République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et la Recherche Scientifique

Ecole Nationale Polytechnique

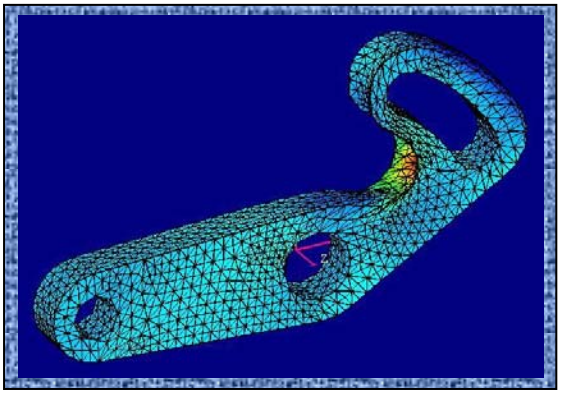
المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département de Génie Mécanique

Mémoire de fin d'étude pour l'obtention
du diplôme Ingénieur d'Etat en Génie Mécanique

THEME

Résolution des problèmes 3D à grand nombre de degrés de liberté par la méthode des éléments de frontières



Proposé et dirigé par :

Professeur S. Rechak **ENP**

Docteur H. Kebir **UTC**

Présenté par :

Atmane LAOUATI

Septembre 2007

Remerciements

Nous remercions avant tout **ALLAH** le tout puissant de nous avoir donné la fois la volonté et le courage de mener à bien ce modeste travail.

Mes sincères remerciements au **Professeur S. Rechak** sans qui le travail accompli, le savoir acquis, l'aventure, n'auraient jamais été possibles.

Sans oublier **Mr H. Kebir** pour nous avoir accueillis au sein de Laboratoire Roberval de l'UTC, et pour nous avoir permis de réaliser ce stage dans si bonnes, Nous le remercions pour ses conseils, l'encadrement dont nous avons bénéficié et la confiance qu'il nous a accordée.

Je remercie aussi **Messieurs Belkacemi et Ammiche** d'avoir pris le temps et le soins de lire ce rapport.

Je remercie les enseignants du département de génie mécanique qui ont participé à ma formation tout au long de ces trois ans.

Je remercie mes amis de la promotion 2007

Dédicaces

A ma chère Maman et mon cher Père

A mon frère Nabil et mon oncle Mahmoud

A mes sœurs, Sabah, Souad, Houda, Meriem et la petite

Hana

A mes frères, Khaled, Ismail, Salah Edine

A mon cousin Adlaine

A toute la famille LAOUATI

A tout mes amis de l'ENPEI et L'ENP

A tous ceux qui me connaissent de prés et de loin

Je dédie chaleureusement ce travail ...

Atmane

ملخص

التمثيل الرقمي في الميكانيك يأخذ أهمية أكبر فأكبر في الوسط الصناعي والمسائل المطروحة تصبح ضخم فأضخم، هذا العمل يتمثل في تطوير طريقة حل تكرارية لتسريع كود للتمثيل الرقمي لسلوك الهياكل ذات العدد لكبير من درجات الحرية باستعمال طريقة العنصر الحدودي.

وبما أن جملة المعادلات المطروحة للحل والناتجة عن استعمال هذه الطريقة تمثل بمصفوفة مملوءة وغير متناظرة فقد احترنا الطريقة العامة للباقي الأقل(GMRES) والتي تعتبر الطريقة الأمثل لهدا النوع من المعادلات.

كلمات مفتاح:التمثيل الرقمي،سلوك الهياكل ذات العدد الكبير من درجات الحرية،طريقة العنصر الحدودي،طرق لحل التكرارية،GMRES .

Résumé

La simulation numérique en mécanique prend de plus en plus d'importance dans le milieu industriel et les problèmes à modéliser deviennent de plus en plus grand, ce travail consiste à développer une méthode de résolution itérative pour accélérer un code de simulation numérique du comportement des structures à grand nombre de degrés de liberté par la méthode des équations intégrales.

Puisque le système à résoudre résultant de cette méthode est représenté par une matrice pleine et non symétrique on a choisi la méthode GMRES qui est la plus adaptée pour ce type de système.

Mots clé : simulation numérique, comportement des structures à très grand nombre de ddl, le méthode d'élément frontière, méthode itérative de résolution, GMRES.

Absract

The numerical simulation in mechanics takes more and more importance in the industrial environment and the problems to be modelled become more and more big, this work consists in developing a method of iterative resolution to accelerate a code of numerical simulation of the conduct of the structures with big number of degrees of freedom by the method of boundary element method.

Because the system to be resolved resulting from this method is represented by a full and not symmetric matrix we chose the method GMRES which is the most adapted for this type of system.

Table des matières

Table des figures

Nomenclature

λ, μ : coefficients de Lamé.

ν : coefficient de Poisson.

E : module d'Young.

$[\varepsilon]$: état de déformation.

ε_{ij} : la composante (i,j)de l' état de déformation.

$[\sigma]$: état de contrainte.

σ_{ij} : la composante (i,j)de l' état de contrainte.

$\vec{u}$: vecteur de déplacement.

$\vec{t}$: vecteur de tension.

$\vec{f_v}$: vecteur des forces de volume.

U_i : déplacement suivant la direction e_j.

$U_{i,j} = \dfrac{\partial U_i}{\partial e_j}$

U_j^i : la composante j du vecteur déplacement suivant la direction e_j.

$U_{j,k}^i = \dfrac{\partial U_j^i}{\partial e_k}$

T_j^i : la composante j du vecteur tension suivant la direction e_j.

σ_{jk}^i : le tenseur de contrainte lié à T_j^i et $U_{j,k}^i$

Ω : domaine à étudier.

$\partial\Omega$: frontière du domaine à étudier.

Γ : contour d'intégration.

ε : rayon d'isolation d'un point pour le contour.

Introduction

Les problèmes de mécanique traités par les ingénieurs sont décrits par les lois physiques qui se présentent sous forme d'équation différentielles et qui conduisent à des formes intégrales permettant de fournir des solutions approchées après une étape de discrétisation du domaine à étudier. Parmi les méthodes numérique utilisable, la méthode des élément finis de frontière (Boundary Element Method) est particulièrement performante pour résoudre les problèmes de grand taille et de la propagation des fissure en élasticité linéaire. En effet, dans ce cas, seul le contour de la structure étant discrétisé, les problèmes de modification de la discrétisation au cours de la propagation des défauts s'avèrent moins délicats à traiter que par exemple dans le cadre de la méthode des éléments finis de domaine.

L'une des différences majeures entre la méthode des équations intégrales et la méthode des éléments finis est que la forme de la matrice du système à résoudre est totalement aléatoire et diffère d'un cas à un autre pour la première méthode alors qu'elle est symétrique et bande pour la deuxième méthode, Ce dernier point l'a défavorise face à la méthode des éléments finis, mais on peut surpasser ce petit inconvénient en utilisant une méthode de résolution adapté à ce type de système, et la plus célèbre parmi les méthodes de résolution connus c'est la méthode GMRES.

On assiste à plusieurs travaux de recherches depuis plus de 10 ans qui se placent essentiellement dans la conception et le développement d'outils informatiques de simulation numérique pour la mécanique basés surtout sur la méthode des équations intégrales.

Dans le cas des problèmes 3D, et c'est souvent le cas, et surtout pour les problèmes de grande taille, cette méthode et plusieurs d'autres méthodes demandent de très grandes capacités d'ordinateurs en mémoire et en vitesse.

Le KSP est un logiciel pour la modélisation numérique en mécanique. Il est développé à l'Université de Technologie de Compiègne, Il permet la résolution de différents types de problème linéaires ou non linéaires, modélisés par la méthode des éléments finis et/ou par la méthode des équations intégrales, en deux ou en trois dimensions, et on a implémenté la méthode GMRES comme méthode de résolution pour le KSP pour pouvoir traité des problèmes de grand taille.

Cette version de KSP reste incapable de résoudre des problèmes qui dépassent les 9 000 degrés de liberté parce qu'il y a un problème de saturation de la RAM de l'ordinateur.

Le but de ce stage est de développé une nouvelle version de KSP capable de traité des problèmes de grands tailles 15 000 , 20 000 ddl etc.

La présente étude est abordée de la manière suivante :

- Dans le chapitre (I) on va voire la méthode des équations intégrales appliquées sur des structures non fissurées et fissurées.

- A travers le chapitre (II), on abordera le traitement numérique des équations intégrales duales.

- On a consacré le chapitre (III) à la présentation du code de calcul KSP.

- Le chapitre (IV) sera réservé à la méthode de résolution GMRES utilisé par le code KSP.

- Le chapitre (V) on présente l'implémentation faite sur le code KSP pour arriver à traité des problèmes de grand taille

On terminerons l'étude par une conclusion générale.

Chapitre 1

La méthode des éléments de frontières

1.1 Introduction

Le principe de la méthode des éléments de frontières consiste à transformer des équations aux dérivées partielles dans le volume de la pièce en équations intégrales sur le contour. En élasticité linéaire, on ramène ainsi l'étude d'une pièce mécanique à l'étude de son comportement en surface. La discrétisation de celle-ci permet de transformer l'équation intégrale en un système d'équations linéaires et de déterminer les déplacements et contraintes en chaque point du contour. On peut ensuite obtenir les grandeurs à l'intérieur de la pièce à l'aide de relations intégrales simples.

Historiquement, la méthode des éléments de frontières appelée aussi méthode des équations intégrales a été développée de deux manières distinctes; l'une des deux est une approche physique intuitive, qu'on appelle la méthode des discontinuités de déplacements. Cette approche consiste à chercher en premier lieu les valeurs des perturbations fictives dont les effets sur le contour sont les conditions aux limites spécifiées, ensuite à calculer le reste des inconnues du contour comme étant les effets de ces perturbations "*fictives*" en ces points. Puisque les inconnues du contour sont obtenues indirectement, cette approche est appelée **Méthode des Equations Intégrales Indirectes**. L'autre approche est une approche plus mathématique, elle est basée sur certains théorèmes fondamentaux qui relient directement les inconnues du contour aux conditions limites, cette approche est appelée la Méthode des Equations Intégrales Directes et c'est elle qui est adopté pour notre étude.

Dans ce chapitre on présente le principe et l'implantation numérique de la méthode des équations intégrales directes en élasticité linéaire. A partir du théorème de réciprocité de Maxwell-Betti et de la solution élémentaire (solution du problème de Kelvin), on obtient les équations intégrales en déplacement qui, après discrétisation et résolution, nous permettent d'estimer les déplacements et les tensions sur tout le contour. Les contraintes et les déplacements des points intérieurs sont alors calculés par une simple intégration.

1.2 Formulation des équations intégrales

1.2.1 Hypothèses :

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. Les problèmes e contraintes planes seront traités en remplaçant le module de Young E et le coefficient de poisson v respectivement par $(1 - v^2)\,E$ et $v / (1 + v)$.

1.2.2 Théorème de réciprocité de Maxwell-Betti :

Le théorème de réciprocité de Maxwell-Betti lie entre les solutions de deux problèmes d'élasticité linéaire sur un même domaine Ω.

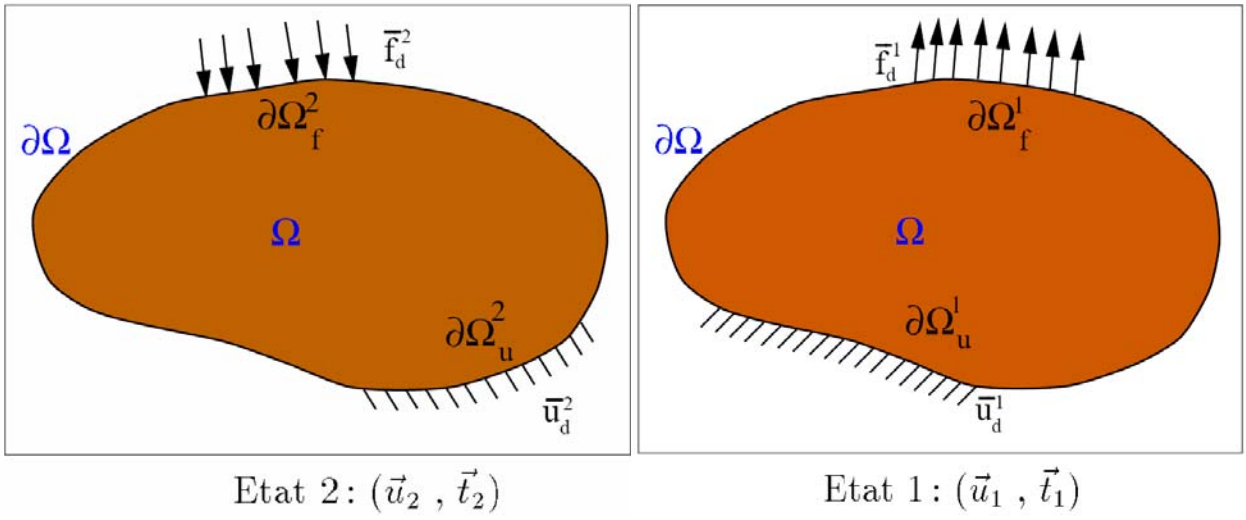


FIG.1.1 LE POINT D'APPLICATION DE LA FORCE EST UN POINT INTERIEUR

Supposons que le premier problème ait comme solution le couple $(\vec{u^1},\vec{t^1})$; champ de déplacements et champ de tentions sur le contour $\partial\Omega$.

Soit un second problème défini sur le même domaine et qui a comme solution le couple $(\vec{u^2},\vec{t^2})$. D'après le théorème des travaux virtuels on peut écrire :

$$\int_\Omega \left[\sigma^1\right]:\left[\varepsilon^2\right]d\Omega - \int_{\partial\Omega} \vec{t^1}\vec{u^2}ds - \int_\Omega \vec{f_\upsilon^1}\vec{u^2}d\Omega = 0 \tag{1.1}$$

et

$$\int_\Omega \left[\sigma^2\right]:\left[\varepsilon^1\right]d\Omega - \int_{\partial\Omega} \vec{t^2}\vec{u^1}ds - \int_\Omega \vec{f_\upsilon^2}\vec{u^1}d\Omega = 0 \tag{1.2}$$

en faisant la différence des deux équations et on a :

$$\int_\Omega \left(\left[\sigma^2\right]:\left[\varepsilon^1\right]-\left[\sigma^1\right]:\left[\varepsilon^2\right]\right)d\Omega - \int_{\partial\Omega}\left(\vec{t^2}\vec{u^1}-\vec{t^1}\vec{u^2}\right)ds - \int_\Omega\left(\vec{f_\upsilon^2}\vec{u^1}-\vec{f_\upsilon^1}\vec{u^2}\right)d\Omega = 0 \tag{1.3}$$

or,d'après la symétrie du comportement élastique, on a :

$$\left[\sigma^2\right]:\left[\varepsilon^1\right]=\left[\sigma^2\right]:\left([D]\left[\sigma^1\right]\right)=\left([D]\left[\sigma^2\right]\right):\left[\sigma^1\right]=\left[\sigma^1\right]:\left[\varepsilon^2\right] \tag{1.4}$$

d'où le théorème de réciprocité :

$$\int_{\partial\Omega}\left(\vec{t^1}\vec{u^2}-\vec{t^2}\vec{u^1}\right)ds - \int_\Omega\left(\vec{f_\upsilon^1}\vec{u^2}-\vec{f_\upsilon^2}\vec{u^1}\right)d\Omega = 0 \tag{1.5}$$

en l'absence des forces de volume, ce théorème s'écrit :

$$\int_{\partial\Omega}\vec{t^1}\vec{u^2}ds = \int_{\partial\Omega}\vec{t^2}\vec{u^1}ds \tag{1.6}$$

Ce théorème permet d'établir de relations intégrales entre les inconnues du problème à résoudre (1) ou (2), l'autre étant connu.

1.2.3 Problème de KELVIN

Pour pouvoir écrire l'équation intégrale on a besoin d'une solution analytique d'un problème particulier sur le domaine Ω. Plusieurs solutions élémentaires existent, par exemple:

- demi -espace avec surface libre ;
- demi-espaces collés;
- plaque épaisse infinie;
- problème de GREEN, etc.

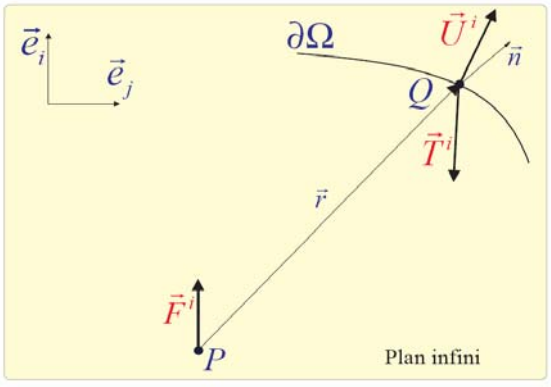


FIG.1.2 PROBLEME DE KELVIN

La solution la plus simple et la plus générale est celle du problème de *KELVIN* décrite ci dessous.

Soit une force unitaire $\overrightarrow{F^i}$ appliquée en un point *P* d'un milieu infini bidimensionnel suivant la direction $\vec{e_i}$. Pour ce problème, l'équation de ***NAVIER*** s'écrit en un point Q du domaine:

$$(\lambda+\mu)u_{k,kj}+\mu u_{j,kk}=-\delta_{PQ}\delta_{ij} \tag{1.7}$$

Avec déplacements nuls à l'infinis comme condition aux limites

$$\lim_{\|PQ\|\to\infty} u_j = 0 \tag{1.8}$$

La résolution de cette équation différentiel conduit à :

- La composante du vecteur déplacement en un point Q suivant la direction $\vec{e_i}$ est:

$$U_j^i(P,Q)=\frac{1}{8\pi\mu(1-\upsilon)}\left[(3-4\upsilon)\ln(\frac{1}{r})\delta_{ij}+r_{,i}r_{,j}\right] \tag{1.9}$$

où:

r: est la distance entre les point *P* et Q, $\mu=\dfrac{E}{2(1+\upsilon)}$

et

$$r_{,i}=\begin{cases}\dfrac{x_Q-x_P}{r} & \text{si } i=1\\[2ex] \dfrac{y_Q-y_P}{r} & \text{si } i=2\end{cases} \tag{1.10}$$

- Le tenseur des contraintes est calculé à partir de la loi de comportement e élasticité linéaire (loi de Hooke) :

$$\sigma_{jk}^i=\lambda\delta_{jk}U_{l,l}^i+\mu(U_{j,k}^i+U_{k,j}^i) \tag{1.11}$$

- la composante du vecteur tension en un point Q sur une facette de normale $\overrightarrow{n(Q)}$ suivant la direction $\vec{e_j}$ est :

$$T_j^i(P,Q)=\sigma_{jk}^i(P,Q)n_k(Q) \tag{1.12}$$

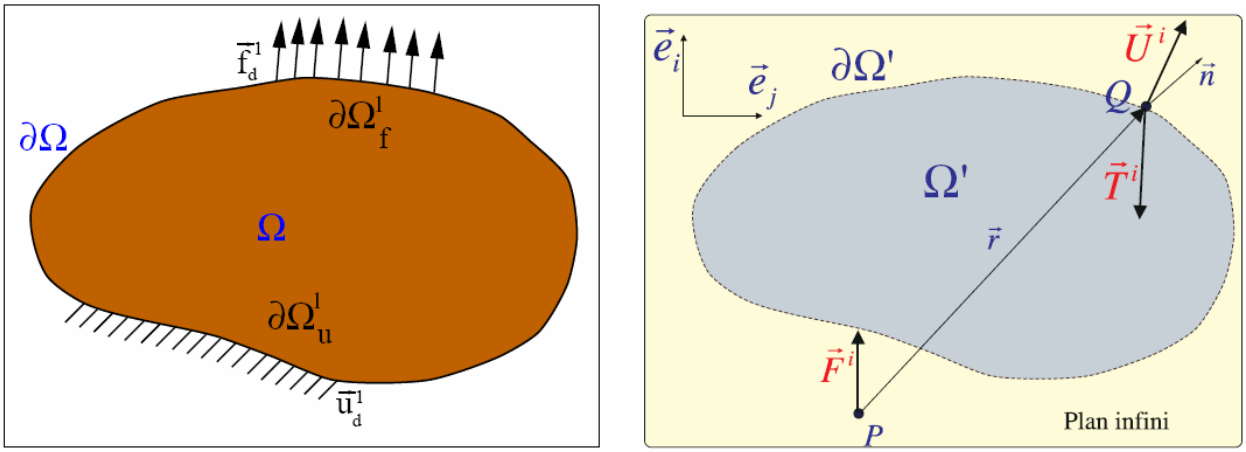
soit sous une forme explicite :

$$T_j^i(P,Q)=\frac{1}{4\pi(1-\nu)r}\left[\frac{\partial r}{\partial n}\left[(1-2\nu)\delta_{ij}+2r_{,i}r_{,j}\right]+(1-2\nu)(n_jr_{,i}-n_ir_{,j})\right] \tag{1.13}$$

où :

$$\frac{\partial r}{\partial n} = n_1 r_{,1} + n_2 r_{,2} \qquad (1.14)$$

1.2.4 Equations intégrales en déplacements



a) Problème à résoudre : $(\vec{u}\ ,\ \vec{t})$ **b)** Problème de KELVIN : $(\vec{U}^i\ ,\vec{T}^i)$

FIG.1.3 DOMAINE Ω ET SA TRACE Ω' SUR LE PLAN INFINI

La figure (Fig.1.3.a) représente le domaine Ω du problème à résoudre :

- $u_j(Q)$ est la composante du vecteur déplacement d'un point Q suivant la direction $\vec{e_j}$;
- $t_j(Q)$ est la composante du vecteur tension suivant la direction $\vec{e_j}$ en un point Q sur une facette de normale $\vec{n}(Q)$.

Pour un problème bien posé, soit $u_j(Q)$, soit $t_j(Q)$, Q variant sur le contour $\partial\Omega$ et (j=1,2), est donné comme conditions aux limites, l'autre étant l'inconnue qu'on cherche. La figure (Fig.1.3.b) représente un plan infini, Ω' étant la trace de Ω sur ce plan. $\vec{F_i}$ est une force unitaire appliquée en un point P suivant la direction $\vec{e_j}$, la composante du vecteur déplacement en un point Q de $\partial\Omega'$ suivant la direction $\vec{e_j}$ est donnée par la solution analytique du problème de KELVIN $(U_j^i(P,Q))$, et la composante du vecteur tension en un point Q de $\partial\Omega'$ sur la droite tangente est $T_j^i(P,Q)$.

- **Si le point P est à l'extérieur de Ω' :**

Dans ce cas, Ω' est en équilibre sous l'action du champ des tensions $T_j^i(P,Q)$. Ainsi on a deux états où le domaine Ω est en équilibre élastostatique : le premier est le problème à résoudre $(u_j(Q), t_j(Q))$, le second est $(U_j^i(P,Q), T_j^i(P,Q))$. Alors d'après le théorème de Maxwell-Betti, on a :

$$\int_{\partial\Omega} T_j^i(P,Q)u_j(Q)ds(Q) = \int_{\partial\Omega} U_j^i(P,Q)t_j(Q)ds(Q) \qquad (1.15)$$

- **Si le point P est sur Ω' :**

Le domaine Ω' n'est plus en équilibre sous l'action du champ $T_j^i(P,Q)$. Par conséquent, l'équation (1.15) ne peut plus être écrite pour un point P appartenant à $\partial\Omega$.

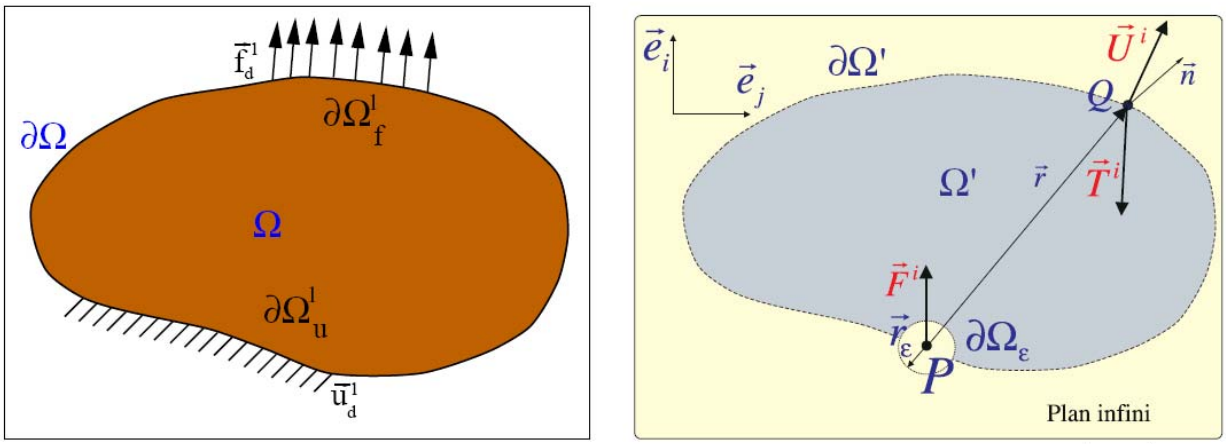


FIG.1.4 CAS OU LE POINT D'APPLICATION DE LA FORCE EST SUR LE CONTOUR $\partial\Omega$

Pour pouvoir écrire le théorème de MAXWELL-BETTI pour un point P du contour, on isole P par un cercle de rayon ε comme le montre la figure (Fig.1.4). Le nouveau contour d'intégration sera :

$$\Gamma = \partial\Omega - (\Omega_\varepsilon \cap \partial\Omega) + (\partial\Omega_\varepsilon \cap \Omega) \qquad (1.16)$$

où Ω_ε est la région du cercle, $\partial\Omega_\varepsilon$ son contour. On peut écrire alors :

$$\int_\Gamma T_j^i(P,Q)u_j(Q)ds(Q) = \int_\Gamma U_j^i(P,Q)t_j(Q)d(Q) \qquad (1.17)$$

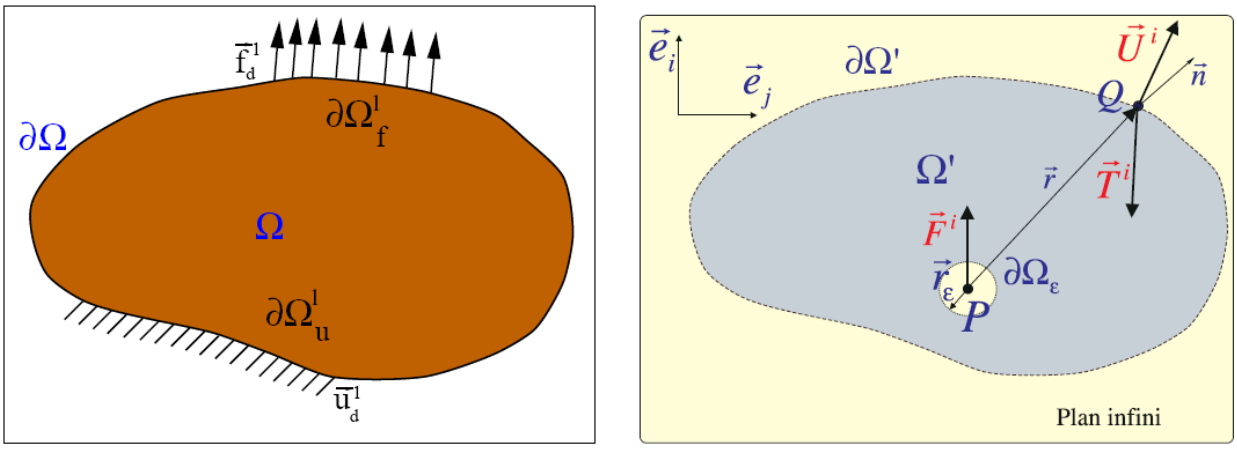
et en faisant tendre ε vers zéro, on obtient l'équation intégrale en déplacements :

$$C_{ij}u_i(P) + \int_{\partial\Omega} T_j^i(P,Q)u_j(Q)ds(Q) = \int_{\partial\Omega} U_j^i(P,Q)t_j(Q)ds(Q) \qquad (1.18)$$

où $C_{ij} = \frac{1}{2}\delta_{ij}$ si la normale en p est continue. Dans ce cas l'équation intégrale en déplacement s'écrit :

$$\frac{1}{2}u_i(P) + \int_{\partial\Omega} T_j^i(P,Q)u_j(Q)ds(Q) = \int_{\partial\Omega} U_j^i(P,Q)t_j(Q)ds(Q) \qquad (1.19)$$

1.2.5 Déplacements des points intérieurs



Problème à résoudre : $(\vec{u}\ ,\ \vec{t})$ Problème de KELVIN : $(\vec{U}^i\ ,\vec{T}^i)$

FIG.1.5 CAS OU LE POINT D'APPLICATION DE LA FORCE EST UN POINT INTERIEUR

- **Si le point P est à l'intérieur de Ω' :**

Pour pouvoir écrire le théorème de Maxwell-Betti, on isole P par un cercle de rayon ε comme l'indique la figure (Fig.1.5). Le nouveau contour d'intégration devient : $\Gamma = \partial\Omega + \partial\Omega_\varepsilon$

On a alors :

$$\int_\Gamma T_j^i(P,Q)u_j(Q)ds(Q) = \int_\Gamma U_j^i(P,Q)t_j(Q)ds(Q) \qquad (1.20)$$

en faisant tendre ε vers zéro, on obtient :

$$u_i(P) + \int_{\partial\Omega} T_j^i(P,Q)u_j(Q)ds(Q) = \int_{\partial\Omega} U_j^i(P,Q)t_j(Q)ds(Q) \qquad (1.21)$$

ou

$$u_i(P) = \int_{\partial\Omega} U_j^i(P,Q)t_j(Q)ds(Q) - \int_{\partial\Omega} T_j^i(P,Q)u_j(Q)ds(Q) \qquad (1.22)$$

Cette équation est connue sous le nom de « **Identité de Somigliana** » pour les déplacements. Elle permet de donner la valeur du déplacement en tous points intérieurs en termes de déplacement et d'efforts (u_i et t_i) et pour tous points P où l'effort unitaire est appliqué, une fois que u_i et t_i sont connus en tous points de la frontière.

1.2.6 Contraintes des points intérieurs

La composante σ_{ij} du tenseur des contraintes est calculée à partir de la loi de comportement de Hooke (1.11) en injectant la valeur $u_i(P)$ de l'équation (1.22) dans (1.11) :

$$\sigma_{ij}(P) = \int_{\partial\Omega} D_{jk}^i(P,Q)t_k(Q)ds(Q) - \int_{\partial\Omega} S_{jk}^i(P,Q)u_k(Q)ds(Q) \qquad (1.23)$$

où

$$D^i_{jk} = \frac{1}{4\pi(1-\nu)r}\Big[(1-2\nu)(\delta_{ik}r_{,j} + \delta_{jk}r_{,i} - \delta_{ik}r_{,j}) + 2r_{,i}r_{,j}r_{,k}\Big] \tag{1.24}$$

et

$$\begin{aligned} S^i_{jk} = \frac{2\mu}{4\pi(1-\nu)r^2}\Bigg\{ & 2\frac{\partial r}{\partial n}\Big[(1-2\nu)\delta_{ij}r_{,k} + \nu(\delta_{ik}r_{,j} + \delta_{jk}r_{,i}) - 4r_{,i}r_{,j}r_{,k}\Big] \\ & + \nu(n_i r_{,j}r_{,k} + n_{,j}r_{,i}r_{,k}) + (1-2\nu)(2n_k r_{,i}r_{,j} + n_j\delta_{ik} + n_i\delta_{jk}) - (1-4\nu)n_k\delta_{ij}\Bigg\} \end{aligned} \tag{1.25}$$

1.2.7 Equations intégrales en tension

Le tenseur des contraintes σ_{ij} est obtenu en appliquant la loi de Hooke aux équations de déplacement. L'application de la loi de Hooke nous permet de relier la composante σ_{ij} du tenseur des contraintes à la solution de Kelvin et aux tensions et déplacements du contour.

L'équation (1.23) donne le tenseur des contraintes pour un point P à l'intérieur du domaine $(r \neq 0)$. En prenant P sur le contour et en suivant le même raisonnement que pour l'équation (1.19), on obtient l'équation intégrale en contraintes :

$$\begin{aligned} & \frac{1}{2}\sigma_{ij}(P) + \int_{\partial\Omega} S^i_{jk}(P,Q)u_k(Q)ds(Q) \\ & = \int_{\partial\Omega} D^i_{jk}(P,Q)t_k(Q)ds(Q) \end{aligned} \tag{1.26}$$

avec les conditions : $u_k(P) \in C^1$ et $t_k(P) \in C^1$

En multipliant l'équation (1.26) par $n_j(P)$, on obtient l'équation intégrale en tensions :

$$\begin{aligned} & \frac{1}{2}t_i(P) + n_j(P)\int_{\partial\Omega} S^i_{jk}(P,Q)u_k(Q)ds(Q) \\ & = n_j(P)\int_{\partial\Omega} D^i_{jk}(P,Q)t_k(Q)ds(Q) \end{aligned} \tag{1.27}$$

1.2.8 Contraintes des points du contour

Pour calculer le tenseur des contraintes pour un point P situé sur le contour, trois méthodes sont envisageables :

- connaissant le champ des déplacements sur le contour, on peut déterminer celui des contraintes variant linéairement sur un élément. Cette méthode nous donne des résultats peu précis.
- En calculant les contraintes pour un point intérieur assez proche de P puis en faisant une extrapolation. Cela ne donne pas de bons résultats pour les zones de fortes variations de contraintes.

- La troisième méthode, la plus précise, est d'utiliser la formule (1.26) comme une formule des contraintes sur le contour. Ainsi on peut écrire :

$$\sigma_{ij}(P) = 2\int_{\partial\Omega} D^i_{jk}(P,Q)t_k(Q)ds(Q) - 2\int_{\partial\Omega} S^i_{jk}(P,Q)u_k(Q)ds(Q) \quad (1.28)$$

1.3 Equations intégrales pour les structures fissurées

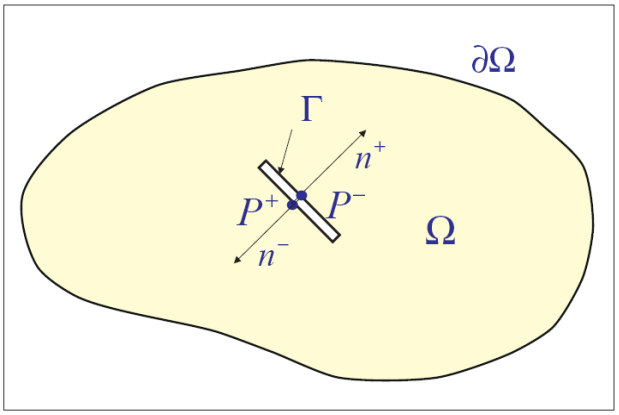


FIG.1.6 STRUCTURE FISSUREE

Les équations intégrales de frontière (1.19) et (1.27) sont valables telles quelles pour un solide élastique non fissuré, la présence d'une fissure entraîne des difficultés supplémentaires. Considérons donc un solide élastique Ω, contenant une fissure Γ ; les deux lèvres Γ^+, Γ^- géométriquement confondus et libres de contraintes, sont munies de normales unitaire n^+ et n^- opposées et choisies de sorte que n^- soit orientée de Γ^+ vers Γ^- (ceci pour rester cohérent avec la convention d'orientation selon laquelle la normale est toujours dirigée vers l'extérieur du domaine élastique).

1.3.1 Equation intégrales en déplacement pour les points de la fissure

Décomposons par la pensée Ω en deux sous domaine Ω^+ et Ω^- séparées par un contour C et la fissure Γ (Fig.1.7).

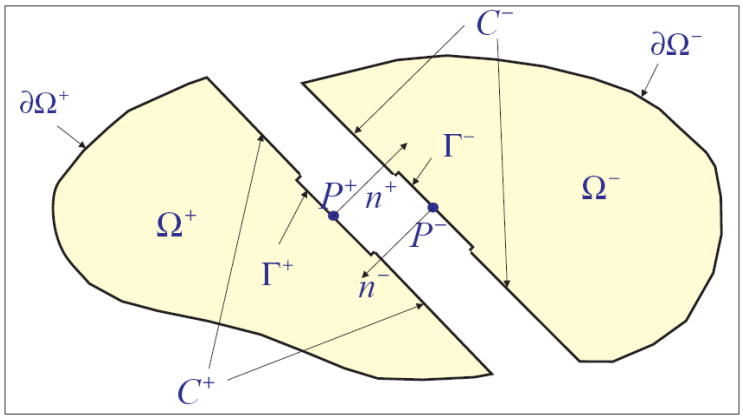


FIG.1.7 DECOMPOSITION DE LA STRUCTURE EN DEUX PARTIES AU NIVEAU DE LA FISSURE

On peut écrire, pour un point P^+ appartenant à la lèvre Γ^+ de la fissure, l'équation intégrale en déplacement (1.19) :

$$\begin{aligned}&\frac{1}{2}u_i(P^+)+\int_{\partial\Omega^+\cup C^+\cup\Gamma^+}T_j^i(P^+,Q)u_j(Q)ds(Q)\\&=\int_{\partial\Omega^+\cup C^+\cup\Gamma^+}U_j^i(P^+,Q)t_j(Q)ds(Q)\end{aligned}\tag{1.29}$$

De la même manière, l'équation intégrale en déplacement (1.19) pour un point P^- appartenant à la lèvre Γ^- de la fissure, s'écrit :

$$\begin{aligned}&\frac{1}{2}u_i(P^-)+\int_{\partial\Omega^-\cup C^-\cup\Gamma^-}T_j^i(P^-,Q)u_j(Q)ds(Q)\\&=\int_{\partial\Omega^-\cup C^-\cup\Gamma^-}U_j^i(P^-,Q)t_j(Q)ds(Q)\end{aligned}\tag{1.30}$$

Tous calculs faits, l'addition membre à membre des deux égalités (1.29) et (1.30) prises pour le même point $P = P^+ = P^-$, donne, compte tenu des conditions de raccord et l'opposition des normales n^+ et n^- sur C :

$$\begin{aligned}&\frac{1}{2}\left(u_i(P^+)+u_i(P^-)\right)+\int_{\partial\Omega\cup\Gamma^+\cup\Gamma^-}T_j^i(P,Q)u_j(Q)ds(Q)\\&=\int_{\partial\Omega\cup\Gamma^+\cup\Gamma^-}U_j^i(P,Q)t_j(Q)ds(Q)\end{aligned}\tag{1.31}$$

Et encore, sachant que $\Gamma=\Gamma^+\cup\Gamma^-$:

$$\begin{aligned}&\frac{1}{2}\left(u_i(P^+)+u_i(P^-)\right)+\int_{\partial\Omega\cup\Gamma}T_j^i(P,Q)u_j(Q)ds(Q)\\&=\int_{\partial\Omega\cup\Gamma}U_j^i(P,Q)t_j(Q)ds(Q)\end{aligned}\tag{1.32}$$

L'équation (1.32) représente l'équation intégrale en déplacement pour les points de la fissure.

1.3.2 Equation intégrale en tension pour les points de la fissure

En suivant le même raisonnement que pour l'équation intégrale en déplacement,l'équation intégrale en contrainte pour un point $P = P^+ = P^-$ situé sur les lèvres de la fissure s'écrit :

$$\begin{aligned}&\frac{1}{2}\left(\sigma_{ij}(P^+)+\sigma_{ij}(P^-)\right)+\int_{\partial\Omega\cup\Gamma}S_{jk}^i(P,Q)u_k(Q)ds(Q)\\&=\int_{\partial\Omega\cup\Gamma}D_{jk}^i(P,Q)t_k(Q)ds(Q)\end{aligned}\tag{1.33}$$

La multiplication des deux membres de l'équation (1.33) par la normale n^+ du point P^+ou n^- du point P^- nous donne l'équation intégrale en tension :

$$\frac{1}{2}\left(t_i(P^+)+t_i(P^-)\right)+n_j(P^+)\int_{\partial\Omega\cup\Gamma} S^i_{jk}(P,Q)u_k(Q)ds(Q)$$
$$=n_j(P^+)\int_{\partial\Omega\cup\Gamma} D^i_{jk}(P,Q)t_k(Q)ds(Q) \qquad (1.34)$$

Pour une fissure non chargée l'équation intégrale en tension se réduit à :

$$n_j(P^+)\int_{\partial\Omega\cup\Gamma} S^i_{jk}(P,Q)u_k(Q)ds(Q)=n_j(P^+)\int_{\partial\Omega} D^i_{jk}(P,Q)t_k(Q)ds(Q) \qquad (1.35)$$

1.3.3 Equation intégrale en déplacement pour les points du contour

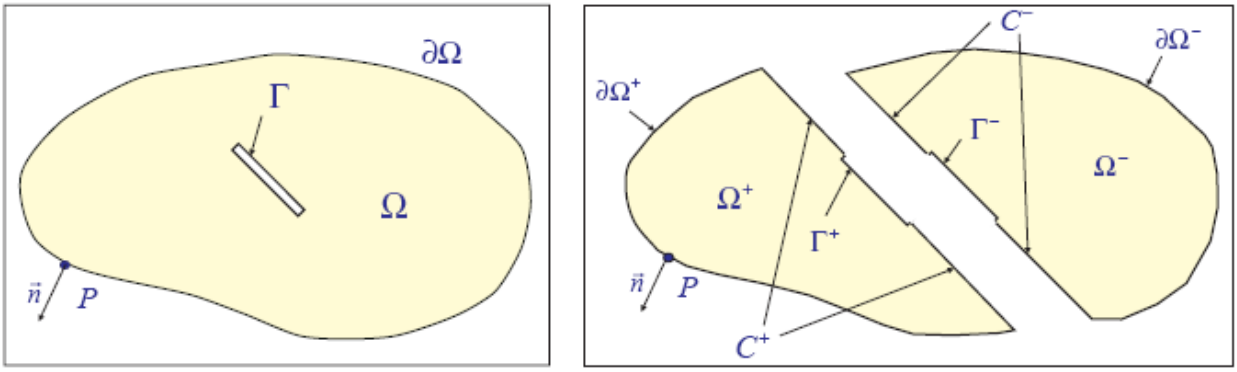


FIG.1.8 STRUCTURE FISSUREE (CAS OU $P \notin \Gamma$)

Si le point P n'appartient aux lèvres de la fissure, P appartenant a $\partial\Omega^+$ par exemple, l'équation intégrale en déplacements (1.19) s'écrit :

$$\frac{1}{2}u_i(P)+\int_{\partial\Omega^+\cup C^+\cup\Gamma^+} T^i_j(P,Q)u_j(Q)ds(Q)=\int_{\partial\Omega^+\cup C^+\cup\Gamma^+} U^i_j(P,Q)t_j(Q)ds(Q) \qquad (1.36)$$

Par contre, puisque $P \notin \Omega^-$, on va utiliser le théorème de Maxwell-Betti (1.15) pour écrire une relation intégrale sur Ω^-, ce qui nous donne :

$$\int_{\partial\Omega^-\cup C^+\cup\Gamma^-} T^i_j(P,Q)u_j(Q)ds(Q)=\int_{\partial\Omega^-\cup C^-\cup\Gamma^-} U^i_j(P^+,Q)t_j(Q)ds(Q) \qquad (1.37)$$

L'addition membre à membre des deux égalités (3.8) et (3.9) donne, compte tenu des conditions de raccord et l'opposition des normales n^+ et n^- sur C :

$$\frac{1}{2}u_i(P)+\int_{\partial\Omega\cup\Gamma} T^i_j(P,Q)u_j(Q)ds(Q)=\int_{\partial\Omega\cup\Gamma} U^i_j(P,Q)t_j(Q)ds(Q) \qquad (1.38)$$

Ce qui représente l'équation intégrale en déplacement pour un point du contour.

1.3.4 Equation intégrale en tension pour les points du contour

La même démarche suivie pour l'obtention de l'équation intégrale en déplacement pour un point P du contour nous conduit à l'équation intégrale en tension appliquée pour le même point P, ainsi on obtient :

$$\frac{1}{2}t_i(P)+n_j(P)\int_{\partial\Omega\cup\Gamma}S_{jk}^{i}(P,Q)u_k(Q)ds(Q)=n_j(P)\int_{\partial\Omega\cup\Gamma}D_{jk}^{i}(P,Q)t_k(Q)ds(Q) \qquad (1.39)$$

1.3.5 Déplacements et contraintes des points du contour

Supposons que le point P soit à l'intérieur du sous domaine Ω^+, on peut alors écrire l'identité de Somigliana (1.22) :

$$u_i(P)=\int_{\partial\Omega^+\cup C^+\cup\Gamma^+}U_j^i(P,Q)t_j(Q)ds(Q)-\int_{\partial\Omega^+\cup C^+\cup\Gamma^+}T_j^i(P,Q)u_j(Q)ds(Q) \qquad (1.40)$$

D'autre part, le théorème de réciprocité de Maxwell-Betti (1.15) s'écrit sur le sous-domaine Ω^- :

$$\int_{\partial\Omega^-\cup C^-\cup\Gamma^-}T_j^i(P,Q)t_j(Q)ds(Q)=\int_{\partial\Omega^-\cup C^-\cup\Gamma^-}U_j^i(P,Q)t_j(Q)ds(Q) \qquad (1.41)$$

L'addition membre à membre à membre des deux égalités (1.40) et (1.41) nous donne l'identité de Somigliana pour une structure fissurée :

$$u_i(P)=\int_{\partial\Omega\cup\Gamma}U_j^i(P,Q)t_j(Q)ds(Q)-\int_{\partial\Omega\cup\Gamma}T_j^i(P,Q)u_j(Q)ds(Q) \qquad (1.42)$$

La composante σ_{ij} du tenseur des contraintes est obtenus à partir de la loi de comportement de Hooke (1.11), en injectant la valeur de $u_i(P)$ de l'équation (1.42) dans (1.11), on obtient :

$$\sigma_{ij}(P)=\int_{\partial\Omega\cup\Gamma}D_{jk}^{i}(P,Q)t_j(Q)ds(Q)-\int_{\partial\Omega\cup\Gamma}S_{jk}^{i}(P,Q)u_j(Q)ds(Q) \qquad (1.43)$$

1.3.6 Contraintes des points de contours

La composante σ_{ij} du tenseur des contraintes d'un point P situé sur le contour $\partial\Omega$ se calcul à partir de l'équation intégrale en contrainte d'une structure fissurée, ce qui nous donne :

$$\sigma_{ij}(P)=2\int_{\partial\Omega\cup\Gamma}D_{jk}^{i}(P,Q)t_k(Q)ds(Q)-2\int_{\partial\Omega\cup\Gamma}S_{jk}^{i}(P,Q)u_k(Q)ds(Q) \qquad (1.44)$$

Chapitre 2

Traitement numérique des équations intégrales

Les solutions analytiques des équations intégrales relatives aux problèmes d'élasticité sont extrêmement rares et n'existent que pour quelques géométries et conditions aux limites simples.

Pour être en mesure de traiter des cas plus complexes qui correspondent à ceux rencontrés dans la pratique, on est amené à la résolution numérique de ces problèmes. On approche alors la solution du problème avec une erreur qui dépend généralement de :

- la représentation de la géométrie ;
- le nombre d'éléments utilisés ;
- la représentation des champs élastiques ;
- la méthode d'intégration.

Cette partie est consacrée à la présentation des principales étapes du traitement numérique de l'équation intégrale en déplacements.

2.1 Traitement numérique des équations intégrales pour les structures non fissurées

2.1.1 Discrétisation de la géométrie

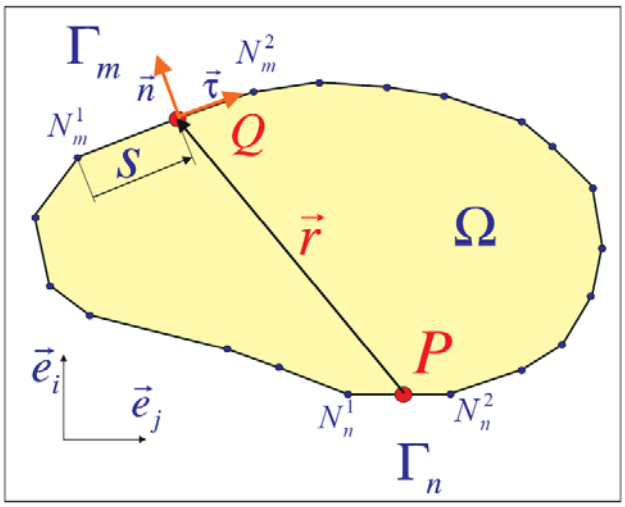


FIG.2.1 DISCRETISATION DE LA GEOMETRIE

La frontière $\partial\Omega$ est discrétisée en NEG (nombre d'éléments de la géométrie)segments de droite Γ_m (m=1, NEG), chaque élément est défini par deux nœuds N_m^1 et N_m^2. Connaissant les coordonnées de ces nœuds, on détermine les coordonnées d'un point quelconque Q de l'élément à l'aide des fonctions de forme linéaire :

$$x_Q(s) = x_{N_m^1} N_1(s) + x_{N_m^2} N_2(s) \tag{2.1}$$

$$y_Q(s) = y_{N_m^1} N_1(s) + y_{N_m^2} N_2(s) \tag{2.2}$$

avec

$$N_1(s) = 1 - \frac{s}{L_m} \tag{2.3}$$

$$N_2(s) = \frac{s}{L_m} \tag{2.4}$$

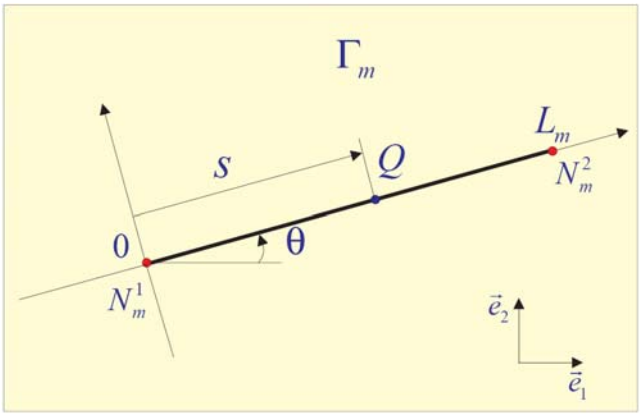


FIG.2.2 ELEMENT LINEAIRE POUR LA REPRESENTATION DE LA GEOMETRIE

où L_m est la longueur de l'élément Γ_m .

Nous avons adopté ce type d'élément pour pouvoir évaluer analytiquement les intégrales qui apparaissent dans la formulation des équations intégrales. Un autre choix de type d'élément, quadratique par exemple, nous obligera à les évaluer numériquement par la méthode de quadrature de Gauss, ce qui est très coûteux en temps de calcul.

2.1. 2 Discrétisation des champs élastiques

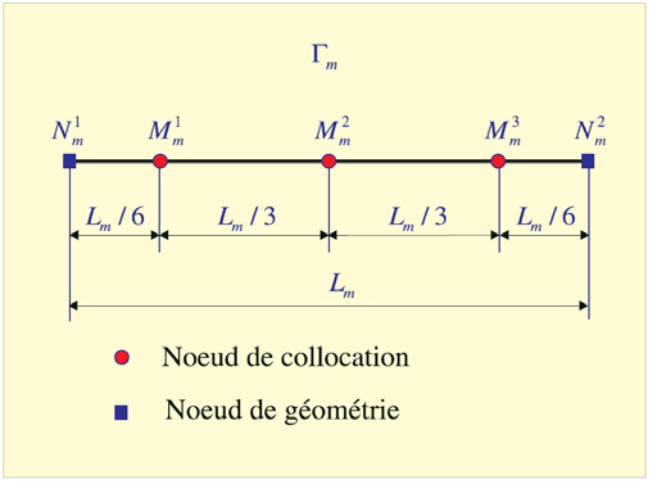


FIG.2.3 ELEMENT QUADRATIQUE NON CONFORME

La formulation des équations intégrales en déplacements n'impose aucune condition sur les champs des déplacements et des tensions, par conséquent, on peut prendre une variation constante sur chaque élément

$$u_j \in C^{-1} \qquad t_j \in C^{-1} \tag{2.5}$$

Pour d'autre cas (problèmes de fissures) on doit prendre une variation au moins quadratique des champs de déplacements et de tensions, soient :

$$u_j \in C^{1} \qquad t_j \in C^{1} \tag{2.6}$$

Les points de collocation ont été choisis à l'intérieur de l'élément de discrétisation pour éviter les problèmes dus à la discontinuité de la normale (le terme libre $C_{ij}(P,Q)$ de l'équation intégrale change en fonction de la discontinuité de la normale), ainsi que pour pouvoir traiter les problèmes de singularité en fond de fissure.

Sur un élément Γ_n on a :

$$u_i(s) = \sum_{l=1}^{3} u_i(M_n^l) N_l(s) \tag{2.7}$$

et

$$t_i(s) = \sum_{l=1}^{3} t_i(M_n^l) N_l(s) \tag{2.8}$$

où M_n^l (l=1,3) sont les points de collocation de l'élément Γ_n, et avec :

$$N_1(s) = \frac{15}{8} - \frac{6}{L_n} s + \frac{9}{2L_n^2} s^2 \tag{2.9}$$

$$N_2(s) = -\frac{5}{4} + \frac{9}{L_n} s + \frac{9}{L_n^2} s^2 \tag{2.10}$$

$$N_3(s) = \frac{3}{8} - \frac{3}{L_n} s + \frac{9}{2L_n^2} s^2 \tag{2.11}$$

2.1.3 Discrétisation de l'équation intégrale

L'équation intégrale en déplacements (1.19) appliquée à un point de collocation ($P = M_n^k$) de l'élément Γ_n s'écrit :

$$\frac{1}{2} u_i(M_n^k) + \int_{\partial\Omega} T_j^i(M_n^k, Q) u_j(Q) ds(Q) = \int_{\partial\Omega} U_j^i(M_n^k, Q) t_j(Q) ds(Q) \tag{2.12}$$

En discrétisant le contour $\partial\Omega$ en NEG éléments $\Gamma_m (m = 1, NEG)$, on aura :

$$\frac{1}{2}u_i(M_n^k)+\sum_{m=1}^{NEG}\int_{\Gamma_m}T_j^i(M_n^k,Q)u_j(Q)ds(Q)=\sum_{m=1}^{NEG}\int_{\Gamma_m}U_j^i(M_n^k,Q)t_j(Q)ds(Q) \tag{2.13}$$

et en remplaçant la valeur des variables par leurs représentations, on obtient :

$$\frac{1}{2}u_i(M_n^k)+\sum_{m=1}^{NEG}\int_{\Gamma_m}T_j^i(M_n^k,Q)\left(\sum_{l=1}^{3}N_l(s)u_j(M_m^l)\right)ds(Q)=\sum_{m=1}^{NEG}\int_{\Gamma_m}U_j^i(M_n^k,Q)\left(\sum_{l=1}^{3}N_l(s)t_j(M_m^l)\right)ds(Q) \tag{2.14}$$

soit :

$$\frac{1}{2}u_i(M_n^k)+\sum_{m=1}^{NEG}\sum_{l=1}^{3}\sum_{j=1}^{2}\left(\int_0^{Lm}T_j^i(M_n^k,s)N_l(s)ds\right)u_j(M_m^l)=\sum_{m=1}^{NEG}\sum_{l=1}^{3}\sum_{j=1}^{2}\left(\int_0^{Lm}U_j^i(M_n^k,s)N_l(s)ds\right).t_j(M_m^l) \tag{2.15}$$

Finalement, on a :

$$\sum_{m=1}^{NEG}\sum_{l=1}^{3}\sum_{j=1}^{2}\left(\frac{1}{2}\delta_{mn}\delta_{kl}\delta_{ij}+\int_0^{Lm}T_j^i(M_n^k,s)N_l(s)ds\right)u_j(M_m^l)=\sum_{m=1}^{NEG}\sum_{l=1}^{3}\sum_{j=1}^{2}\left(\int_0^{Lm}U_j^i(M_n^k,s)N_l(s)ds\right)t_j(M_m^l) \tag{2.16}$$

C'est une équation à $(12\times NEG)$ variables. En faisant varier le point de collocation M_n^k sur tous les éléments (n=1,NEG), sur chaque point de collocation k(k=1,3) et suivant chaque direction $\vec{e_i}(i=1,2)$, on obtient ainsi un système de $(6\times \mathrm{NEG})$ équations à $(12\times NEG)$ variables de la forme :

$$[A]\{u\}=[B]\{t\} \tag{2.17}$$

2.1.4 Calcul des intégrales

C'est l'étape la plus importante de la méthode des équations intégrales. En effet il faut déterminer avec précision les coefficients des matrices [A] et [B] pour obtenir une bonne approximation de la solution du problème. Le nombre de coefficients à déterminer est égal à $2\times(6\times NEG\times 6\times NEG)$, soit $72\times(NEG)^2$ coefficients, ce qui prend une grande partie du temps de calcul. La solution idéale est de les évaluer analytiquement. C'est ce

que nous avons finalement choisi. Pour cela il est nécessaire de discrétiser la géométrie en segment de droite pour pouvoir effectuer une intégration analytique.

2.1.5 Constitution du système d'équations

La construction du système d'équations consiste à écrire l'équation intégrale en déplacement discrétisée (1.39) pour les points de collocation M_m^k(k=1,N_{pc}) de chaque élément du contour Γ_m(m=1, NEG) suivant les deux directions, N_{pc} étant le nombre de nœuds de collocation par élément. Ainsi, on obtient un système d'équations algébriques de $(6\times NEG)$ équations à $(12\times NEG)$ variables :

Après discrétisation de l'équation intégrale en déplacements on est arrivé à un système

$$[A]\{u\}=[B]\{t\} \tag{2.18}$$

Le coefficient des matrices [A] et [B] ne sont pas du même ordre de grandeur. En effet, A_{ij} est du même ordre de grandeur que r tandis que B_{ij} est du même ordre de grandeur que $\frac{r}{2G}$ (G étant le module de cisaillement). Pour obtenir un système bien conditionné, on doit multiplier les coefficients de la matrice [B] par 2G. On aura alors à résoudre :

$$[A]\{u\}=(2G)[B]\left\{t'=\frac{t}{2G}\right\} \tag{2.19}$$

Les vecteurs {u} et {t} contiennent les valeurs du déplacement et des tentions avant l'application des conditions aux limites. $(6\times NEG)$ de ces variables sont données comme conditions aux limites. Ces conditions peuvent être introduites par un réarrangement dans les colonnes de [A] et [B] ce qui nous permet de transformer le système à résoudre en mettant les variables connues d'un coté et les inconnues de l'autre coté et obtenir un système d'équations de $(6\times NEG)$ équations à $(6\times NEG)$ inconnues de la forme :

$$[K]\{x\}=\{y\} \tag{2.20}$$

où {x} est le vecteur qui contient toutes les inconnues en déplacement et en tension.

Après la résolution par une méthode approprié, les valeurs des tentions trouvées seront multipliées par 2G pour avoir la solution du problème.

2.1.6 Résolution du système d'équations :

La matrice [K] du système à résoudre est une matrice pleine, non symétrique et à diagonale dominante, ce qui représente une différence considérable par rapport aux méthodes des éléments finis, pour lesquelles il est bien connu que la matrice de rigidité est symétrique et bande. Pour les matrices de taille moyenne, les méthodes directes sont très efficaces. Par contre, pour les matrices de grande taille, et du fait que la matrice [K] est à diagonale dominante les méthodes itératives sont les plus efficaces. Cela représente le cœur de notre travail qui consiste à implémenter une des plus célèbres méthodes itératives adaptée à ce type de système d'équations, à savoir la méthode GMRES.

2.1.7 Calcul des déplacements des points intérieurs

Le champ des déplacements des points intérieurs P se calcule par une simple intégration, de l'équation (1.22), on a :

$$u_i(P) = \int_{\partial\Omega} U_j^i(P,Q)t_j(Q)ds(Q) - \int_{\partial\Omega} T_j^i(P,Q)u_j(Q)ds(Q) \tag{2.21}$$

En discrétisant cette équation, on obtient :

$$\begin{aligned} u_i(P) &= \sum_{m=1}^{NEG}\sum_{l=1}^{3}\sum_{j=1}^{2}\int_0^{L_m}\left(U_j^i(P,s)N_l(s)ds\right)t_j(M_m^l) \\ &-\sum_{m=1}^{NEG}\sum_{l=1}^{3}\sum_{j=1}^{2}\int_0^{L_m}\left(T_j^i(P,s)N_l(s)ds\right)u_j(M_m^l) \end{aligned} \tag{2.22}$$

2.1.8 Calcul des contraintes des points intérieurs

Le tenseur des contraintes d'un point intérieur P se calcule par une simple sommation, de l'équation (1.21) on a :

$$\sigma_{ij}(P) = \int_{\partial\Omega} D_{jk}^i(P,Q)t_k(Q)ds(Q) - \int_{\partial\Omega} S_{jk}^i(P,Q)u_k(Q)ds(Q) \tag{2.23}$$

On peut écrire alors après discrétisation :

$$\begin{aligned} \sigma_{ij}(P) &= \sum_{m=1}^{NEG}\sum_{l=1}^{3}\sum_{k=1}^{2}\left(\int_0^{L_m} D_{jk}^i(P,s)N_l(s)ds\right)t_k(M_m^l) \\ &-\sum_{m=1}^{NEG}\sum_{l=1}^{3}\sum_{k=1}^{2}\left(\int_0^{L_m} S_{jk}^i(P,s)N_l(s)ds\right)u_k(M_m^l) \end{aligned} \tag{2.24}$$

2.1.9 Calcul des contraintes des points du contour

Le tenseur des contraintes d'un point P du contour se calcule par simple sommation à partir de l'équation (1.28) :

$$\sigma_{ij}(P) = 2\left\{\int_{\partial\Omega} D_{jk}^i(P,Q)t_k(Q)ds(Q) - \int_{\partial\Omega} S_{jk}^i(P,Q)u_k(Q)ds(Q)\right\} \tag{2.25}$$

Soit :

$$\sigma_{ij}(P) = 2\left(\sum_{m=1}^{NEG} \sum_{l=1}^{3} \sum_{k=1}^{2} \left(\int_0^{L_m} D_{jk}^{i}(P,s) N_l(s) ds \right) t_k(M_m^l) \right.$$
$$\left. - \sum_{m=1}^{NEG} \sum_{l=1}^{3} \sum_{k=1}^{2} \left(\int_0^{L_m} S_{jk}^{i}(P,s) N_l(s) ds \right) u_k(Q_m^l) \right) \quad (2.26)$$

2.1.10 Algorithme :

1. Lecture des données: Géométrie du problème, conditions limites et propriétés du matériau.
2. Maillage: discrétisation du contour en *NEG* éléments Γ_m sinon on récupère des structure déjà maillée avec d'autre logiciel comme l'ABAQUS
3. remplissage des matrices [A] et [B] :

Boucle sur les éléments Γ_n (*n=1, NEG*)

Boucle sur les nœuds de collocation M_k^n (*k=1,3*)

Boucle sur les directions $\vec{e_i}$ (*i=1,2*)

Ecriture de l'équation intégrale (..)

Boucle sur les éléments Γ_m (*m=1, NEG*)

Boucle sur les nœuds de collocation M_m^l (*l=1,3*)

Boucle sur les directions $\vec{e_j}$ (j =*1,2*)

Evaluation de l'intégrale :

$$\int_0^{L_m} T_j^i(M_n^k, s) N_l(s) ds \,,\; \int_0^{L_m} U_j^i(M_n^k, s) N_l(s) ds$$

4. Introduction des conditions aux limites et constitution du système d'équations :

$$[K]\{x\} = \{y\}$$

5. Résolution du système d'équations par la méthode de décomposition de Gauss et obtention des inconnues du contour : $\{u\}$ et $\{t\}$
6. Calcul des déplacements des points intérieurs par la formule (1.22)
7. Calcul des contraintes des points intérieurs par la formule (1.23)
8. Calcul des contraintes des points du contour par la formule (1.24)
9. Affichage des résultats.

2.1.11 Organigramme

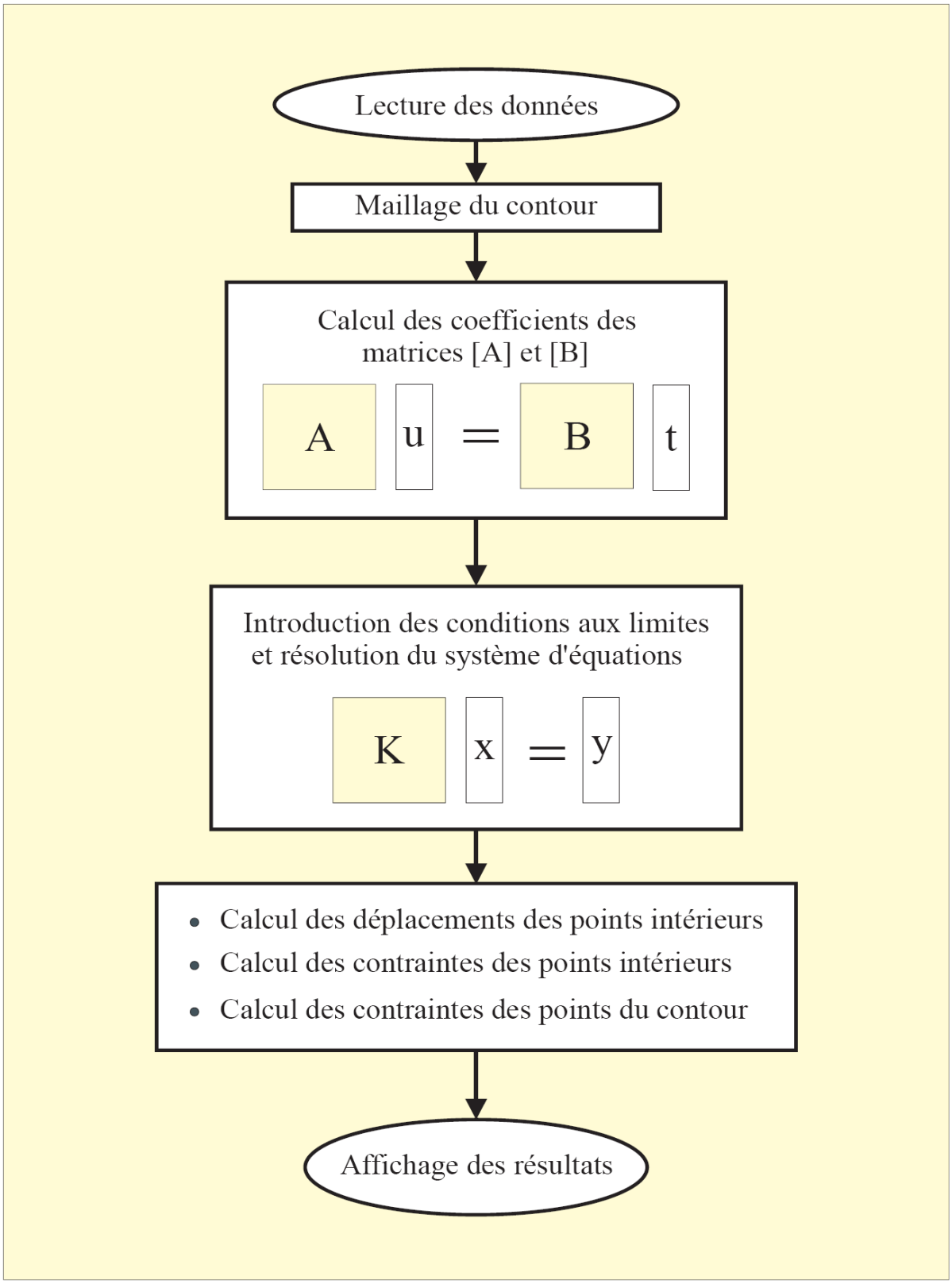


FIG.2.4 ORGANIGRAMME

2.2 Traitement numérique des équations intégrales duales pour les structures fissurées

2.2.1 Discrétisation de la géométrie et des champs élastiques

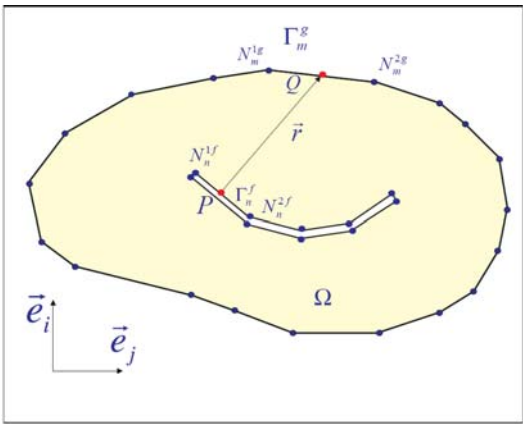


FIG.2.5 DISCRETISATION DE LA GEOMETRIE

Le contour $\partial\Omega$ de la structure est discrétisé en NEG segments de droite Γ_m^g ($m=1, NEG$), chaque élément est défini par deux nœuds de géométrie N_m^{1g} et N_m^{2g}. La fissure est discrétisée en $2\times NEF$ segments de droite Γ_m^f (m=$1, 2\times NEF$), les lèvres Γ^+ et Γ^- contenant NEF éléments chacune. Les éléments Γ_{2i+1}^f et Γ_{2i+2}^f ($i=1, NEF$-1) sont géométriquement confondus et de normales opposées :

$$\vec{n}\left(\Gamma_{2i+1}^f\right) = -\vec{n}\left(\Gamma_{2i+2}^f\right) \tag{2.27}$$

Les éléments sont définis par deux nœuds de géométrie N_m^{1f} et N_m^{2f}. La position d'un point quelconque Q sur l'élément de frontière Γ_m^k (k= f , g) est définie par la donnée des nœud de géométrie de l'élément et des fonction de forme linéaires $N_1(s)$ et $N_2(s)$ (2.4)

$$\begin{cases} x_Q(s) = x_{N_m^{1k}} N_1(s) + x_{N_m^{2k}} N_2(s) \\ y_Q(s) = y_{N_m^{1k}} N_1(s) + y_{N_m^{2k}} N2(s) \end{cases} \qquad (\text{k} = f, g) \tag{2.28}$$

2.2.2 Discrétisation des champs élastiques

Les champs élastiques de déplacements $\vec{u}(Q)$ et de la tension $\vec{t}(Q)$ ($Q \in \partial\Omega \cup \Gamma$) sont interpolés par des fonctions de forme quadratiques (2.9),(2.10)et(2.11). Les trois points de collocation ont été choisis à l'intérieur de l'élément de discrétisation. Dans un premier temps le fond de la fissure est discrétisé par le même type d'élément (élément quadratique non conforme), un autre type d'élément (élément singulier) sera utilisé pour la discrétisation du fond de fissure.

2.2.3 Discrétisation des équations intégrales duales

Discrétisation de l'équation intégrale en déplacement pour un point *P* de la fissure :

L'équation intégrale en déplacement (1.32) appliquée à un point de collocation ($P = M_n^{fk}$) de l'élément de fissure Γ_n^f située sur la lèvre Γ^+ s'écrit :

$$\frac{1}{2}u_i\left(M_n^{fk}\right)+\frac{1}{2}u_i M_{n+1}^{f(4-k)}+\int_{\partial\Omega\cup\Gamma} T_j^i\left(M_n^{fk},Q\right)u_j(Q)ds(Q)=\int_{\partial\Omega\cup\Gamma} U_j^i\left(M_n^{fk},Q\right)t_j(Q)ds(Q) \quad (2.29)$$

En discrétisant le contour $\partial\Omega$ en *NEG* éléments Γ_m^g (*m=1, NEG*) et la fissure Γ en $2\times NEF$ éléments Γ_m^f (m=1, $2\times NEF$) on aura :

$$\begin{aligned}&\frac{1}{2}u_i\left(M_n^{fk}\right)+\frac{1}{2}u_i M_{n+1}^{f(4-k)}+\sum_{m=1}^{NEG}\int_{\Gamma_m^g} T_j^i\left(M_n^{fk},Q\right)u_j(Q)ds(Q)\\&+\sum_{m=1}^{2\times NEF}\int_{\Gamma_m^f} T_j^i\left(M_n^{fk},Q\right)u_j(Q)ds(Q)\\&=\sum_{m=1}^{NEG}\int_{\Gamma_m^g} U_j^i\left(M_n^{fk},Q\right)t_j(Q)ds(Q)+\sum_{m=1}^{2\times NEF}\int_{\Gamma_m^f} U_j^i\left(M_n^{fk},Q\right)t_j(Q)ds(Q)\end{aligned} \quad (2.30)$$

En remplaçant la valeur des variables $u_j(Q)$ et $t_j(Q)$ par leurs représentations respectives (2.7) et (2.8) on obtient :

$$\begin{aligned}&\frac{1}{2}u_i\left(M_n^{fk}\right)+\frac{1}{2}u_i M_{n+1}^{f(4-k)}+\sum_{m=1}^{NEG}\int_{\Gamma_m^g} T_j^i\left(M_n^{fk},Q\right)\left(\sum_{l=1}^{3}N_l(s)u_j\left(M_m^{gl}\right)\right)ds(Q)\\&+\sum_{m=1}^{2\times NEF}\int_{\Gamma_m^f} T_j^i\left(M_n^{fk},Q\right)\left(\sum_{l=1}^{3}N_l(s)u_j\left(M_m^{fl}\right)\right)ds(Q)\\&=\sum_{m=1}^{NEG}\int_{\Gamma_m^g} T_j^i\left(M_n^{fk},Q\right)\left(\sum_{l=1}^{3}N_l(s)t_j\left(M_m^{gl}\right)\right)ds(Q)\\&+\sum_{m=1}^{2\times NEF}\int_{\Gamma_m^f} T_j^i\left(M_n^{fk},Q\right)\left(\sum_{l=1}^{3}N_l(s)t_j\left(M_m^{fl}\right)\right)ds(Q)\end{aligned} \quad (2.31)$$

Soit :

$$\begin{aligned}&\frac{1}{2}u_i\left(M_n^{fk}\right)+\frac{1}{2}u_i M_{n+1}^{f(4-k)}+\sum_{m=1}^{NEG}\sum_{l=1}^{3}\sum_{j=1}^{2}\left(\int_0^{L_m^g} T_j^i(M_n^{fk},s)N_l(s)ds\right)u_j\left(M_m^{gl}\right)\\&+\sum_{m=1}^{2\times NEF}\sum_{l=1}^{3}\sum_{j=1}^{2}\left(\int_0^{L_m^f} T_j^i(M_n^{fk},s)N_l(s)ds\right)u_j\left(M_m^{fl}\right)\\&=\sum_{m=1}^{NEG}\sum_{l=1}^{3}\sum_{j=1}^{2}\left(\int_0^{L_m^g} U_j^i(M_n^{fk},s)N_l(s)ds\right)t_j\left(M_m^{gl}\right)\\&+\sum_{m=1}^{2\times NEF}\sum_{l=1}^{3}\sum_{j=1}^{2}\left(\int_0^{L_m^f} U_j^i(M_n^{fk},s)N_l(s)ds\right)t_j\left(M_m^{fl}\right)\end{aligned} \quad (2.32)$$

Où L_m^g est la longueur de l'élément du contour Γ_m^g et L_m^f la longueur de l'élément de fissure Γ_m^f.

Finalement on a :

$$\begin{aligned}
&\sum_{m=1}^{NEG}\sum_{l=1}^{3}\sum_{j=1}^{2}\left(\int_0^{L_m^g} T_j^i(M_n^{fk},s)N_l(s)ds\right)u_j\left(M_m^{gl}\right)\\
&+\sum_{m=1}^{2\times NEF}\sum_{l=1}^{3}\sum_{j=1}^{2}\left(\frac{1}{2}\alpha_{mnklij}+\frac{1}{2}\beta_{mnklij}+\int_0^{L_m^f} T_j^i(M_n^{fk},s)N_l(s)ds\right)u_j\left(M_m^{fl}\right)\\
&=\sum_{m=1}^{NEG}\sum_{l=1}^{3}\sum_{j=1}^{2}\left(\int_0^{L_m^g} U_j^i(M_n^{fk},s)N_l(s)ds\right)t_j\left(M_m^{gl}\right)\\
&+\sum_{m=1}^{2\times NEF}\sum_{l=1}^{3}\sum_{j=1}^{2}\left(\int_0^{L_m^f} U_j^i(M_n^{fk},s)N_l(s)ds\right)t_j\left(M_m^{fl}\right)
\end{aligned} \tag{2.33}$$

Avec $$\alpha_{mnklij}=\delta_{mn}\delta_{kl}\delta_{ij} \text{ et } \beta_{mnklij}=\delta_{m(n+1)}\delta_{(4-k)l}\delta_{ij} \tag{2.34}$$

C'est une équation à ($12\times NEG+24\times NEF$) variables.

Discrétisation de l'équation intégrale en tension pour un point *P* de la fissure :

Pour une fissure non chargée, l'équation intégrale en tension (1.35) appliquée à un point de collocation ($P=M_n^{fk}$) de l'élément de fissure Γ_n^f situé sur la lèvre Γ^- s'écrit :

$$n_j\left(M_n^{fk}\right)\int_{\partial\Omega\cup\Gamma} S_{j\alpha}^i(M_n^{fk},Q)u_\alpha(Q)ds(Q)=n_j(M_{j\alpha}^i)\int_{\partial\Omega} D_{j\alpha}^i(M_n^{fk},Q)t_\alpha(Q)ds(Q) \tag{2.35}$$

La même démarche de discrétisation nous conduit à :

$$\begin{aligned}
&\sum_{j=1}^{2}\sum_{m=1}^{NEG}\sum_{l=0}^{3}\sum_{\alpha=1}^{2}n_j(M_n^{fk})\left(\int_0^{L_m^g} S_{j\alpha}^i(M_n^{fk},s)N_l(s)ds\right)u_\alpha(M_m^{gl})\\
&+\sum_{j=1}^{2}\sum_{m=1}^{2\times NEF}\sum_{l=0}^{3}\sum_{\alpha=1}^{2}n_j(M_n^{fk})\left(\int_0^{L_m^f} S_{j\alpha}^i(M_n^{fk},s)N_l(s)ds\right)u_\alpha(M_m^{fl})\\
&=\sum_{j=1}^{2}\sum_{m=1}^{NEG}\sum_{l=0}^{3}\sum_{\alpha=1}^{2}n_j(M_n^{fk})\left(\int_0^{L_m^g} S_{j\alpha}^i(M_n^{fk},s)N_l(s)ds\right)t_\alpha(M_m^{gl})
\end{aligned} \tag{2.36}$$

Discrétisation de l'équation intégrale en déplacement pour un point P du contour

L'équation intégrale en déplacement (1.31) appliquée à un point de collocation ($P=M_n^{gk}$) de l'élément du contour Γ_n^g :

$$\frac{1}{2}u_i\left(M_n^{gk}\right)+\int_{\partial\Omega\cup\Gamma}T_j^i\left(M_n^{gk},Q\right)u_j(Q)ds(Q)=\int_{\partial\Omega\cup\Gamma}U_j^i\left(M_n^{gk},Q\right)t_j(Q)ds(Q) \tag{2.37}$$

Après la discrétisation, on obtient :

$$\begin{aligned}&\sum_{m=1}^{NEG}\sum_{l=1}^{3}\sum_{j=1}^{2}\left(\frac{1}{2}\delta_{mn}\delta_{kl}\delta_{ij}+\int_0^{L_m^g}T_j^i\left(M_n^{gk},Q\right)u_j(Q)ds(Q)\right)u_j(M_m^{gl})\\&+\sum_{m=1}^{2\times NEF}\sum_{l=1}^{3}\sum_{j=1}^{2}\left(\int_0^{L_m^f}T_j^i(M_n^{gk},s)N_l(s)ds\right)u_j(M_m^{fl})\\&=\sum_{m=1}^{NEG}\sum_{l=1}^{3}\sum_{j=1}^{2}\left(\int_0^{L_m^g}T_j^i\left(M_n^{gk},Q\right)N_l(Q)ds(Q)\right)t_j(M_m^{gl})\\&+\sum_{m=1}^{2\times NEF}\sum_{l=1}^{3}\sum_{j=1}^{2}\left(\int_0^{L_m^f}U_j^i\left(M_n^{gk},Q\right)N_l(Q)ds(Q)\right)t_j(M_m^{fl})\end{aligned} \tag{2.38}$$

Cette équation à ($12\times NEG+24\times NEF$) variables.

2.2.4 Constitution du système d'équations

La discrétisation des champs élastique nous a conduit à avoir ($12\times NEG+24\times NEF$) variables, la moitié de ces variables sont connus comme conditions aux limites. Pour pouvoir résoudre le problème, il nous faut donc ($6\times NEG+24\times NEF$) équations indépendantes.

La méthode de collocation consiste en l'écriture de l'équation intégrale (1.32) pour tous les points de collocation de la structure. L'écriture de cette équation pour un point de collocation P^+ d'une lèvre de la fissure conduit à la même équation écrite pour le point P^- de l'autre lèvre de la fissure. Pour surmonter ce problème, on utilisera la méthode des équations intégrales duales qui consiste à écrire l'équation intégrale en déplacement (1.32) pour les points de collocation d'une lèvre de la fissure, et l'équation intégrale en tension (1.35) (l'équation duale de celle en déplacement) sur l'autre lèvre de la fissure, ce qui nous donne deux équations indépendantes écrites pour les point géométriquement confondus P^+ et P^-

la construction du système d'équations consiste à écrire :

- L'équation intégrale en déplacement discrétisée (2.38) pour les trois points de collocation M_n^{gk} (k=1.3) de chaque élément du contour Γ_n^g (n=1,NEG) suivant les deux directions $\vec{e_i}$ (i=1,2)

- L'équation intégrale en déplacement discrétisée (2.33) pour les trois points de collocation M_n^{fk}(k=1,3) de chaque élément d'une même lèvre de fissure Γ_n^f ($n = 1 + 2\alpha, \alpha = 1, NEF - 1$) suivant les directions $\vec{e_i}$ (i=1,2)
- L'équation intégrale en tension discrétisée (2..36) pour les trois points de collocation M_n^{fk}(k=1.3) de chaque élément de l'autre lèvre de fissure M_n^{fk} (k=1.3) de chaque élément de lèvre de fissure Γ_n^f ($n = 2\alpha, \alpha = 1, NEF$)

suivant les deux directions $\vec{e_i}$ (i=1,2)

Ainsi, on obtient un système de $(6 \times NEG + 6 \times NEF + 6 \times NEF)$ équations à $(12 \times NEG + 24 \times NEF)$ variables de la forme :

$$[A]\{u\} = [B]\{t\} \tag{2.39}$$

Les coefficients des matrices $[A]$et $[B]$ne sont pas du même ordre de grandeur. En effet, A_{ij} et du même ordre de grandeur que *r* tandis que B_{ij} et du même ordre que $\frac{r}{2G}$ (*G* étant le module de cisaillement).

Pour obtenir un système bien conditionné, on doit multiplier les coefficients de la matrice $[B]$ par 2*G*, on aura alors à résoudre :

$$[A]\{u\} = (2G)[B]\left\{t' = \frac{t}{2G}\right\} \tag{2.40}$$

$(6 \times NEG + 12 \times NEF)$ de ces variables sont données comme conditions aux limites. En mettant les variables connues d'une coté et les inconnues de l'autre, on obtient un système d'équations de $(6 \times NEG + 12 \times NEF)$ équations à $(6 \times NEG + 12 \times NEF)$ inconnues de la forme :

$$[K]\{x\} = \{y\} \tag{2.41}$$

Après la résolution par la méthode de Gauss, les valeurs des tensions trouvées seront multipliées par 2G pour avoir la solution du problème.

2.3 Calcul des facteurs d'intensité de contraintes

Il existe plusieurs techniques pour calculer les facteurs d'intensité des contraintes. Dans notre étude, deux méthodes seront utilisées :

- La méthode d'extrapolation des déplacements au voisinage du fond de la fissure.
- La méthode d'intégrale de Rice (Intégrale J)

Dans ce qui suit, on présente les principes de ceux deux techniques, une comparaison entre les deux sera faite au (1.34).

2.3.1 Méthode d'extrapolation des déplacements

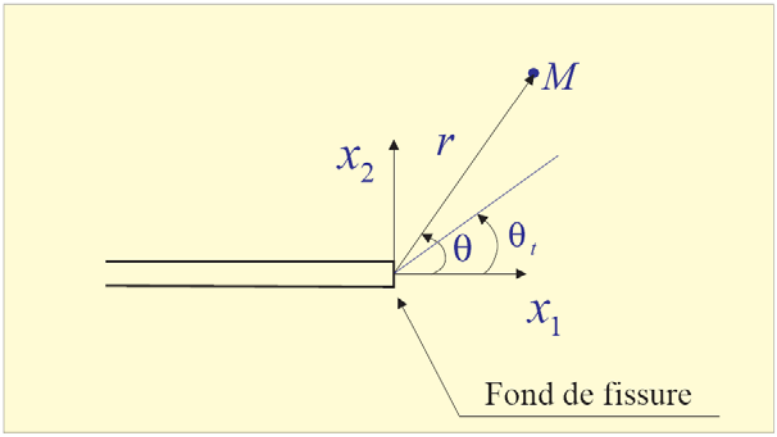


FIG.2.6 REPERE CARTESIEN ET POLAIRE EN FOND DE FISSURE

Au voisinage du fond de la fissure, dans le cadre de l'élasticité linéaire, les champs de déplacements et de contraintes sont définis par une série infinie découplée en mode I et II. En ne considérant que le premier terme de la série, les déplacements et les contraintes d'un point M(r, θ) sont :

$$\begin{cases} u_1 = \dfrac{K_I}{2\mu}\sqrt{\dfrac{r}{2\pi}}\cos\dfrac{\theta}{2}(k-\cos\theta)+\dfrac{K_{II}}{2\mu}\sqrt{\dfrac{r}{2\pi}}\sin\dfrac{\theta}{2}(\kappa-\cos\theta+2) \\ u_2 = \dfrac{K_I}{2\mu}\sqrt{\dfrac{r}{2\pi}}\cos\dfrac{\theta}{2}(k-\cos\theta)-\dfrac{K_{II}}{2\mu}\sqrt{\dfrac{r}{2\pi}}\sin\dfrac{\theta}{2}(\kappa-\cos\theta+2) \end{cases} \tag{2.42}$$

$$\begin{aligned} \sigma_{11} &= \frac{K_I}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left(1-\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right)-\frac{K_{II}}{\sqrt{2\pi r}}\sin\frac{\theta}{2}\left(2+\cos\frac{\theta}{2}\cos\frac{3\theta}{2}\right) \\ \sigma_{12} &= \frac{K_I}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\sin\frac{\theta}{2}\cos\frac{3\theta}{2}+\frac{K_{II}}{\sqrt{2\pi r}}\sin\frac{\theta}{2}\left(2+\cos\frac{\theta}{2}\cos\frac{3\theta}{2}\right) \\ \sigma_{11} &= \frac{K_I}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left(1-\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right)+\frac{K_{II}}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\sin\frac{\theta}{2}\cos\frac{3\theta}{2} \end{aligned} \tag{2.43}$$

avec $\kappa = \dfrac{3-\upsilon}{1+\upsilon}$ en contraintes planes et $\kappa = 3-4\upsilon$ en déformations planes, υ étant le coefficient de Poisson et μ le module de cisaillement.

Le saut des déplacements des lèvres de la fissure pour les points $P^+(r, \theta = \pi)$ et $P^-(r, \theta = -\pi)$ se calcule à partir des formules (3.34) et (3.35)

$$u_2(\theta=\pi)-u_2(\theta=-\pi)=\frac{\kappa+1}{\mu}K_I\sqrt{\frac{r}{2\pi}}$$
$$u_1(\theta=\pi)-u_1(\theta=-\pi)=\frac{\kappa+1}{\mu}K_{II}\sqrt{\frac{r}{2\pi}} \qquad (2.44)$$

La connaissance des déplacements des lèvres de la fissure nous permet d'évaluer directement les facteurs d'Intensité de contraintes à partir des formules (2.44).

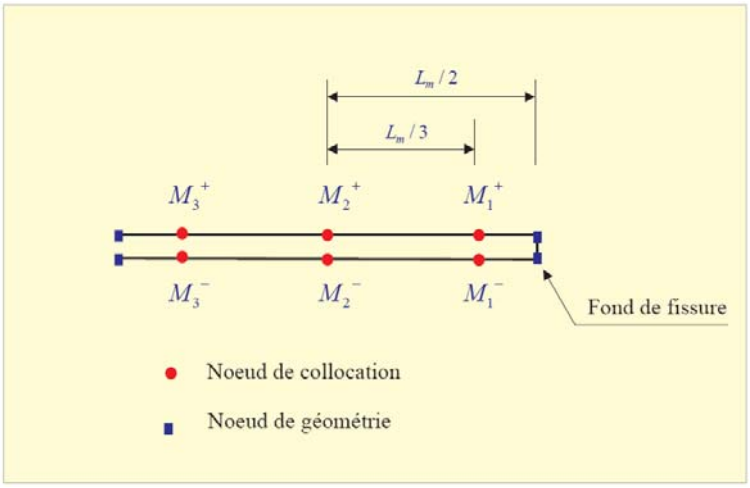


FIG.2.7 ELEMENTS DE FOND DE FISSURE

Après résolution du problème, les vecteurs déplacements sont connus en tout point de collocation, notamment pour ceux des deux éléments du fond de la fissure. Donc, d'après les équations (2.44), on peut écrire :

$$\begin{cases} K_I^{M_2}=\left(u_2^{M_2^+}-u_2^{M_2^-}\right)\dfrac{2\mu}{\kappa+1}\sqrt{\dfrac{\pi}{L}} \\ K_{II}^{M_2}=\left(u_1^{M_2^+}-u_1^{M_2^-}\right)\dfrac{2\mu}{\kappa+1}\sqrt{\dfrac{\pi}{L}} \end{cases} \qquad (2.45)$$

$$\begin{cases} K_I^{M_3}=\left(u_2^{M_3^+}-u_2^{M_3^-}\right)\dfrac{2\mu}{\kappa+1}\sqrt{\dfrac{\pi}{L}} \\ K_{II}^{M_3}=\left(u_1^{M_3^+}-u_1^{M_3^-}\right)\dfrac{2\mu}{\kappa+1}\sqrt{\dfrac{\pi}{L}} \end{cases} \qquad (2.46)$$

Si on fait une extrapolation linéaire des valeurs des facteurs d'Intensité de contraintes entre les points M_3 et M_2 vers le fond de la fissure on aura :

$$\begin{cases} K_I=\left[5\left(u_2^{M_2^+}-u_2^{M_2^-}\right)-\dfrac{3\sqrt{15}}{5}\left(u_2^{M_3^+}-u_2^{M_3^-}\right)\right]\dfrac{\mu}{\kappa+1}\sqrt{\dfrac{\pi}{L}} \\ K_{II}=\left[5\left(u_1^{M_2^+}-u_1^{M_2^-}\right)-\dfrac{3\sqrt{15}}{5}\left(u_1^{M_3^+}-u_1^{M_3^-}\right)\right]\dfrac{\mu}{\kappa+1}\sqrt{\dfrac{\pi}{L}} \end{cases} \qquad (2.47)$$

2.3.2 Intégrale de Rice (intégrale J)

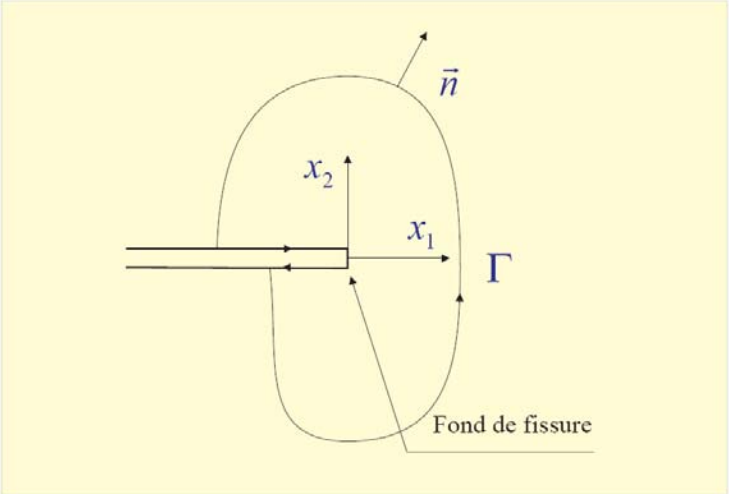


FIG.2.8 CONTOUR DE Γ DE L'INTEGRALE J

La méthode de l'intégrale-J est une méthode efficace pour l'évaluation des Facteurs d'Intensité de contraintes, parce que la méthode des éléments de frontière nous permet de calculer avec une grande précision les champs élastiques au voisinage du fond de fissure. Considérons un repère cartésien défini en fond de fissure comme le montre la figure (Fig.2.7) (on suppose que les lèvre des fissure ne sont pas chargées). En négligeant les forces de volume, Rice a introduit l'intégrale de contour suivante :

$$J_k = \int_{\Gamma} (Wn - t_j u_{j,k}) d\Gamma \tag{2.48}$$

Où

- W est un contour orienté ouvert entourant le fond de la fissure dont les extrémités se trouvent sur les lèvres de la fissure (Fig3.6).
- Γ est un contour orienté ouvert entourant le fond de la fissure dont les extrémités se trouve sur les lèvres de la fissure (Fig.3.6)
- t_j est la composante du vecteur tension suivant la direction x_j, définie le long du contour. $t_j = \sigma_{ij} n_i$ (2.49)
- n_i est la composante de la normale suivante la direction x_i

Les relations entre l'intégrale J et les facteurs d'intensité de contrainte sont données par :

$$J_1 = \frac{K_I^2 + K_{II}^2}{E'} \tag{2.50}$$

et

$$J_2 = \frac{2K_I K_{II}}{E'} \tag{2.51}$$

Où E' est le module d'élasticité de Young ;E'=E en contraintes planes et $E' = E/(1-\upsilon^2)$ en déformations planes.

Ces relations ne nous permettent pas d'obtenir séparément les facteurs d'Intensité de contraintes K_I et K_{II}, **Bui** à présenté une méthode élégante de décomposition des modes qui consiste à décomposer l'intégrales J en la somme de deux intégrales

$$J_1 = J_1^I + J_1^{II} \tag{2.52}$$

Où les exposants sont associés aux modes de rupture.

Pour que cette présentation soit possible, il a introduit la décomposition suivante des champs élastiques :

$$\begin{Bmatrix} \sigma_{11}^I \\ \sigma_{22}^I \\ \sigma_{12}^I \end{Bmatrix} = \frac{1}{2}\begin{Bmatrix} \sigma_{11} + \sigma'_{11} \\ \sigma_{22} + \sigma'_{22} \\ \sigma_{12} - \sigma'_{12} \end{Bmatrix}, \begin{Bmatrix} \sigma_{11}^{II} \\ \sigma_{22}^{II} \\ \sigma_{12}^{II} \end{Bmatrix} = \frac{1}{2}\begin{Bmatrix} \sigma_{11} - \sigma'_{11} \\ \sigma_{22} - \sigma'_{22} \\ \sigma_{12} + \sigma'_{12} \end{Bmatrix} \tag{2.53}$$

et

$$\begin{Bmatrix} u_1^I \\ u_2^I \end{Bmatrix} = \frac{1}{2}\begin{Bmatrix} u_1 + u'_1 \\ u_2 + u'_2 \end{Bmatrix}, \begin{Bmatrix} u_1^{II} \\ u_2^{II} \end{Bmatrix} = \frac{1}{2}\begin{Bmatrix} u_1 - u'_1 \\ u_2 + u'_2 \end{Bmatrix} \tag{2.54}$$

où

$$\sigma'_{ij}(x_1, x_2) = \sigma_{ij}(x_1, -x_2) \tag{2.55}$$

et

$$u'_i(x_1, x_2) = u_i(x_1, -x_2) \tag{2.56}$$

Les équations (2.53) et (2.54) mène à la décomposition suivante des champs élastiques :

$$\sigma_{ij} = \sigma_{ij}^I + \sigma_{ij}^{II} \quad \text{et} \quad u_i = u_i^I + u_i^{II} \tag{2.57}$$

En introduisant (2.53) et (2.54) dans l'équation (2.48), on obtient la forme (2.52) de l'intégrale de J avec :

$$J_i^M = \int_\Gamma \left(W^M n_i - t_j^M u_{j,i}^M \right) d\Gamma \tag{2.58}$$

Ou M = I , II .

Finalement, on obtient :

$$J_1^{II} = \frac{K_1^2}{E'} \ , \ J_1^{II} = \frac{K_{II}^2}{E'} \tag{2.59}$$

La mise en œuvre de cette procédure dans le code KSP en faite avec un Contour circulaire Γ ,

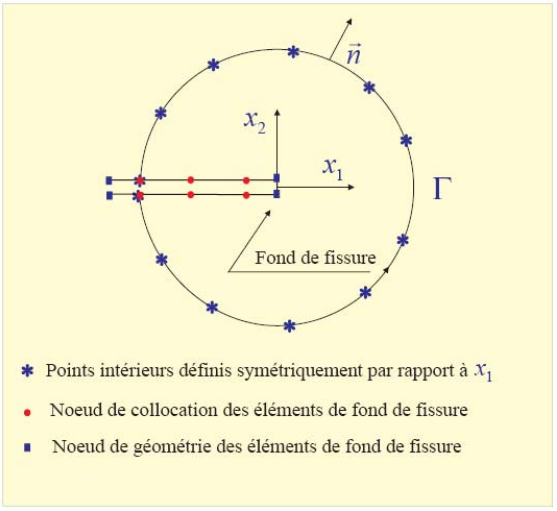


FIG.2.9 CHOIX DES POINTS INTERIEURS POUR LE CALCUL DE L'INTEGRALE J

de centre le fond de fissure, et de rayon $\frac{5L_m}{6}$, où L_m est la longueur de l'élément en fond de fissure(cela correspond au nœud la plus loin du fond de fissure se situant sur le dernier élément, donnant une meilleure précision). Sur le cercle, un nombre pair de points intérieurs sont positionnés symétriquement autour de l'axe de la fissure (Fig.2.7)

2.3.3 Algorithme

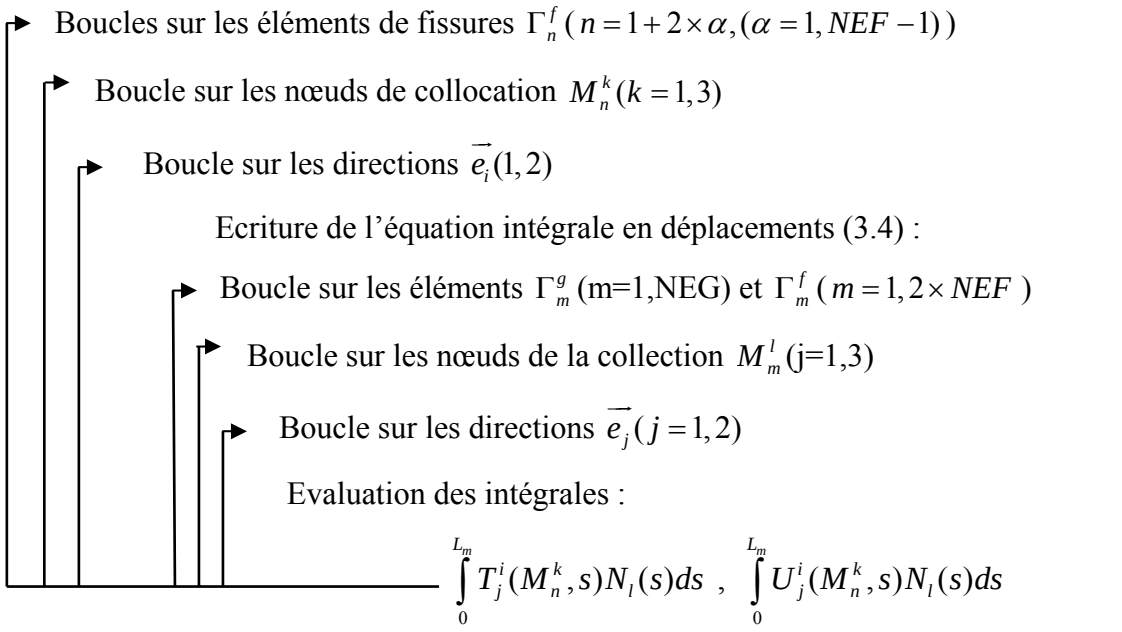
1. Lecture des données : géométrie du problème, conditions aux limites et propriété du matériau ;
2. Maillage : discrétisation du contour en NEG éléments Γ_m^g et discrétisation des fissures en $2\times NEF$ éléments Γ_m^f ;
3. Construction des matrices [A] et [B] :

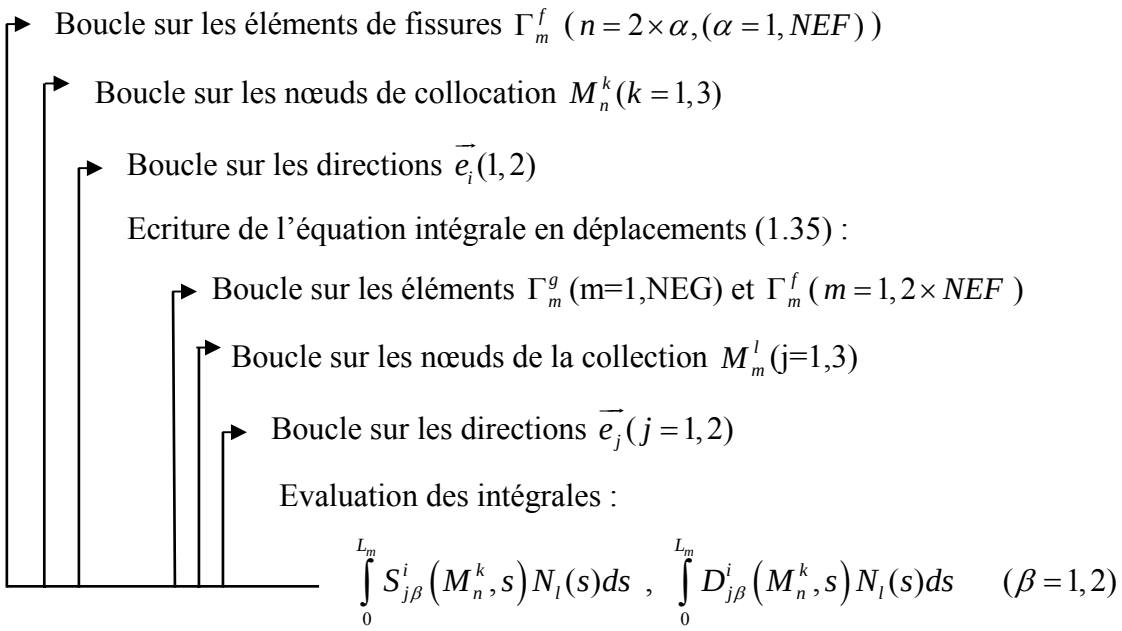
- Boucle sur les éléments du contour Γ_m^f (n = 1, NEG)
  - Boucle sur les nœuds de collocation M_k^n (k=1,3)
    - Boucle sur les directions $\vec{e_i}$ (i=1,2)

      Ecriture de l'équation intégrale en département (1.38) :
      - Boucle sur les éléments Γ_m^g (m=1,NEG) et Γ_m^f ($m=1,2\times NEF$)
        - Boucle sur les nœuds de la collection M_m^l (j=1,3)
          - Boucle sur les directions $\vec{e_j}(j=1,2)$

            Evaluation des intégrales :

$$\int_0^{L_m} T_j^i(M_n^k,s)N_l(s)ds \ , \ \int_0^{L_m} U_j^i(M_n^k,s)N_l(s)ds$$





4. introduction des conditions aux limites et constitution du système d'équations :

$$[K]\{x\}=\{y\}$$

5. Résolution du système d'équations par la méthode de décomposition de gauss et obtention des inconnues du contour : $\{u\}et\{t\}$
6. Calcul des déplacements des points intérieurs par la formule (1.42) ;
7. Calcul des contraintes des points intérieurs par la formule (1.43) ;
8. Calcul des contraintes des points contours par la formule (1.44) ;
9. Calcul des Facteurs d'Intensité de Contraintes par les formules (2.47) ou (2.59).
10. Affichage des résultats.

Chapitre 3
Le code KSP

3.1 Introduction

En terme de programmation, il y a eu plusieurs évolutions successives. Une des principales fut la programmation structurée, dont le principe premier était de diviser un programme en sous-programmes, afin de pouvoir en gérer la complexité. Ce type de programmation tient avant tout compte des traitements et peut être résumé par la question "Que doit faire le programme ?".

Les langages orientés objets (L.O.O) sont une nouvelle méthode de programmation qui tend à se rapprocher de notre manière naturelle d'appréhender le monde. Les L.O.O. se sont surtout posé la question "Sur quoi porte le programme ?". En effet, un programme informatique comporte toujours des traitements, mais aussi et surtout des données. Si la programmation structurée s'intéresse aux traitements puis aux données, la conception objet s'intéresse d'abord aux données, auxquelles elle associe ensuite les traitements. L'expérience a montré que les données sont ce qu'il y a de plus stable dans la vie d'un programme, il est donc intéressant d'architecturer le programme autour de ces données

3.2 La programmation orientée objet

La programmation par objet (du terme anglo-saxon Object-Oriented Programming ou OOP), est un paradigme de programmation, il consiste en la définition et l'assemblage de briques logicielles appelées objets ; un objet représente un concept, une idée ou toute entité du monde physique, comme une voiture, une personne ou encore une page d'un livre.

Orthogonalement à la programmation par objet, afin de faciliter le processus d'élaboration d'un programme, existent des méthodologies de développement logiciel objet dont la plus connue est USDP (Unified Software Development Process).

Il est possible de concevoir par objet une application informatique sans pour autant utiliser des outils dédiés. Il n'en demeure pas moins que ces derniers facilitent de beaucoup la conception, la maintenance, et la productivité. On en distingue plusieurs sortes :

- les langages de programmation (Eiffel, Python, C++, Smalltalk...)
- les outils de modélisation qui permettent de concevoir sous forme de schémas semi-formels la structure d'un programme (Objecteering, UMLDraw, Rapsody…)

- les bus distribués (COM, CORBA, RMI, Pyro...)
- les ateliers de génie logiciels (ou AGL) (Rational Rose XDE, Objecteering...)

Des langages à objets, il en existe actuellement deux catégories : les langages à classes et ceux à prototypes, que ceux-ci soient sous forme fonctionnelle (CLOS, OCaml…) ou impérative (C++, Java…) ou les deux (Python).

3.3 Microsoft visuel C++

Microsoft Visual C++ offre des outils et un environnement de développement très puissant et très flexible pour créer et développer différentes applications. Il est constitué des composants suivants:

- Les outils du compilateur de Visual C++. Le compilateur possède de nouvelles fonctionnalités destinées à assister les développeurs dans les différentes étapes de leurs travails, et surtout pour ceux qui ciblent des plates-formes de machine virtuelle, comme le CLR (Common Language Runtime).

- Les bibliothèques Visual C++. Il s'agit de la bibliothèque ATL (Active Template Library), de la bibliothèque MFC (Microsoft Fondation Class) et des bibliothèques standard, telles que la bibliothèque C++ standard et la bibliothèque Runtime C (CRT). Une nouvelle bibliothèque, la bibliothèque de prise en charge C++, a été conçue pour simplifier les programmes qui visent le CLR.

- L'environnement de développement Visual C++. Bien que les outils de compilateur et les bibliothèques C++ puissent être utilisés depuis la ligne de commande, l'environnement de développement aide puissamment à la gestion et la configuration des projets, ainsi que pour modifier ou parcourir le code source, et offre des outils de débogage. Cet environnement prend également en charge IntelliSense, qui effectue des suggestions contextuelles intelligentes au cours de la création du code.

Outre des applications classiques d'interface graphique utilisateur, Visual C++ permet aux développeurs de générer des applications Web, des applications Windows clientes intelligentes, ainsi que des solutions pour client basique et pour périphériques mobiles clients intelligents. Visual C++ appartient à la suite de logiciel Visual Studio.

Mais avant tout le Visual C++ est du C++ qui est un langage de programmation dit "haut niveau", c'est le langage de niveau système le plus répandu au monde créé en 1983 par Bjarne Stroustrup des laboratoires Bell. Il se distingue de son prédécesseur (le langage C) en proposant une programmation orientée objet. Ce langage est particulièrement utilisé dans les applications demandant de hautes performances.

3.4 Le KSP

3.4.1 Introduction

Le KSP est un logiciel pour la modélisation numérique en mécanique. Il est développé en langage orienté objet sous ***Visual C++.***

Il permet la résolution de différents types de problème linéaires ou non linéaires, modélisés par la méthode des éléments finis et/ou par la méthode des équations intégrales, en deux ou en trois dimensions. Il permet, notamment, la résolution des problèmes suivants :

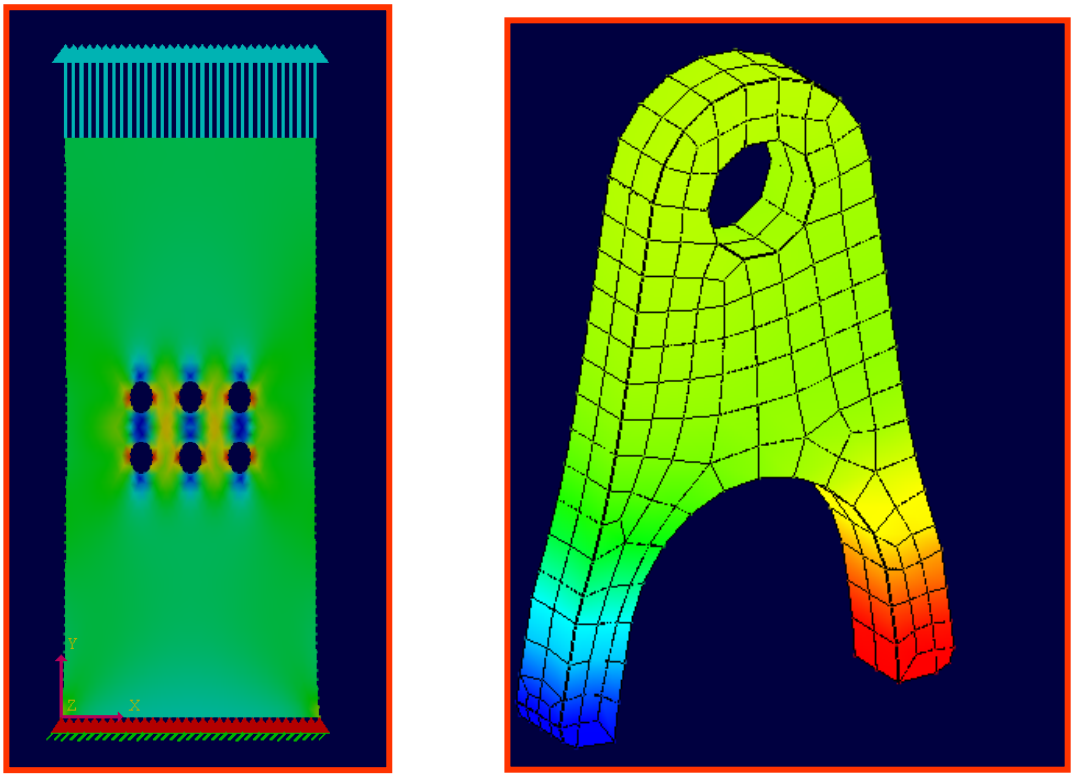


FIG .3.1 ELASTICITE EN 2 ET 3 DIMENSIONS PAR LA METHODE DES ELEMENTS FRONTIERES

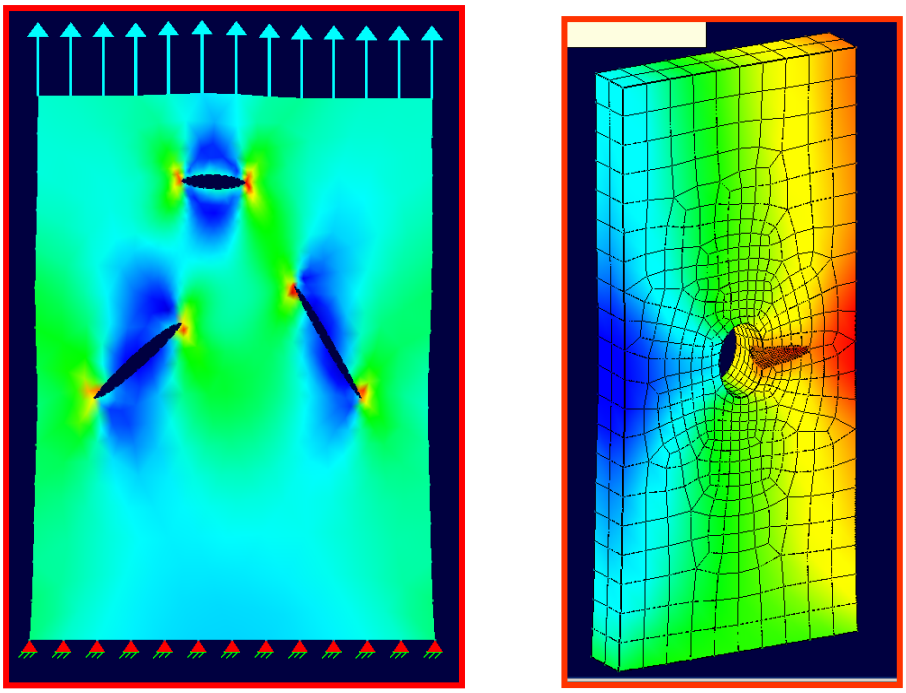


FIG.3.2 MULTI FISSURATION PAR LA METHODE DES ELEMENTS DE FRONTIERES

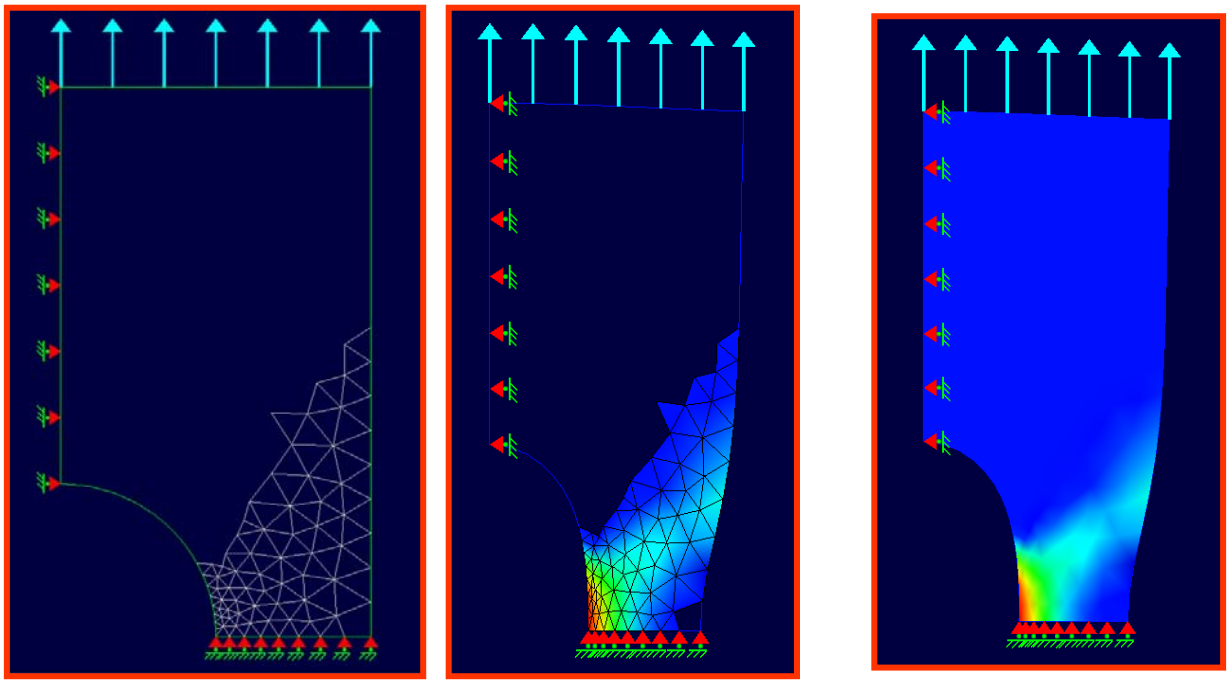


FIG.3.3 ELASTOPLASTICITE EN 2 DIMENSIONS PAR LA METHODE DES ELEMENTS FRONTIERES

3.4.2 Environnement de développement du logiciel

KSP est développé en langage orienté objet sous ***Visual C++***, il utilise quelques classes de la bibliothèque **MFC**. Actuellement, il est constitué de 56 classes, chacune d'elles représente un objet bien spécifique.

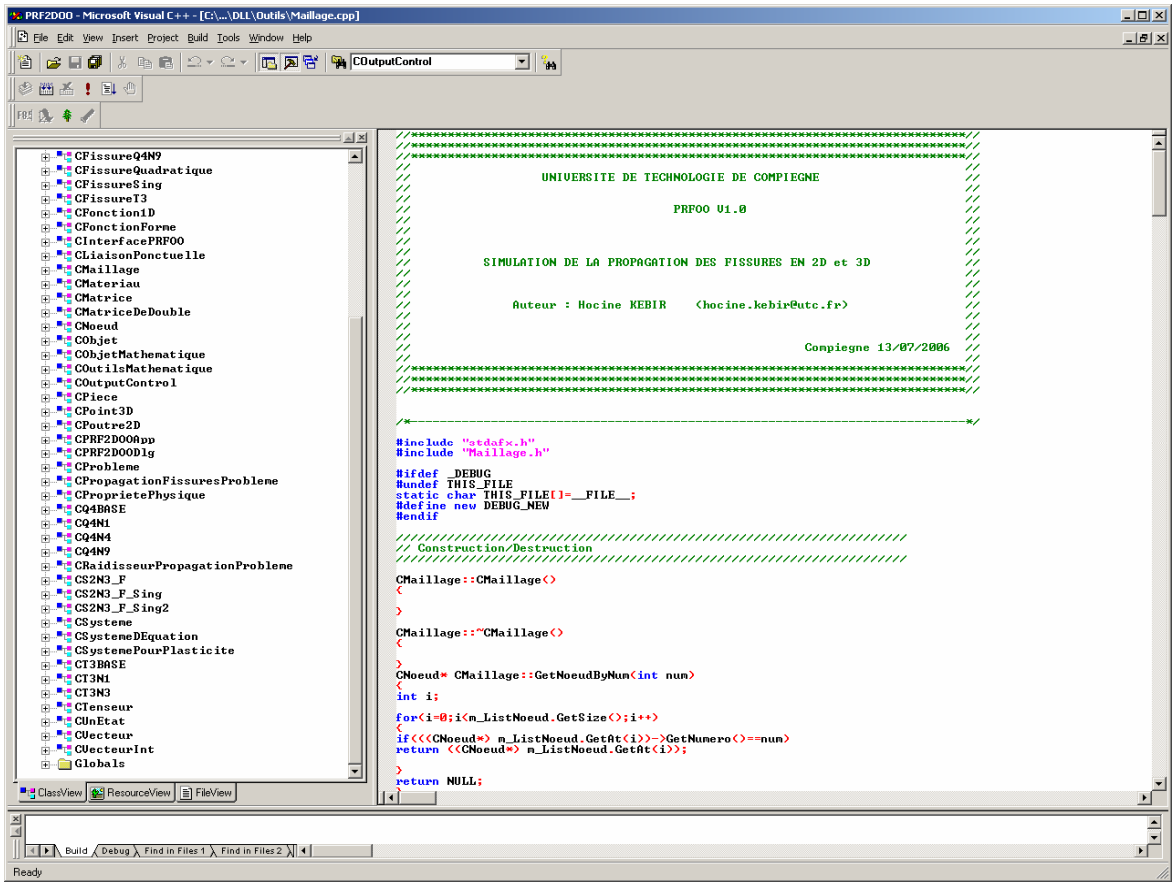


FIG.3.4 LES CLASSES DU KSP

3.5 Décomposition des problèmes dans KSP

3.5.1 Les classes CProbleme et l'héritage

Lorsqu'une classe est définie à partir d'une autre (ou plus généralement, à partir de plusieurs autres), la nouvelle classe est appelée classe dérivée. Une classe dérivée contient automatiquement tous les membres donnés des classes ayant permis de la définir ainsi que les fonctions membres. On dit que la classe hérite des membres donnés et des fonctions membres des classes de base. Ainsi, Les classes **`C***Problème`** dérivent de la classe **`CProblème.`**

Le code KSP utilise la notion de fonction virtuelle. Il s'agit de la plus importante fonctionnalité du C++.

Une fonction virtuelle permet aux classes dérivées de remplacer l'implémentation fournie par la classe de base. Le compilateur s'assure que la méthode remplacée est bien appelée quand l'objet en question est bien du type de la classe dérivée, même si l'objet est accédé par un pointeur de base plutôt qu'un pointeur dérivé. Cela permet aux algorithmes de la classe de base d'être remplacé dans la classe dérivée.

Ainsi, lorsque KSP appelle la fonction mère CProblème ::SolveYourSelf() c'est les fonctions filles C**`***Problème ::solveYourSelf()`** qui vont s'exécuter.

3.5.2 Présentation des classes CProbleme

KSP est un programme pluridisciplinaire de calculs mécaniques numériques. Il peut traiter aussi bien les problèmes mécaniques par la méthode des éléments finis, que par la méthode des équations intégrales ou en couplant les deux méthodes. Il est prévu pour contenir une bibliothèque de classes de problèmes qui peuvent être complètement différents.
Parmi ces classes :

- **CPropagationFissure** : cette classe permet la résolution de problèmes mécaniques en élasticité linéaire en deux et trois dimensions par la méthode des équations intégrales. Elle permet notamment la simulation de la propagation des multifissures.

- **CRaidisseurPropagationProbleme :** cette classe permet la résolution de problèmes mécaniques en élasticité linéaire en deux dimensions par couplage de la méthode des équations intégrales et la méthode des éléments finis. Elle permet notamment la simulation de la propagation des fissures dans des panneaux raidies.

- **CBEMElastoPlasticite :** Cette classe permet la résolution des problèmes élastoplastiques par la méthode des équations intégrales duales. Elle permet notamment la simulation de la déchirure ductile des tôles minces.

- **CContactProbleme** Cette classe permet la résolution des problèmes de contact par la méthode des équations intégrales duales. Elle permet notamment l'étude des assemblages boulonnés.

- **CFEMProblme :** Cette classe permet la résolution des problèmes mécaniques par la méthode des éléments finis.

Seules les deux premières classes seront décrites dans ce manuel.

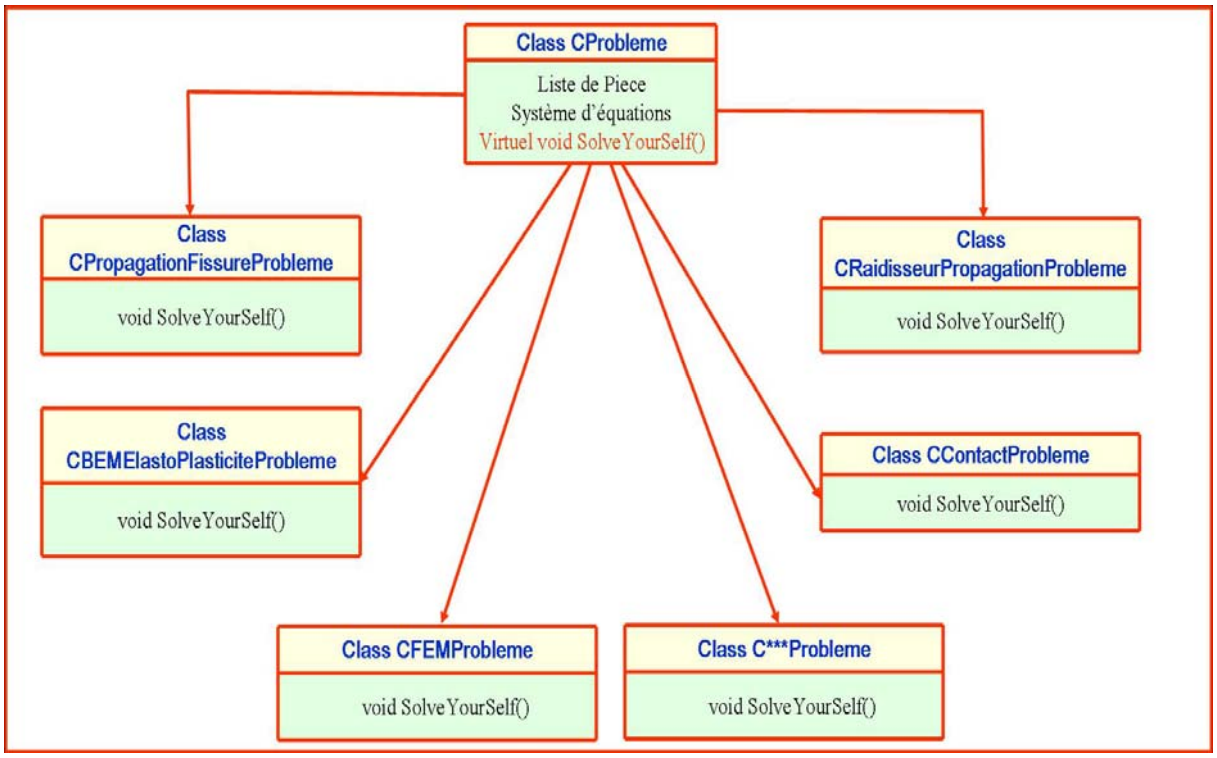


FIG.3.5 LES CLASSES DE PROBLEME DU KSP

3.6 L'introduction des données

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.

3.6.1 L'introduction des données géométriques

Elles sont introduites dans le cas de problèmes 2D par un sketch 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 donne 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 leur importation depuis d'autre logiciel telle que ABAQUS et IDEAS.

3.6.2 L'introduction des conditions aux limites

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

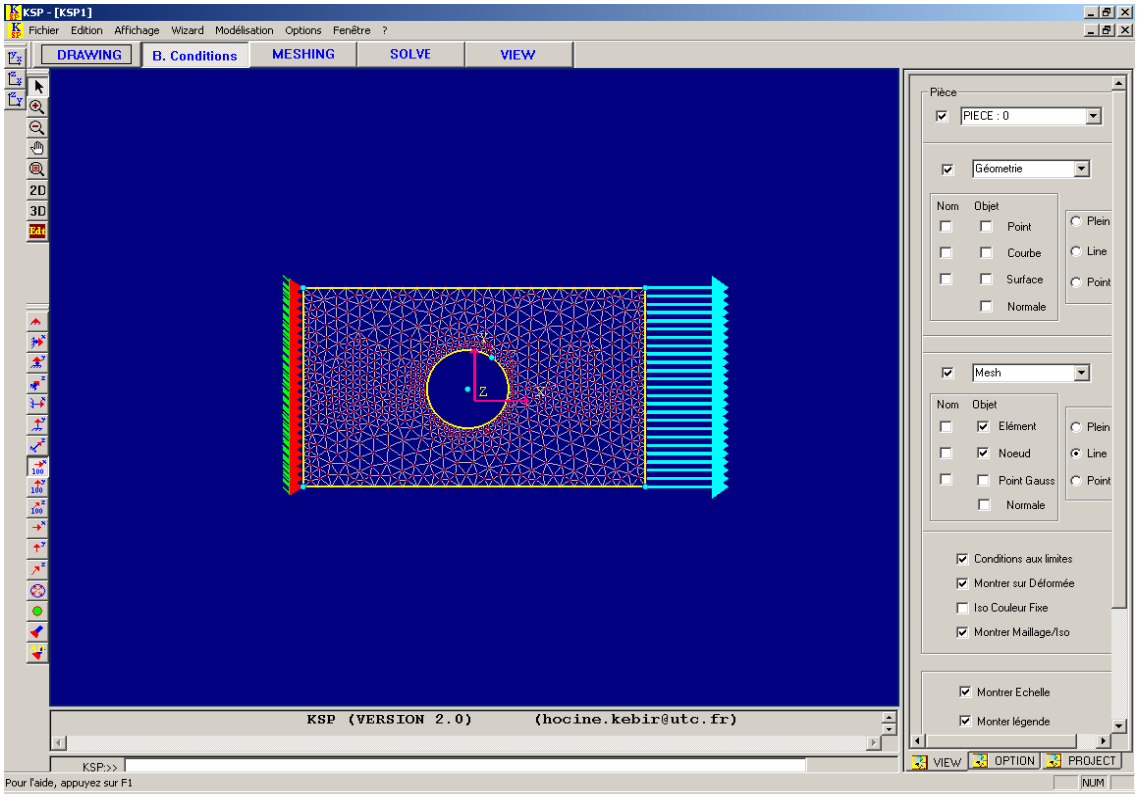


FIG.3.6 EXEMPLE D'INTRODUCTION DES DONNES SUR LE KSP

3.6.3 Le traitement

C'est l'étape la plus intéressante, elle aussi est constitué de deux étape : la création du système et la résolution.

3.6.3.1 La création du système

Elle est effectuée par la méthode des équations intégrales expliquées dans le premier chapitre.

3.6.3.2 La résolution

C'est la partie où intervient notre travail. Le KSP utilisait la méthode de décomposition de gauss pour la résolution du système d'équations résultant de l'algorithme de la méthode des équations intégrales.

On a choisi l'algorithme GMRES qu'on va l'implémenter dans le KSP pour pouvoir résoudre les problèmes de grandes tailles de l'ordre de 10 000 ddl et 20 000 ddl etc.

3.6.4 La sortie des résultats

Elle est assurée dans le KSP par une interface graphique qui permet de voire la distribution des contraintes clairement à l'aide d'un jeu 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.

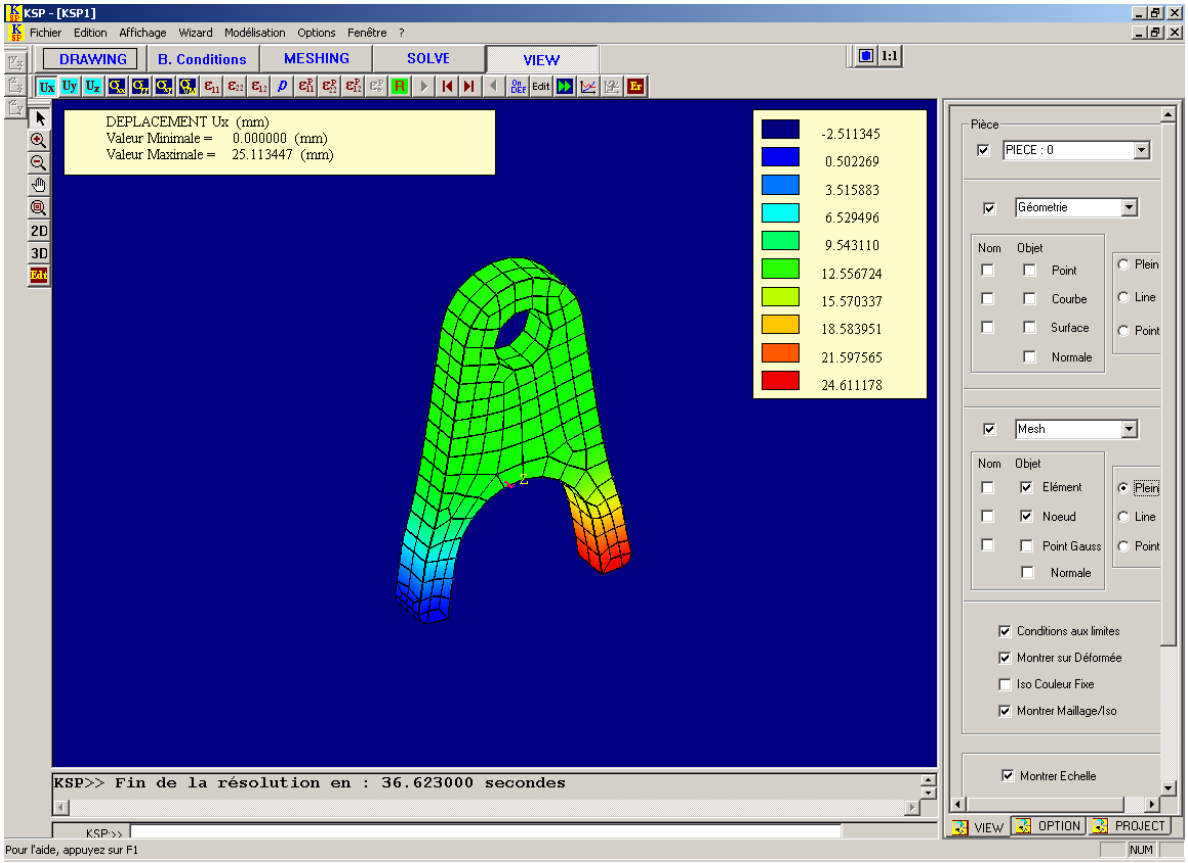


FIG.3.7 EXEMPLE DE SORTIE DES RESULTATS SUR LE KSP

Chapitre 4
La méthode itérative GMRES

4.1 Introduction

Pour la résolution d'un système $Ax = b$ où $A \in R^{n \times n}$ est une matrice carrée d'ordre $n \times n$ et x et b sont des vecteurs de longueur n, et quand la taille du système est très grande ($n > 10^3$), plusieurs auteurs ont proposé des généralisations différentes de la méthode du gradient conjugué et du gradient biconjugué ces 25 dernière années.

Paige et Saunders ont proposé la méthode MINRES caractérisé par la minimisation de la norme du résidu via un sous espace de Krylov. Pour généraliser MINRES, en 1986, Saad et Schultz ont formulé la très populaire méthode GMRES (Generalised Minimized RESidual) pour la résolution des problèmes non symétrique. Ils présentent une implémentation pratique basée sur le processus d'Arnoldi et les rotations de Givens. Elle construit une base orthonormale du sous espace de Krylov $K_k(r_0) = span\{r_0, Ar_0, ..., A^{k-1}r_0\}$ où $r_0 = b - Ax_0$ et l'initie avec le vecteur $v_1 = r_0 / \| r_0 \|$.

4.2 Les méthodes de Krylov

4.2.1 Définition de l'espace de Krylov

L'espace de Krylov de dimension k lié à une matrice A et un vecteur v est le sous espace engendré par $\{ v, A v, \ldots, A^{k-1} v \}$, on le note K_k.

$$K_k(A,v) = \text{span}\{ v, Av, ..., A^{k-1}v\} \tag{4.1}$$

4.2.2 La propriété de minimisation

Contrairement aux méthodes itératifs stationnaires les méthodes de Krylov n'utilisent pas une matrice d'itération, et puisque la matrice n'est pas hermitienne définie positive, des méthodes relativement simples ne peuvent pas être appliquée car on ne dispose pas de la norme $\|.\|_A$. L'idée va être de construire des méthodes itératives consistant à minimiser la norme $\|.\|_2$ du résidu. A la $k^{ième}$ itération, une minimisation de l'erreur est effectuée envers l'espace voisin. La solution sera dans

$$x_0 + K_k \tag{4.2}$$

où x_0 est la solution initiale et K_k est le $k^{ième}$ sous espace de Krylov et le résidu est

$$r = b - Ax \tag{4.3}$$

alors $\{r_k\}_{k\geq 0}$ représente le résidu séquentiel

$$r_k = b - Ax_k \tag{4.4}$$

La m^{ième} itération des algorithmes des méthodes de Krylov donne la solution du problème aux moindre carrée

$$\min_{x\in x_0+K_m} \| b - Ax \| \tag{4.5}$$

4.3 Décomposition de Hessenberg

4.3.1 Définition d'une matrice de Hessenberg supérieure

La matrice $H \in R^{n\times n}$ est sous forme Hessenberg supérieure si et seulement si :

$$h_{i,j} = 0 \text{ pour } i > j + 1$$

Cette matrice est dite irréductible si et seulement si :

$$h_{i+1,i} \neq 0 \text{ pour } i=1,\ldots,n-1$$

On appelle matrice non dérogatoire une matrice dont l'indice de toute valeur propre est égal à sa multiplicité algébrique. Alors on peut affirmer qu'une matrice de Hessenberg irréductible est non dérogatoire.

4.3.2 Présentation des décompositions de Hessenberg

Une décomposition de Hessenberg d'une matrice $A \in R^{n\times n}$ consiste à trouver deux matrices, la première Q unitaire et la deuxième H de forme Hessenberg telles que : $A=QHQ^T$. Les matrices A et H sont unitairement équivalentes. Et il n'y a pas unicité de la décomposition de Hessenberg.

Il existe plusieurs familles de méthodes de décomposition de Hessenberg. Nous nous intéressons principalement à la méthode itérative d'Arnoldi pour obtenir une décomposition de Hessenberg. Elle effectue une projection orthogonale de A sur le sous-espace de Krylov engendré par $\{v_1, Av_1, ..., A^{i-1}v_1\}$. A l'étape i, on obtient la matrice V_i que l'on note $V_i = \{v_1, \ldots, v_i\}$.

La matrice $V_i^T AV_i = H_i$ est une matrice de Hessenberg d'ordre i. La matrice $H_{i+1} = V_{i+1}^T AV_{i+1}$ d'ordre i+1 s'obtient en rajoutant une colonne puis une ligne à H_i :

$$H_{i+1} = \left[\begin{array}{ccc|c} & & & \times \\ & H_i & & . \\ & & & . \\ & & & \times \\ \hline 0 & \ldots & 0 & \times \quad \times \end{array}\right]$$

FIG.4.1 LA MATRICE DE HESSENBERG

Il existe plusieurs manières pour calculer la base Vi dans la méthode d'Arnoldi en utilisant essentiellement la factorisation QR à l'aide d'une des trois implantations : Gram Schmidt classique, Gram Schmidt modifiée ou Householder.

4.3.3 Méthode d'Arnoldi pour la décomposition incomplète de Hessenberg

Cette méthode associe à une matrice $A \in R^{n \times n}$

- une base orthonormale $V_k = \{v_1,\ldots, v_k\}$ du sous-espace de Krylov

$$K_k(A,v_1) = \text{span}\{ v_1, Av_1, \ldots, A^{i-1}v_1\}, \qquad (4.6)$$

et

- une matrice $\overline{H_k}$ de taille (k+1,k) qui a pour bloc principal une matrice de Hessenberg supérieure H_k, et qui est augmentée d'une ligne supplémentaire dont le seul élément non nul est $h_{k+1,k}$.

La matrice $\overline{H_k}$ a donc la forme suivante :

$$\overline{H}_k = \begin{pmatrix} h_{11} & \cdots & \cdots & \cdots & \cdots & h_{1k} \\ h_{12} & \ddots & & & & \vdots \\ 0 & h_{23} & & & & \vdots \\ \vdots & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & & & \vdots \\ \vdots & & & & h_{kk-1} & h_{kk} \\ 0 & \cdots & & \cdots & 0 & h_{k+1k} \end{pmatrix} = \begin{pmatrix} \boxed{\begin{array}{ccc} & & \\ \ddots & & H_k \\ & \ddots & \\ & & \ddots \end{array}} & & \\ & & \\ & & \underline{\qquad} \\ & & h_{k+1k} \end{pmatrix}$$

FIG.4.2 LA FORME DU MATRICE $\overline{H_k}$

On peut montrer que, à l'itération k de l'algorithme d'Arnoldi, les matrices A, V_k et H_k vérifient l'égalité :

$$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T \tag{4.7}$$

où le vecteur e_k est le $k^{ième}$ vecteur colonne de la matrice identité d'ordre n.
Cette égalité peut également s'écrire :

$$A V_k = V_{k+1} \overline{H_m} \tag{4.8}$$

En multipliant l'égalité (4.7) à gauche par V_k^H, nous pouvons écrire

$$V_k^H A V_k = H_k \tag{4.9}$$

H_k est la matrice quotient de Rayleigh de A par V_k. C'est une autre manière de voir que H_k est la matrice de l'application A projetée orthogonalement sur K_k dans la base V_k.

Un grand intérêt de cette méthode pour les très grandes matrices ($n > 10^3$) est de pouvoir stopper le processus au pas $m << n$ et de travailler avec H (respectivement $\overline{H}$) de taille $m \times m$ (respectivement $(m+1) \times m$). Cela s'appelle une décomposition incomplète de Hessenberg, avec H la matrice qui représente la projection orthogonale de A sur le sous-espace de Krylov de dimension m dans la base V_m.

L'algorithme d'Arnoldi s'arrête lorsque $h_{k+1,k}=0$. La matrice V_k est alors une base d'un sous espace invariant associé à A. Cela se produit pour $k_0 \leq n$ et on a :

$$AV_{k0} = V_{k0} H_{k0} \tag{4.10}$$

Pour $k \geq k0$, on a $K_k(A,v_1) = K_{k0}(A,v_1)$. Plus précisément :
pour $k < k_0$, $rg(K_k)=k$,
pour $k \geq k_0$, $rg(K_k)=k_0$
où $rg(K_k)$ est le rang de K_k.
Nous qualifions le phénomène $h_{k+1,k}=0$ à l'itération $k=k_0$ par le terme d'arrêt heureux, qui est traduit du terme anglais : « ***happy break-down*** ». Une justification de cette appellation sera expliquée dans le paragraphe suivant.

4.4 Processus d'Arnoldi

4.4.1 Effectuer une factorisation QR

L'algorithme d'Arnoldi effectue une factorisation QR récursive de la matrice $[v_1, Av_1, ..., Av_m] = [v_1, Av_m]$ pour $m = 1, ..., n-1$:

$$[v_1, AV_m] = V_{m+1}[1, \overline{H_m}] = V_{m+1} R_{m+1} \ . \tag{4.11}$$

avec R_{m+1} la matrice triangulaire supérieure d'ordre m+1.

Ainsi, tant que $h_{m+1,m} \neq 0$, pour $m = 1,...,n-1$.

$$AV_m = V_{m+1}\overline{H_m} \tag{4.12}$$

Pour m=n, se situe un arrêt mathématique qui correspondrait à $h_{n+1,n}=0$ et

$$AV_n = V_n H_n \Leftrightarrow A = V_n H_n V_n^T \tag{4.13}$$

4.4.2 Trois façons classiques de réaliser la factorisation QR

Nous donnons trois façons classiques de réaliser cette factorisation QR équivalentes en arithmétique exacte avant l'arrêt :

- **Algorithme *d'Arnoldi-Gram Schmidt classique*** **(CGS)**

- Choisir un vecteur v_1 de norme 1.
  - Pour k=1,…,n
  - Calculer $h_{i,k}=(Av_k,v_i)$ pour i=1,…,k
  - Calculer $w_k=Av_k-\sum_{i=1}^{k} h_{ik} v_i$
  - $h_{k+1,k} =\|w_k\|$. Si $h_{k+1,k}= 0$, Arrêt.
  - $v_{k+1}=w_k/h_{k+1,k}$
- Fin Pour

- **Algorithme *d'Arnoldi-Gram Schmidt modifié*** **(MGS)**

La variante MGS qui est présentée comme une modification de l'algorithme de Gram Schmidt classique a en fait été utilisée directement par le Marquis Pierre Simon de Laplace dès le début du 19ème siècle, lors de ses calculs astronomiques de moindres carrés.

- Choisir un vecteur v_1 de norme 1.
- Pour k=1,…,n
  - $w_k^{(1)} = Av_k$
  - Pour *i*=1,…,k
    - calculer $h_{ik}=(w_k^{(i)} , v_i)$
    - $w_k^{(i+1)} = w_k^{(i)} - h_{ik} v_i$
  - fin pour
  - $h_{k+1,k} =\|w_k\|$. Si $h_{k+1,k}= 0$, Arrêt.
  - $v_{k+1}=w_k/h_{k+1,k}$
- Fin Pour

- **Algorithme *d'Arnoldi-Householder***

Pour cet algorithme, le calcul de $V_m=[v_1,…,v_m]$ utilise des matrices P_i, i=1,…,n, hermitiennes unitaires (donc carrées $n \times n$) construites à partir de matrices de Householder.

1. Choisir un vecteur v_1 de norme 1 et calculer P_1 tel que $P_1v_1=e_1$.

2.

- Pour m=2,…,n
  - $z_m=P_{m-1}\ldots P_1 A v_{m-1}$
  - Partitionner z_m en $z_m=[\, z_m^{(1)\,T} \;\; z_m^{(2)\,T}]^T$ avec $z_m^{(1)}$ de taille m-1.
  - Si $z_m^{(2)}=0$,
    - $P_m=I_n$
    - $h_{m-1}=z_m$
    - Arrêt des itérations ou continuation par $P_m=I_n$ et $v_m=P_1\ldots P_{m-1}e_m$.
  - Sinon
    - Calculer Pm à partir de $z_m^{(2)}$.
    - $h_{m-1}= P_m z_m$
    - $v_m=P_1\ldots P_m e_m$
- fin pour

3. $h_n=P_n...P_1 A v_n$

4.5 Les rotations de Givens

Quand k est très grand (k étant la dimension de sous espace de Krylov K_k), l'implémentation en utilisant les rotations de Givens est plus efficace et plus rapide qu'une approche directe.

Une rotation de 2×2 est une matrice de la forme :

$$G=\begin{pmatrix} c & -s \\ s & c \end{pmatrix} \tag{4.14}$$

où $c=\cos(\theta), s=\sin(\theta)$ pour $\theta\in[-\pi,\pi]$.La matrice orthogonale G fait une rotation du vecteur(c,-s), qui fait un angle de $-\theta$ avec l'axe x, d'un angle de θ se qui le fait coïncidé avec l'axe x :

$$G\begin{pmatrix} c \\ -s \end{pmatrix}=\begin{pmatrix} c & -s \\ s & c \end{pmatrix}\begin{pmatrix} c \\ -s \end{pmatrix}=\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Une rotation de Givens de $n\times n$ remplace un bloc de 2×2 d'une diagonale d'une matrice identité $n\times n$ par un bloc d'une rotation de Givens de 2×2.

$$G = \begin{pmatrix} 1 & 0 & & & \cdots & \cdots & \cdots & & & 0 \\ 0 & \ddots & \ddots & & & & & & & \\ & \ddots & \ddots & \ddots & & & & & & \\ & & \ddots & 1 & 0 & & & & & \vdots \\ \vdots & & & 0 & c & -s & & & & \vdots \\ \vdots & & & & s & c & 0 & & & \vdots \\ \vdots & & & & & 0 & 1 & \ddots & & \\ & & & & & & \ddots & \ddots & \ddots & \\ & & & & & & & \ddots & \ddots & 0 \\ 0 & & & \cdots & \cdots & \cdots & & & 0 & 1 \end{pmatrix}. \qquad (4.15)$$

FIG.4.3 LA ROTATION DE GIVENS

on note une rotation de Givens de $n \times n$ $G_j(c,s)$ une rotation de type (4.14) avec une rotation de Givens de 2×2 pour la colonne et la ligne j et j+1.

Les rotations de Givens sont utilisées pour réduire la matrice de Hessenberg supérieur à une matrice triangulaire supérieure et par conséquent à résoudre le problème au moindre carrée résulter de l'algorithme de GMRES. Cette méthode peut être aussi utilisé pour la résolution des problèmes aux valeurs propres par la méthode de QR.

On réduit H à la forme triangulaire en premier par sa multiplication par une matrice de rotation de Givens qui annule $h_{2,1}$(et bien sur qui change la valeur de $h_{1,1}$ et les colonnes suivantes).

On défini $G_1 = G_1(c_1, s_1)$ par :

$$c_1 = h_{11} / \sqrt{h_{11}^2 + h_{21}^2} \quad \text{et} \quad s_1 = -h_{21} / \sqrt{h_{11}^2 + h_{21}^2} \ .$$

Si on remplace H par G_1H, alors la première colonne de H ne contient maintenant qu'un seul élément non nul h_{11}. D'une manière similaire on peut appliquer $G_2(c_2, s_2)$ à H où :

$$c_2 = h_{22} / \sqrt{h_{22}^2 + h_{32}^2} \quad \text{et} \quad s_2 = -h_{32} / \sqrt{h_{22}^2 + h_{32}^2} \ .$$

et qui annule h_{32}. Notons que l'application de G_2 n'affecte pas la première colonne, et similairement l'application de G_i n'affecte pas les i-1 colonnes précédentes.

En continuant ainsi et en mettant : $Q = G_n G_{n-1} .... G_2 G_1$.

On peut voir que $QH = R$ est une triangulaire supérieure.

4.6 Méthode GMRES de base

Soit le problème (*P*) : résoudre le système linéaire *Ax=b*

4.6.1 Principe de la méthode de GMRES (utilisation de $\overline{\mathrm{H}}$)

Le principe de la méthode GMRES de base consiste à approcher x par un vecteur de forme $x_0 + z$ du problème (*P*) où :

- x_0 est un vecteur initial choisi par l'utilisateur,
- $z \in K_k(A, r_0) = lin\{r_0, Ar_0, ..., A^{k-1}r_0\}$n un sous-espace du Krylov avec $r_0 = b - Ax_0$ le résidu initial et k le numéro de l'itération.

Pour ce faire,

1. on construit une base orthonormale $V_k = \{v_1, v_2, ..., v_k\}$ de l'espace de Krylov $K_k(A, r_0)$ et une matrice $\overline{H_k} \in C^{k+1 \times k}$ provenant d'une décomposition de Hessenberg. Cette décomposition s'effectue par la méthode d'Arnoldi étudiée dans la section précédente. Pour un vecteur v_1 choisi tel que $v_1 = \frac{r_0}{\|r_0\|}$, les méthodes de GMRES suivant le principe d'Arnoldi de décomposition incomplète de Hessenberg par l'orthogonalisation de Householder, Gram Schmidt classique et Gram Schmidt modifiée seront équivalentes en arithmétique exacte. On rappelle que les matrices $A, V_K et \overline{H_k}$ satisferont la relation

$$AV_k = V_{k+1}\overline{H_k}$$

2. A chaque étape, on résout le problème de moindres carrés : $\min_{z \in K_k} \|b - A(x_0 + z)\|$

c'est-à-dire $\min_{z \in K_k} \|r_0 - Az\|$. Si on pose $z = V_k y \in K_k$, alors on peut écrire $\|r_0 - Az\| = r^{(1)}(y)$ avec $r^{(1)}(y) = \|\beta v_1 - AV_k y\|$

Où on a posé $v_1 = \frac{r_0}{\|r_0\|}, \beta = \pm\|r_0\|$ (le signe de β dépend de l'orthogonalisation utilisée par la méthode GMRES : positif pour Gram Schmidt, résultant de $P_1 z = \beta e_1$ pour Householder).

On peut encore écrire que $\|\beta v_1 - AV_k y\| = r^{(2)}(y)$ avec $r^{(2)}(y) = \|V_{k+1}(\beta e_1 - \overline{H}_k y)\|$ où e_1 est la première colonne de la matrice identité appartenant à $R^{(k+1)\times(k+1)}$ puisque V_{k+1} est orthonormal, il vient que $\|V_{k+1}(\beta e_1 - \overline{H_k y})\| = r^{(3)}(y)$ avec $r^{(3)}(y) = \|\beta e_1 - \overline{H}_k y)\|$

La solution approchée du problème $\min_x \|b - Ax\|$ est donnée par $x_k = x_0 + V_k y_k$ où y_k minimise la fonction $r^{(3)}(y) = \|\beta e_1 - \overline{H}_k y)\|$ avec $y_k \in \square^k$

4.6.2 GMRES algorithmes

Par l'application de la construction de Gram-Schmidt dans le processus d'Arnoldi on aura la matrice $k+1\times k$ H_k de coefficients h_{ij} une matrice de Hessenberg supérieure.

En gardant le problème au moindre carrée telle qu'il est, et en se basant sur les informations et les explications fournis par les sections précédentes, et où le teste d'arrêt est que $\rho = ||r_k||$ soit inférieur à une valeur ε fixé par l'utilisateur, on aura l'algorithme GMRES de base avec orthogonalisation de Gram-Schmidt classique qu'on note ***gmres_cgs*** qui à la forme suivante :

Algorithme 1 : algorithme *gmres_cgs* (x, b, A, ε , k*max*)

1. $r = b - Ax, v_1 = \dfrac{r}{\| r \|}, \rho = \| r \|, \beta = \rho, k = 0.$

2. tant que $\rho > \varepsilon$ et k$\leq$ k*max*

   (a) $k = k + 1.$

   (b) pour j = 1,….,k $\quad h_{jk} = (Av_k)^T v_j$

   (c) $v_{k+1} = Av_k - \sum_{j=1}^{k} h_{jk} v_j$

   (d) $h_{k+1,k} = \| v_{k+1} \|$

   (e) $v_{k+1} = \dfrac{v_{k+1}}{\| v_{k+1} \|}$

   (f) minimise $\| \beta e_1 - H_k y^k \|_{R^{k+1}}$ envers R^k pour obtenir y^k

   (g) $\rho = \| \beta e_1 - H_k y^k \|_{R^{k+1}}$

3. $x_k = x_0 + V_k y^k.$

Une importante propriété de GMRES est qu'il est indispensable de mémoriser la base de l'espace de Krylov au fur et à mesure que la progression des itérations. Pour réaliser k GMRES itérations on doit sauvegarder k vecteurs de tailles n. Pour les très grands systèmes cela peut devenir prohibitive. Une manière de procéder est d'itérer jusqu'à un k maximum qui représente le nombre maximum de vecteurs à sauvegarder. GMRES teste la norme du résidu $b - Ax_k$ et la compare à la valeur tolérée ε, si celle si est toujours supérieur GMRES redémarrera avec comme solution initiale le résultat de la dernière série d'itérations x_{kmax}. Cette version de GMRES avec redémarrage de l'algorithme est dénoté ***gmres(m)***. Il n'y a aucun théorème pour la convergence de cet algorithme et le redémarrage entraînera un retardement de la convergence qui est une résultante d'un autre

problème plus gênant pour l'implémentation de l'algorithme gmres_cgs est le fait que les vecteurs v_i peuvent perdre leur orthogonalité à cause des calculs en précision finie qui font accumuler les erreurs, s'il arrive que cette perte soit un peut plus importante l'évaluation du résidu peut ne pas être juste et la solution peut ne pas être atteinte. Ce qui veut dire que l'algorithme va stagné et ne fournira jamais de résultat.

Un remède partiel à ce problème est de modifier l'orthogonalisation classique de Gram-Schmidt : on remplace l'étape 2c de l'algorithme ***gmres_cgs*** par :

- $v_{k+1} = Av_k$
- pour $j = 1,....,k$

$$v_{k+1} = v_{k+1} - (v_{k+1}^T v_j)v_j.$$

Ce qui nous conduit à l'algorithme dit avec orthogonalisation de Gram-Schmidt modifié présenté si après et noté ***gmres_mgs*** :

Algorithme 2 : algorithme *gmres_mgs*(x, b, A, ε , k*max*)

1. $r = b - Ax, v_1 = \dfrac{r}{\| r \|}, \rho = \| r \|, \beta = \rho, k = 0.$

2. tant que $\rho > \varepsilon$ et k$\leq$ k*max*

   (a) $k = k + 1.$

   (b) $v_{k+1} = Av_k$

   pour j = 1,….,k

   i. $h_{jk} = v_{k+1}^T v_j$

   ii. $v_{k+1} = v_{k+1} - h_{jk} v_j$

   (c) $h_{k+1,k} = \| v_{k+1} \|$

   (d) $v_{k+1} = \dfrac{v_{k+1}}{\| v_{k+1} \|}$

   (e) minimise $\| \beta e_1 - H_k y^k \|_{R^{k+1}}$ envers R^k pour obtenir y^k

   (f) $\rho = \| \beta e_1 - H_k y^k \|_{R^{k+1}}$

3. $x_k = x_0 + V_k y^k.$

Cette algorithme représente seulement l'introduction de l'orthogonalisation de Gram-Schmidt modifié dans le processus d'Arnoldi et pas la version redémarré de la méthode qui peut être appliquée pour n'importe quelle type d'orthogonalisation, et que nous présenterons un peu plus loin.

Une simple application dans l'algorithme **gmres_mgs** des idées exposée précédemment a comme optique la résolution du problème au moindre carrée par l'application du produit de k rotations de Givens qui donne Q.

Mettons $g = \beta Q e_1$ et notons que la résolution du problème se réduit à :

$$\| \beta e_1 - H_k y^k \|_{R^{k+1}} = \| Q(\beta e_1 - H_k y^k) \|_{R^{k+1}} = \| g - R_k y^k \|_{R^{k+1}}, \qquad (4.16)$$

où R_k est une matrice triangulaire supérieure de $(k+1)\times k$ résultante de la factorisation QR de H_k.

On peut performer la factorisation QR de H au fur et à mesure que la réalisation des itérations de l'algorithme ***gmres_mgs***. En fait, si $R_k = Q_k H_k$ et après orthogonalisation, on rajoute une colonne h_{k+2} à H_k, on peut faire une mis à jour de Q_k et R_k en premier lieu par une multiplication de h_{k+2} par Q_k (se qui va appliqué la première rotation de Givens à h_{k+2}), et puis réaliser la rotation de Givens G_{k+1} qui élimine le $(k+2)^{\text{ième}}$ élément de $Q_k h_{k+2}$, et finalement $Q_{k+1} = G_{k+1} Q_k$ et former R_{k+1} par l'augmentation de R_k par $G_{k+1} Q_k h_{k+2}$.

Ainsi après l'utilisation des rotations de Givens l'algorithme ***gmres_mgs*** sera noté ***gmres*** tout court. Pour nous c'est l'algorithme de GMRES de base, et il prendra la forme suivante :

Algorithme 3 : algorithme *gmres* (x, b, A, ε , k*max*)

1. $r = b - Ax, v_1 = \dfrac{r}{\| r \|}, \rho = \| r \|, \beta = \rho, k = 0, g = \rho(1, 0...., 0)^T \in R^{k\max+1}$.

2. tant que $\rho > \varepsilon$ et k≤ k*max*

(a) $k = k + 1$.

(b) $v_{k+1} = A v_k$

pour j = 1,….,k

i. $h_{jk} = v_{k+1}^T v_j$

ii. $v_{k+1} = v_{k+1} - h_{jk} v_j$

fin pour

(c) $h_{k+1,k} = \| v_{k+1} \|$

(d) $v_{k+1} = \dfrac{v_{k+1}}{\| v_{k+1} \|}$

(e)i. si k>1 appliquer Q_{k-1} à la $k^{\text{ième}}$ colonne de H.

ii. $\upsilon = \sqrt{h_{k,k}^2 + h_{k+1,k}^2}$.

iii. $c_k = h_{k,k} / \upsilon, s_k = -h_{k+1,k} / \upsilon$

$h_{k,k} = c_k h_{k,k} - s_k h_{k+1,k}, h_{k+1,k} = 0$.

vi. $g = G_k(c_k, s_k) g$.

(f) $\rho = |(g)_{k+1}|$.

3. $r_{i,j} = h_{i,j}$ pour $1 \le i, j \le k$.

$(w)_i = (g)_i$ pour $1 \le i \le k$.

résoudre le système triangulaire supérieure $Ry^k = w$.

4. $x_k = x_0 + V_k y^k$.

On close cette section par la présentation de la variante de l'algorithme dite avec redémarrage ou réinitialisation dénoté ***gmres(m)*** et proposé par Saad lui même pour détourné le problème de la taille du sous espace de Krylov quand le système initiale est de très grande taille. La convergence de cette variante est loin d'être assurée et même qu'elle est beaucoup moins rapide que les versions de base appelées ***full gmres*** et qui ont la structure de l'algorithme (4.13) avec des différences par fois intéressantes au niveau techniques de programmations, de minimisation des opérations à effectué, de gestion de la mémoire, de calcul de l'erreur et au niveau testes d'arrêt.

L'idée principale de cette variante et la structure générale est représentée si après :

Algorithme 4 : algorithme *gmres(m)* (x, b, A, ε , k*max*, m)

1. ***gmres*** (x, b, A, ε , m)
2. $k = m$.
3. tant que $\rho > \varepsilon$ et k$\le$ k*max*

(a) ***gmres*** (x, b, A, ε , m)

(b) $k = k + m$.

4.7 Organigramme de la méthode GMRES

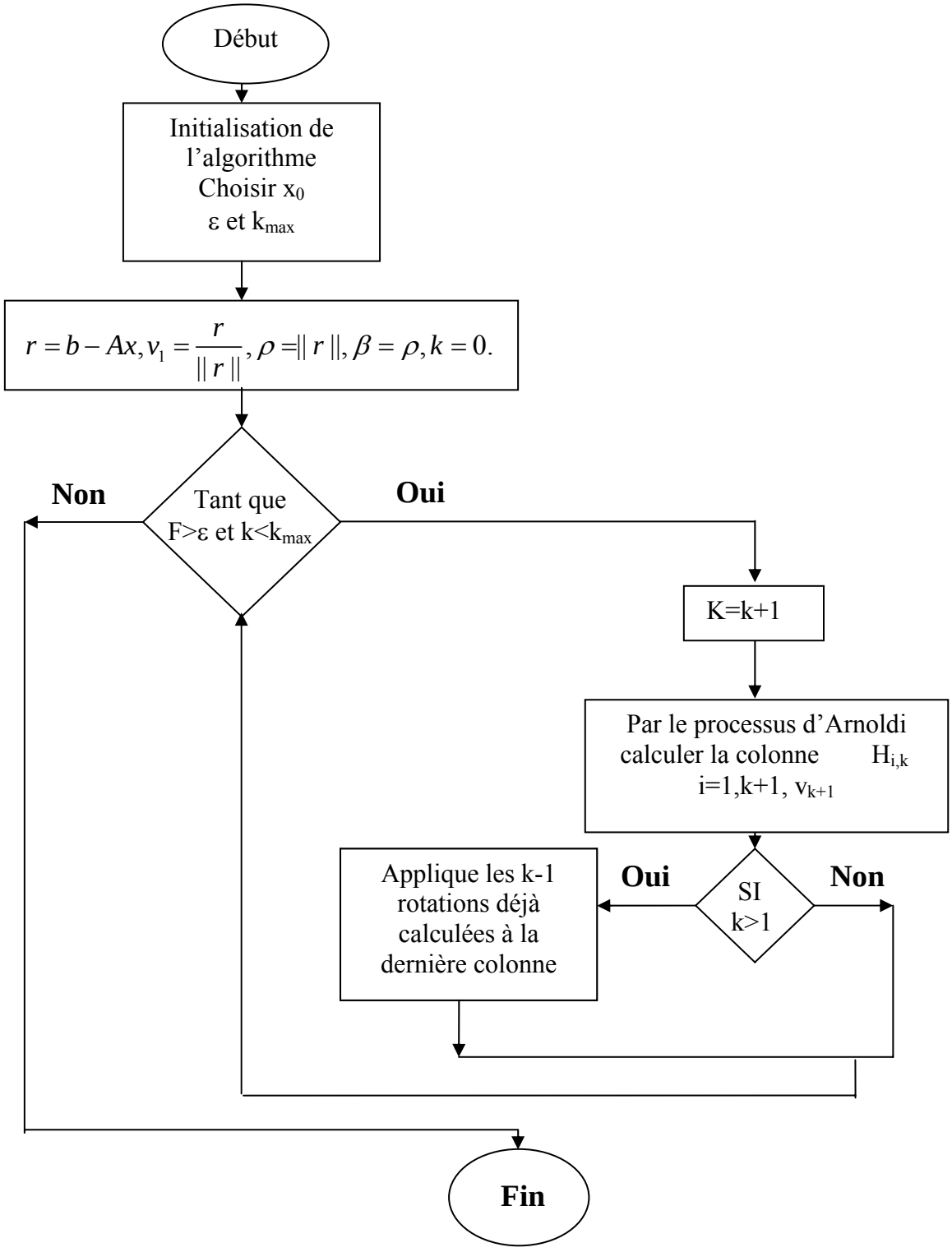


FIG.4.4 ORGANIGRAMME DE LA METHODE GMRES

Chapitre 5
Implémentation du code GMRES pour les systèmes de grande taille

5.1 Introduction

Une méthode itérative est une méthode qui résout un problème (comme une équation ou un système d'équations) en trouvant une succession d'approximations en commençant par une valeur initiale. Cette approche est en contraste avec les méthode directes qui résolvent les problèmes en une fois (comme résoudre un système linéaire $Ax = b$ en calculant la matrice inverse de A). Parmi les méthodes itératives les plus connues ces dernières années la méthode GMRES,la majorité des implémentations sont basées sur le cette algorithme, à savoir l'algorithme GMRES avec redémarrage (restart) représenté dans le chapitre précédant.

Dans notre étude, on utilise le même algorithme comme la méthode de résolution pour le KSP.

Cela implique qu'on n'a pas d'interface de manipulation, tout doit être bien réglé et bien adapté pour notre cas, et notre but principale est de mettre en œuvre un code qui peuvent résoudre des systèmes de grand nombre de degrés de liberté (l'ordre de 20 000 ddl, 30 000 ddl etc.)

5.2 Comparaison de GMRES avec la méthode directe de Gauss :

La méthode du pivot de Gauss est une méthode pour transformer un système en un autre système équivalent (ayant les mêmes solutions) qui est triangulaire et est donc facile à résoudre, c'est une méthode parmi les méthodes directes les plus connues.

Dans la figure suivante, on présente une comparaison de temps de calculs entre la méthode GMRES et Gauss avec pivot

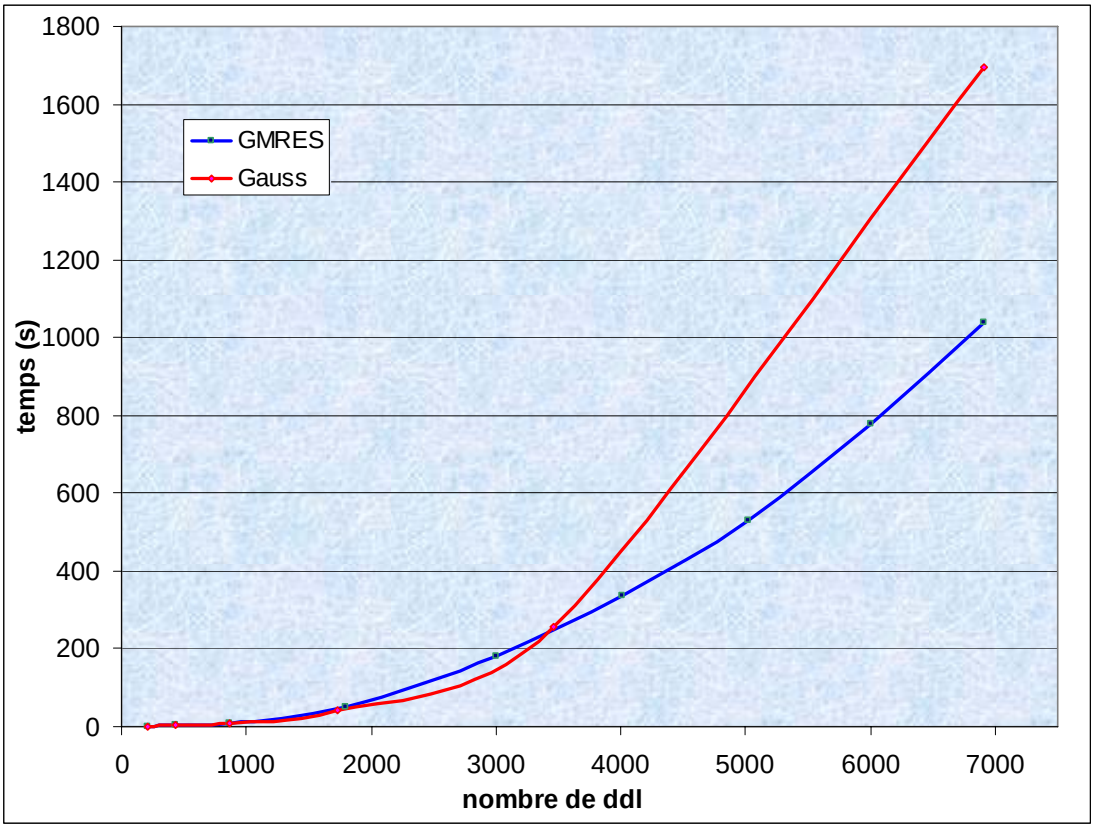


FIG. 5.1 COMPARAISON DE TEMPS DE CALCUL ENTRE LA METHODE GMRES ET GAUSS

On distingue dans le graphe la présence de deux zones :

La première zone où la méthode de Gauss est plus rapide que la méthode GMRES pour un nombre de degré de liberté inférieur à 3500.

La deuxième zone, à partir d'un nombre de degré de liberté supérieur à 3500, GMRES devient très rapide par rapport à Gauss, on remarque bien, dans le cas d'un problème de 7000 ddl il y a une différence de plus de 700 seconds, ce qui nous donne un gain très important sur le temps de calcul (40 % comme gain)

D'après ce graphe, on peut dire que la méthode de Gauss reste très lente devant la méthode GMRES pour les systèmes de grande taille.

5.3 Influence de la précision sur le nombre d'itérations

Dans cette section, on va présenter l'influence la précision ε sur la convergence de l'algorithme pour trois exemples différents, on trace le nombre d'itérations en fonction de moins logarithme décimale de la tolérance :

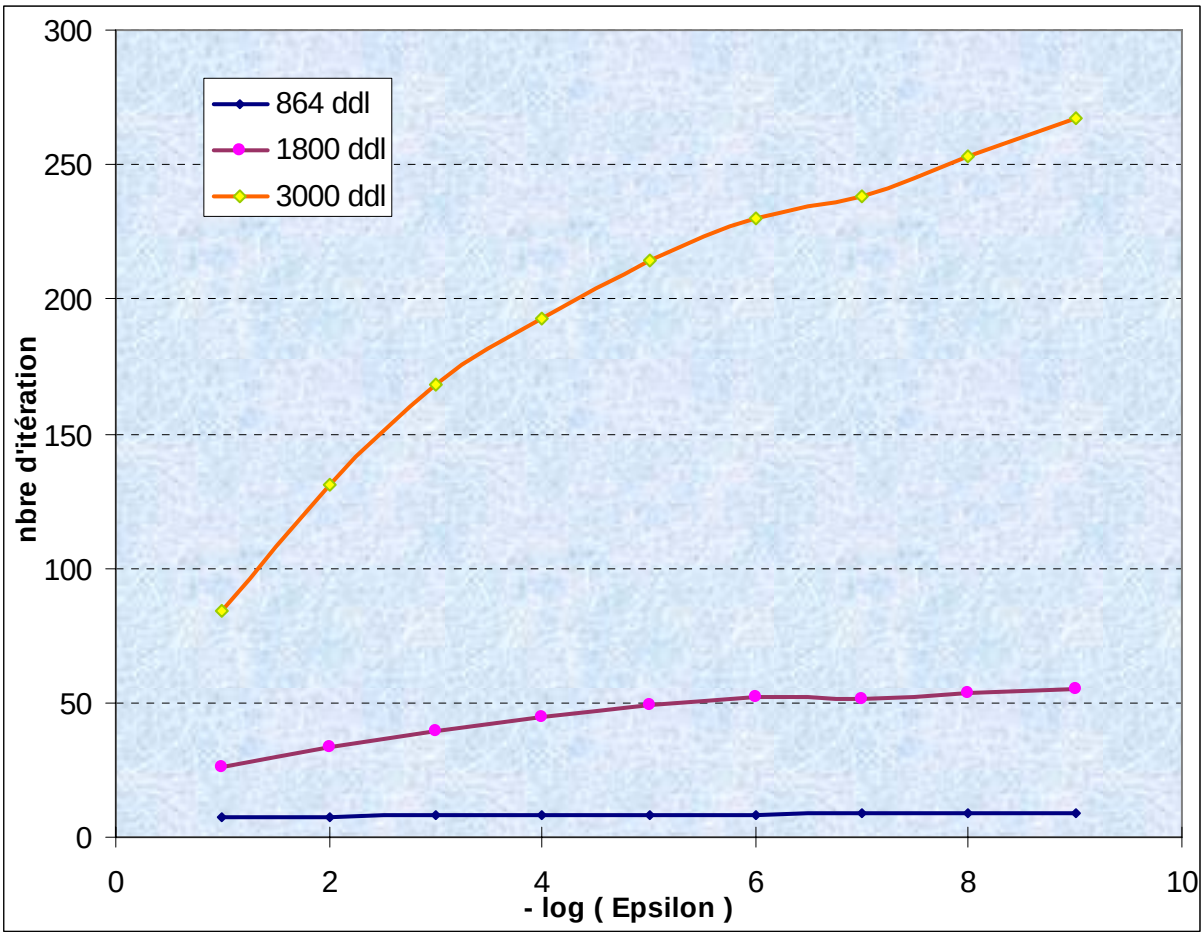


FIG. 5.2 INFLUENCE DE LA PRECISION SUR LE NOMBRE D'ITERATIONS

Pour les deux cas, 864 ddl et 1800 ddl on remarque que l'influence de la précision n'est pas très important sur le nombre d'itérations.

Par contre dans le cas de 3000 ddl , pour $\varepsilon = 0,1$ et $\varepsilon = 0,0000000001$ on a une différence du nombre d'itérations presque égale à 200 itérations de plus , ce qui est très coûteux sur le temps de calcul

Vu qu'on va traiter les problèmes de grande taille on doit bien choisir la précision selon le type de problème.

5.4 Le Choix de la solution initiale

Dans cette section on va voir l'influence du choix de la solution initiale dans le cas de la propagation des fissures, pour cela, on a fait des calculs en prenant la solution initiale égale à zéro et d'autre coté, en prenant comme solution initiale la solution trouvée dans l'itération précédente :

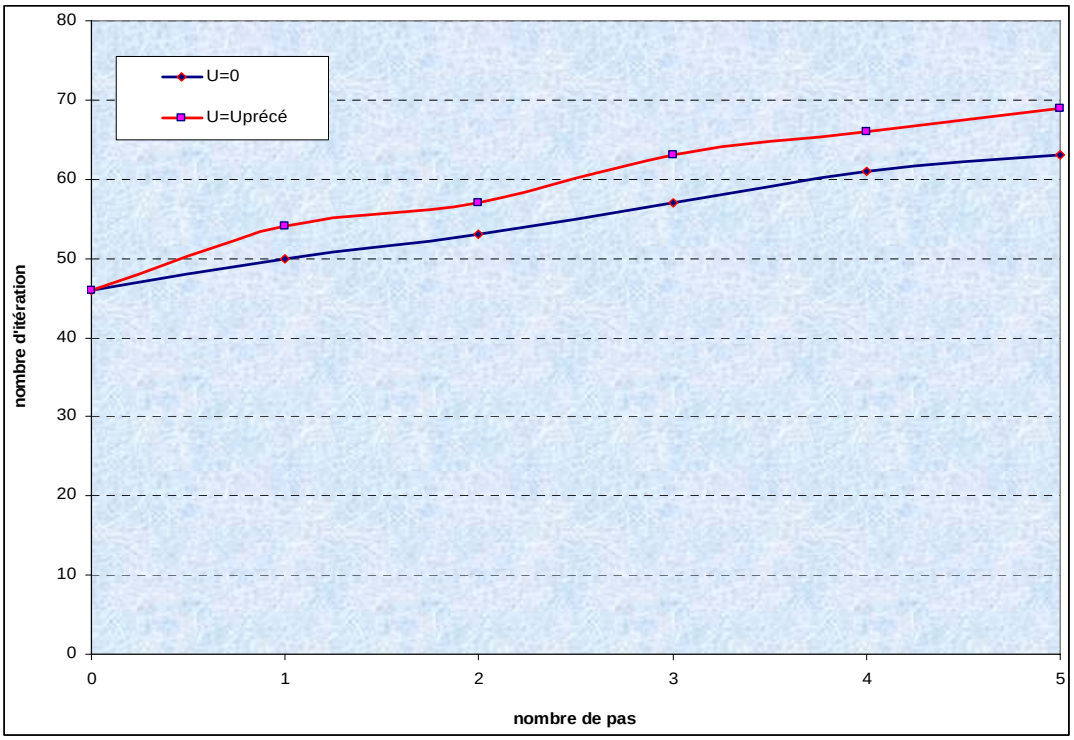


FIG. 5.3 COMPARAISON POUR U=0 ET U=U_{PREC} POUR UN PROBLEME DE 540 DDL

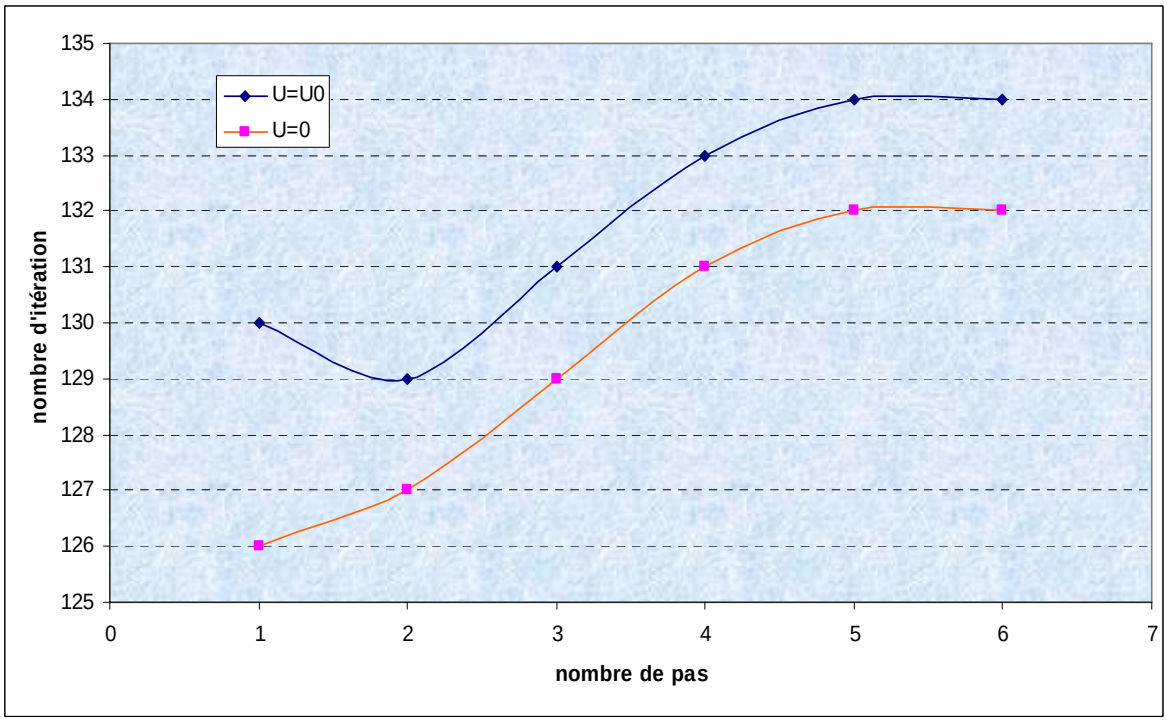


FIG. 5.4 COMPARAISON POUR U=0 ET U=U_{PREC} POUR UN PROBLEME DE 1764 DDL

Dans les graphes précédents on a présenté le tracé du nombre d'itérationS on fonction du nombre de pas pour deux problèmes différents 540 ddl, 1764 ddl

On remarque bien que choisir la solution initiale comme la solution précédente ne réalise pas un gain du coté nombre d'itérations et aussi le calcul devient instable , c'est pour cette raison qu'on as choisi U=0 comme solution initiale pour notre code calcul

5.5 Implémentation du code GMRES

Pour les matrices de taille moyenne, les méthodes directes sont très efficaces,par contre, pour les matrices de grande taille, et du fait que la matrice [K] est à diagonale dominante les méthodes itératives sont les plus efficaces.

Le essentiel problème pour la résolution des systèmes de grande taille (les systèmes qui dépassent les 10 000 ddl) est le sauvegarde de la matrice à résoudre parce qu'elle prend une taille énorme pour un titre d'exemple, un problème de 10 000 ddl occupe (8 octets x 10 000 x 10 000) 800 méga octets de la Ram, et il y a aussi le problème de la convergence des méthodes itératives qui devient instable pour les problèmes de grande taille.

On doit développer la méthode itérative célèbre GMRES pour la résolution des systèmes de grande nombre de degrés de liberté (20 000 ddl , 30 000 ddl,….)
Pour cela, on doit faire un :

- **Préconditionnement de la matrice**

- **Stockage de la matrice K dans un fichier au fur et à mesure de la création de système d'équations et vider la mémoire chaque fois.**

5.5.1 Préconditionnement de la matrice K

Le préconditionnement utilisé est le suivant :

1- sauvegarder les valeurs du pivot de la matrice
2- diviser chaque colonne de la matrice sur son pivot

alors le nouveau système à résoudre est :

$$[K']\{x'\} = \{y\}$$

Tel que : $K_{ij}' = \frac{K_{ij}}{K_{jj}}$, $x' = x \times K_{ij}$

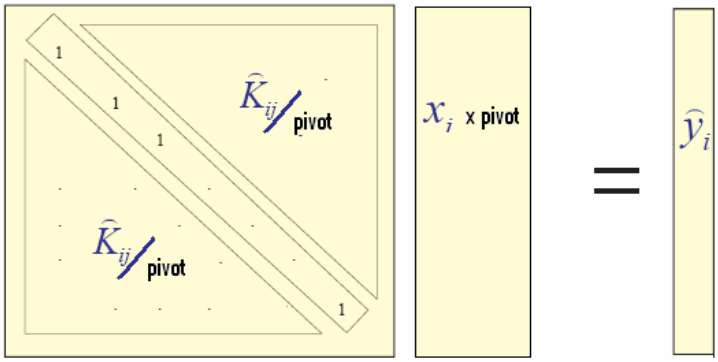


FIG. 5.5 LA MATRICE K APRES LE PRECONDITIONNEMENT

Dans la figure suivante on va voir l'effet du préconditionnement sur le temps de calcul :

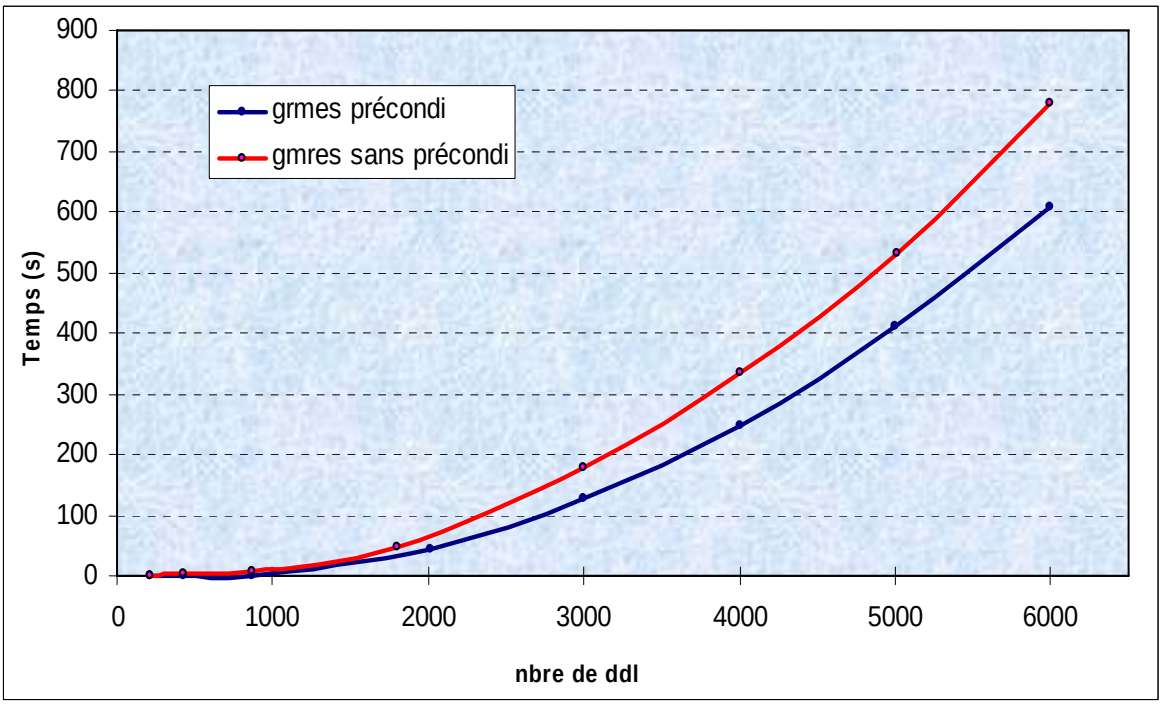


FIG. 5.6 EFFET DU PRECONITIONNEMENT SUR LE TEMPS DE CALCUL

On voit bien sur le graphe ci dessus, où on a tracé le temps de calcul en fonction du nombre de degrés de liberté pour GMRES et GMRES avec préconditionnement , plus que le nombre de degrés de libertés augmente on aura un gain de temps plus important , pour un exemple de 6000 ddl , on a un gain de 25 % de temps calculs.

Dans la figure suivante on va voir l'influence du préconditionnement sur le nombre d'itérations :

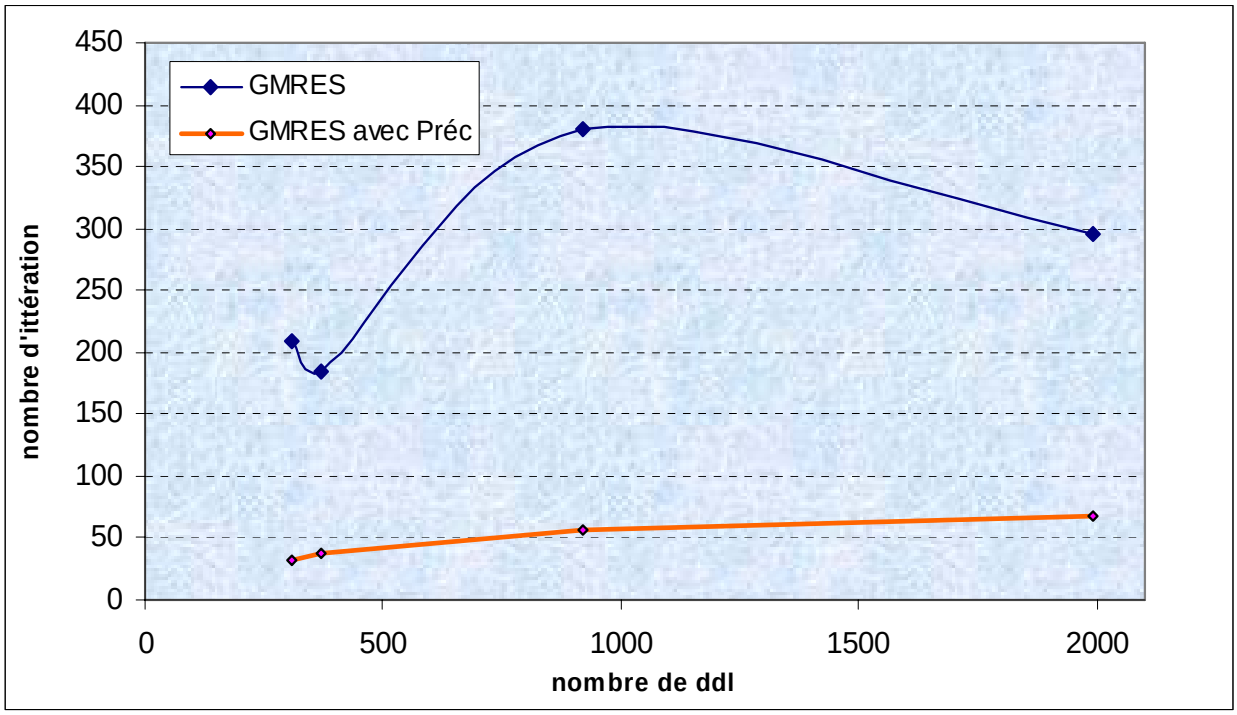


FIG. 5.7 EFFET DU PRECONITIONNEMENT SUR LE NOMBRE D'ITERATIONS

Dans ce graphe on a présenté le tracé du nombre d'itérations en fonction du nombre de degrés de liberté pour la méthode GMRES sans et avec préconditionnement, on peut distinguer deux choses essentielles :

- la méthode GMRES sans préconditionnement est instable par rapport GMRES avec préconditionnement
- un gain très important qui atteint le 700 % sur le nombre d'itérations

D'après les deux graphes précédents, on remarque bien la très grande influence d'un préconditionneur pour rendre GMRES plus stable, rapide, convergente et aussi pour avoir un gain très important sur le nombre d'itérations

Maintenant , après qu'on a fait toutes les optimisations sur la méthode GMRES du coté programmation , on utilisant un bon préconditionneur permet à GMRES de converger vers la solution plus rapidement, le choix d'une solution initiale U = 0 et le choix approprié de ε selon le problème à traité réduit le temps de calcul d'une façon considérable.

5.5.2 Stockage de la matrice K

Une matrice de grande taille est impossible de la résoudre sur un micro-ordinateur ordinaire, et le problème c'est la taille énorme que la matrice va l'occuper , par exemple

une matrice de 50 000 ddl et vue qu'on stock les cœfficients de la matrice K comme des Doubles (un Double en C++ occupe 8 octets de mémoire), ça veut dire que chaque cœfficient occupe 8 octets de mémoire , un problème de 50 000 ddl , il va occuper (50 000 x 50 000 x 8 octets) = 20 Giga octets , ce qui y est impossible de l'avoir sur la RAM d'un micro-ordinateur ordinaire.

Pour cela on doit :

- Stoker la matrice dans un fichier dans le disque dure du PC au moment de la création de la matrice et vider la RAM chaque fois

- Dans l'algorithme de résolution, on doit lire les coefficients de K directement à partir du fichier et sans faire appel à la matrice K.

La façon de stocker la matrice est très importante pour réduire le temps de calculs, le graphe suivant nous montre l'importance de sauvegarder la matrice colonne par colonne

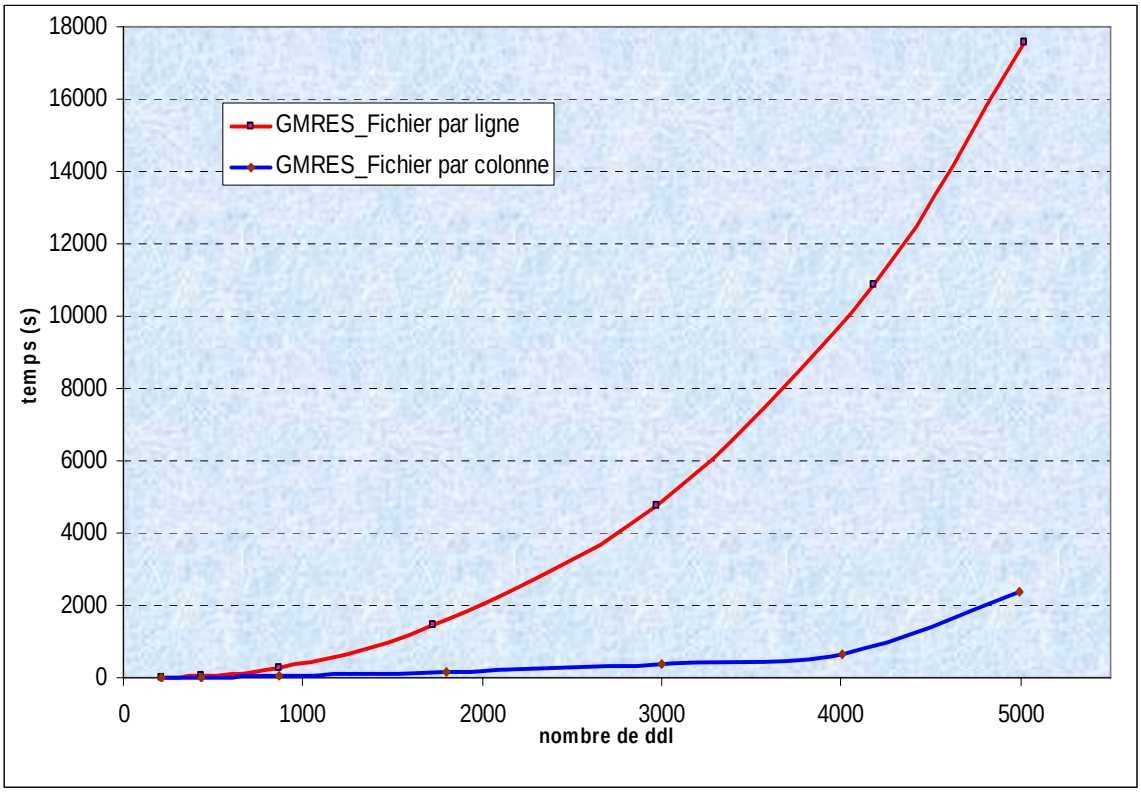


FIG. 5.8 COMPARAISON ENTRE GMRES SANS ET AVEC FICHIER

D'après la figure ci-dessus, on peut bien remarquer que sauvegarder la matrice du système à résoudre ligne par ligne va prendre beaucoup de temps pour la résolution , par contre si on sauvegarde le matrice colonne par colonne même s'on aura pas un gain du temps par

rapport la méthode GMRES directe , mais l'avantage c'est que la méthode GMRES directe ne peut pas résoudre des systèmes qui dépassent le 9 000 degré de liberté sur un micro-ordinateur ordinaire (1 Giga de RAM et un microprocesseur Pentium 4 de 2 Giga Hertz) par contre GMRES par fichier pour aller à des problèmes d'une taille très grande de l'ordre de 50 000 ddl .

5.6 Exemple d'un calcul de grand taille :

Dans l'exemple suivant on va traiter une Chape de 15 504 degré de liberté en utilisant la dernière version de KSP et comme méthode de résolution la méthode GMRES préconditionnée avec fichier,

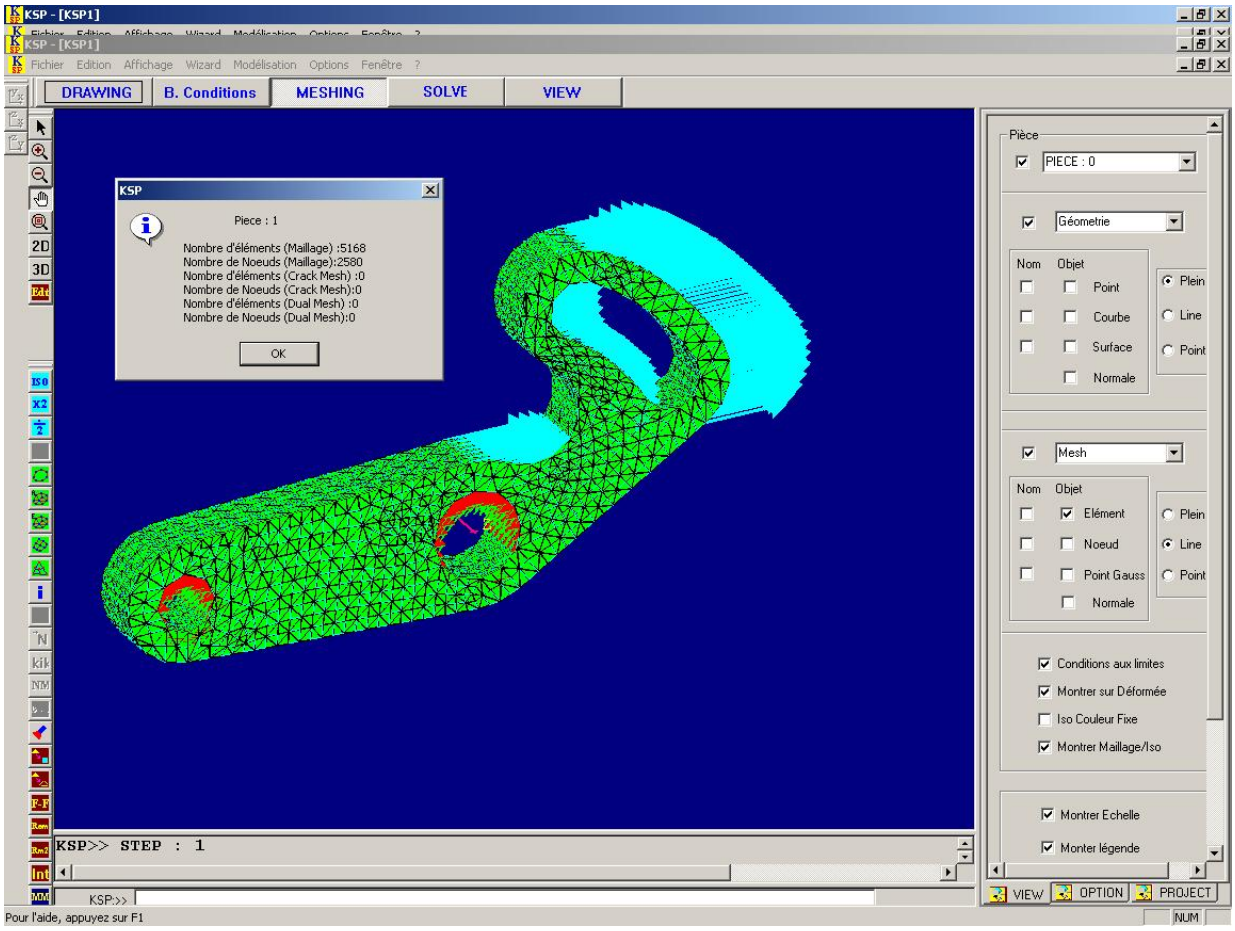


FIG.5.9 LE CHARGEMENT DE LA CHAPE

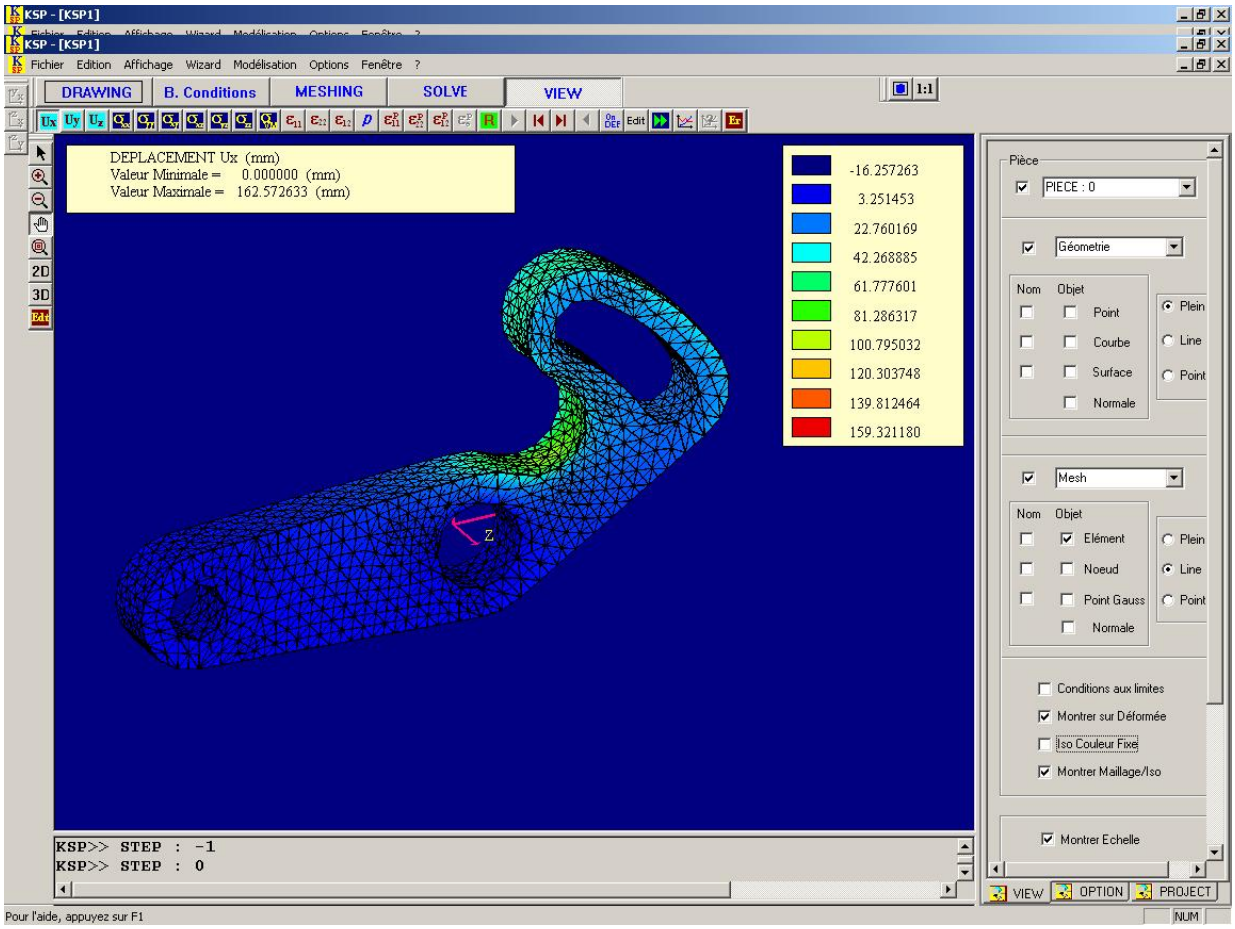


FIG.5.10 LA DEFORME DE LA CHAPE

Pour notre exemple la taille du fichier de sauvegarde de la matrice du système est 1,78 Giga octets, et c'est une énorme taille, on voit bien que c'est impossible de la stocker dans la RAM (dans notre cas la RAM du micro-ordinateur est 1 Giga octets)

La précision prise dans notre cas $\varepsilon = 0{,}00001$, ce qui donne une tolérance de 0,1 micro mètre.

Le nombre d'itérations égale à 40 itérations. On a comme temps de calcul global 13232,677 seconds, ça veut dire 3 heurs et 40 minutes.

Avec le nouveau code de GMRES qui s'appelle « GMRES Par Fichier », on a pu résoudre des systèmes impossible de les résoudre avec l'ancien version de GMRES sur des micro-ordinateurs ordinaires, cela conduit à la réalisation d'une nouvelle version de code KSP capable de traité même des problèmes de grande taille (plus de 10 000 ddl) .

Conclusion

L'objectif au départ de ce travail était de pouvoir traiter des problèmes de grande taille, d'un code utilisant la méthode des équations intégrales par l'implémentation de la méthode GMRES.

Dans le chapitre précédant, on remarque bien que notre objectif est atteint, vu qu'on a pu résoudre des systèmes de l'ordres de 15 000 degrés de liberté, ce qui y était impossible avec l'ancien version du KSP.

Ce travail a conduit à la réalisation d'un DLL (Direct Link Library) de résolution d'un système d'équations de très grande taille issu d'une modélisation par la méthode des équations intégrales où les seules entrées sont la taille du système à résoudre, la matrice du système à résoudre et le vecteur des coefficients à droite des équation, et une seule sortie qui est le vecteur de la solution recherchée

Le fait de stocker la matrice de système dans un fichier et l'utilisation un bonne préconditionneur nous a conduit à réaliser une nouvelle version de « GMRES par Fichier », qui peut résoudre des systèmes de très grande taille (30 000 , 50 000 ddl etc.) ce qui n'était jamais fait sur des calculateurs ordinaires (micro-ordinateur simple).

Le temps de calcul avec cette version de GMRES ne dépend pas seulement de la vitesse du microprocesseur et la capacité de RAM, mais il dépend aussi de la vitesse du lecture-écriture du disque dure, plus qu'on a un disque dure plus performant et plus rapide, il va réduire considérablement le temps de calcul avec la méthode de « GMRES Par Fichier ».

Bibliographe

[1] H. Kebir. ‘‘Approche déterministe et probabiliste de la prévision de la durée de vie aéronautique à l’aide des équations intégrales duales’’ . Université de Technologie de Compiègne. 1998

[2] H. Kebir, J. Marc. Roelandt, J. Gaudin, ‘‘ Simulation de la propagation de fissures dans les solides élastiques en mode mixtes par la méthode des équations intégrales duales’’. Revue européenne des éléments finis. 2000. Vol 9

[3] D. François, A. Pineau, A. Zaoui . ‘‘ Comportement Mécanique des matériaux’’. Edition Hermès . 1993 .

[4] C. Delannoy. ‘‘Programmer en langage C++ ’’.Editions Eyrolles .2000.5ème édition

[5] C. BATHIAS, J. BAILON. ‘‘La fatigue des matériaux et des structures’’. 2ème édition.Edition HERMES. 1997.

[6] Y. Saad. ‘‘ Iterative methods for sprase lineair system’’. PWS. Minnesota,1995

[7] Y. Saad and M. Schultz. ‘‘GMRES : A generalized minimal residual algorithm for solving nonsymmetric lineair systems’’. SLAM J. Sci. Stat. Comput. 1986.

[8] N. Recho ‘‘ Rupture par fissuration des structures’’. Editions Hermès. 1995

[9] J. Liberty ‘‘ Le languague C++ en 21 jours’’ CampusPress. 2001.

[10] I. Horton ‘‘Visual C++’’ Edition Eyrolles. 1999.

[11] Z. Gaiech, H. Kebir, J. – M.Roelandt, L. Chambon ‘‘ Calcul des structures fissurées en présence de contraintes résiduelles par la méthode des équations intégrales duales’’ . 6ème colloque nationale en calcul des structures, Giens. 2003.

Annexe A
Le code GMRES finale avec Fichier

```
void CSystemeDEquation::FileGMRESFinal(int &restart, int &iter)
{
int m;
double ** K=m_K.m_Valeur;
double *F=m_F.m_Valeur;
double *U=m_U.m_Valeur;

double epsilon;

int n=m_Taille;

m=int(0.15*n);//la taille maximale de la base du sous espace de Krylov

epsilon=0.0001;//la précision demandée sur les résultats


int i,j,jh,restartmax,northog;
double beta,bn,bea,be,dloo,dnormw,dnormx,dnormres,sic,
                dvi,aux,auxhjj,auxhjp1j;

double *w=new double[n], *r0=new double[n],*D=new double[n],*ycurrent=new
double[m], *xcurrent=new double[n];
double *rotcos=new double[m], *rotsin=new double[m],*h=new double[m+1],**H,**V;
H=new double*[m+1]; for (i=0;i<m+1;i++) H[i]= new double[m+1];
V=new double*[n]; for (i=0;i<n;i++) V[i]= new double[m];


restartmax=20;// le nombre maximum de redémarrage de l'algorithme
iter=0;//initialisation du compteur des itérations
restart=0;//initialisation du compteur des redemarrages


bn=dnrm2(n,F);// calcul de la norme 2 de vecteur F  , bn= la norme2 de F
if(bn==0)
  {
  for(j=0;j<n;j++)U[j]=0;
  delete[] w;delete[]r0;delete[]D;delete[]ycurrent;delete[]xcurrent;
  delete[]rotcos;delete[]rotsin;delete[]h;
  for (i=0;i<m+1;i++) delete[]H[i];delete[] H;
  for (i=0;i<m;i++) delete[] V[i];delete[] V;
  return;// si F est nul alor la solution est nul aussi
  }

for(j=0;j<n;j++)D[j]=m_D.m_Valeur[j];//K[j][j];//la construction du préconditionneur

double *tab;
```

```
tab = (double*)malloc(n*sizeof(double));

  for(j=0;j<n;j++)r0[j]=F[j];
Filedgemv('N',n,n,-1,K,U,1,r0,tab);
beta=dnrm2(n,r0);
if(beta==0)
{
        delete[] w;delete[]r0;delete[]D;delete[]ycurrent;delete[]xcurrent;
   delete[]rotcos;delete[]rotsin;delete[]h;
   for (i=0;i<m+1;i++) delete[]H[i];delete[] H;
   for (i=0;i<m;i++) delete[] V[i];delete[] V;
        return;//la solution initiale represente le solution exacte
}


//appliquer GMRES à K*D*y=b
for(j=0;j<n;j++)V[j][0]=r0[j]/beta;


do{

H[0][m]=beta;
for(j=0;j<m;j++)H[j+1][m]=0;



for(jh=1;jh<=m;jh++)

{

for(i=0;i<n;i++)        w[i]=0;


for(j=0;j<n;j++)
{
        lire_j(j,n,tab);

for(i=0;i<n;i++)


        {
                                        w[i]+=tab[i]*V[j][jh-1];
        }
}


for(j=0;j<jh;j++)H[j][jh-1]=0;
northog=0;

for(j=0;j<jh;j++)
```

```
{
        H[j][jh-1]=0;
        for(i=0;i<n;i++)H[j][jh-1]+=V[i][j]*w[i];
        daxpy(n,-H[j][jh-1],V,j,w);
}

dnormw=dnrm2(n,w);

H[jh][jh-1]=dnormw;
if(jh<m)
{
        aux=1/dnormw;
        for(j=0;j<n;j++)V[j][jh]=0;
        for(j=0;j<n;j++)V[j][jh]+=aux*w[j];
}

for(j=0;j<jh-1;j++)drot(H[j][jh-1],H[j+1][jh-1],rotcos[j],rotsin[j]);
auxhjj=H[jh-1][jh-1];
auxhjp1j=H[jh][jh-1];
drotg(auxhjj,auxhjp1j,rotcos[jh-1],rotsin[jh-1]);
drot(H[jh-1][m],H[jh][m],rotcos[jh-1],rotsin[jh-1]);
drot(H[jh-1][jh-1],H[jh][jh-1],rotcos[jh-1],rotsin[jh-1]);
H[jh][jh-1]=0;
dnormres=fabs(H[jh][m]);

 dcopy(jh,H,m,ycurrent);
 dtrsv('U','N',jh,H,ycurrent);
 dgemv('N',n,jh,1,V,ycurrent,0,r0);
 for(i=0;i<n;i++)xcurrent[i]=U[i]+r0[i];
 dnormx=dnrm2(n,xcurrent);

bea=dnormres;
if((bea<=epsilon)||(restart>=restartmax))
 {


         dcopy(jh,H,m,ycurrent);
         dtrsv('U','N',jh,H,ycurrent);
         dgemv('N',n,jh,1,V,ycurrent,0,r0);
         for(i=0;i<n;i++)xcurrent[i]=U[i]+r0[i];

 for(i=0;i<n;i++)r0[i]=xcurrent[i];
 Filedgemv('N',n,n,1,K,r0,0,w,tab);

 for(i=0;i<n;i++)w[i]=F[i]-w[i];

 for(i=0;i<n;i++)r0[i]=w[i];
 be=dnrm2(n,r0);

 if((be<=epsilon)||(restart>=restartmax))
```

```
        {
         for(i=0;i<n;i++)U[i]=D[i]*xcurrent[i];//réaliser x=D*y


delete[] w;delete[]r0;delete[]D;delete[]ycurrent;delete[]xcurrent;
delete[]rotcos;delete[]rotsin;delete[]h;
for (i=0;i<m+1;i++) delete[]H[i];delete[] H;
for (i=0;i<m;i++) delete[] V[i];delete[] V;

        return;
        }
  }
FILE * fe;

fe=fopen("matricek.txt","a");
fprintf(fe,"iter = %ld be=%lf bea=%lf \n",iter,be,bea);
fclose( fe);

iter++ ;



}

restart++;

  dcopy(jh-1,H,m,ycurrent);
  dtrsv('U','N',jh-1,H,ycurrent);
  dgemv('N',n,jh-1,1,V,ycurrent,0,r0);
  for(i=0;i<n;i++)U[i]=U[i]+r0[i];

  Filedgemv('N',n,n,1,K,U,0,r0,tab);
  for(i=0;i<n;i++)r0[i]=F[i]-r0[i];

  beta = dnrm2(n,r0);


for(j=0;j<n;j++)V[j][0]=r0[j]/beta;

}while(restart<=restartmax);
}
```

Annexe B

Les fonctions de GMRES

void CSystemeDEquation::daxpy(int n, double da, double **X, int j, double *Y)

```
{
for(int i=0;i<n;i++)Y[i]+=da*X[i][j];return;
}
```

void CSystemeDEquation::dcopy(int n, double **X, int m, double *Y)

```
{
for(int i=0;i<n;i++)Y[i]=X[i][m];return;

}
```

void CSystemeDEquation::dgemv(char trans, int m, int n, double alfa, double **A, double *X, double beta, double *Y)

```
{
double temp;
int i,j,jx,jy,kx=1,ky=1,leny;
if((m==0)||(n==0)||((alfa==0)&&(beta==1)))return;
leny=n;
if(trans=='N')leny=m;

for (i=0;i<leny;i++)Y[i]=beta*Y[i];

if(alfa==0)return;

if(trans=='N')
  {
  jx=kx;


  for(j=0;j<n;j++)
          {
          if(X[jx-1]!=0)
                  {
                  temp=alfa*X[jx-1];
                  for(i=0;i<m;i++)Y[i]+=temp*A[i][j];
                  }
          jx++;
          }
  }
else{
         jy=ky;
```

```
        for(j=0;j<n;j++)
                {
                temp=0;
                for(i=0;i<m;i++)temp+=A[i][j]*X[i];
                Y[jy-1]+=alfa*temp;
                jy++;
                }
        }
return;
}
```

double CSystemeDEquation::dnrm2(int n, double *X)

```
{
double s=0;
for(int i=0;i<n;i++)
        {
        s+=X[i]*X[i];
        }
return sqrt(s);
}
```

void CSystemeDEquation::drot(double &X, double &Y, double c, double s)

```
{
double dtemp;
dtemp=c*X+s*Y;
Y=c*Y-s*X;
X=dtemp;
return;
}
```

void CSystemeDEquation::drotg(double &da, double &db, double &c, double &s)

```
{
double roe,scale,r,z;
roe = db;
if(fabs(da)>fabs(db)) roe=da;
scale = fabs(da) + fabs(db);
if(scale==0)
  {c=1;s=0;r=0;z=0;}
else
        {
        r= scale*sqrt(pow((da/scale),2) + pow((db/scale),2));
        r= (roe/fabs(roe))*r;
        c= da/r;
        s= db/r;
        z= 1;
```

```
        if(fabs(da)>fabs(db))z=s;
        if((fabs(db)>=fabs(da))&&(c!=0))z=1/c;
        }
da= r;db= z;return;
}
```

void CSystemeDEquation::dtrsv(char uplo, char trans, int n, double **A, double *X)

```
{
double temp;
int i,j;


if(trans=='N')
  {
 if(uplo=='U')
        {
        for(j=n-1;j>=0;j--)
                {
                if(X[j]!=0)
                        {
                        if(trans=='N')X[j]=X[j]/A[j][j];
                        temp=X[j];
                        for(i=j-1;i>=0;i--) X[i]-=temp*A[i][j];
                        }
                }
        }
 else
                {
                for(j=0;j<n;j++)
                        {
                        if(X[j]!=0)
                          {
                          if(trans=='N')X[j]=X[j]/A[j][j];
                          temp=X[j];
                          for(i=j+1;i<n;i++) X[i]-=temp*A[i][j];
                          }
                        }
                }
 }

else
        {
        if(uplo=='U')
                {
                for(j=0;j<n;j++)
```

```
                        {
                        temp=X[j];
                        for(i=0;i<j;i++)
                                {
                                temp-=A[i][j]*X[i];
                                }
                        if(trans=='N')temp=temp/A[j][j];
                        X[j]=temp;
                        }
                }
        else
                {
                for(j=n-1;j>=0;j--)
                        {
                        temp=X[j];
                        for(i=n-1;i>j;i--)temp-=A[i][j]*X[i];
                        if(trans=='N')temp=temp/A[j][j];
                        X[j]=temp;
                        }
                }
        }
return;
}
```

void CSystemeDEquation::Filedgemv(char trans, int m, int n, double alfa, double **A, double *X, double beta, double *Y, double *tab)

```
{
double temp;
int i,j,jx,jy,kx=1,ky=1,leny;
if((m==0)||(n==0)||((alfa==0)&&(beta==1)))return;
leny=n;
if(trans=='N')leny=m;

for (i=0;i<leny;i++)Y[i]=beta*Y[i];

if(alfa==0)return;

if(trans=='N')
  {
  jx=kx;


  for(j=0;j<n;j++)
          {
          if(X[jx-1]!=0)
                  {
```

```
                lire_j(j,n,tab);



                temp=alfa*X[jx-1];

for(i=0;i<m;i++)Y[i]+=temp*tab[i];//for(i=0;i<m;i++)Y[i]+=temp*A[i][j];
                }
          jx++;
          }
  }
else{
          jy=ky;
          for(j=0;j<n;j++)
                {
                lire_j(j,n,tab);
                temp=0;
                for(i=0;i<m;i++)temp+=tab[i]*X[i];
//for(i=0;i<m;i++)temp+=A[i][j]*X[i];
                Y[jy-1]+=alfa*temp;
                jy++;
                }
          }
return;
}
```

void CSystemeDEquation::lire_j(int j, int n ,double *get_col)

```
{

   for (int i=0;i<n;i++) get_col[i]=0;
        int tmp=j*n;

   fseek(m_fichierMatriceK,tmp*sizeof(double),SEEK_SET);
   fread(get_col, sizeof(double),n, m_fichierMatriceK);

return;
}
```