Élimination des conditions aux limites dualisées

Résumé :
La prise en compte des conditions aux limites dans *Code_Aster* peut être réalisée soit par une technique directe d'élimination dans des cas simples (AFFE_CHAR_CINE), soit grâce à une technique de dualisation (AFFE_CHAR_MECA), et à l'introduction de multiplicateurs de Lagrange, pour les cas les plus généraux.

Néanmoins, cette approche présente deux inconvénients. D'une part, l'ajout de multiplicateurs de Lagrange augmente le nombre de degré de liberté du problème global, et donc la taille du système à résoudre. Ce point peut devenir pénalisant dès lors que le nombre de conditions aux limites augmente (nombreuses interfaces, liaisons cinématiques, impositions de mouvements rigides, etc.). D'autre part, la technique particulière de dualisation conduit à la perte du caractère semi-défini positif de la matrice de raideur. Ce changement de nature peut conduire à des défauts de robustesse, voir à l'échec de la résolution dans le cas des solveurs itératifs.

On présente dans ce document une méthode alternative à l'introduction de multiplicateurs de Lagrange pour prendre en compte les conditions aux limites affines. Cette approche ne se substitue pas à l'introduction des multiplicateurs de Lagrange, en particulier pour les études présentant du contact. En revanche, dans le cas des études où les conditions aux limites sont figées, cette approche permet des gains de robustesse, et de performance dans le cas d'un nombre important de conditions aux limites.

L'élimination des conditions aux limites est réalisée soit par l'utilisation du mot-clef ELIM_LAGR dans SOLVEUR [U4.50.01], soit directement avec l'opérateur ELIM_LAGR [U4.55.03] qui construit de nouveaux concepts matr_asse.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Table des Matières

1 Notations....3
2 Introduction....4
3 Principe de la résolution par élimination....4
3.1 Définition du problème à résoudre....4
3.2 Recherche de la solution particulière....5
3.3 Recherche de la solution générale....6
3.3.1 Obtention d'une base du noyau....6
3.3.2 Projection et résolution....6
4 Algorithme retenu pour la construction du noyau de la « matrice des contraintes »....8
4.1 Objectifs....8
4.2 Algorithme global....8
4.3 Construction itérative du noyau d'un vecteur....10
4.4 Calcul de la décomposition « QR-Qless »....11
5 Modes propres et élimination....13
6 Bibliographie....13
7 Annexe : code matlab de la construction du noyau (section 4.2)....14

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

1 Notations

$[K]_{Nt\,x\,Nt}$	Matrice de raideur contenant les conditions aux limites dualisées
$[A]_{NxN}$	Matrice de raideur du problème sans conditions aux limites
$[C]_{Nc\,x\,N}$	Matrice de définition des conditions aux limites ou « matrice des contraintes »
$[I_d]_{Nc\,x\,Nc}$	Matrice identité
$[T]_{N\,x\,Nl}$	Matrice associée au noyau (Ker) de C
$\{u\}_{Nx1}$	Vecteur des déplacements solutions
$\{u_0\}_{Ncx1}$	Vecteurs des déplacements imposés
$\{u_N\}_{Nlx1}$	Fonctions de forme vérifiant les conditions aux limites
$(\omega_0, \{\varphi\}_{Nlx1})$	Modes propres vérifiant les conditions aux limites
$\{f\}_{N\,x\,1}$	Vecteur du chargement imposé
$\{\lambda\}_{Nc\,x\,1}$, $\{\lambda_1\}_{Nc\,x\,1}$, $\{\lambda_2\}_{Nc\,x\,1}$	Vecteurs des multiplicateurs de Lagrange
N	Nombre de degrés de liberté du problème non contraint
Nc	Nombre de contraintes
Nt	Nombre total d'inconnues • $N+Nc$ pour une dualisation simple, • $N+2\mathrm{Nc}$ pour une dualisation double,
Nl	Nombre d'inconnues indépendantes

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

2 Introduction

La prise en compte des conditions aux limites dans *Code_Aster* peut être réalisée soit par une technique directe d'élimination dans des cas simples (AFFE_CHAR_CINE), soit grâce à une technique de dualisation (AFFE_CHAR_MECA), et à l'introduction de multiplicateurs de Lagrange, pour les cas les plus généraux. Cette dernière approche est très générale et permet de traiter sous un même formalisme tous les types de conditions aux limites.

Néanmoins, cette approche présente deux inconvénients. D'une part, l'ajout de multiplicateurs de Lagrange augmente le nombre de degré de liberté du problème global, et donc la taille du système à résoudre. Ce point peut devenir pénalisant dès lors que le nombre de conditions aux limites augmente (nombreuses interfaces, liaisons cinématiques, impositions de mouvements rigides, etc.). D'autre part, la technique particulière de dualisation conduit à la perte du caractère semi-défini positif de la matrice de raideur. La recherche de la solution n'est alors plus une recherche de minimum, mais une recherche de point selle [1]. Ce changement de nature peut conduire à des défauts de robustesse, voir à l'échec de la résolution dans le cas des solveurs itératifs.

On présente dans ce document une méthode alternative à l'introduction de multiplicateurs de Lagrange pour prendre en compte les conditions aux limites affines. Cette approche ne se substitue pas à l'introduction des multiplicateurs de Lagrange, en particulier pour les études présentant du contact. En revanche, dans le cas des études où les conditions aux limites sont figées, cette approche permet des gains de robustesse, et de performance dans le cas d'un nombre important de conditions aux limites. On rappelle dans un premier temps la technique générale qui préside à l'élimination des conditions aux limites, puis on présente la démarche retenue pour éliminer en pratique les conditions aux limites, ainsi que les différents algorithmes associés. Enfin, on présente la technique d'élimination dans le cas d'un problème aux valeurs propres.

3 Principe de la résolution par élimination

3.1 Définition du problème à résoudre

Le système à résoudre peut se mettre sous la forme générale suivante :

$$[A]\{u\}=\{f\} \tag{1}$$

sous la contrainte

$$[C]\{u\}=\{u_0\} \tag{2}$$

.

Avec l'introduction de multiplicateurs de Lagrange, le problème devient

$$\begin{cases}[A]\{u\}+[C]^T\{\lambda\}=\{f\}\\ [C]\{u\}=\{u_0\}\end{cases}, \tag{3}$$

et se met sous la forme générale

$$\begin{bmatrix}A & C^T\\ C & 0\end{bmatrix}\begin{Bmatrix}u\\ \lambda\end{Bmatrix}=\begin{Bmatrix}f\\ u_0\end{Bmatrix} \tag{4}$$

Dans le cas de la double dualisation (AFFE_CHAR_MECA), cette matrice est augmentée, et le système à résoudre devient

$$\begin{bmatrix}A & C^T & C^T\\ C & -\alpha I_d & \alpha I_d\\ C & \alpha I_d & -\alpha I_d\end{bmatrix}\begin{Bmatrix}u\\ \lambda_1\\ \lambda_2\end{Bmatrix}=\begin{Bmatrix}f\\ u_0\\ u_0\end{Bmatrix}. \tag{5}$$

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Après avoir posé naturellement $\lambda_1=\lambda_2=\lambda/2$.
Pour rendre son caractère semi-défini positif au problème générique $Ku=f$, dans le cas où la matrice K présente une topologie découlant de la mise sous la forme (4) ou (5), on va chercher à décomposer la solution u du problème sous la forme $u=u_p+u_g$, où

- u_p est la solution particulière du problème,
- u_g est la solution générale du problème, associée aux conditions aux limites $[C]\{u_g\}=\{0\}$.

3.2 Recherche de la solution particulière

La recherche de la solution particulière consiste simplement à trouver u_p tel que $[C]\{u_p\}=\{u_0\}$. Pour calculer u_p , on considère la décomposition sur l'image et le noyau de C , soit $u_p=u_{p-i}+u_{p-n}$, avec u_{p-i} appartenant à l'image de C et u_{p-n} appartenant au noyau de C .

Ce problème peut être résolu au sens des moindres carrés, et dans ce cas, on considère la solution de (2) de norme L_2 minimale, soit $u_p=u_{p-i}$. $\{u_{p-i}\}$ est la solution du problème de minimisation :

$$\{u_{p-i}\}=\underset{\{u_p\}}{\mathrm{ArgMin}}\left(\left\|[C]\{u_p\}-\{u_0\}\right\|^2\right) . \tag{6}$$

Pour calculer efficacement u_{p-i} , sous réserve que C soit de rang plein (rang égal à Nc , donc pas de contraintes redondantes), on exploite la propriété suivante :

$$\left([C][C]^T\right)\left([C][C]^T\right)^{-1}=[I_d]_{NcxNc} , \tag{7}$$

et donc

$$\left([C][C]^T\right)\left([C][C]^T\right)^{-1}\{u_0\}=\{u_0\} . \tag{8}$$

La solution recherchée est naturellement donnée par :

$$\{u_p\}=\{u_{p-i}\}=[C]^T\left([C][C]^T\right)^{-1}\{u_0\} \tag{9}$$

En pratique, la solution de (9) n'est pas donnée par le calcul direct de $C^T\left(CC^T\right)^{-1}$, mais par le calcul de la matrice R associée à la décomposition QR de C^T . Le calcul de Q , matrice orthogonale pleine, est cher et n'apporte pas d'information intéressante. En revanche, le calcul de R est facile et peu coûteux, et fournit toute l'information nécessaire, puisque :

$$C^T\left(CC^T\right)^{-1}=[C]^T\left([R]^T[Q]^T[Q][R]\right)^{-1}=[C]^T[R]^{-1}[R]^{-T} . \tag{10}$$

Par construction, R est une matrice de taille $NxNc$, avec un bloc triangulaire supérieur sur les Nc premières lignes :

$$[R]_{NxNc}=\begin{bmatrix}[R_{ts}]_{Nc\ x\ Nc}\\ [0]_{(N-Nc)xNc}\end{bmatrix} \text{ Avec } [R_{ts}]=\begin{bmatrix}R_{1,1} & \cdots & R_{1,Nc}\\ 0 & \ddots & \vdots\\ 0 & 0 & R_{Nc,\,Nc}\end{bmatrix} . \tag{11}$$

En pratique, les calculs successifs de l'effet de R^{-1} et R^{-T} sur $R^{-T}u_0$ et u_0 respectivement se font par simple descente remontée.

Remarque :

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Le calcul de R par décomposition QR permet de gérer le cas des contraintes redondantes. Si C n'est pas de rang plein, alors des zéros apparaissent sur la diagonale de R. Il suffit alors de supprimer les colonnes correspondantes, ce qui correspond à la suppression de la contrainte redondante. On construit alors une solution particulière vérifiant (2), de norme minimale, même dans le cas d'une matrice C de rang déficient.

3.3 Recherche de la solution générale

La solution générale u_g vérifiant $[C]\{u_g\}=\{0\}$, elle appartient au noyau T de C. La recherche de la solution générale passe donc par une étape de construction du noyau de C. On peut ensuite projeter le problème sur le noyau, et résoudre ce problème, mieux posé et de taille réduite.

3.3.1 Obtention d'une base du noyau

Formellement, la construction du noyau de C peut se faire de plusieurs façons :

- Décomposition en valeurs singulières de C, soit $C=USV^T$. $Ker(C)$ correspond aux colonnes de V^T associées à une valeur singulière nulle.
- Décomposition QR de C^T, soit $C=R^TQ^T$. $Ker(C)$ correspond alors aux colonnes de Q associées à une valeur nulle sur la diagonale de R^T.
- Décomposition LU de C^T. Le calcul du noyau n'est pas direct, et s'obtient à partir de sous-blocs de L.

On pose :

$$[L]=\begin{bmatrix}[L_1]_{NcxNc} & 0\\ [L_2]_{(N-Nc)xNc} & [L_3]_{(N-Nc)x(N-Nc)}\end{bmatrix} \tag{12}$$

Où L_1 et L_3 sont des matrices triangulaires inférieures. $Ker(C)$ est obtenu par

$$Ker(C)=\begin{bmatrix}-([L_2][L_1]^{-1})^T\\ [I_d]_{(N-Nc)x(N-Nc)}\end{bmatrix} \tag{13}$$

On fait ici l'hypothèse que C est de rang plein. Si tel n'est pas le cas, alors il suffit de construire L_1 à partir du bloc de L associé aux termes non nuls de la diagonale de U.

En pratique, ces techniques ne sont adaptées que pour des problèmes de dimensions réduites. La décomposition en valeur singulière est l'approche la plus coûteuse, suivie par la décomposition QR, puis la décomposition LU. Dans le cas présent, cette décomposition est réalisée par un algorithme particulier, présenté dans la section 4, prenant en compte la spécificité du problème à résoudre.

3.3.2 Projection et résolution

Une fois que la base du noyau est calculée, la résolution est directe. On pose $u_g=T\,\bar{u}$, et la solution du problème est donc directement donnée par

$$[T]^T[A][T]\{\bar{u}\}=T^T\{f\}\ , \tag{14}$$

puisque, par construction,

$$[T]^T[C]^T\{\lambda\}=\{0\}\ . \tag{15}$$

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Dans certains cas, en revanche, il est nécessaire d'accéder aux multiplicateurs de Lagrange. Il faut donc les reconstruire, à partir des solutions u_p et u_g. Pour ce faire, on doit résoudre la première ligne du système (3), soit

$$[C]^T\{\lambda\}=\{f\}-[A]\left([T]\{\bar{u}\}+\{u_p\}\right) \quad (16)$$

Si C est de rang plein, la solution s'écrit formellement :

$$\{\lambda\}=\left([C][C]^T\right)^{-1}[C]\{f\}-[A]\left([T]\{\bar{u}\}+\{u_p\}\right), \quad (17)$$

et se calcule en pratique en réutilisant la décomposition QR de C^T calculée pour la recherche de la solution particulière. De façon analogue, si C n'est pas de rang plein, il est possible de supprimer les contraintes redondantes.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

4 Algorithme retenu pour la construction du noyau de la « matrice des contraintes »

4.1 Objectifs

Pour gérer l'élimination des contraintes cinématiques, un algorithme de calcul du noyau a été développé pour prendre en compte les spécificités du problème à résoudre. Dans le cadre des éléments finis, la matrice A correspond à la matrice de raideur du problème, assemblée sans prise en compte des conditions aux limites. Cette matrice, même dans le cas de problèmes de grande taille, reste très creuse.

Par ailleurs, indépendamment des problèmes de coût de calcul liés à l'obtention du noyau de C par les techniques de décomposition, les matrices obtenues par ces procédés sont en général pleines. L'opération de projection (14) conduit donc, dans ce cas, à un problème de taille réduite, mais très dense, qui est donc inadapté pour les solveurs utilisés.

La méthodologie développée ici permet de construire, de façon séquentielle, une base T du noyau de C tout en assurant un remplissage minimal, afin de conserver un caractère creux pour le problème (14). Cette approche est d'autant plus efficace que le caractère creux de la matrice C est important.

4.2 Algorithme global

La construction de la base se fait de façon séquentielle. On initialise le processus en construisant une matrice $T_0 = I_d$ de la taille $N \times N$, soit le nombre de degrés de liberté du problème non contraint. On initialise également la matrice finale $T = T_0$. On initialise enfin le nombre de contraintes éliminées $Nbce$ à 0. On boucle ensuite sur les lignes de la matrice C, et on réalise les opérations suivantes :

Tant que $Nbce < N$ **, on fait :**
* Itération sur les lignes de la matrice C

- On initialise à zéro un vecteur $CONT$ contenant les indices des DDL déjà impliqués dans d'autres contrainte
- On initialise $NbCont = 0$

- Première itération :

  - On initialise à zéro un vecteur CE de taille Nc pour stocker les numéros des contraintes éliminées.
  - On initialise à zéro un vecteur NZ de taille N pour stocker les indices des degrés de liberté déjà impliqués dans une relation linéaire.
  - On extrait la première ligne C_1
  - On norme la ligne extraite : $\|C_1\| = 1$
  - On calcule $\|C_1 T_0\|$

  - Si $\|C_1 T_0\| < \mathrm{Tol}$, la contrainte est déjà vérifiée
    - On indique que la première contrainte est éliminée : $CE(1) = 1$
    - On passe à l'itération suivante

  - Si $\|C_1 T_0\| \geqslant \mathrm{Tol}$ (dans le code, $\mathrm{Tol} = 10^{-12}$ par défaut), la contrainte n'est pas vérifiée. On cherche les indices i_{nz} associés aux termes non nuls de la première ligne C_1.
    - Si $Card(i_{nz}) = 1$, alors la condition cinématique n'implique qu'un seul degré de liberté.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- On met donc à 0 le terme associé sur la diagonale de T_0 : $T_0(i_{nz},i_{nz})=0$

- Sinon, on construit une base du noyau de $C_1(i_{nz})$. Cette base est construite de façon itérative, pour aboutir à une matrice triangulaire. Le détail est donné dans la section 4.3. On note Ker_1 cette matrice, de taille $Card(i_{nz})\times Card(i_{nz})$
  - On affecte Ker_1 dans T_0 : $T_0(i_{nz},i_{nz})=Ker_1$. On notera que, par construction, un des éléments de la diagonale de Ker_1 est nul.
- On stocke les positions i_{nz} des éléments non nuls de C_1 dans NZ : $NZ(i_{nz})=1$
- On indique que la première contrainte est éliminée : $CE(1)=1$
- On indique que les DDL i_{nz} sont impliqués dans une contrainte : $CONT(NbCont+(1:Card(i_{nz})))=i_{nz}$
- On incrémente le nombre de DDL déjà contraints : $NbCont=NbCont+Card(i_{nz})$
- On passe à l'itération suivante

* Itérations suivantes :

- On extrait la ligne courante C_k
- On norme la ligne courante : $\|C_k\|=1$
- On calcule $\|C_k T_0\|$
- Si $\|C_k T_0\|<\mathrm{Tol}$:
  - On indique que la contrainte est éliminée : $CE(k)=1$
  - On passe à l'itération suivante

- Si $\|C_k T_0\|\geqslant \mathrm{Tol}$, on cherche les indices i_{nz} associés aux termes non nuls de la ligne C_k .
- On détermine si certains degrés de libertés sont déjà impliqués dans d'autres relations linéaires, et on récupère les indices associés : $i_{cs}=\{i \mid NZ(i)=1, i\in i_{nz}\}$
- On construit l'ensemble complémentaire i_{lb} . On a donc $i_{nz}=\{i_{cs};i_{lb}\}$
  - Si $Card(i_{lb})=0$: Tous les degrés de liberté sont déjà impliqués dans des contraintes linéaires. En conséquence, le bloc $T_0(i_{nz},i_{nz})$ a déjà été rempli, et le modifier conduirait à ne plus vérifier une des contraintes précédentes.
    - On ne fait rien, et on passe à l'itération suivante. La contrainte C_k n'est pas éliminée pour l'instant.

  - Si $Card(i_{nz})=1$, et $i_{lb}=i_{nz}$, alors la condition cinématique n'implique qu'un seul degré de liberté, non encore impliqué dans une précédente relation.
    - On met donc à 0 le terme associé sur la diagonale de T_0 : $T_0(i_{nz},i_{nz})=0$
    - On indique que le DDL i_{nz} est impliqué dans une contrainte : $CONT(NbCont+1)=i_{nz}$
    - On incrémente le nombre de DDL déjà contraints : $NbCont=NbCont+1$

  - Sinon, on construit une base du noyau de $C_k(i_{nz})$ sous la contrainte de ne pas modifier les blocs de T_0 déjà initialisés, soit $T_0(i_{cs},CONT(1:NbCont))$. Le problème à résoudre s'écrit formellement

Trouver $T_0(i_{cs},i_{lb})$, $T_0(i_{lb},CONT(1:NbCont))$ et $T_0(i_{lb},i_{lb})$ tels que

$$[C_k(i_{cs})\ \ C_k(i_{lb})]\begin{bmatrix} T_0(i_{cs},CONT(1:NbCont)) & T_0(i_{cs},i_{lb}) \\ T_0(i_{lb},CONT(1:NbCont)) & T_0(i_{lb},i_{lb}) \end{bmatrix}=[0] \quad (18)$$

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- Si $Card(i_{cs})=0$, on ne construit que $T_0(i_{lb},i_{lb})$, les autres termes sont dégénérés, avec l'algorithme présenté dans la section 4.3, qui calcule la solution du problème $C_k(i_{lb})T_0(i_{lb},i_{lb})=0$

- Si $Card(i_{cs})>0$:
  - On construit $T_0(i_{lb},i_{lb})$ avec l'algorithme présenté dans la section 4.3, qui calcule la solution du problème $C_k(i_{lb})T_0(i_{lb},i_{lb})=0$
  - On calcule $T_0(i_{lb},CONT(1:NbCont))$ comme solution du problème aux moindres carrés

$$T_0(i_{lb},CONT(1:NbCont))=\underset{T}{\mathrm{Arg\,Min}}\left(\left\|C_k(i_{cs})T_0(i_{cs},CONT(1:NbCont))+C_k(i_{lb})T\right\|^2\right) \quad (19)$$

  - On calcule $\left\|C_k(i_{cs})T_0(i_{cs},i_{cs})+C_k(i_{lb})T\right\|$
  - Si $\left\|C_k(i_{cs})T_0(i_{cs},i_{cs})+C_k(i_{lb})T\right\|\geqslant Tol$, la contrainte ne peut pas être éliminée à cette étape, on passe à l'itération suivante.
  - On impose $T_0(i_{cs},i_{lb})=0$

- On stocke les positions i_{nz} des éléments non nuls de C_k dans NZ : $NZ(i_{nz})=1$
- On indique que les DDL i_{lb} sont impliqués dans une contrainte : $CONT(NbCont+(1:Card(i_{lb})))=i_{lb}$
- On incrémente le nombre de DDL déjà contraints : $NbCont=NbCont+Card(i_{lb})$
- On indique que la contrainte courante est éliminée : $CE(k)=1$
- On passe à l'itération suivante

* Après la dernière itération, on calcule la somme des termes de CE : $Nbne=Nbne+\sum_{k=1}^{Nc}CE(k)$.

- On met à jour la base du noyau à partir des contraintes déjà éliminées $T=T_0\ T$
- On met à jour la matrice des contraintes pour construction itérative : $C=C\ T_0$
- Si $Nbne<N$
  - On réinitialise $T_0=I_d$
  - On élimine les lignes de C correspondant à des contraintes déjà éliminées
  - On réinitialise le vecteur NZ .

Fin

Le processus tel que décrit ci dessus converge, puisqu'on assure d'éliminer au moins la première contrainte à chaque passage dans la boucle globale. En revanche, la qualité de la matrice T (nombre de termes non nuls) se dégrade rapidement quand le nombre d'itérations global augmente.

4.3 Construction itérative du noyau d'un vecteur

L'algorithme global présenté dans la section 4.2 repose sur les constructions successives de noyaux de vecteurs, correspondants aux lignes extraites de la matrice C . Le processus d'extraction des lignes de C s'assurant de ne récupérer que les valeurs non nulles de la ligne courante, un algorithme a été construit pour assembler une base orthogonale sous la forme d'une matrice triangulaire supérieure, afin de limiter le remplissage de la matrice T finale. L'idée consiste à construire itérativement une base orthogonale, en assurant que chaque nouveau vecteur est orthogonal :
- au vecteur V ,
- aux colonnes de Ker déjà remplies.

On suppose, pour la suite, que le vecteur V est de norme unitaire. Ce traitement est réalisé avant l'appel à cette routine.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

La topologie triangulaire supérieure permet d'assurer facilement la construction d'une telle matrice. L'algorithme se présente comme suit :
Soit V le vecteur donné en entrée, de longueur Nnz .

- On initialise la matrice Ker à zéro
- On boucle sur les valeurs de V pour déterminer la position i_{nz} du premier élément dont la norme est supérieure à une tolérance Tol fixée à la précision machine (R8PREM [D6.01.01])
- Si $i_{nz}=1$, alors $Ker(1,1)=0$. Sinon, $Ker(1,1)=1$
- On boucle sur les valeurs de V , en commençant par la deuxième valeur : $i=2:Nnz$
  - On teste la valeur de $V(i)$
    - Si $V(i)<Tol$, on fait $Ker(i,i)=1$
    - Sinon, si $i=i_{nz}$, on fait $Ker(i,i)=0$
    - Sinon,
      - on boucle sur $j=1:i-1$
        - On calcule le produit scalaire $ps=\left(Ker(1:j-1,j),Ker(1:j-1,i)\right)$
        - On remplit $Ker(j,i)=\frac{-ps}{Ker(j,j)}$
      - On calcule le produit scalaire $ps=\left(Ker(1:i-1,i),V(1:i-1)\right)$
      - On remplit $Ker(i,i)=\frac{-ps}{V(i)}$
      - On norme $Ker(1:i,i)=\frac{Ker(1:i,i)}{\|Ker(1:i,i)\|}$

4.4 Calcul de la décomposition « QR-Qless »

Pour calculer la solution particulière, et reconstruire les multiplicateurs de Lagrange, on fait appel à une décomposition QR de la matrice des contraintes. Pour ce faire, on a utilisé l'algorithme standard, publié dans [2]. Cet algorithme, dont la forme a été adaptée pour son utilisation à travers la librairie PETSc, est simplement reproduit ici. La matrice C est donnée en entrée. On stocke la diagonale de la matrice R dans un vecteur d . La décomposition écrase C , puisque R est stockée dans la partie triangulaire supérieure. La partie inférieure correspond aux termes non nuls des vecteurs w servant à la transformation de Householder. La matrice C est de taille $m\times n$.

for j = 1 to n do
$s=\sqrt{\|C(j:m,j)\|^2}$;
# calcul de la diagonale de R
if $C(j,j)>0$ then
$d(j)=-s$;
else
$d(j)=s$;
endif
# calcul de la partie triangulaire supérieure de R
$\alpha=\sqrt{(s(s+|C(j,j)|))}$;
$C(j,j)=C(j,j)-d(j)$;
$C(j:m,j)=C(j:m,j)/\alpha$;
# calcul des vecteurs w_i pour la transformation de Householder : $P_i=I_d-w_i w_i^T$
for i := j + 1 to n do
$s=C(j:m,j)^T C(j:m,i)$;
$C(j:m,i)=C(j:m,i)-s\,C(j:m,j)$;
end for i;

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

end for j;

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

5 Modes propres et élimination

Cette section fait écho à la section 7 de la documentation R3.03.01, et justifie la méthode d'élimination pour le calcul des modes propres. On s'intéresse ici au problème général suivant :

Trouver les couples $\left(\omega_0^2,\{\varphi\}\right)$ vérifiant $\left([K_c]-\omega_0^2[M_c]\right)\{\varphi\}=0$ (20)

Les matrices M_c et K_c doivent prendre en compte les conditions aux limites, et sont ici supposées obtenues à partir de fonctions de formes u_N vérifiant $Cu_N=0$. Or, dans le cas présent, le problème assemblé n'est pas le problème (20), mais un problème de taille plus importante, pour lequel les fonctions de formes ne vérifient pas les contraintes cinématiques. Les matrices M_c et K_c sont de dimension $(N-Nc)\times(N-Nc)$.
En partant des matrices assemblées M et K , de taille $N\times N$, construites sur la base de fonctions de formes u ne vérifiant pas $Cu=0$, il existe deux alternatives pour résoudre le problème (20). La première consiste à augmenter la matrice K pour faire apparaître les contraintes cinématiques, en ajoutant des inconnues sous la forme de multiplicateurs de Lagrange. C'est l'approche retenue jusqu'à présent dans le code. Le problème à résoudre s'écrit, avec la technique de double dualisation :

$$\left(\begin{bmatrix} K & C^T & C^T \\ C & -Id & Id \\ C & Id & -Id \end{bmatrix}-\omega_0^2\begin{bmatrix} M & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right)\begin{Bmatrix} \varphi \\ \lambda_1 \\ \lambda_2 \end{Bmatrix}=\begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \quad (21)$$

L'approche par ajouts de multiplicateurs de Lagrange, dans le cas de la recherche des modes propres, n'est pas satisfaisante, puisqu'en plus de la perte des propriétés de positivité de l'opérateur, on ajoute un nombre important de degrés de liberté, et on élargit donc le spectre du problème. Cet élargissement de spectre pose des problèmes numériques, qui obligent à un travail important dans ce cas.
La seconde approche, celle proposée ici, est plus naturelle, et se rapproche du problème initial. Les matrices de masse et de raideur sont certes assemblées sur la base de fonctions de formes ne vérifiant pas les conditions limites, mais la recherche des modes propres et des valeurs propres peut se faire dans un sous espace adapté. Il suffit de construire une base du sous espace des vecteurs v vérifiant $Cv=0$. Ce sous espace est naturellement le noyau de C . Il suffit alors de rechercher les modes du problème non contraint projeté dans le noyau de C . Soit T une base du noyau, on cherche alors les couples $\left(\omega_0^2,\{\psi\}\right)$ qui vérifient

$$\left[T^T\left(K-\omega_0^2M\right)T\right]\{\psi\}=\{0\} \quad (22)$$

On identifie alors $M_c=T^T M T$ et $K_c=T^T K T$.

Les efforts de réactions f_0 , calculés par $f_0=\lambda_1+\lambda_2$ dans la méthode avec double dualisation (voir relation 21), sont simplement fournis par

$$\left[C\left(K-\omega_0^2M\right)T\right]\{\psi\}=\{f_0\} \quad (23)$$

6 Bibliographie

[1] Benzi, M. and Golub, G.H., and Liesen, J. :Numerical solution of saddle point problems, Acta Numerica, Vol. 14, pp1 - 137 (2005).

[2] Gander, W. :Algorithms for the QR-Decomposition, Research report n° 80-02 (1980).

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

7 Annexe : code matlab de la construction du noyau (section 4.2)

```
function T=algo_elim(C);

C_ini=C;
T=speye(size(C,2));
CONT=zeros(size(C,2),1);
NbCont=1;
nv_const=[]; %-- handling of non verified constrainsts
boucle=1;
Tf=T;
while boucle==1
    boucle=0;
    for i1=1:size(C,1)
        C(i1,:)=C(i1,:)./norm(C(i1,:));
        if norm(C(i1,:)*T)>0  %-- contrainte non verifiee
            ind=find(C(i1,:));
            if length(ind)==1
                CONT(NbCont)=ind;
                NbCont=NbCont+1;
                T(ind,ind)=0;
            else
                lib=[];
                cont=[];
                for j1=1:length(ind)
                    if ~isempty(find(CONT==ind(j1)))
                        cont=[cont ind(j1)];
                    else
                        lib=[lib ind(j1)];
                    end ;
                end ;
                if isempty(cont)
                    [q,r]=qr(C(i1,ind)');
                    q(:,1)=0;
                    T(ind,:)=q*T(ind,:);
                    CONT(NbCont+(0:length(ind)-1))=ind;
                    NbCont=NbCont+length(ind);
                else
                    if ~isempty(lib)

                        out=(CONT(1:NbCont-1));
                        T(lib,out)=-(C(i1,lib))\(C(i1,cont)*T(cont,out));
                        [q,r]=qr(C(i1,lib)');q(:,1)=0;
                        T(lib,lib)=q;

                        lev=max(max(abs(C(i1,:)*T)));
                        if abs(lev)>1e-10
                               disp([ 'contrainte ' num2str(i1) ' mal eliminee...'
]);
                            disp([ ' max(|Ci.T|) = ' num2str(full(lev))]);
                            T(lib,lib)=speye(length(lib));
                            T(lib,out)=0;
                        else
                            CONT(NbCont+(0:length(lib)-1))=lib;
                            NbCont=NbCont+length(lib);
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
                        end ;

                    else
                        lev=max(max(abs(C(i1,:)*T)));
                         if abs(lev) < 1e-15
                            disp([ 'contrainte ' num2str(i1) ' eliminee...' ]);
                        else
                            disp([ 'contrainte ' num2str(i1) ...
                                ' non verifiee ! ' ,num2str(full(lev))]);
                            nv_const=[nv_const i1];
                            boucle=1;
                        end ;
                    end
                end ;
            end ;
        end ;

    end ;

    if boucle ==1
        disp( ' Redemarrage...' );
        %-- on ne garde que les colonnes actives
        ii=find(sum(abs(T),1));
        T=T(:,ii);

        %-- Si ca n'a pas abouti la premiere fois, on projette les contrainte
        %-- dans le nouvel espace, et on recommence...
        C=C*T;
        C=C(nv_const,:);
        Tf=Tf*T;
        T=speye(size(C,2));
        nv_const=[];
        CONT=zeros(size(C,2),1);
        NbCont=1;
    end ;
end ;

%-- derniere projection
T=Tf*T;

%-- on ne garde que les colonnes des ddl actifs
ii=find(sum(abs(T),1));
T=T(:,ii);

res=max(max(abs(C_ini*T)));
s2=sprintf( '%s %0.5g' , 'Max. residu : ' ,full(res));
disp(s2);
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)