} w_{k_2} f\left(x_1(\xi_1, \xi_2), x_2(\xi_1, \xi_2)\right) J(x,\xi) \qquad (5.5)$$

Avec

$$J(x,\xi) = \begin{vmatrix} \dfrac{\partial x_1}{\partial \xi_1} & \dfrac{\partial x_1}{\partial \xi_2} \\ \dfrac{\partial x_2}{\partial \xi_1} & \dfrac{\partial x_2}{\partial \xi_2} \end{vmatrix} \qquad (5.6)$$

En général, les bornes d'intégration n'ont pas besoin d'être constantes; en d'autres termes, la région physique n'a pas besoin d'être définie par des frontières parallèles au axes du repère global x_1 et x_2. Cette intégration bidimensionnelle pourrait également être utilisée pour intégrer une fonction au-dessus d'une surface *3D* si les axes x_1 et x_2 constituent un système d'axes local (tangentiel).

Retournons à notre sujet principal, il devrait maintenant être évident que le problème principal est de déterminer le nombre minimum de points d'intégration sur un élément donné qui puisse réduire au maximum les temps de calculs tout en procurant une précision acceptable. Ceci dépendra clairement de la distance de l'élément au point de collocation

(point *p*) et des propriétés des noyaux puisque ces derniers possèdent des singularités fortes voir des hyper- singularités dans le cas 3D.

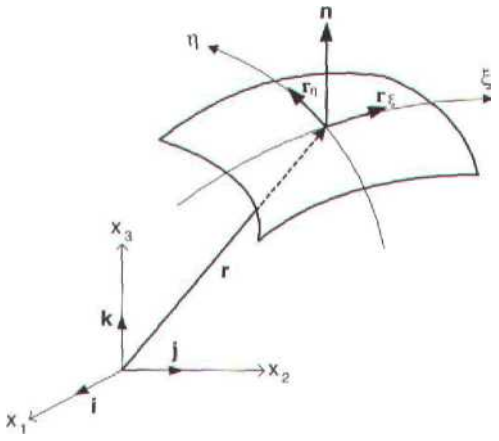


Figure 5.1 représentation du vecteur normal extérieur à l'élément.

Concrètement, les intégrales qui nous concernent sont du type

$$I_i = \sum_{\alpha=1}^{M} t_j^{\alpha} \int_{-1}^{1}\int_{-1}^{1} U_{ij}\left[x^p, x(\xi,\eta)\right] N_\alpha(\xi,\eta) J(\xi,\eta) d\xi d\eta \qquad (5.9)$$

Ici, x^p est le point de collocation et U_{ij} représente le noyau à intégrer au-dessus de l'élément. Nous excluons ici toutes les intégrales singulières et par conséquent il n'y a rien de particulier au sujet du choix des U_{ij} pour des buts d'illustration plutôt que T_{ij}. D'abord, nous devons déterminer le Jacobien de la transformation du système de coordonnées global tridimensionnel au système de coordonnées intrinsèque bidimensionnel.

A partir des résultats trouvés en géométrie différentielle (Burke, 1985) le Jacobien d'une telle transformation est égal au module du vecteur résultant du produit vectoriel des deux vecteurs r_ξ *et* r_η, qui sont portés par les plans tangent à surface de l'élément (figure 5.1). Sous forme explicite :

$$J(\xi,\eta) = \left| r_\xi \times r_\eta \right| \qquad (5.10)$$

Où

$$r_\xi = \frac{\partial x_1(\xi,\eta)}{\partial \xi} i + \frac{\partial x_2(\xi,\eta)}{\partial \xi} j + \frac{\partial x_3(\xi,\eta)}{\partial \xi} k$$
$$r_\eta = \frac{\partial x_1(\xi,\eta)}{\partial \eta} i + \frac{\partial x_2(\xi,\eta)}{\partial \eta} j + \frac{\partial x_3(\xi,\eta)}{\partial \eta} k \qquad (5.11)$$

Ici, ***i,j*** et ***k*** forment des vecteurs unitaires orthogonaux du système de coordonnées global. Le produit vectoriel par définition nous donne un vecteur n^* (qui est égale à $n^* = r_\xi \times r_\eta$) qui est normal à la surface et ainsi cette opération nous donne également les composants, des cosinus directeur, le (n_1, n_2, n_3) du vecteur normal unitaire n (= n*/|n*|). Ces quantités sont nécessaires pour calculer le noyau T_{ij}. En deux dimensions, où l'intégration est effectuée sur un élément linéaire avec les coordonnées intrinsèques ξ, le même procédé s'applique. Ici le Jacobien représente le module du vecteur normal n* et est donné par la relation suivante :

$$J(\xi) = \left| n^* \right| \qquad (5.12.a)$$

avec

$$n^* = \frac{\partial x_2}{\partial \xi} i - \frac{\partial x_1}{\partial \xi} \qquad (5.12)$$

Par conséquent, les composantes du vecteur normal unitaire *n* sont donnée tout simplement par :

$$n_1 = \frac{1}{J(\xi)} \frac{\partial x_2}{\partial \xi}$$
$$n_1 = -\frac{1}{J(\xi)} \frac{\partial x_1}{\partial \xi} \qquad (5.13)$$

La formule qui donne la quadrature de Gauss pour une surface 3D peut être exprimée dans un système de coordonnées intrinsèque par l'équation :

$$I = \int_{-1}^{1}\int_{-1}^{1} f(\xi_1, \xi_2) d\xi_1 d\xi_2 = \sum_{i=1}^{m_1}\sum_{j=1}^{m_2} w_i^1 w_j^2 f(\xi_1^i, \xi_2^j) + E_1 + E_2 \qquad (5.14)$$

(ξ_1^i, ξ_2^i) sont les points de Gauss, (w_i^1, w_j^2) sont les poids, et (m_1, m_2)sont les ordres d'intégration, et E_1, E_2 sont les erreurs d'intégration dans les deux directions.

Le tableau suivant donne le nombre de points et poids de Gauss calculés par les relations (5.3) et (5.4) respectivement.

Tableau 4.1. Points et poids de Gauss (Stroud & Secrest, 1966)

Ordre ***(n)***	Ordonnées (x_k)	Poids ***(w_k)***
1	0	2
2	±0.57735 026920	1
3	0	0.88888 88889
	±0.77459 66692	0.55555 55556
4	±0.33998 10436	0.65214 51549
	±0.86113 63116	0.34785 48451
5	0	0.56888 88889
	±0.53846 931010	0.47862 86705
	±0.90617 98459	0.23692 68851
6	±0.23861 91861	0.46791 39346
	±0.66120 93865	0.36076 15730
	±0.93246 95142	0.17132 44924

5.3 Optimisation de la technique d'intégration

D'après ce qu'on a vu dans les paragraphes précédents, il est clair que le nombre de points de Gauss nécessaire pour obtenir des résultats fiables dépend directement de la distance entre le point ***p*** et le point ***Q***, mais comme nous allons le voir, il dépend aussi de la taille des éléments du maillage. Notre premier travail dans ce mémoire était de voir la variation de la distance entre les deux points ***p*** et ***Q,*** et le nombre de points de Gauss requis pour converger vers la solution exact. Ainsi nous pourrions formuler une équation reliant le nombre de points de gauss à la taille de l'élément et à la distance entre le point champs et le point source.

En implémentant cette formule dans le code source de KSP nous pourrions ainsi réaliser des gains important en temps de calcul. Pour cette fin nous avons lancé plusieurs calculs dans KSP en prenant comme application en premier temps des géométries 2D puis ensuite des cas plus complexes en utilisant des géométries de pièces industrielles. Ces calculs

ont permit de trouver une relation entre le nombre de points de gauss nécessaire à la convergence des calculs et les grandeurs qui sont directement liées, à savoir la taille de l'élément de maille et la distance entre le point source et le point champ.

La méthode « traditionnelle » utilisée dans KSP pour intégrer les noyaux consiste à prendre un nombre de points de Gauss égal à 30. Ce nombre reste constant quelque soit la distance entre les points *p* et *Q* et quelque soit la taille de l'élément considéré, un tel nombre (élevé) de points de gauss garantit l'exactitude des résultats avec une précision allant jusqu'à douze chiffres après la virgule, cependant cette méthode possède l'inconvénient d'être assez lente, en particulier dans le cas de gros calculs ou il est nécessaire d'intégrer les noyaux jusqu'à des millions de fois pour former le système à résoudre. Il serait donc avantageux de trouver une relation qui permette de donner un nombre de points de gauss minimum pour chaque élément considéré.

Notons que la préparation des modèles de géométries complexes à été réalisé avec le code commerciale IDEAS, Il est donc possible d'importer des géométries de différents logiciels de simulation à partir de KSP. Cette compatibilité avec les logiciels commerciaux de type IDEAS ou ABAQUS permet de gagner du temps et se révèle de ce fait assez maniable.

La figure 5.5 représente les différents paramètres influant sur le nombre de points de Gauss.

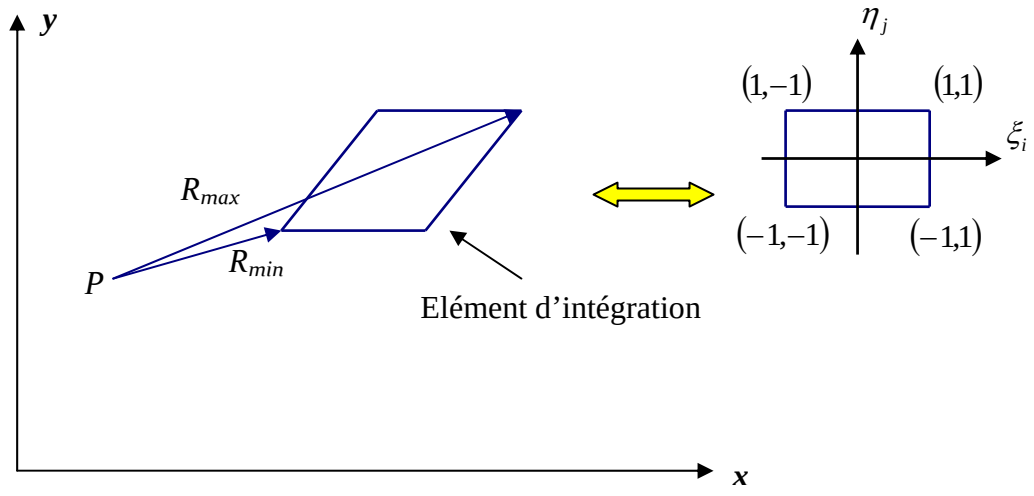


Figure 5.5 : représentation des paramètres influant le nombre de point de Gauss

Nous proposons une relation heuristique linéaire déterminée à partir simulations numériques sur des géométries simples en prenant en compte la variation des paramètres

influant sur le nombre de points de Gauss (taille de l'élément et distance entre les points *p* et Q). Ces simulations nous on permit de retrouver la relation suivante

$$N = 0.1\rho + 2.9 \qquad (5.15) \qquad \text{Avec} \qquad \rho = \frac{R_{\max}}{R_{\min}} \qquad (5.16)$$

N : nombre de points de gauss nécessaire à l'intégration.
Rmax : distance du nœud le plus éloigné d'un l'élément de maillage par rapport au point source.
Rmin : distance du nœud le plus proche d'un élément de maillage par rapport au point source.

Afin de réaliser des gains de temps important nous n'allons pas implémenter cette formule directement sur le code KSP, car cela risque de pénaliser le temps de réponse du code. Cependant, nous allons introduire cette formule sous forme de « paquets » d'éléments. Le schéma suivant représente l'organigramme d'optimisation du nombre de point de Gauss.

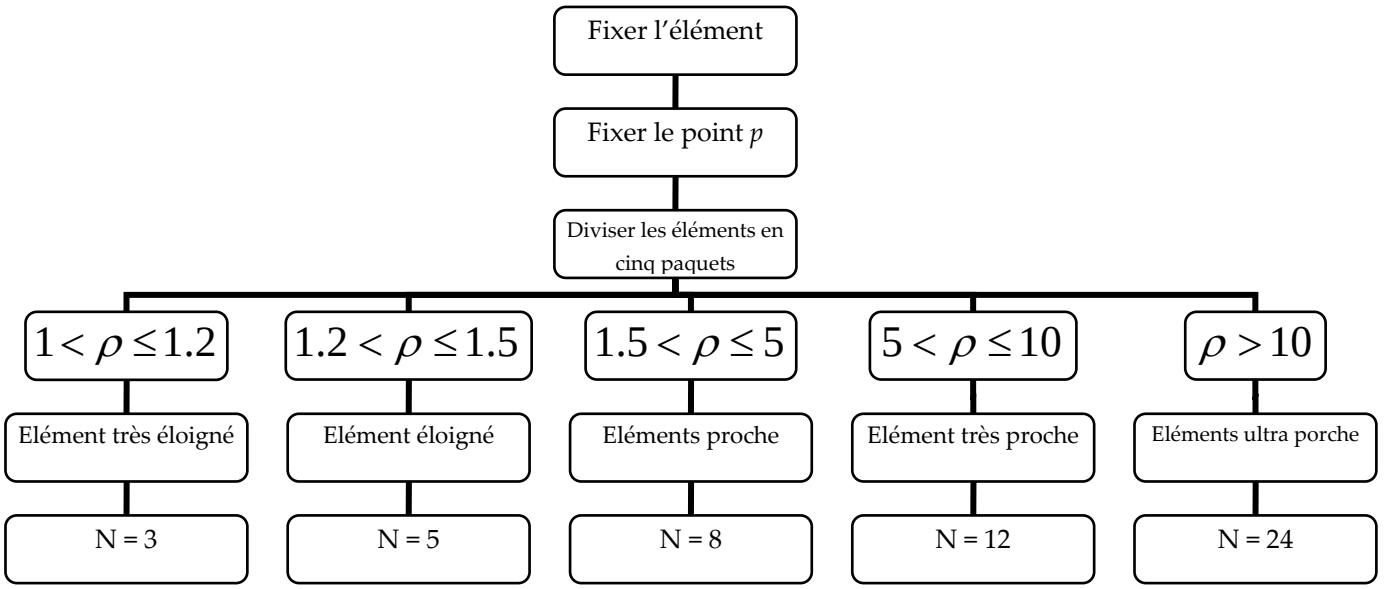


Figure 5.6 : Organigramme pour l'optimisation de l'intégration numérique

Dans ce qui suit on se propose de valider la formule (5.15), nous allons traiter des cas test afin de voir la variation du temps de calcul par rapport à la méthode classique. On se propose d'étudier pour commencer un cas simple qui consiste en un barreau cylindrique soumis à une traction simple sur son axe longitudinale comme on peut le voir sur la figure 5.7.

Afin d'analyser la diminution du temps de calcul en fonction de le taille des éléments du maillage nous prenons le même cas avec un nombre d'éléments différent comme le montre la figure 4.8

Cas test N°1 : barreau cylindrique sollicité en traction uni-axiale

Afin de valider le formule heuristique proposée pour le calcul du nombre de points d'intégration des noyaux réguliers (les points p et Q appartenant à deux différents éléments), on se propose de comparer les résultats obtenus par les deux méthodes (avec et sans formule heuristique) et ce par rapport à deux types de grandeurs importantes :

1. Les champs de contraintes et de déplacements obtenus
2. Les temps de calculs qui ont été nécessaires pour terminer le calcul.

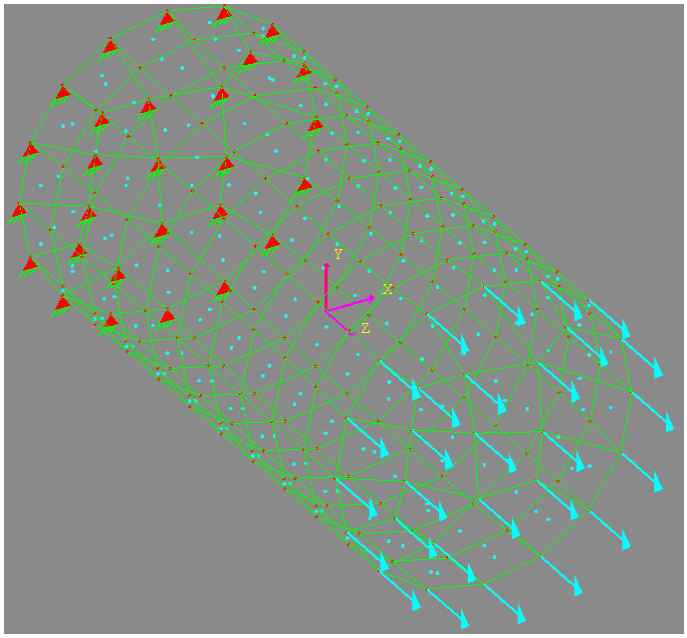
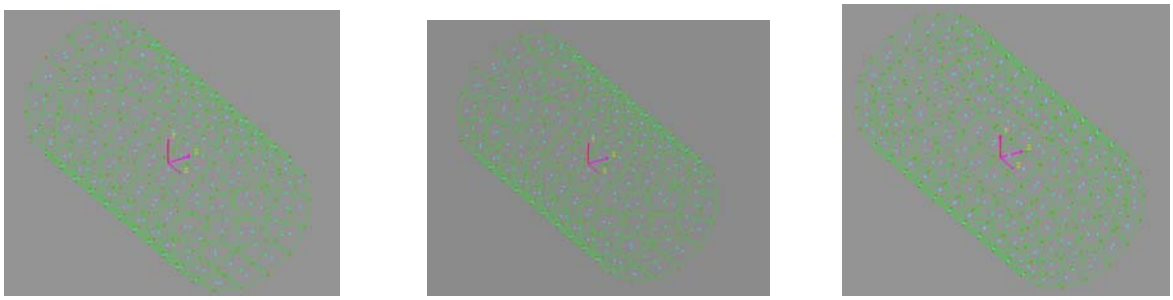


Figure 5.7 : Barreau cylindrique soumis à un effort de traction suivant l'axe z



Maillage à 256 éléments Maillage à 256 éléments Maillage à 880 éléments

Figure 5.8 : Différents maillage utilisés pour quantifier la variation du temps de calcul

Résultats cas test n°1

Le tableau suivant représente les résultats obtenus pour le premier cas test, qui on le rappel consiste en un barreau cylindrique soumis à un effort de traction. Ici les champs de déplacements suivant l'axe z suffisent pour pouvoir faire une appréciation sur la précision de la méthode proposée ci-avant.

Nombre d'éléments du maillage	198	256	880
Uzz Méthode normale	0.286331 (16.78 secondes)	0.284632 (28 secondes)	0.284620 (361.9 secondes)
Uzz Méthode optimisée	0.286318 **(3.25 secondes)**	0.284635 **(6.39 secondes)**	0.284619 **(95.92 secondes)**
Tnorm/Topti	**5.16**	**4.38**	**3.77**

Tableau 5.1 : représentation des résultats obtenus pour le premier cas test

Notons le fait que les calculs obtenus par KSP ont été validés que ce soit en élasticité, en plasticité ou en mécanique de la rupture (Gaiech & Kebir, 2007; Kebir & Roelandt, 1999) ici nous considérons les résultats de références comme étant les résultats obtenus par la méthode d'intégration normale, (nous n'aurons pas besoin de recourir à des solutions analytiques pour la comparaison des résultats obtenus).

Cas test N°2 : plaque trouée sollicitée en traction

Dans le deuxième cas test, on présente une plaque épaisse soumise à une contrainte uniforme sur sa face supérieur dans la direction de l'axe y et encastré en sa partie inférieur. Les forces volumiques sont nulles. La plaque contient trois trous circulaires alignés sur l'axe des x (figure 5.), et est supposée rester dans le domaine élastique avec un module d'Young E et un coefficient de Poisson ν. Le but est d'obtenir les champs de contraintes et de déplacements dans toute la plaque, et par la suite comparer les champs obtenus dans le cas de la méthode d'intégration normale et de la méthode d'intégration optimisée afin de valider cette dernière, par la suite nous dresserons un bilan des temps de calculs pour chaque méthode.

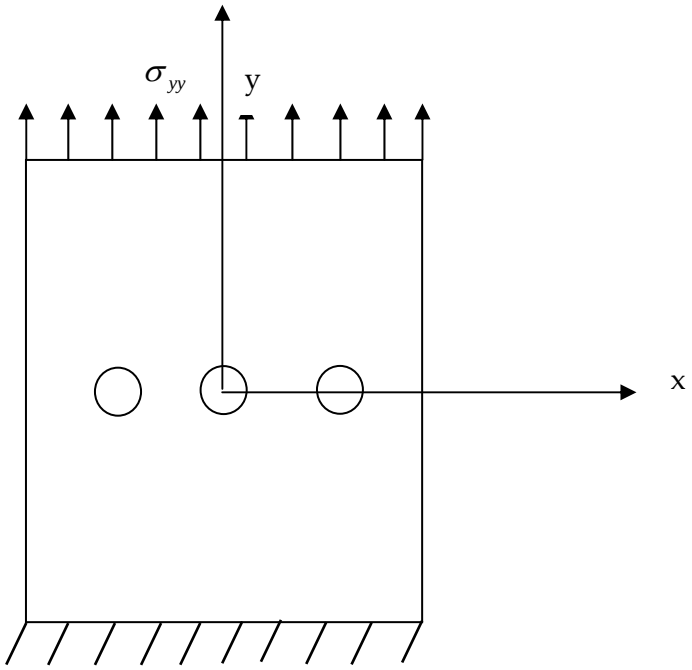
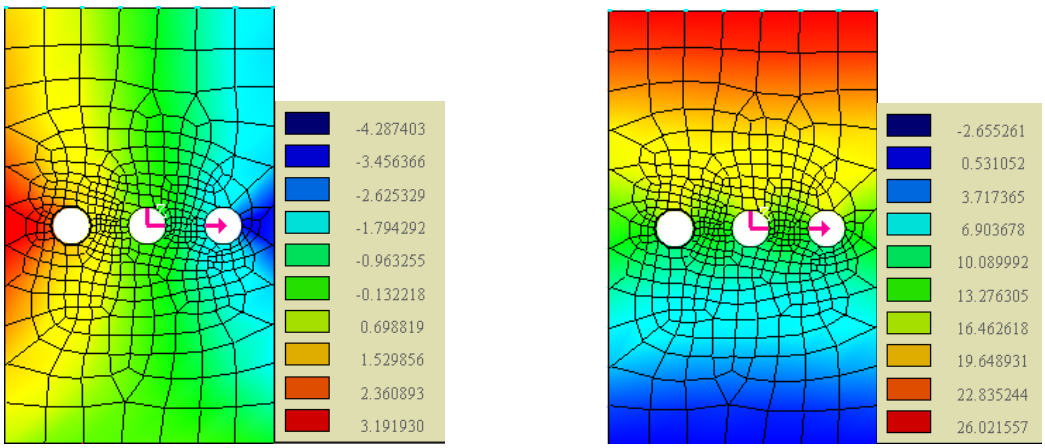


Figure 5.9 cas test n°2 : plaque trouée soumise un effort de traction

La figure suivante représente le champ de déplacements sur la plaque suivant les deux directions x et y.



(a) **(b)**

Figure 5.10 : représentation des champs de déplacements sur le cas test n°2

(a) : champs de déplacement sur l'axe des x ; **(b)** : champs de déplacement sur l'axe y.

La figure 5.5 représente les temps de calcul en secondes pour les deux méthodes étudiées, en considérant différents maillages avec un nombre d'éléments allant de 642 à 1555 éléments.

Nous illustrons ensuite dans la figure 5. la variation du rapport Tnorm/Topti pour différents nombre d'éléments.

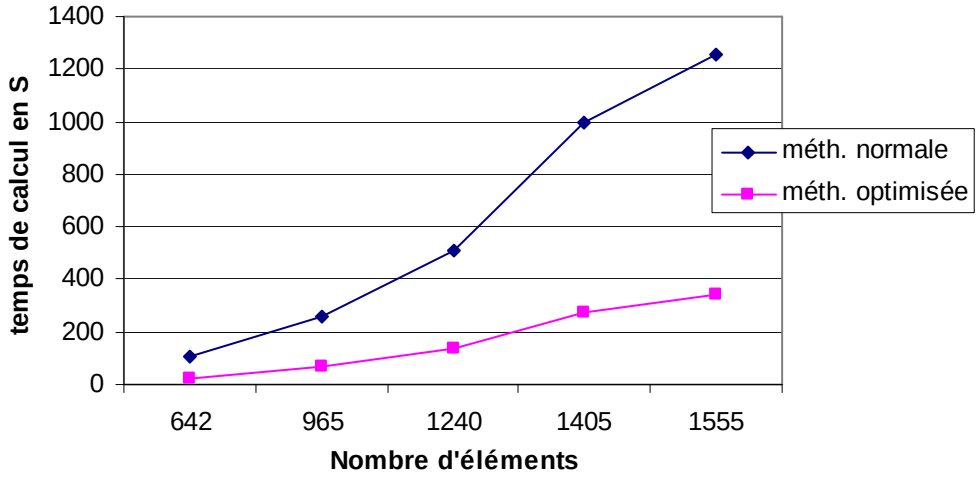


Figure 5.11 : temps de calculs pour différents maillages

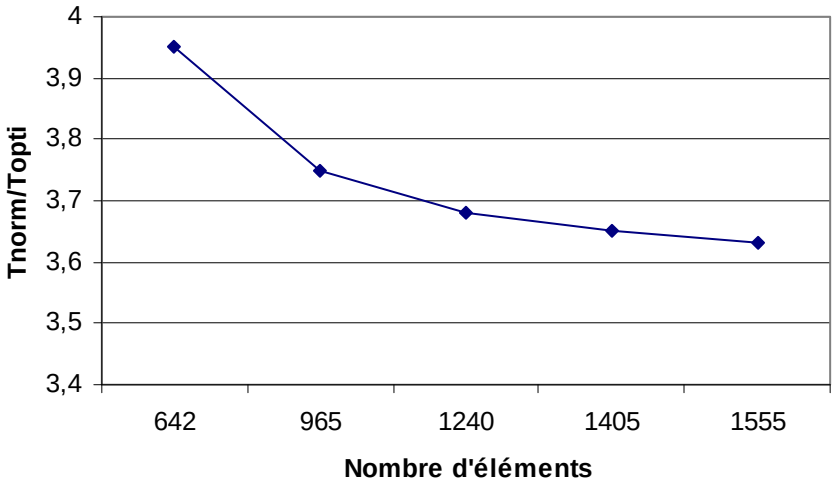
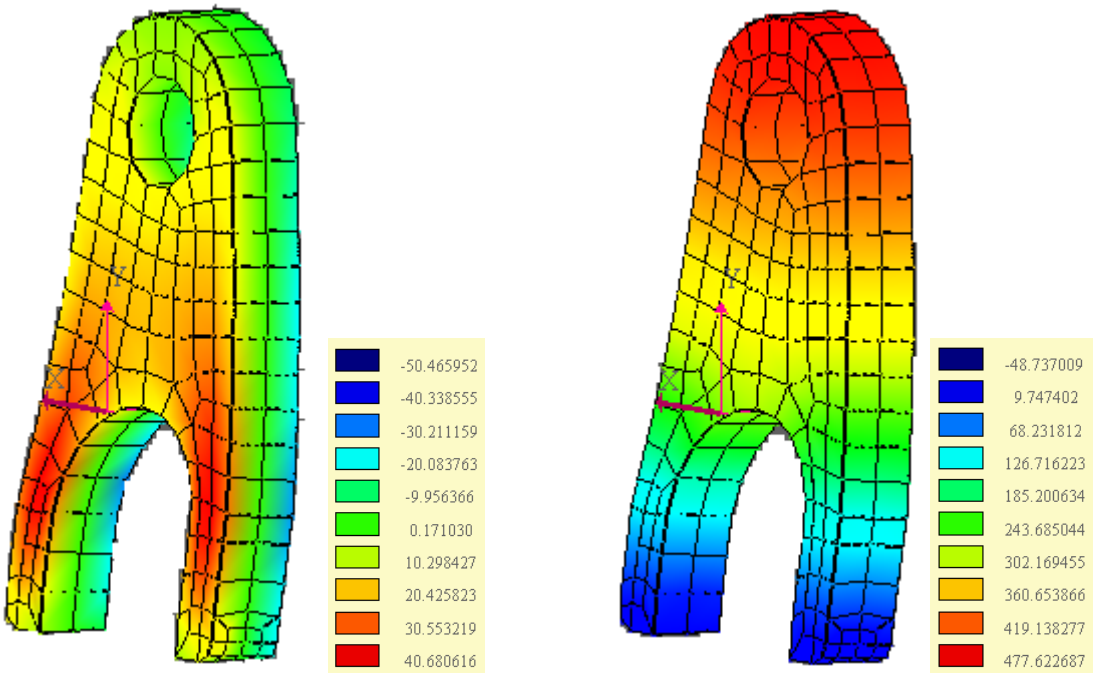


Figure 5.12: rapport des temps de calculs pour différents maillages

Cast test N°3: flexion d'une chape

Le troisième cas test représente une pièce industrielle en forme de chape encastrée sur ses deux extrémités et soumise à un chargement de flexion (figure 5.12). La pièce est supposée rester dans le domaine élastique avec un module d'Young E et un coefficient de Poisson ν .



(a) (b)

Figure 5.12 : rerpésentation des champs de déplacements pour chape sollicitée en flexion (a) : déplacement suivant l'axe des YY ; (b) : déplacement suivant l'axe ZZ (tangentiel au plan de la chape)

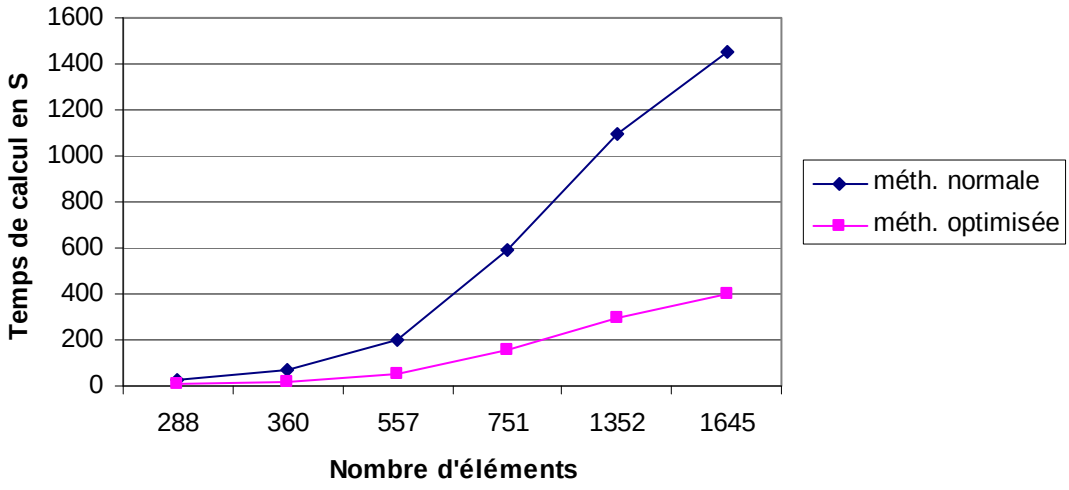


Figure 5.13 : temps de calculs pour différents maillages

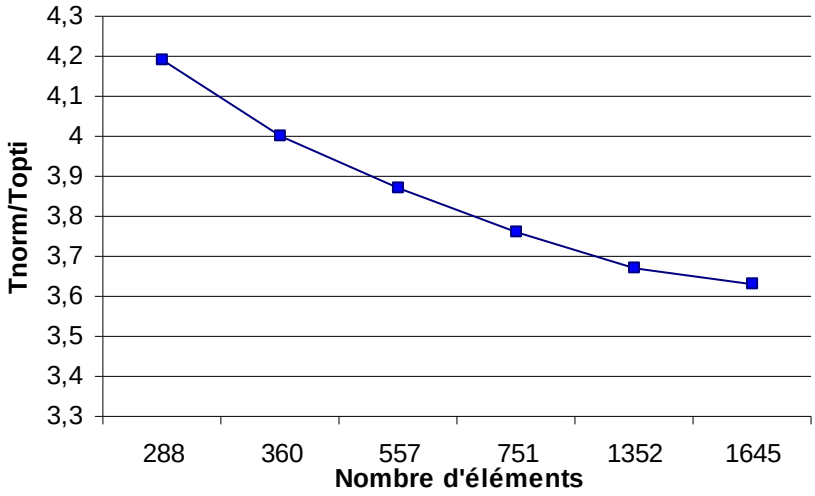


Figure 5.14: rapport des temps de calculs pour différents maillages

Dans la section qui suit nous allons discuter les résultats obtenus pour les différents cas test de validation que nous avons présenté auparavant.

5.4 Discussion :

Nous avons effectué trois cas de validation de la nouvelle approche permettant la réduction des temps de calcul par l'optimisation du nombre de points de Gauss. Rappelons que ce dernier est obtenu par une formule heuristique que nous avons proposée et qui prend en compte la taille des éléments du maillage et la distance entre le point source au point champ.

Le premier cas de validation consistait en un barreau cylindrique soumis à un effort de traction sur une des ces faces, et libre de rotation sur l'autre. Nous avons utilisé trois différents maillages de 198, 256 et 880 éléments pour chaque calcul. Les résultats obtenus en terme de champs de déplacements sont très proches (l'erreur se situe au niveau du cinquième chiffre après la virgule), en terme de temps de calcul, la méthode que nous proposons se révèle plus rapide, avec des rapports de temps de calcul (Tnorm/Topti) allant de 3.77 à 5.16 pour les différentes discrétisations géométriques considérées.

Les deux autres cas de validation traitent respectivement une plaque trouée en traction plane et une chape sollicitée en flexion. De façon analogique au premier cas test, nous avons considérés, plusieurs discrétisations géométriques afin de prendre en compte l'effet du nombre d'éléments sur la variation du temps de calcul pour les deux méthodes (normale et optimisée). Les figures 5.11 et 5.13 représentent respectivement la variation des temps de calculs en fonction du nombre d'éléments pour le deuxième et troisième cas test. Pour un

nombre d'élément allant de 288 à 1645 éléments, le rapport Tnorm/Topti (rapport entre le temps de calcul par la méthode normale sur le temps de calcul par la méthode optimisée) varie entre 4.2 et 3.62, ces résultats se révèlent assez satisfaisant, notons aussi que les courbes sur les figures 5.12 et 514 ont tendance à se stabilisé avec l'augmentation du nombre d'éléments.

Deuxième partie : Implémentation d'un élément conforme

Résumé

Dans cette deuxième partie du chapitre cinq, nous allons implémenter des éléments conformes dans le code KSP toujours afin de réduire au maximum les temps de calculs. La principale difficulté réside ici réside dans le calcul du terme cij de l'équation intégrale de frontière. Nous commençons donc par donner une approche pour le calcul de ce terme. A la fin de ce chapitre, on illustrera quelques cas test pour valider notre approche.

5.5 Introduction

Dans cette deuxième partie du chapitre cinq, nous proposons d'implémenter des éléments conformes (triangles à trois nœuds) dans le code KSP, toujours dans un but de minimiser les temps de calculs. Nous présentons donc ici le type d'éléments utilisés pour des problèmes de mécanique de la rupture, à savoir les éléments non-conformes, puis les nouveaux éléments implémentés dans le cadre de ce travail, on terminera par la validation de ces éléments par des cas test afin d'illustrer la performance de ces derniers.

5.6 Discontinuités des tractions aux points situés sur les bords et arrêtes

Quand un nœud est situé en un point où la frontière n'est pas plane, c.-à-d les bords pour des problèmes bidimensionnels ou les bords et arrêtes dans des cas tridimensionnels, des discontinuités dans les tractions vont apparaitre en ce nœud résultant des forts gradients du vecteur normal en ces endroits. Ceci implique que si les tractions nodales sont inconnues le nombre d'équations en ce nœud va être inférieur au nombre d'inconnus, on sera donc confronté à un problème mal posé.

Afin d'expliquer ce qui se produit au niveau de ces zones, considérons un coin bidimensionnel pour la simplicité (figure 5.15). Quand les tractions sont connues sur les deux côtés du nœud faisant le coin, seul les deux composantes des déplacements nodaux sont inconnues et aucun traitement spécial du nœud faisant le coin n'est exigé. Il peut également se produire, qu'un déplacement et une des tractions « avant » ou « après » le nœud soit connue; puis l'autre traction « après » ou « avant » le nœud soit inconnu. Dans ce cas le problème est résolu sans difficultés comme nous le verrons dans la section suivante.

Cependant, quand deux valeurs différentes (« avant » et « après » le nœud) de n'importe quel composant de la traction sont inconnues et que seul le déplacement est connut un traitement spécial du coin (ou de l'arrête) serra exigé.

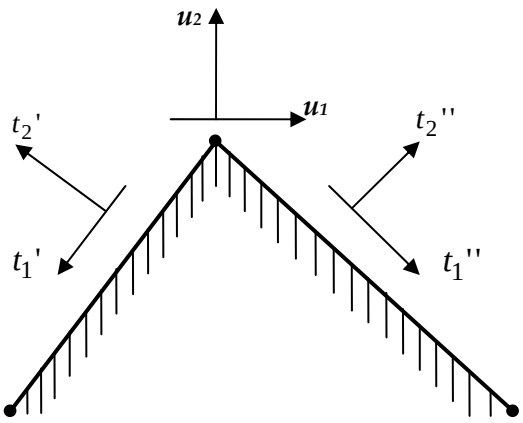


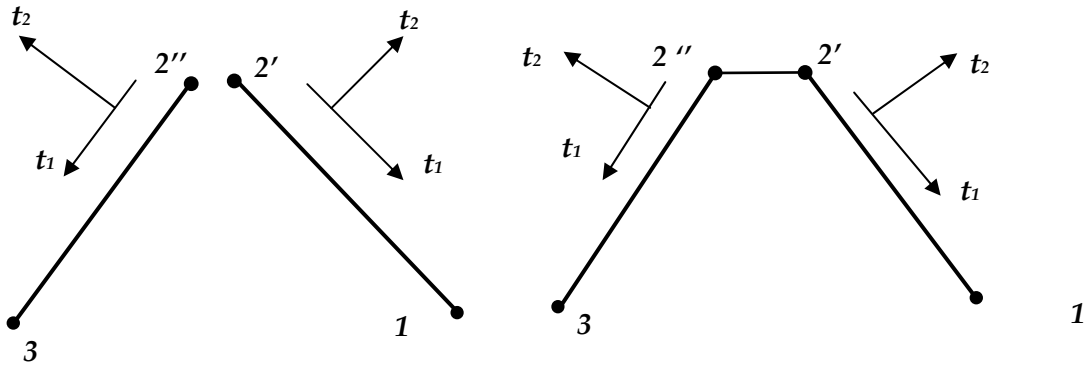
Figure 5.15 coin possédant un seul nœud en son sommet

Plusieurs techniques, connues aussi sous le nom de « treatement of corner and edges technics » ont été élaborées et on se propose d'expliciter leurs principes, on abordera par la suite le calcul de l'angle solide au niveau de ces régions.

1. la géométrie du coin ou de l'arrête est tout simplement arrondie par un élément courbé, cette technique a été proposé par Jaswon et Symm (Jaswon M.A & Symm G.T, 1977), et permet d'avoir des résultats même si ces dernier sont tout à fait discutable. Dans le cas de problèmes multi-domaines, cette technique est unanimement impossible à mettre en place (Becker A.A, 1992); (Gao X.W & Davies T.G, 2002).

2. la deuxième technique appelée méthode des éléments discontinus propose d'utiliser deux nœuds au niveau du coin où de l'arrête avec un petit espacement ε entre les deux nœuds comme le montre la figure (5.16.a). Cependant, si la valeur de ε est trop petite (de l'ordre de l'erreur commise par la machine), alors des erreurs numérique vont apparaitre et la précision de calcul sera fortement touchée.

3. la troisième technique est connue sous le nom de « small corner approach », dans cette méthode, un petit élément est introduit au niveau de du coin où de l'arrête comme le montre la figure (1.16.b), cependant on constate les mêmes inconvénients que la technique précédente au niveau de la dépendance de l'erreur par rapport à la taille de l'élément.



(a) Méthode des éléments discontinus (b) « Small corner approach »

Figure 5.16 séparation des nœuds de bords

4. la méthode la plus récente et la plus élaborée est connue sous le nom de la méthode de dédoublement nodal (Zhang & Mukherjee, 1991); (Wilde, 1998); (Gao & Davies, 2000.a) et consiste à introduire des nœuds additionnels au niveau des bords et des arrêtes (figure 5.17), on développe ainsi des équations additionnelles auxiliaires pour déterminer les inconnus additionnels. Sion définit $\tilde{N}$ comme étant le nombre de nœuds additionnels, alors en 3D, $3\tilde{N}$ équations additionnelles doivent êtres établies.

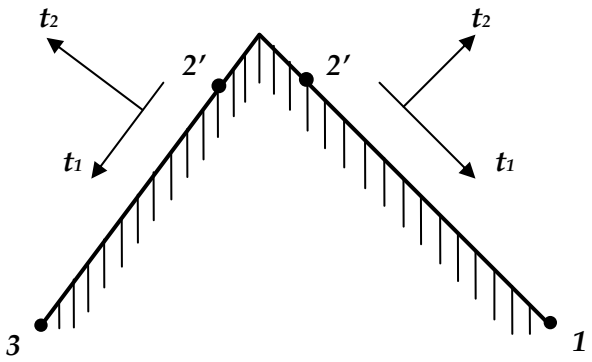


Figure 5.17 : coin avec un élément non-conforme

Dans cette approche, on commence par considérer l'équation d'équilibre comme suit

$$t_i = \sigma_{ij} n_j \qquad (5.15)$$

Dans l'équation précédente les n_j sont les composantes du vecteur normal. En se basant sur le système d'axe local définit précédemment, nous introduisons un système de coordonnées cartésien local définit par x_i' avec les axes x_1' et x_2' tangentiels à l'élément et x_3' pris dans la direction normale à ce dernier. Les quantités globales, coordonnés et tractions peuvent êtres transférées dans le système de coordonnées local par la formule suivante :

$$\begin{aligned} x_i^{'} &= L_{ij} x_j \\ t_{'i} &= L_{ij} t_j \end{aligned} \qquad (5.16)$$

Les variable en primes nous donnes des grandeurs par rapport au repère local et le tenseur de transformation est donné par :

$$L_{ij} = \frac{\partial x_i^{'}}{\partial x_j} \qquad (5.17)$$

À partir de ces équations nous obtenons immédiatement la relation suivante :

$$\begin{aligned} \frac{\partial t_k^{'}}{\partial x_k^{'}} &= L_{ki} \frac{\partial \sigma_{ij}}{\partial x_k^{'}} n_j \\ &= \frac{\partial \sigma_{ij}}{\partial x_i} n_j \end{aligned} \qquad (5.18)$$

À partir des équations d'équilibre, nous obtenons directement les relations suivantes :

$$\frac{\partial \sigma_{ij}}{\partial x_i} = 0 \qquad (5.19)$$

Il s'ensuit que :

$$\frac{\partial t_k^{'}}{\partial x_k^{'}} = 0 \qquad (5.20)$$

Cette équation auxiliaire nous permet de relier les composantes tangentielles (k=1 et 2 seulement) du vecteur contraintes, et reste valable quand même les champs de contraintes sont discontinus. Il est facile de remarquer que cette équation peut aussi provenir d'un simple remaniement de l'équation d'équilibre écrite dans le repère local. Maintenant, réécrivons l'équation (5.19) en termes de quantités globales, et des valeurs nodales du vecteur contrainte. La relation de transformation (5.20) permet de passer du repère cartésien local à un repère curviligne local définit par les variables ξ_i , ceci permet d'écrire que :

$$\frac{\partial t_k^{'}}{\partial x_k^{'}} = L_{pi} \frac{\partial t_i}{\partial \xi_k} \frac{\partial \xi_k}{\partial x_p^{'}} \qquad (5.21)$$

Avec L_{pi} ($p=1,2$) représente les cosinus directeurs du système d'axes local, x_1' , x_2' respectivement (définit dans l'équation 5.16). Maintenant, si nous introduisant la relation $t_i = N_i t_i^{\alpha}$, nous obtenons :

$$\frac{\partial \xi_k}{\partial x_p^{'}} L_{pi} \frac{\partial N_{\alpha}}{\partial \xi_k} t_i^{\alpha} = 0 \qquad (5.22)$$

Avec t_i^{α} étant la ième composante du vecteur contrainte au nœud α . Cette équation peut facilement être implémentée sur un code éléments de frontières. L'approche illustrée ci-dessus présente un nombre d'équations auxiliaires suffisant pour le cas 2D, cependant ce n'est pas le cas en 3D. Dans (Gao & Davies, 2000), les auteurs proposent de « dériver » une équation supplémentaire en faisant la supposition que le tenseur de contraintes est continu sur un coin ou une arrête, ce qui n'est pas forcément vrai, mais qui par contre donne des résultats intéressant et acceptables. La méthode est explicitée dans ce qui suit :

Soit un bord formé par l'intersection de deux surfaces S_a et S_b (possédant respectivement des normales unitaires n^a et n^b), la multiplication de l'équation d'équilibre par le vecteur normal unitaire de chaque surface nous donne :

$$\begin{aligned} n_i^b t_i^a &= n_i^b \sigma_{ij} n_j^a \\ n_i^a t_i^b &= n_i^a \sigma_{ij} n_j^b \end{aligned} \qquad (5.23)$$

La première expression des équations ci-dessus sont égales du fait que le tenseur de contraintes est symétrique. Donc nous pouvons écrire l'égalité suivante :

$$n_i^a t_i^b = n_i^b t_i^a \qquad (5.24)$$

Bien que les équations intégrales de frontière fournissent seulement neuf équations, six équations, plus trois équations auxiliaires provenant de la relation (5.21), trois autres équations sont disponibles à partir de la relation (5.23). L'expérience montrent que les premières (relation 5.21) offres de meilleurs résultats parce qu'elle évite l'hypothèse de continuité de la contrainte entre les éléments ce qui se révèle une contrainte assez restrictive. Le nombre d'équations auxiliaires qui doit être soit appelé aux coins et aux bords

$$m = d \times (n-2) \qquad (5.25)$$

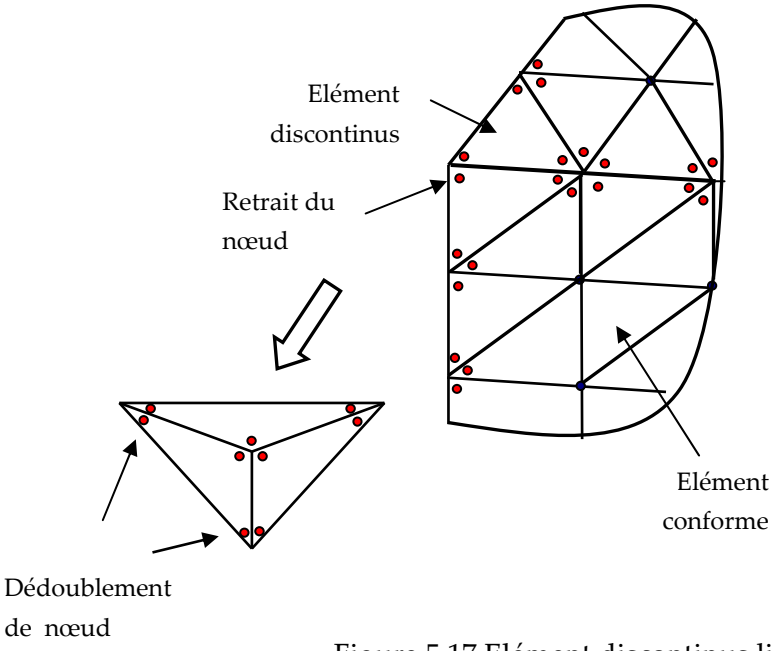


Figure 5.17 Elément discontinus linéaire

5.7 Calcul du terme c_{ij} de l'équation intégrale de frontière

Pour des frontières planes le terme c_{ij} présenté au chapitre deux est simplement une matrice diagonale avec une valeur 0.5 sur la diagonale. Cependant quand le point p est sur un coin ou une arrête (Figure 4.6) la limite des tractions fondamentaux de solution, c.-à-d. (équations (2.29) et (2.26) du chapitre 2)

$$c_{ij} = \delta_{ij} + \beta_{ij}(p) \qquad (5.26)$$

Avec

$$\beta_{ij}(p) = \lim_{\varepsilon \to 0} \left\{ \int_{\Gamma_2} T_{ij}(Q, p) d\Gamma(Q) \right\} \qquad (5.27)$$

Pour des problèmes bidimensionnels le terme $\beta_{ij}(p)$ est calculé tout simplement à partir de l'angle que fait le coin, et peut être calculé par la formule suivante (Brebbia, Telles, & Worbel, 1984).

$$\beta_{ij} = \frac{-1}{8\pi(1-\nu)} \begin{bmatrix} 4(1-\nu)(\pi + \theta_2 - \theta_1) + \sin 2\theta_1 - \sin 2\theta_2 & \cos 2\theta_2 - \cos 2\theta_1 \\ \cos 2\theta_2 - \cos 2\theta_1 & 4(1-\nu)(\pi + \theta_2 - \theta_1) + \sin 2\theta_1 - \sin 2\theta_2 \end{bmatrix}$$

Il est beaucoup plus complexe d'obtenir une expression générale de β_{ij} en trois dimensions puisque la discontinuité du coin peut être de différents types. Rappelons aussi que nous n'avons pas le droit d'utiliser la technique du mouvement de corps rigide comme énoncé précédemment. Dans ce qui suit nous allons faire de simples développements géométriques en se basant sur les travaux de Mantic (Mantic, 1995), afin de présenter un approche qui permette de calculer l'angle solide. Nous écrivons ensuite cette approche sous forme de programme informatique afin de l'introduire dans le code KSP.

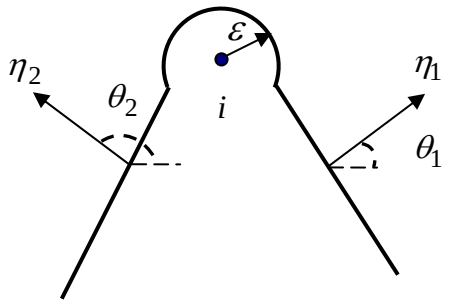


Figure 5.18 nœud de coin

5.8 Eléments conformes versus Eléments non-conformes

Nous présentons ici deux différents types d'éléments, les éléments non-conformes implémentés dans KSP (CT3N3 par exemple). Ce type d'éléments représentés sur la figure (5.19.b) possède des nœuds collocations différents des nœuds géométriques. Ces éléments permettent d'éviter le calcul du terme libre qui en trois dimensions peut se révéler compliqué. Cependant, les éléments non conformes coutes chère en temps de calcul pour la

simple raison qu'ils utilisent un plus grand nombre de point de collocation comparé aux éléments conformes. Les éléments conformes quand à eux possèdent leurs nœuds de collocation superposés aux nœuds géométriques (figure 5.19.a)

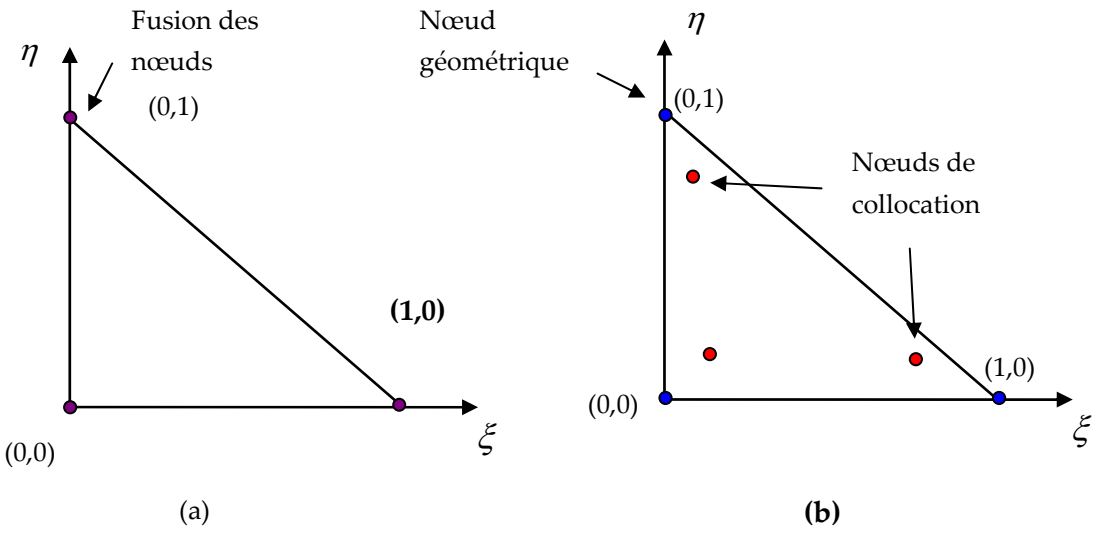


Figure 5.19 Représentation d'un élément triangulaire conforme (a) et d'un élément triangulaire non-conforme (b)

Dans ce qui suit nous présenterons d'abord l'implémentation des éléments non conformes, ensuite celle des éléments conformes.

5.9 Calcul de l'angle solide

L'angle solide est le rapport entre la surface en rose (figure 6.) de la projection d'un objet sur une sphère et le carré du rayon de celle-ci. Ici, l'objet dont est mesuré l'angle solide est une surface quadrilatère (en bleu).

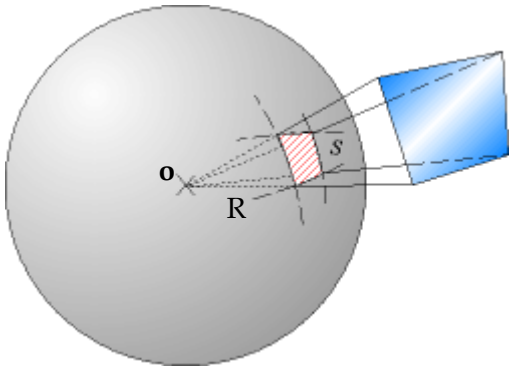


Figure 5.20 représentation de l'angle solide

Pour calculer l'angle solide sous lequel on voit un objet à partir d'un point donné, on projette l'objet sur une sphère de rayon R centrée en ce point. Si la surface que cette projection fait sur la sphère est S, l'angle solide sous lequel l'observateur voit l'objet est par définition :

$$\Omega = \frac{S}{R^2} \qquad (5.28)$$

Pour une sphère de rayon r, l'angle solide est défini pour un élément de surface élémentaire dS, c'est-à-dire engendré par des variations angulaires infinitésimales des zénith θ et azimut ϕ (la surface élémentaire est assimilée à un plan) :

$$dS = rd\theta.r\sin\theta d\theta = r^2.d\theta\sin\theta d\phi \qquad (5.29)$$

D'où :

$$d\Omega = \sin\theta d\phi.d\theta \qquad (5.30)$$

Par intégration dans les domaines angulaires des coordonnées sphériques et en notant θ' et ϕ' les variables d'intégration :

$$\Omega = \int_0^{2\pi} d\phi' \int \sin\theta' d\theta' = 2\pi\int_0^{\alpha} \sin\theta' d\theta' = 2\pi\left[-\cos\theta'\right]_0^{\alpha} = 2\pi(1-\cos\alpha) \qquad (5.31)$$

Ceci définit un cône d'angle $\Omega = 2\alpha$ et de base A.

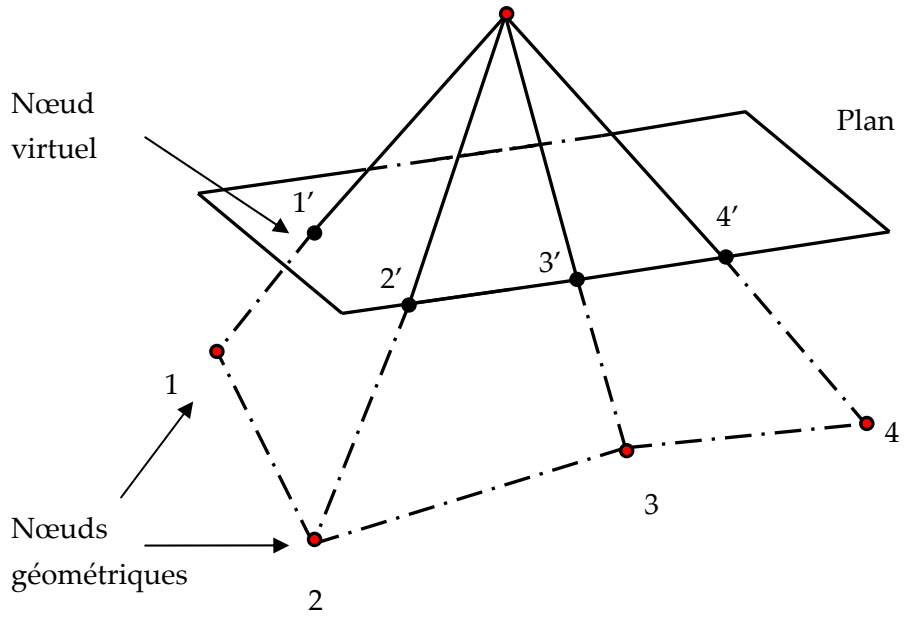


Figure 5.21-a : représentation de l'approche adoptée pour le calcul de l'angle solide

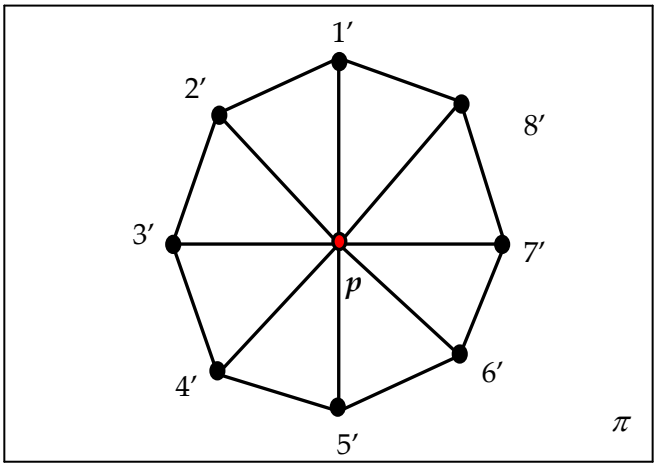


Figure 5.21-b : représentation de l'approche adoptée pour le calcul de l'angle solide

Cependant dans notre cas il est difficile d'implémenter cette formule directement dans le logicielle KSP pour la simple raison que notre maillage constitué d'éléments triangulaires à trois nœuds et peut être de ce fait un maillage de forme quelconque.

Dans le cas d'un maillage quelconque, la distance entre le point $\boldsymbol{p}$ et les autres nœuds appartenant au même élément que le point $\boldsymbol{p}$ n'est pas la même, cependant le nœud coupés par le plan π (figure 5.21-a, b.) sont à la même hauteur.

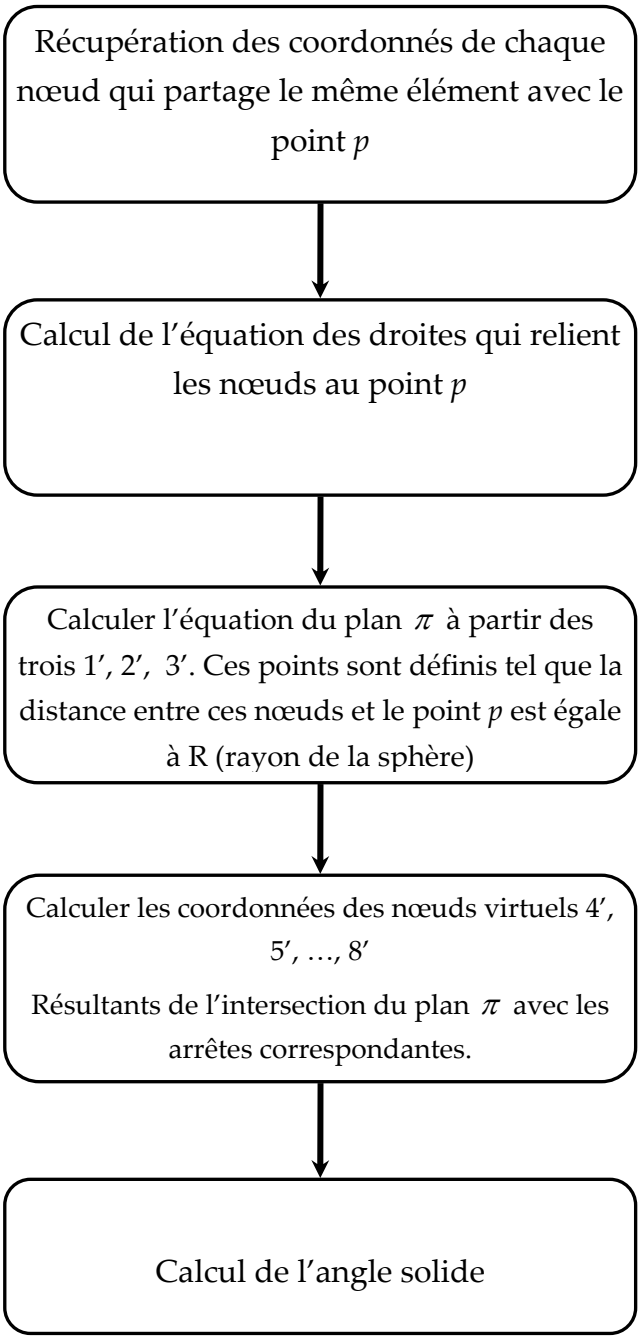


Figure 5.23 : organigramme de calcul de l'angle solide programmé dans KSP.

La stratégie employée pour le calcul de l'angle solide est illustrée dans l'organigramme de la figure 5.23.

Le programme de calcul de l'angle solide est donné en Annexe

5.10 Formulation d'un élément conforme dans KSP

Nous somme maintenant en mesure de calculer l'angle solide en 3D, ce terme doit être calculé avant la résolution du système. Nous pouvons maintenant introduire les éléments conformes dans le code KSP sous forme d'une classe qui dérive de la classe **ElementFrontières**

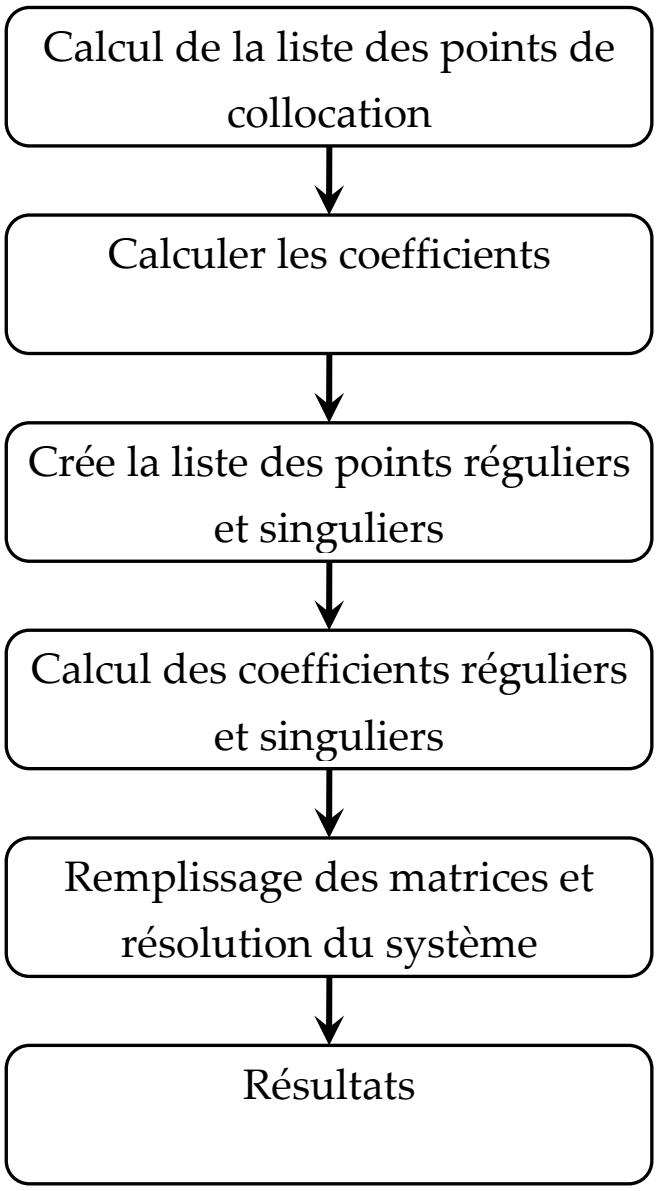


Figure 5.24 : organigramme pour l'implémentation des éléments conformes

La figure précédente représente l'organigramme de l'implémentation des éléments conformes dans KSP,

5.11 Validation des éléments conformes

Dans cette section nous passons à la validation des éléments conformes. Par analogie à la partie intégration adaptative de ce chapitre, nous commençons par un cas une géométrie simple

5.11.1 Cas test N°1 : plaque trouée sollicitée en traction

Dans ce premier cas test, nous reprenons l'exemple considéré dans la première partie du chapitre 5, à savoir le cas test constitué d'une plaque trouée soumise à un chargement uniformément répartie sur sa face supérieur, la plaque est encastrée sur sa partie inférieure. Le maillage utilisé est un maillage triangulaire à trois nœuds, le but ici est de comparer les temps de calculs obtenus par les éléments triangulaire non-conformes et conformes.

Le tableau suivant illustre les résultats obtenus en terme de champs de déplacements suivant l'axe de sollicitation (zz), aussi nous donnons les temps de calculs qu'il a fallu pour aboutir aux résultats

Nbr d'éléments	674	885	1060	1378	1546	1880
Uyymax Mét. Normale	5.254628	5.254610	5.254573	5.254524	5.254524	5.254524
Uyymax Mét. Optimisée 1	5.254745	5.254230	5.254910	5.254621	5.254575	5.254510
Uyymax Mét. Optimisée 2	5.265402	5.264738	5.24440	5.259875	5.259829	5.258713
Temps (s) Mét. Normale	320.260	420.36	784.276	1416.94	2148.34	3080.045
Temps (s) Mét. Optimisée 1	**17.793**	**25.100**	**49.325**	**90.829**	**139.50**	**211.541**
Temps (s) Mét. Optimisée 2	**9.42**	**15.56**	**33.374**	**62.975**	**98.097**	**146.66**
Tnorm/Topti1	**18**	**16.8**	**15.9**	**14.9**	**14.1**	**13.9**
Tnorm/Topti2	**28**	**25.1**	**23.05**	**21.53**	**20.9**	**20.13**

Tableau 5.2 : résultats obtenus par les trois méthodes (normale, avec éléments conforme et combinée) pour la plaque en traction simple

Mét. Optimisée 1 : calcul utilisant les éléments conformes ;

Mét. Optimisée 2 : calcul qui combine intégration adaptative et les éléments conformes ;

Tnorm/Topti1 : rapport des temps de calculs de la méthode normale sur la méthode utilisant les éléments conformes ;

Tnorm/Topti2 : rapport des temps de calculs de la méthode normale sur la méthode utilisant les éléments conformes et l'intégration adaptative.

5.11.2 Cas test n° 2 : porte-outils

Dans ce deuxième cas test, nous étudions une pièce représentant un porte-outil utilisé dans les machines-outils de type tournage, fraisage. Le tournage (fraisage) est un procédé de fabrication par coupe (par enlèvement de matière) mettant en jeu des outils à arrête unique (multiples). La pièce est animée d'un mouvement de rotation (mouvement de coupe), qui est le mouvement principale du procédé.

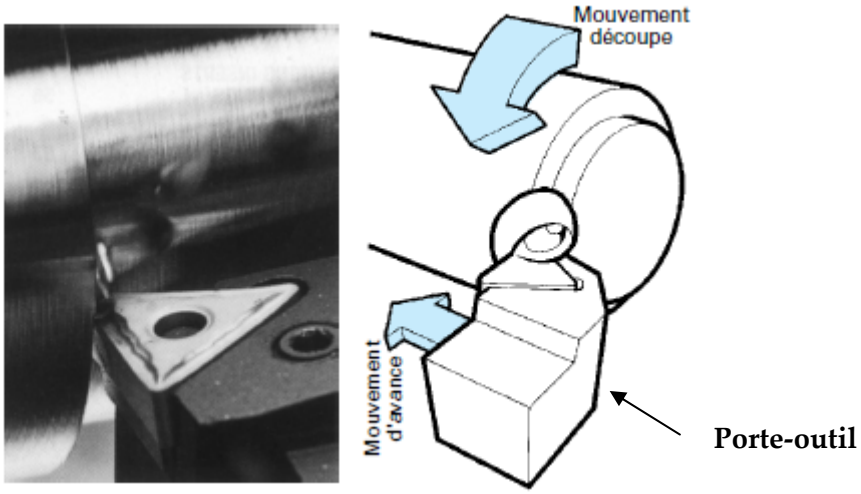
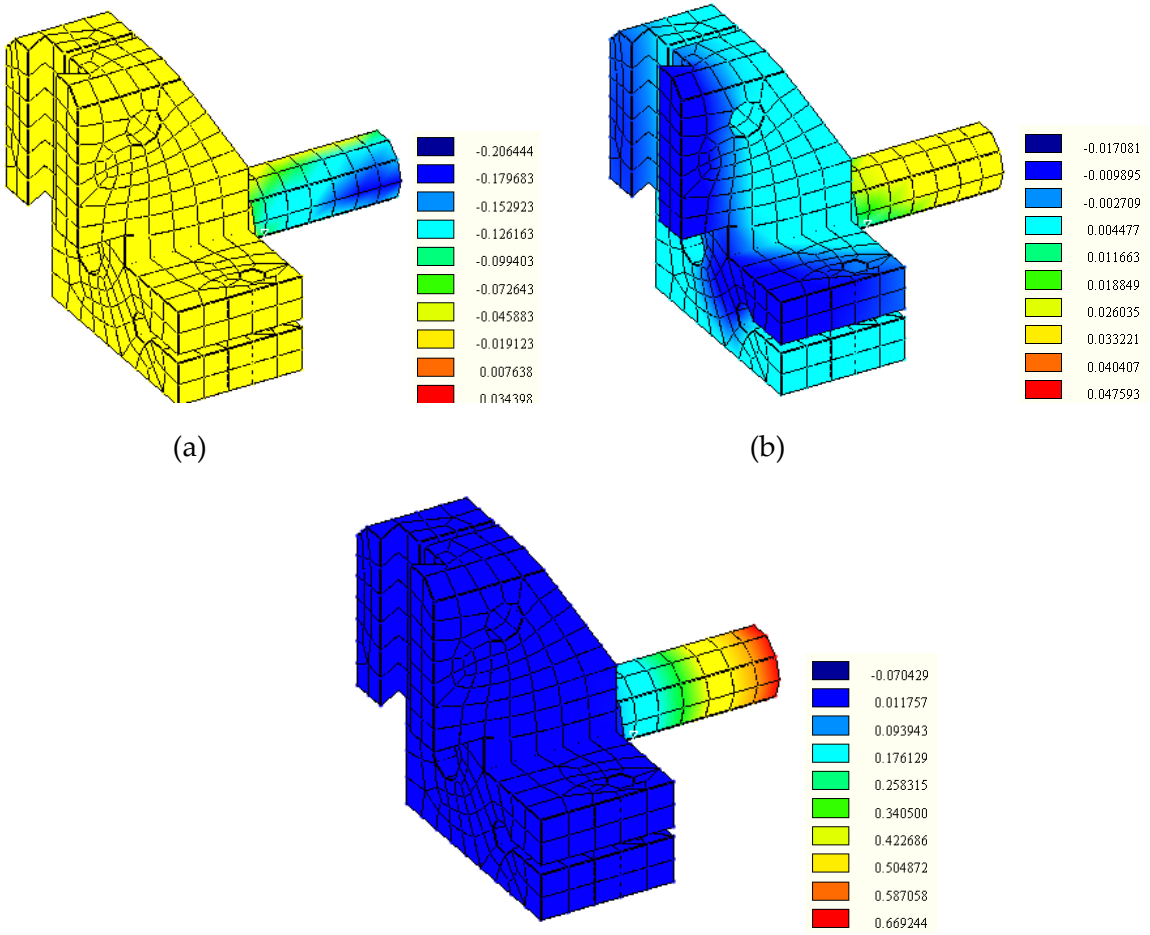


Figure 5.25 : Tournage avec arrête unique (selon Passeron A, 2006)

La figure 5. Illustre la cinématique du procédé, nous pouvons déduire ainsi les efforts repris par l'outil. L'outil est soumis à divers contraintes dynamiques, qui sont constitués principalement d'efforts de compression dus à sa pénétration dans la matière. Il aussi est soumis des efforts de cisaillement dus au contact avec la surface de la pièce. Dans notre test de validation, nous nous mettons dans les hypothèses d'un chargement quasi-statiques afin de simplifier la modélisation, car l'objectif étant de connaitre la contribution des éléments conformes sur la réduction des temps de calculs.

Le porte-outil reprend les efforts transmis par l'outil lui-même et les achemine vers les fondations via la broche. De ce fait, la pérennité et le bon fonctionnement de la machine-outil passe aussi par celle du porte-outil.

La figure suivante montre la distribution des champs de déplacement sur le porte-outil soumis aux charges de services :



(a) (b)

(c)

Figure 5.26 : représentation des champs de déplacements dans le cas d'un porte-outils soumis à chargement de service

(a) : uxx ; (b) : uyy ; (c) : uzz.

(b)

Pour plus de clarté et de visibilité des données nous donnons la représentation des résultats obtenus pour les deux méthodes optimisées sur deux figures (5.27 et 5.28) afin de mieux comparer la combinaison des deux méthodes discutés plus haut. La figure suivante montre la distribution des champs de déplacement sur le porte-outil soumis aux charges de services :

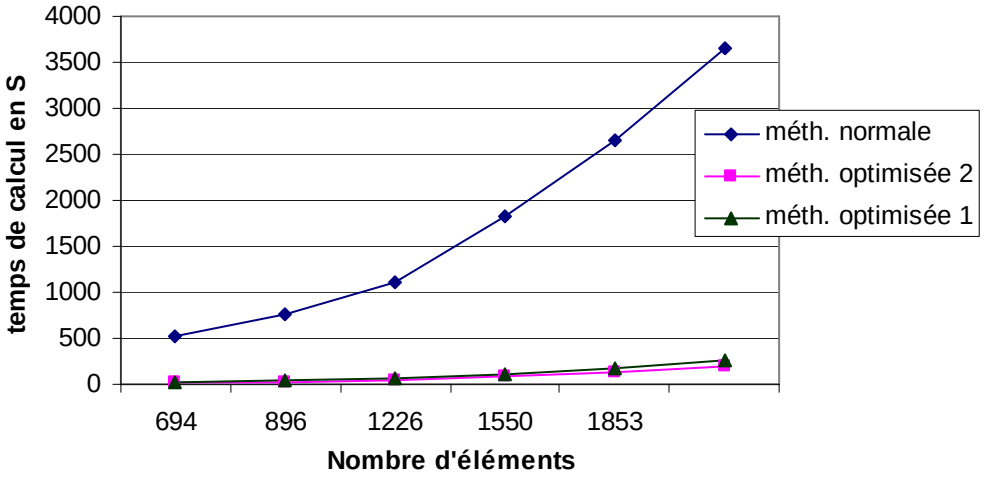


Figure 5. 27 : variation des temps de calculs pour les trois méthodes considérées

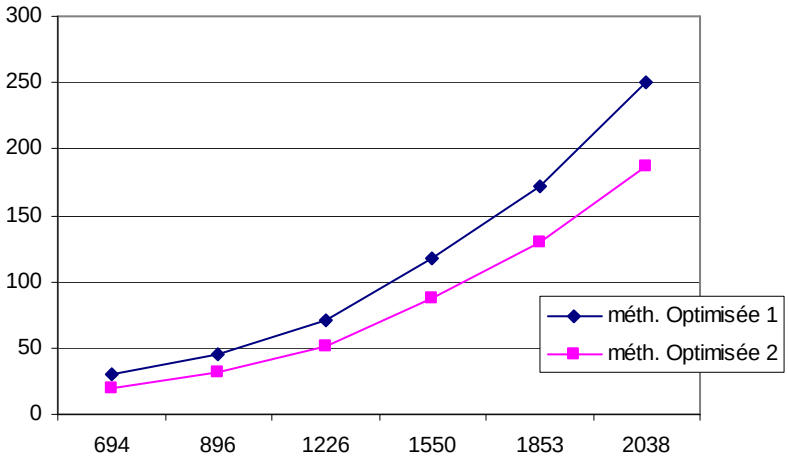


Figure 5.28 : variation des temps de calculs dans le cas éléments conformes et combiné

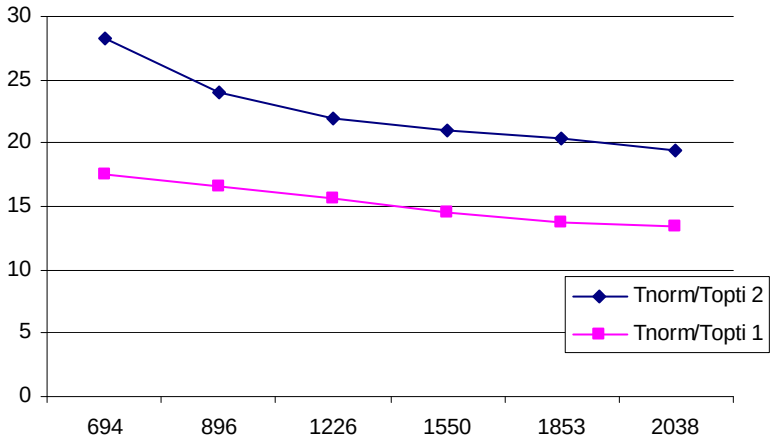


Figure 5.29 : rapport des temps de calculs pour le cas éléments conformes et combiné

5.11.3 Cas test n°3 : comportement mécanique simplifié d'un fémur lors de la marche

La modélisation numérique (le plus souvent par éléments finis) a été intégrée à la recherche en biomécanique pour sa capacité à reproduire le comportement d'un os, d'une articulation ou d'un implant, et évaluée comme alternative aux expérimentations in-vitro, couteuses et parfois difficiles a mettre en place.

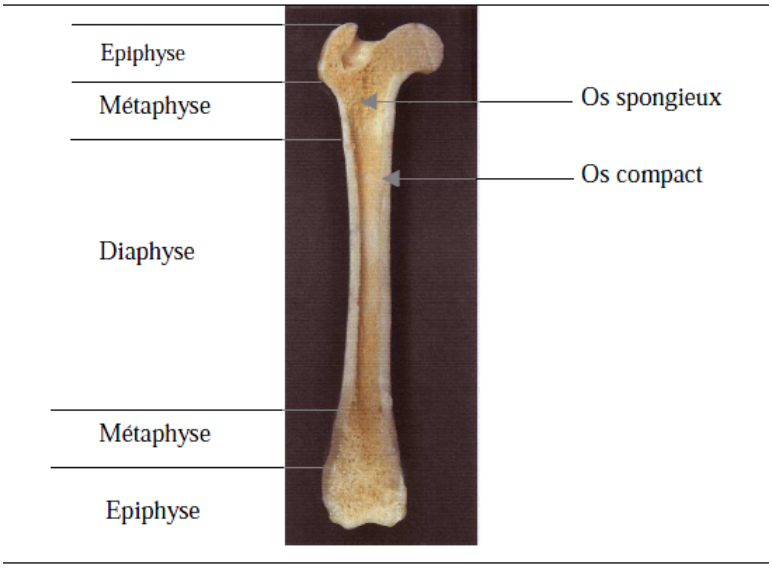
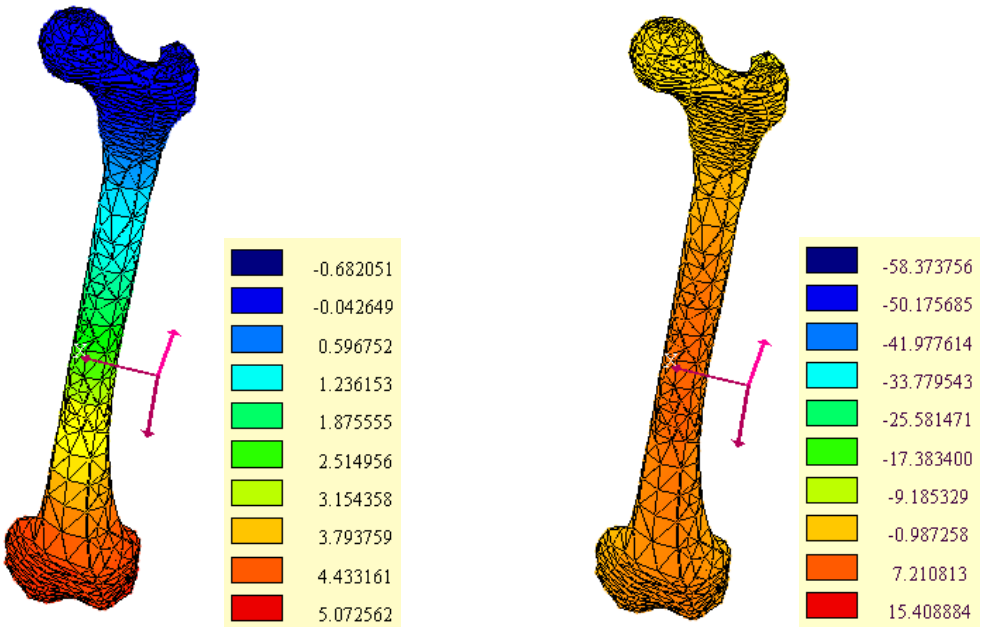


Figure 5.30 : coupe d'un fémur humain (selon Paulin A.M)

Dans ce troisième test de validation, on cherche à simuler la réponse d'un fémur (figure 5.29) aux sollicitations mécanique durant la marche. Durant cet exercice, le fémur est sollicité principalement en compression et en flexion afin de reprendre le poids propre de la personne. La plus part des études considèrent l'os comme un matériau isotrope (Duchemin L, 2006). La répartition des propriétés mécaniques est considéré homogène, la littérature propose des valeurs de module de Young unique dans le cas d'une répartition homogène une valeur (15000 à 16700 MPa)



(a) **(b)**

Figure 5.31 : comportement mécanique simplifié du fémur lors de la marche
(a) : champs de déplacements uxx (axe vertical passant par le fémur) ;
(b) : champs de déplacements uzz (axe portant les efforts de cisaillement appliqués).

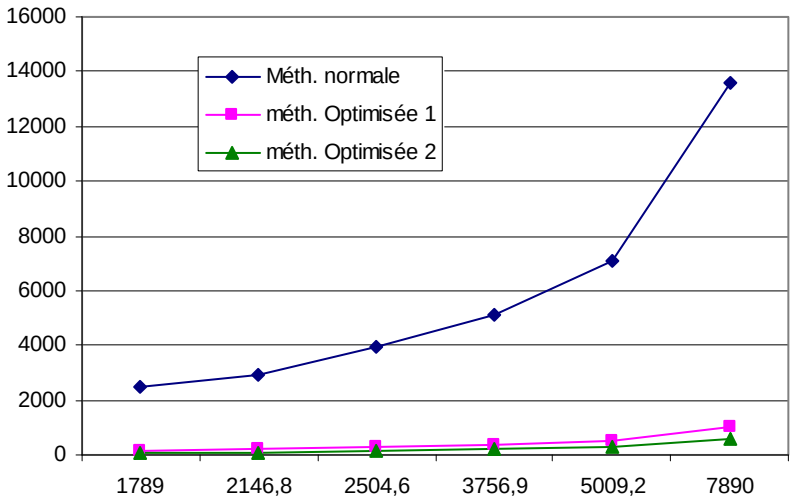


Figure 5.32 : représentation des temps de calculs pour différents maillages

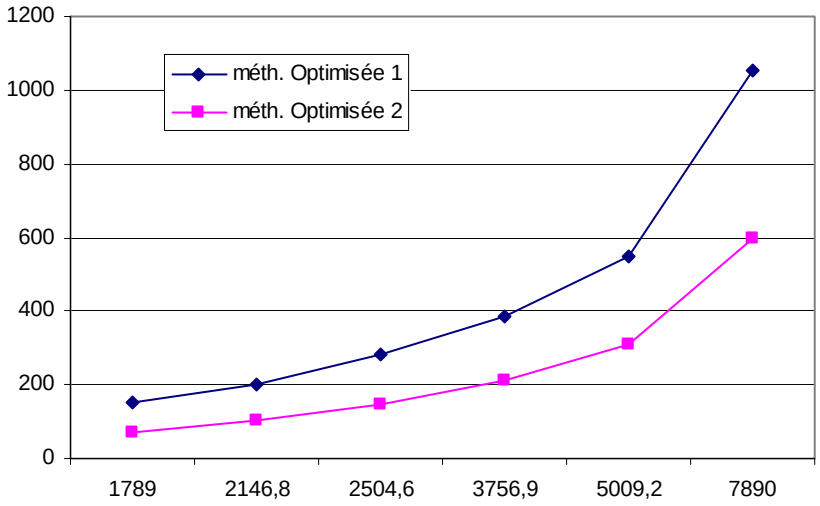


Figure 5.33 : variation des temps de calculs en considérant les deux méthodes d'optimisation

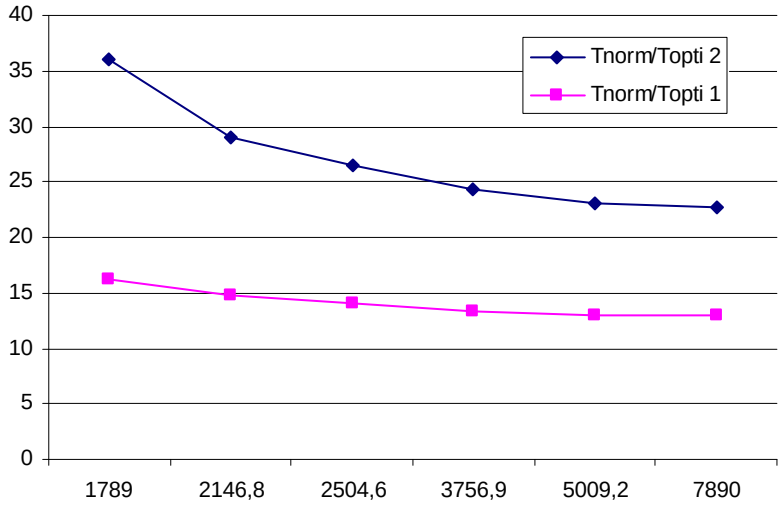


Figure 5.34 : rapport des temps de calculs pour différents maillages du fémur

Remarque

Nous donnons en annexe B des tableaux dans lequel nous avons mis les différentes valeurs exactes que nous avons illustrées sur les figures 5.14, 5.15, 5.26, 5.27. Les lecteurs en quête de détails au sujet des résultats pourront s'y référer.

5.12 Discussion :

Dans cette section, nous allons discuter les résultats obtenus pour les trois tests de validation que nous avons menés pour montrer la validité des éléments conformes que nous avons implémentés dans le code KSP.

Le premier cas test consistait en une plaque épaisse et trouée, soumise à un effort de traction sur sa face supérieur, et encastré sur l'autre face. Les résultats obtenus lors de la simulation de cette pièce sont illustrés dans le tableau 5. . Nous remarquons dans ce tableau que les champs de déplacements (qui constituent nos champs de références pour la validation de notre approche) sont pratiquement les mêmes pour le cas des éléments conformes, l'erreur est plus importante lorsque nous considérons, combiné (de 1% à 0.4%), ceci est selon nous dus à l'accumulation des erreurs commises par la formule heuristique et l'implantation des éléments conformes, l'erreur que nous retrouvons pour les éléments conformes doit provenir selon nous du calcul du terme *cij*. Cependant, ce qui est à noter est la forte réduction des temps de calcul particulièrement pour le cas combiné (intégration adaptative et éléments conformes). Le rapport Tnorm/Topti1 varie de 18 à 13.8 pour des maillages allant de 694 à 1880 éléments alors que Tnorm/Topti2 varie lui de 28 à 20.13. Sur la figure 5.29, nous remarquons que les rapports Tnorm/Topti1 Tnorm/Topti1 ont diminuent avec l'augmentation du nombre d'éléments, cependant cette diminution tendent à se réduire pour des maillages de plus en plus fins.

Dans le deuxième cas de validation, nous avons considéré un porte-outil soumis à un chargement de service appliqué à la partie en contact direct avec le bec. Nous avons considéré cinq discrétisation de la pièce avec un nombre d'éléments variant de 694 à 2038 éléments. La figure 5.27 représente la variation des temps de calcul pour les cas : normale, avec éléments conformes et combiné (éléments conformes et intégration adaptative). Les rapports Tnorm/Topit1 Tnorm/opit2 varient de 17.6 à 13.8 et 28.21 à 19.5 respectivement.

Le dernier cas correspondait à la simulation du comportement mécanique d'un fémur humain lors de la marche. Le nombre de discrétisation pris en compte est le même que le précédent cas, cependant le nombre d'éléments variait entre 1789 et 7890. Les rapports Tnorm/Topti1 et Tnorm/Topti2 varient de 16.2 à 12.9 et de 36 à 22.66 respectivement. La valeur de Tnorm/Topti1 est plus élevée que les cas test précédents, la raison pour nous est que dans le cas du fémur, nous somme en présence d'une géométrie élancée, qui permet de profiter au mieux de la l'intégration adaptative.

5.13 Conclusion

Dans ce chapitre nous avons réussi à élaborer des techniques qui avaient pour objectif la réduction des temps de calculs lors de simulations numériques via le code KSP. KSP étant

un code de calcul par éléments de frontières, il est évident que la consommation des temps de calculs se situe au niveau du remplissage et de la résolution du système (équation 3.9). Le travail concernant l'optimisation à été donc scindé en deux différentes parties:

La première partie s'attaquait à la procédure de remplissage du système, elle consistait en la minimisation du nombre de points de gauss pris en compte lors de l'intégration des noyaux réguliers. Le nombre de point de gauss était fixé (30 points) ce qui causait des ralentissements de vitesses de calculs, nous avons proposé donc une formule heuristique obtenue grâce à une série de simulation. Cette formule permet de calculer le nombre de points de gauss minimal en fonction de la taille de l'élément de maillage et de la distance entre le point source et le point champ. En terme de résultats, les champs de déplacements calculés par les deux configurations (normale et avec intégration adaptative) se révèlent les mêmes, par contre les temps de calculs sont revus à la baisse pour le deuxième cas avec des rapports Tnorm/Topti variant entre 5.2 et 3.6 pour tous les cas test considérés.

La deuxième partie proposait d'implémenter des éléments conformes dans le code KSP. Les éléments conformes ayant leurs nœuds de collocation superposés sur les nœuds géométriques, le nombre de point de collocation appartement au maillage se voit réduire grâce à ces éléments. Notre intervention se situait principalement dans le calcul du terme *cij* (angle solide). Nous avons élaboré un programme qui permettait à partir d'un bloc d'éléments connectés de calculer ce terme.

Après avoir implémenté ces éléments sur KSP, nous avons effectué des tests de validation. Nous avons utilisé trois techniques afin quantifier les gains en temps de calcul, à savoir la méthode normale qui jouait le rôle de référence, la méthode utilisant les éléments conformes, et la méthode combinée (utilisant les éléments conformes et l'intégration adaptative). Nous avons pu par ces deux méthodes réduire considérablement les temps de calculs avec des simulations jusqu'à 36 fois plus rapides.

Annexe A

Dérivation des noyaux

A.1 Dérivation des noyaux de déformation

Par soucis de simplicité, la solution fondamentale de Kelvin (chapitre 2) peut être réécrite dans la forme suivante

$$u_j = U_{ij} e_i \qquad \text{(A.1)}$$

u_j est la jième composante du déplacement (en coordonnées cartésiennes) et e_i est un vecteur force unitaire dans a ième direction. Nous rappelons maintenant les relations déformation déplacement énoncées au chapitre 1 :

$$\varepsilon_{jk} = \left(u_{j,k} + u_{k,i}\right)/2 \qquad \text{(A.2)}$$

Nous obtenons ainsi

$$E_{ijk} = \left(U_{ij,k} + U_{ik,j}\right)/2 \qquad \text{(A.3)}$$

La différentiation spatiale est calculée par rapport au point champs Q, puisque c'est à ce point ou les déplacements sont calculés. Pour la différentiation de la solution fondamentale, la relation suivante est requise :

$$\left(\frac{1}{r}\right)_{,j} = \frac{\partial}{\partial x_j}\left(\frac{1}{r}\right) = \frac{\partial}{\partial r}\left(\frac{1}{r}\right)\frac{\partial r}{\partial x_j}$$
$$= \frac{-1}{r^2}\frac{r_j}{r} = \frac{-r_j}{r^3} \qquad \text{(A.4)}$$

De la même façon,

$$\left(\frac{1}{r^3}\right)_{,j} = \frac{-3r_j}{r^5} \qquad \text{(A.5)}$$

Notons aussi que :

$$r_{i;j} = \delta_{ij} \quad et \quad r_{,i} = r_i / r$$

Maintenant, à partir de l'équation (A.1), ou plus précisément sa forme explicite dans le chapitre trois, nous avons

$$U_{ij} = A\left[B\frac{\delta_{ij}}{r} + \frac{r_i r_j}{r^3} \right] \qquad \text{(A.7)}$$

Avec,

$$A = 1/\left[16\pi G(1-\nu)\right]$$
$$B = 3 - 4\nu \qquad \text{(A.8)}$$

En utilisant les équations (A.5) à (A.10), nous obtenons

$$U_{ij,k} = \frac{A}{r^3}\left[-B\delta_{ik}r_j + r_i\delta_{kj} + r_k\delta_{ij} - 3r_i r_j r_k / r^2 \right] \qquad \text{(A.9)}$$

En inversant les indices *j* et *k*, on obtient

$$U_{ik,j} = \frac{A}{r^3}\left[-B\delta_{ij}r_k + r_i\delta_{jk} + r_j\delta_{ik} - 3r_i r_k r_j / r^2 \right] \qquad \text{(A.10)}$$

Donc, en utilisant l'équation(A.4), nous obtenons le résultat cité dans le chapitre trois :

$$E_{ik,j} = \frac{-A}{r^2}\left[C\left(\delta_{ik}r_{,j} + \delta_{ij}r_{,k}\right) - \delta_{jk}r_{,i} + 3r_{,i}r_{,k}r_{,j} \right] \qquad \text{(A.11)}$$

Avec

$$C = 1 - 2\nu \qquad \text{(A.12)}$$

A.2 Dérivation des noyaux des contraintes

Le noyau de contrainte Σ_{ijk} est définit dans le chapitre deux par l'équation

$$\sigma_{jk} = \Sigma_{ijk} e_i \qquad \text{(A.13)}$$

e_i représente la ième composante du vecteur force unitaire. La loi de Hooke qui relie le tenseur de contraintes au tenseur de déplacements est défini au chapitre 1 par l'équation suivante :

$$\sigma_{jk} = \lambda \delta_{jk} \varepsilon_{mm} + 2G\varepsilon_{jk} \qquad \text{(A.14)}$$

Parce que la constante de Lamé λ est liée au module de cisaillement G par l'équation

$$\lambda = 2G\nu/(1-2\nu) \qquad \text{(A.15)}$$

Nous pouvons réécrire la loi de Hooke dans la forme

$$\sigma_{jk} = 2G\left[\frac{\nu}{1-2\nu}\delta_{jk}\varepsilon_{mm} + \varepsilon_{jk}\right] \qquad \text{(A.16)}$$

Donc, en utilisant les équations(A.3) à (A.15), et (A.18), nous obtenons :

$$\Sigma_{ijk} = 2G\left[\frac{\nu}{1-2\nu}\delta_{jk}E_{imm} + E_{ijk}\right] \qquad \text{(A.17)}$$

Notons que $\delta_{mm} = 3,\ r_{,m}r_{,m} = 1,\ et\ \delta_{mi}r_{,m} = r_{,i}$ nous posons maintenant $j = k = m$ dans l'équation (A.13) pour obtenir :

$$E_{imm} = -2ACr_{,i}/r^2 \qquad \text{(A.18)}$$

Finalement, en substituant cette équation et l'équation(A.13) dans l'équation (A.19), nous obtenons le résultat cité au chapitre deux, c.-à-d. :

$$\Sigma_{ijk} = \frac{-2GA}{r^2}\left[C\left(\delta_{ik}r_{,j} + \delta_{ij}r_{,k} - \delta_{jk}r_{,i}\right) + 3r_{,i}r_{,j}r_{,k}\right] \qquad \text{(A.19)}$$

A.3 Dérivation du noyau de traction

Le noyau de traction est définit dans le chapitre 2 par l'équation

$$t_j = T_{ij}e_i \qquad \text{(A.20)}$$

Avec t_j étant la composante de traction (dans un plan de normale n_k) résultant d'un vecteur force unitaire e_i. En raison du fait que les tractions sont reliées au tenseur de contraintes par la relation énoncée au chapitre 1 et qui la suivante :

$$t_j = \sigma_{jk} n_k \quad \text{(A.21)}$$

En utilisant l'équation(A.15), nous obtenons

$$T_{ij} = \frac{-2GA}{r^2}\left[C\left(n_i r_{,j} - n_j r_{,i}\right) + \left(3r_{,i} r_{,j} + C\delta_{ij}\right) n_m r_{,m}\right] \quad \text{(A.22)}$$

L'indice muet dans le dernier terme est arbitraire, et il peut prendre n'importe quelle valeur.

A.4 Noyaux pour des états de contraintes et de déformations planes

En intégrant la solution de Kelvin, nous obtenons le noyau de déplacement pour une déformation plane et qui est donné par l'équation :

$$U_{ij} = -2A[B\delta ij \log(r) - r_{,i} r_{,j} \quad \text{(A.23)}$$

Avec

$$A = 1/\left[16\pi G(1-\nu)\right]$$
$$B = 3 - 4\nu \quad \text{(A.24)}$$

Naturellement, dans cette équation, les indices i et j prennent les valeurs *1* et *2* seulement. En suivant la même procédure démontrée au paravent pour un cas tridimensionnel, ici en substituant l'équation.(A26) dans l'équation(A.4), nous obtenons le noyau de déformation

$$E_{ijk} = \frac{-2A}{r}\left[C\left(\delta_{ik} r_{,j} + \delta_{ij} r_{,k}\right) - \delta_{jk} r_{,i} + 2r_{,i} r_{,j} r_{,k}\right]$$

Avec

$$C = 1 - 2\nu$$

Le noyau de contrainte, peut être obtenu de la même façon

$$\Sigma_{ijk} = \frac{-4GA}{r}\left[C\left(n_i r_{,j} + \delta_{ij} r_{,k} - \delta_{jk} r_{,i}\right) + 2r_{,i} r_{,j} r_{,k}\right]$$

Ainsi les noyaux de traction sont donnés par

$$T_{ij} = \frac{-4GA}{r}\left[C\left(n_i r_{,j} - n_j r_{,i}\right) + \left(3r_{,i} r_{,j} + C\delta_{ij}\right) n_m r_{,m}\right]$$

Pour déterminer les contraintes internes sous un état plan de déformation, en ayant calculé au préalable les déplacements et les tractions sur la frontière, les noyaux U_{ij} et T_{ij} sont requis. En suivant la même procédure décrite plus haut, nous obtenons

$$U_{ijk} = \frac{1}{4\pi(1-\nu)}\frac{1}{r}\left[C\left(n_i r_{,j} + \delta_{ij} r_{,k} - \delta_{jk} r_{,i}\right) + 2r_{,i} r_{,j} r_{,k}\right]$$

Et

$$T_{ijk} = \frac{G}{2\pi(1-\nu)}\frac{1}{r^2}\left\{2r_{,m}n_m\left[C\delta_{ij}r_{,k} + \nu\left(\delta_{ik}r_{,j} + \delta_{jk}r_{,i}\right) - 4r_{,i}r_{,j}r_{,k}\right]\right.$$
$$\left. + 2\nu\left(n_i r_{,j} r_{,k} + n_j r_{,i} r_{,k}\right) + C\left(2n_k r_{,i} r_{,j} + n_j\delta_{ik} + n_i\delta_{jk}\right) - Dn_k\delta_{ij}\right\}$$

Avec

$D = 1 - 4\nu$

Pour un état de contraintes plan, les résultats correspondant peuvent êtres obtenus en remplaçant le coefficient de Poisson par la valeur $\nu/(1+\nu)$.

Annexe b

B.1 Listing du Programme permettant le calcul de l'angle solide

Dans le chapitre 5, nous avons présenté un organigramme qui nous montre comment nous avons procédé pour le calcul de l'angle solide, dans ce qui suit nous donnons le programme principale qui permet de calculer ce dernier. Notons que dans le programme suivant, le mot mis en italique sont des mots clés que Visual C++ reconnait.

PROGRAMME PRINCIPAL

```
PI =  3.1415926535897932384          // introduire la valeur de PI comme paramètre
                                     // permet un gain de temps et une lisibilité du programme

# include "norm.h"          // inclure le fichier entête norme.h
# include "point.h"         // permet d'inclure le fichier entête contenat la classe point
# include <iostream>        // include la bibliothèque standard
using namespace std;     // permet d'éviter d'écrire avant les commande I/O std ::

int main()
 {
// allocation dynamique
int l,m, Nbrpoints;
double rmin=100000;
double alphaT=0;   // NU=0.3;
point p(0,0,0);
Nbrpoints=6;
int *dim=&Nbrpoints;
point *b= new point[*dim];
point *a= new point[*dim];
vecteur *r= new vecteur [*dim];
double *sgn= new double [*dim];
double *alpha= new double [*dim];
```

```
double *k= new double [*dim];
vecteur **s;
s= new vecteur*[*dim];      // allocation dynamique d'un vecteur

//
for(l=0;l<*dim;l++)
        {
    s[l]= new vecteur[*dim];
        }
vecteur **N = new vecteur *[*dim];
        for(l=0;l<*dim;l++)
        {
N[l]= new vecteur[*dim];
        }
//P1
for (l=0;l<Nbrpoints;l++)
{
r[l].setx(p.getx()-b[l].getx());
r[l].sety(p.gety()-b[l].gety());
r[l].setz(p.getz()-b[l].getz());
double module= sqrt(pow(r[l].getx(),2)+pow(r[l].gety(),2)+pow(r[l].getz(),2));
if (module<rmin) rmin=module;
}
for (l=0;l<Nbrpoints;l++)
{
k[l]=sqrt(pow(r[l].getx(),2)+pow(r[l].gety(),2)+pow(r[l].getz(),2))/rmin;
}
for (l=0;l<Nbrpoints;l++)
{
a[l].m_x= p.m_x - (p.m_x-b[l].m_x)/k[l];
a[l].m_y=p.m_x - (p.m_y-b[l].m_y)/k[l];
a[l].m_z=p.m_z - (p.m_z-b[l].m_z)/k[l];
}
```

```
for (l=0;l<*dim;l++)
{
for (m=0;m<*dim;m++)
  {
          s[l][m].setx(r[l].getx()-r[m].getx());
          s[l][m].sety(r[l].gety()-r[m].gety());
          s[l][m].setz(r[l].getz()-r[m].getz());
          N[l][m]=N[l][m].normale(p,a[l],a[m]);
          N[l][m].normer();
 }
}
sgn[1-1]= acos(N[*dim-1][1-1].prodscal(N[1-1][2-1]));
sgn[*dim-1]= acos(N[*dim-1-1][*dim-1].prodscal(N[*dim-1][1-1]));
if (sgn[1-1]>0) sgn[1-1]= -sgn[1-1];
if (sgn[*dim-1]>0) sgn[*dim-1]= -sgn[*dim-1];
for(l=2-1;l<*dim-1;l++)
{
   sgn[l]=acos(N[l-1][l].prodscal(N[l][l+1]));
    if (sgn[l]>0) sgn[l]= -sgn[l];
 }
for(l=1-1;l<*dim;l++)
{
    alpha[l]=PI + sgn[l];
    alphaT += alpha[l];
}
 alphaT= alphaT - (Nbrpoints-2)*PI;
 double phi= alphaT/(4*PI);
delete [] k;
return 0;
 }
```

B.2 Résultats obtenus pour les différents cas test de validation:

Cas test 1 : porte-outils

Nbr d'éléments	694	896	1060	1378	1546	1880
Uyymax Mét. Normale	0.669137	0.669358	0.669297	0.669244	0.669244	0.669244
Uyymax Mét. Optim. 1	0.669738	0.66641	0.669400	0.669244	0.669244	0.669244
Uyymax Mét. Optim. 2	0,668465	0,66509	0,670741	0,668575	0,66870911	0,66870911
Temps (s) Mét. Normale	532,09	754,57	1115,31	1832,97	2645,72	3645,15
Temps (s) Mét. Optim. 1	**30.227**	**44.880**	**70.145**	**117.43**	**171.753**	**250.353**
Temps (s) Mét. Optim. 2	**18.858**	**31.416**	**50.695**	**87.238**	**129.656**	**186.930**
Tnorm/Topti1	**17,6**	**16.6**	**15.6**	**14.5**	**13.8**	**13.5**
Tnorm/Topti2	**28.21**	**24.12**	**22**	**21.05**	**20.4**	**19.5**

Cas test 2 : fémur

Nbr d'éléments	694	896	1060	1378	1546	1880
Uyymax Mét. Normale	0.669137	0.669358	0.669297	0.669244	0.669244	0.669244
Uyymax Mét. Optim. 1	0.669738	0.66641	0.669400	0.669244	0.669244	0.669244
Uyymax Mét. Optim. 2	0,668465	0,66507	0,6707415	0,668575	0,668709	0,668709
Temps (s) Mét. Normale	2450.32	2940.15	3920	5145.87	7105.18	13564
Temps (s) Mét. Optim. 1	**151.23**	**198.64**	**280.08**	**383.95**	**546.53**	**1051.47**
Temps (s) Mét. Optim. 2	**68.055**	**101.484**	**148.204**	**212.077**	**306.91**	**599.64**
Tnorm/Topti1	**16.2**	**14.8**	**14**	**13.4**	**13**	**12.9**
Tnorm/Topti2	**36**	**28.97**	**26.45**	**24.26**	**23.15**	**22.62**

Bibliographie

Banerjee, P., & Davies, T. (1984). Advancd implementation of the boudary element method for three dimensional problems of elasto-plasticity. *Elservier Applied Science ublisher* , London.

Becker, A. (1992). *The boundary element method in engineering.* London: McGraw-Hill.

Bigot, D., & Kebir, H. (2009). Numerical method coupling finite elements and boundary elements to metal forming tools. *Journal of Material Processing Technology* , 3226-3235.

Brebbia, C., Telles, J., & Worbel, L. (1984). *boundary Element Technics* . Berlin: Springer-Verlag.

Burke, W. (1985). *Applied Diffenrential Geometry.* Cambridge: Cambrige University Press.

Cruse, T. (1974). An improved boundary-integral equation method for three dimensional elastostatics. *Computer and Structures* , 741-754.

Cruse, T. (1969). Numerical solution in three-dimensional elastostatics. *International Journal of Solids and Structures* , 1259-1274.

Fung, Y. (1965). *Foundation of Soild mechanics.* New Jersey: Pentice-Hall.

Gao. (s.d.).

Gao X.W & Davies T.G. (2002). *Boundary Element Programing in Mechanics.* Cambridge: Cambridge University Press.

Gao, X., & Davies, T. (2000.a). 3D multi-region BEM with corners and edges. *International Journa of Solids and Structures* , 1549-1560.

Gao, X., & Davies, T. (2000). Adaptative algorithme in elasto-plastic boundary element analysis. *Journal of Chinese Engineering* , 349-356.

Guiggiani, M., & Gigante, A. (1990). A general algorithme for multidimensional cauchy principal value in the boundary element method. *Applied Mechanics* , 906-915.

Guiggiani, M., & Krishnasamy, G. (1992). General algorithme for numerical solution of hyper-singular boundary integral equation. *Journal of Applied Mechanics (ASME)* , 604-614.

Jaswon M.A & Symm G.T. (1977). *integral Equation Methods in potential Theory and Elastostatics.* London: Academic Press.

Kane, J. (1994). *Boundary Element Analysis in Engineering Continuum Mechanics.* Englwood cliffs, New Jersey: Pentice-Hall.

Kebir, H. (1999). *Approches déterministe et probabiliste de la prévision de la durée de vie des structures aéronautiques à l'aide de la méthode des équation integrales duale.* Compiègne, France: Thèse de Doctora: 'Université de Technologie de Compiègne.

Kebir, H., & Roelandt, J. (1999). A new singular boundary element for crack problems: Application to bolted joints. *Engineering Fracture Mechanics* , 497-510.

Lachat, J. (1975). *A Further Development of the Boundary Integral technics for Elastostatics.* Southampton, UK: Phd Thesis, University of Southampton .

Lachat, J., & Watson, J. (1976). Effective numerical treatement of boundary integral equation. *international Journal of Numerical Methods* , 991-1005.

Leung C.Y & Walker S.P. (1997). Iterative solution of large three-dimensionnal BEM elastostatic analysis using the GMRES technique. *International journal of Numerical Methods Engineering* , 2227-2236.

Mantic. (1995).

Mustoe, G. (1984). Advanced integration schemes over boundary elements and volume cells for two and three dimensionnal non-linear analysis. *Devolpements in boundary element methods, Elsevier* , 213-270.

Rizzo, F. (1967). An integral equation approach to boundary value problems of classical elastostatics. *Journal of Applied Mathematics* , 83-95.

Somigliana, C. (1885). Sopra l'équilibro di un corpo elastico isotropo. *Nuova Cimento* .

Stroud, A., & Secrest, D. (1966). *Gausian Quadrature Formulas.* Englewood Cliffs, New Jersey: Prentice-Hall.

Telles, J., & Brebbia, C. (1979). On the application of the boundary element method to plasticity. *Applied Mathematics modeling* , 466-470.

Thomson, W. (. (1848). A note on the integration of the equation of equilibirium of an elastic solid. *Cambridge and Dublin Mathematical journal* .

Timoshenko, S., & Goodier, J. (1970). *Theory of Elasticity.* New York: McGraw-Hill.

Wilde, A. (1998). *A Hypersingular Dual Bounary Element Formulation for Three-Dimensional Fracture Analysis.* University of Wales, UK: Phd Thesis, Wessex Institute of Technologie .

Zhang, Q., & Mukherjee, S. (1991). Designe sensitivity coefficients for linear elastic bodies with zones and corners by the derivative boundary element method. *International Journal of Solids and Structures* , 983-998.

Zienkiewicz, O. (1977). *The Finite Element Method.* Maidenhead, UK: McGraw-Hill.