Mise en œuvre d'un calcul de modes propres d'une structure

Résumé

Ce document présente une vue globale des différentes approches disponibles dans *Code_Aster* pour calculer les modes propres de vibration d'une structure mécanique. Ces approches sont décrites en partant de la plus simple à mettre en œuvre, pour des études standard, et en allant progressivement vers des mises en œuvre plus élaborées pour des études avancées.

On présente les enchaînements nécessaires des opérateurs de *Code_Aster*, sans entrer dans le détail de chaque opérateur.

Table des Matières

1 Rappels : formulation du problème

On considère une structure mécanique représentée, dans le cadre d'une modélisation par éléments finis, par ses matrices de raideur K, de masse M et éventuellement d'amortissement C. L'équation régissant l'évolution de la structure s'écrit $M\ddot{x}+C\dot{x}+Kx=0$.

On veut caractériser les vibrations libres de la structure mécanique, définies par des fréquences propres $f_i=\frac{\omega_i}{2\pi}$ (ω_i : pulsation propre du mode n° i) et les déformées modales x_i associées (et les amortissements modaux ζ_i si le modèle contient de l'amortissement).

En l'absence d'amortissement (cas le plus simple et le plus fréquent), le calcul modal consiste à trouver les couples $\left(\omega_i, x_i\right)$ tels que $\left(K-\omega_i^2 M\right)x_i=0$.

2 Mise en donnée du problème

La mise en donnée pour un calcul de modes propres de vibrations est classique et commune à la plupart des calculs de mécanique dans *Code_Aster* :

- lecture du maillage (opérateur LIRE_MAILLAGE),
- affectation des caractéristiques du modèle : modèle de type poutre ou 3D ou … ? (AFFE_MODELE),
- définition et affectation des matériaux (DEFI_MATERIAU et AFFE_MATERIAU) et / ou affectation des caractéristiques des éléments de structure (AFFE_CARA_ELEM),
- imposition éventuelle de conditions aux limites (étape absente si la structure est complètement libre). La particularité du calcul modal dans *Code_Aster* actuellement est qu'il faut en général que les conditions aux limites de déplacement, s'il y en a, soient imposées par dualisation (AFFE_CHAR_MECA, mot-clé facteur DDL_IMPO – le plus courant – ou FACE_IMPO ou ARETE_IMPO) plutôt que par des charges cinématiques (AFFE_CHAR_CINE).

La Figure 2-a schématise la mise en donnée d'un problème de calcul modal.

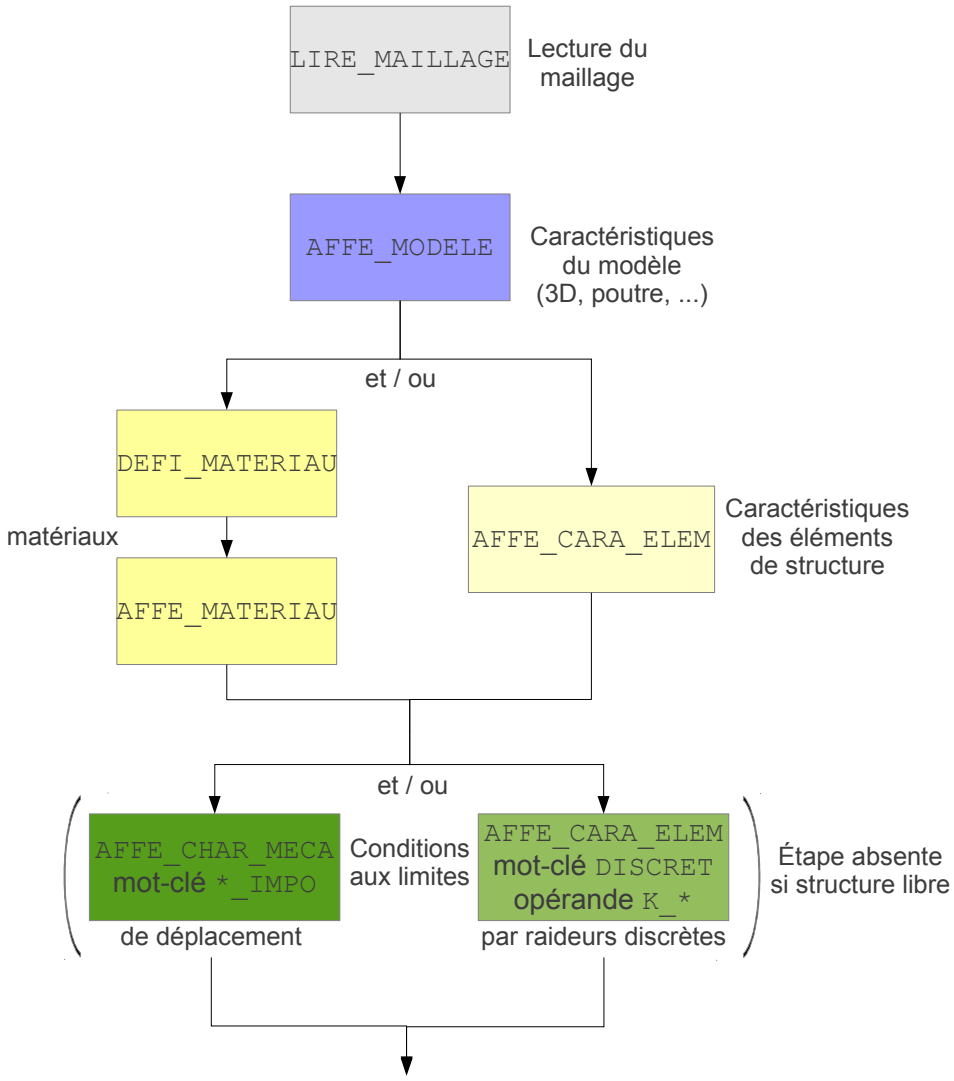

Figure 2-a : Mise en donnée d'un problème de calcul modal.

Remarques :

- *Pour un calcul modal, aucune excitation n'est nécessaire, sauf si on veut prendre en compte l'effet de raideur géométrique apporté par un chargement statique (étude avancée). Le cas échéant, le paragraphe 3.2.2.3 indique la démarche à adopter (étude avancée).*
- *Pour une étude simple, la structure n'est pas précontrainte : les éventuelles conditions aux limites en déplacement sont généralement nulles. Si on veut prendre en compte la précontrainte engendrée par des déplacements non nuls, il faut adopter là aussi la démarche indiquée au paragraphe 3.2.2.3.*

3 Calcul des modes propres de vibration d'une structure

À partir des données d'entrée vues au paragraphe précédent, on présente ici les différentes possibilités offertes par *Code_Aster* pour calculer les modes propres d'une structure, en allant de la plus simple de mise en œuvre, à des enchaînements plus compliqués.

3.1 Études les plus simples

Pour les études simples (on considère ici une structure modélisée sans amortissement, sans précontrainte, sans interaction fluide – structure, sans gyroscopie, …, et avec un nombre de degrés de liberté « raisonnable »), la solution la plus ergonomique est d'utiliser l'opérateur CALC_MODAL dont la syntaxe est très synthétique. Cet opérateur réalise le calcul des modes propres directement à partir des données d'entrée du problème mécanique, en réalisant, de manière transparente pour l'utilisateur, le calcul des matrices assemblées représentant la structure.

Pour un premier calcul, on conseille de laisser les paramètres par défaut de l'algorithme de résolution et de vérification des résultats : l'utilisateur doit seulement à renseigner sa zone de recherche des fréquences propres grâce au mot-clé facteur CALC_FREQ.

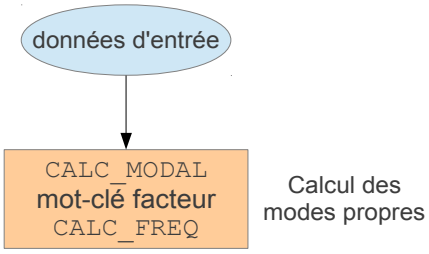

Figure 3.1-a : Calcul des modes propres par la procédure la plus simple.

Exemple :

calcul du mode propre le plus proche de $50\,Hz$:

```
modes = CALC_MODAL( MODELE = modele,
                    CHAM_MATER = ch_mat,
                    CARA_ELEM = cara_el,
                    CHARGE = c_limite,
                    CALC_FREQ = _F( OPTION = 'CENTRE',
                                    FREQ = 50.,
                                    NMAX_FREQ = 1 ));
```

Remarque :

L'opérateur CALC_MODAL permet aussi de traiter des structures avec amortissement visqueux. Il faut pour cela renseigner le mot-clé facteur AMORTISSEMENT='OUI'.

3.2 Fonctionnalités plus avancées : *via* un calcul préalable des matrices assemblées

L'opérateur CALC_MODAL est en réalité une macro-commande qui enchaîne certaines commandes élémentaires de manière prédéfinie. Son champ d'application est donc nécessairement limité à des études relativement simples.
Pour des études avancées, on aura besoin de connaître les matrices assemblées (raideur, masse, amortissement) représentant la structure. Des exemples de leur utilité sont donnés au paragraphe 3.2.2.

3.2.1 Enchaînement des commandes *Code_Aster*

À partir des données d'entrée, on procède ainsi :

- calcul des matrices assemblées (opérateur ASSEMBLAGE avec les options 'RIGI_MECA' et 'MASS_MECA' ; d'autres options existent pour calculer des matrices plus spécifiques, par exemple la rigidité géométrique, l'amortissement, la gyroscopie, etc.) ;
- calcul des modes propres. Pour cela, on dispose de deux opérateurs qui diffèrent par leurs algorithmes de résolution : MODE_ITER_SIMULT et MODE_ITER_INV.

MODE_ITER_SIMULT est à privilégier pour ses performances CPU si on cherche un nombre relativement important de modes (jusqu'à 50 à 80 ; au-delà, on recommande découper la recherche en plusieurs sous-bandes, cf paragraphe 3.2.4.1), notamment avec l'option de recherche sur un bande donnée (OPTION='BANDE') pour une meilleure robustesse.

Au contraire, MODE_ITER_INV, beaucoup plus coûteux, est à utiliser plutôt pour calculer quelques modes avec une très bonne qualité, par exemple si on veut affiner des premières estimations de modes propres (cf. paragraphe 3.2.3).

Dans un premier temps, il est conseillé de laisser les paramètres par défaut de ces opérateurs, et de préciser seulement la zone de recherche des fréquences propres.

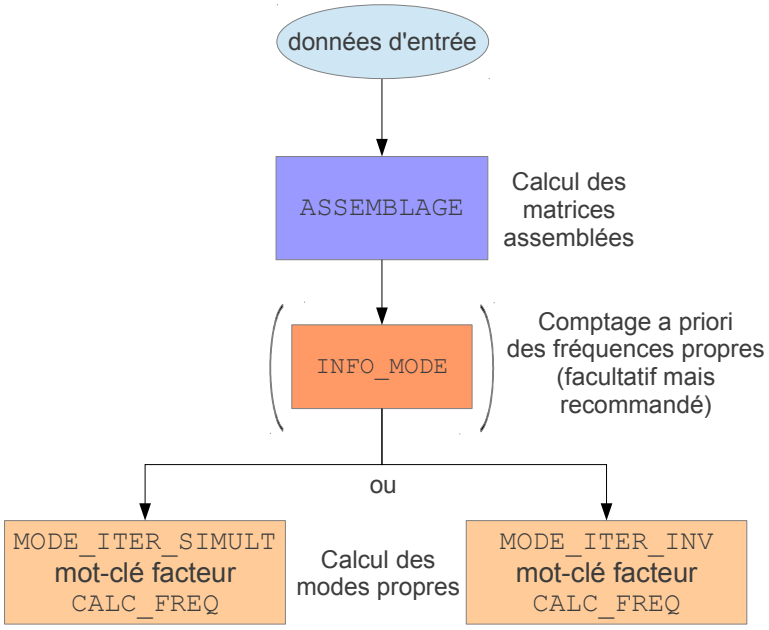

Figure 3.2.1-a : Calcul des modes en passant par les matrices assemblées.

Exemple :

calcul des modes propres sur la bande $[20\ ;300]\,Hz$:

```
ASSEMBLAGE( MODELE = modele,
            CHAM_MATER = ch_mat,
            CARA_ELEM = cara_el,
            CHARGE = c_limite,
            NUME_DDL = CO("numerota"), # création d'une numérotation des
                                       # DDL
            MATR_ASSE =(
                        _F( MATRICE= CO("matr_k"), OPTION= 'RIGI_MECA'),
                        _F( MATRICE= CO("matr_m"), OPTION= 'MASS_MECA'),
                       )
          )

modes = MODE_ITER_SIMULT( MATR_RIGI = matr_k,
                          MATR_MASS = matr_m,
                          CALC_FREQ =_F( OPTION = 'BANDE',
                                         FREQ = (20., 300.) )
                         )
```

3.2.2 Utilité d'un calcul intermédiaire des matrices assemblées : quelques exemples

3.2.2.1 Comptage préalable des fréquences propres

Avant le calcul proprement dit des modes propres, il est fortement recommandé de réaliser un comptage des modes propres contenus dans une ou des bandes de fréquences données (dans le cas standard de modes réels ; si les modes à calculer sont complexes, il s'agira d'un comptage autour d'un point du plan complexe). Ce comptage est beaucoup plus rapide que le calcul à proprement parler des modes propres.
Le comptage des modes propres est réalisé par l'opérateur INFO_MODE .
La connaissance *a priori* du nombre de fréquences propres contenues dans la bande recherche a une double utilité de vérification et d'optimisation des performances CPU du calcul modal :

- vérification : on peut vérifier que le nombre de modes propres calculées par le solveur modal est effectivement égal au nombre de modes propres compté *a priori* ;
- optimisation CPU : si le nombre de fréquences propres comptées sur la bande fréquentielle de recherche est trop élevé (un seuil compris entre 50 et 80 est couramment constaté), l'utilisateur pourra découper sa bande de recherche en plusieurs sous-bandes, grâce à l'opérateur MACRO_MODE_MECA (cf paragraphe 3.2.4.1).

Dans une première approche, l'utilisateur peut se contenter de renseigner

- dans le cas standard de modes réels : les matrices de la structure avec les mot-clés MATR_* ainsi que sa (ses) bande(s) de recherche avec le mot-clé FREQ ;
- dans le cas de modes complexes (par exemple : structures avec amortissement, …) : il faut préciser TYPE_MODE='COMPLEXE' et renseigner le disque de recherche dans le plan complexe par RAYON_CONTOUR (et éventuellement CENTRE_CONTOUR) à la place de FREQ.

3.2.2.2 Structures avec amortissement hystérétique

L'opérateur simple CALC_MODAL ne permet pas de calculer les modes propres d'une structure avec amortissement hystérétique. Il faut donc calculer explicitement la matrice assemblée de rigidité totale incluant la contribution hystérétique (matrice complexe).
L'enchaînement des opérateurs est le suivant :

- calcul des matrices assemblées de rigidité totale (rigidité classique + rigidité hystérétique) et de masse (ASSEMBLAGE avec les options 'RIGI_MECA_HYST' et 'MASS_MECA' respectivement) ;
- calcul modal avec comme entrée la matrice de rigidité totale (complexe) et la matrice de masse (MODE_ITER_SIMULT ou MODE_ITER_INV).

On se reportera à la documentation [U2.06.03] pour plus d'informations sur la prise en compte de l'amortissement hystérétique dans *Code_Aster*.

3.2.2.3 Prise en compte de pré-contraintes

La prise en compte de pré-contraintes (conditions aux limites non nulles, chargements extérieurs statiques, ...) nécessite de calculer les matrices assemblées de rigidité mécanique et géométrique. On peut alors les combiner pour former la matrice assemblée de rigidité totale qui est celle utilisée pour le calcul modal.
L'enchaînement des opérateurs est le suivant, dans un cas relativement simple de chargement extérieur statique :

- définition du chargement extérieur (opérateur AFFE_CHAR_MECA, avec par exemple le mot-clé FORCE_NODALE),
- calcul du champ de contraintes associé à ce chargement (opérateur MECA_STATIQUE ou STAT_NON_LINE ou MACRO_ELAS_MULT pour calculer la réponse statique, puis CREA_CHAMP avec le mot-clé OPERATION='EXTR' pour récupérer le champ de contraintes),
- calcul des matrices assemblées de rigidité mécanique, rigidité géométrique associée au champ de contrainte, et de masse (ASSEMBLAGE),
- combinaison des matrices de rigidité mécanique et rigidité géométrique pour former la matrice de rigidité totale (COMB_MATR_ASSE),

- calcul modal avec comme entrée la matrice de rigidité totale et la matrice de masse (MODE_ITER_SIMULT ou MODE_ITER_INV).

Exemple :

Le cas-test SDLL101 présente un exemple de calcul des modes d'une poutre soumise à des forces statiques.

3.2.2.4 Prise en compte de la gyroscopie (machines tournantes)

En plus des autres matrices assemblées (de raideur, masse et éventuellement amortissement autre que gyroscopique), il faut calculer la matrice d'amortissement gyroscopique avec l'option 'MECA_GYRO'. L'opérateur CALC_MODE_ROTATION permet alors de calculer les modes propres de la structure pour différentes vitesses de rotation définies par l'utilisateur sous le mot-clé VITE_ROTA.
On peut alors tracer le diagramme de Campbell (évolution des fréquences propres en fonction de la vitesse de rotation) de la structure tournante grâce à l'opérateur IMPR_DIAG_CAMPBELL.

Remarque :

L'opérateur CALC_MODE_ROTATION est en réalité une macro-commande apppelant MODE_ITER_SIMULT : au besoin, l'utilisateur peut donc réaliser les différentes étapes élémentaires de CALC_MODE_ROTATION « à la main » mais de manière beaucoup moins ergonomique. Le cas-test SDLL129 illustre la démarche dans le cas d'un rotor avec paliers dont les caractéristiques dépendent de la vitesse de rotation.

3.2.2.5 Utilisation des modes pour un calcul dynamique sur base modale

Il faut là aussi avoir accès aux matrices assemblées : leur projection sur une base modale fournit les matrices généralisées utilisables pour un calcul dynamique, avec des performances CPU bien meilleures que l'utilisation directe des matrices assemblées. Cette méthode de réduction de modèle est décrite dans les documentations de référence [R5.06.01] et d'utilisation [U2.06.04] .

3.2.3 Améliorer la qualité des modes propres

On attire l'attention sur le fait que **la qualité d'un calcul modal dépend avant tout de la qualité des données d'entrée et de la modélisation physique**. On peut notamment citer :

- le choix des conditions aux limites : sont-elles représentative de la réalité ? Leur influence est forte sur le résultat du calcul ;
- la finesse du maillage : une étude de convergence du maillage est nécessaire, comme pour toute étude numérique ;
- le choix de la modélisation : par éléments de structure (poutre, coque, …) ou en 3D ? Par exemple, pour une structure élancée, une modélisation en poutre sera généralement meilleure qu'une modélisation 3D même avec une bonne finesse de maillage.

Si les données d'entrée et la modélisation sont figées, il est possible d'améliorer la qualité « informatique » du résultat.
Pour cela, un premier calcul par la méthode de sous-espace (opérateur MODE_ITER_SIMULT) donne une première estimation des modes propres (fréquences propres, déformées modales, …) d'une structure. Cette première estimation est généralement déjà bonne et satisfaisante. Pour des modèles plus compliqués, il est toutefois conseillé d'affiner cette estimation par un second calcul par la méthode des puissances inverses (opérateur MODE_ITER_INV).
L'enchaînement des commandes est alors le suivant :

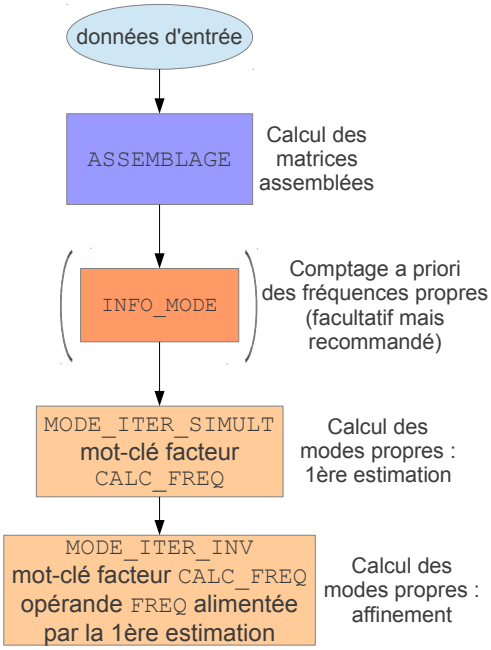

Figure 3.2.3-a : Amélioration de la qualité des modes propres.

Exemple :

Le premier calcul de modes sur la bande [0 ; 2000] Hz avec la commande ci-dessous :

```
mode1 = MODE_ITER_SIMULT( MATR_RIGI = k_asse,
                          MATR_MASS = m_asse,
                          CALC_FREQ =_F(
                                        OPTION = 'BANDE',
                                        FREQ= (0., 2000.),
                                        ),
                        )
```

donne les résultats suivants, visibles dans le fichier MESSAGE :

```
-------------------------------------------------------------------------

   LES FREQUENCES CALCULEES INF. ET SUP. SONT:
      FREQ_INF :  4.65661E+01
      FREQ_SUP :  1.60171E+03


-------------------------------------------------------------------------
       CALCUL MODAL:  METHODE D'ITERATION SIMULTANEE
                      METHODE DE SORENSEN

    NUMERO     FREQUENCE (HZ)     NORME D'ERREUR
         1       4.65661E+01        1.82405E-07
         2       2.91827E+02        3.47786E-09
         3       8.17182E+02        9.83625E-11
         4       1.60171E+03        4.31692E-11
 NORME D'ERREUR MOYENNE:  0.46506E-07


-------------------------------------------------------------------------

          VERIFICATION A POSTERIORI DES MODES

   DANS L'INTERVALLE ( 4.64496E+01, 1.60571E+03)
```

```
   IL Y A BIEN    4 FREQUENCE(S)
-----------------------------------------------------------------------
```

On peut alors affiner par exemple les deux premiers modes propres, en lançant le second calcul à partir des fréquences propres précédemment calculées :

```
mode2 = MODE_ITER_INV( MATR_RIGI = k_asse,
                       MATR_MASS = m_asse,
                       CALC_FREQ =_F( OPTION = 'PROCHE',
                                      FREQ = ( 46.6, 291.8 ),
                                    ),
                     )
```

ce qui donne

```
-----------------------------------------------------------------------
    CALCUL MODAL:  METHODE D'ITERATION INVERSE
                                                          INVERSE
NUMERO  FREQUENCE (HZ) AMORTISSEMENT  NB_ITER  PRECISION   NORME D'ERREUR
    1    4.65661E+01    0.00000E+00       3    3.33067E-16     3.99228E-08
    2    2.91827E+02    0.00000E+00       3    2.22045E-16     1.23003E-09
```

On observe que la norme d'erreur est légèrement améliorée (certes faiblement mais il s'agit ici d'un cas très simple).

Remarque :

On peut aussi automatiser la récupération des fréquences propres issues de la première estimation pour alimenter le second calcul, grâce au langage Python :

```
# récupération de la la liste des fréquences propres estimées dans la
variable Python f_estimation :
f_estimation = MODE1.LIST_VARI_ACCES()['FREQ']

mode2 = MODE_ITER_INV( MATR_RIGI = k_asse,
                       MATR_MASS = m_asse,
                       CALC_FREQ =_F( OPTION = 'PROCHE',
                                      FREQ = f_estimation,
                                    ),
                     )
```

3.2.4 Optimisation des performances CPU

3.2.4.1 Découpage de la bande fréquentielle de recherche

Si on recherche beaucoup de modes propres (soit car la bande de recherche est très large, soit car la densité modale est forte), les performances du calcul modal seront meilleures en découpant la bande de recherche globale $[f_{min};f_{max}]$ en plusieurs (n) sous-bandes : $[f_{min};f_2]$, $[f_2;f_3]$, ..., $[f_n;f_{max}]$. Cela est fait grâce à l'opérateur MACRO_MODE_MECA en précisant le découpage fréquentiel avec le mot-clé FREQ=(fmin,f2,...,fn,fmax). Pour définir les sous-bandes, l'utilisateur peut s'appuyer sur le comptage *a priori* des fréquences propres fourni par l'opérateur INFO_MODE qui peut lui aussi fonctionner par sous-bandes.

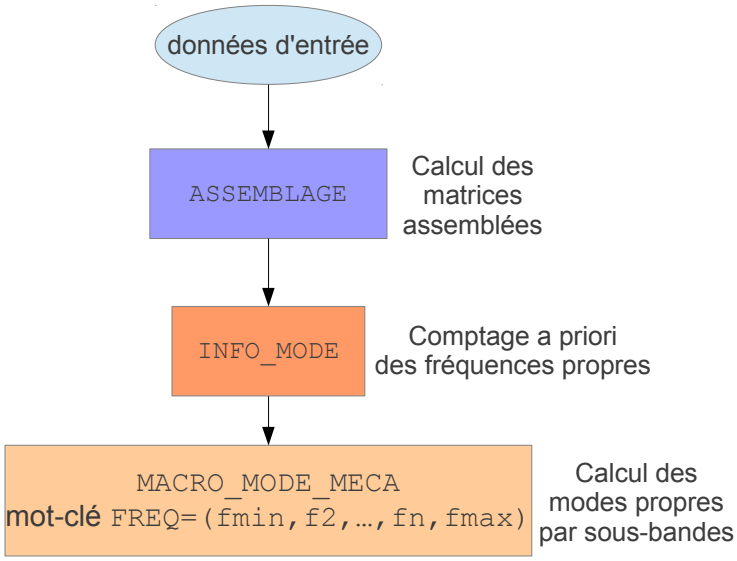

Figure 3.2.4.1-a : Calcul des modes propres par découpage en sous-bandes.

Exemple :

identique au paragraphe 3.2.1 en découpant la bande $[20\ ;300]Hz$ en trois sous-bandes :

```
ASSEMBLAGE( MODELE = modele,
            CHAM_MATER = ch_mat,
            CARA_ELEM = cara_el,
            CHARGE = c_limite,
            NUME_DDL = CO("numerota"), # création d'une numérotation des
                                       # DDL
            MATR_ASSE =(
                        _F( MATRICE= CO("matr_k"), OPTION= 'RIGI_MECA'),
                        _F( MATRICE= CO("matr_m"), OPTION= 'MASS_MECA'),
                       )
          );

nb_modes = INFO_MODE( MATR_RIGI = matr_k,
                      MATR_MASS = matr_m,
                      FREQ = (20.,300.),
                    );

modes = MACRO_MODE_MECA( MATR_RIGI = matr_k,
                         MATR_MASS = matr_m,
                         CALC_FREQ =_F( FREQ = (20., 100., 200., 300.) )
                        );
```

Remarques :

- *Il y a un gain en performance CPU même lorsque les sous-bandes sont traitées séquentiellement (ce qui est le cas par défaut). La mise en œuvre du parallélisme (cf paragraphe suivant) permet d'améliorer encore plus les performances.*
- *Pour des performances optimales, il est conseillé d'avoir des sous-bandes les plus équilibrées possibles (soit avec un nombre de modes recherchés par sous-bande relativement uniforme).*

3.2.4.2 Parallélisme

Pour le calcul modal, le parallélisme peut être mis en œuvre à deux niveaux :

- parallélisation des calculs modaux menés sur chaque sous-bande, dans les opérateurs INFO_MODE et MACRO_MODE_MECA ;

- parallélisme au niveau du solveur linéaire MUMPS, dans les opérateurs INFO_MODE, MODE_ITER_SIMULT, MODE_ITER_INV et MACRO_MODE_MECA.

Pour mettre en œuvre le parallélisme, il faut :

- disposer d'une version de *Code_Aster* construite avec un compilateur parallèle (par exemple : OpenMPI, …). Sur le serveur centralisé Aster4, des versions parallèles existent déjà : STAxx_impi ;
- sélectionner dans ASTK une version parallèle de *Code_Aster* ;

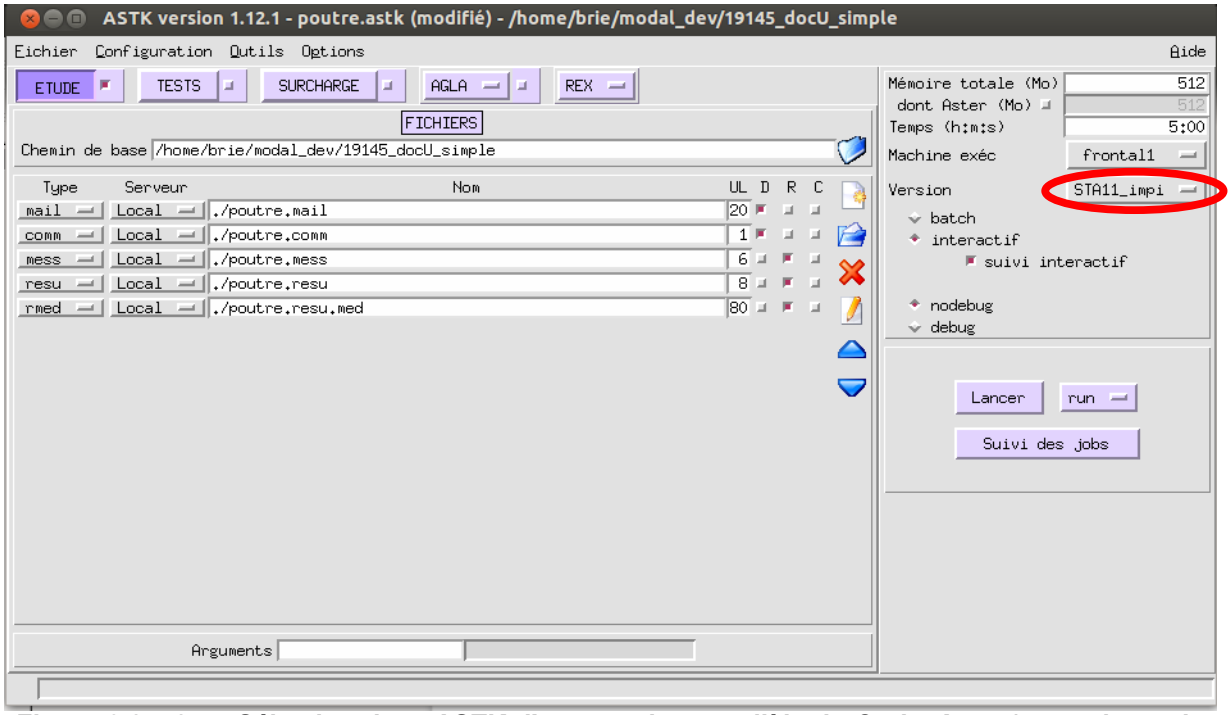

Figure 3.2.4.2-a : Sélection dans ASTK d'une version parallèle de *Code_Aster* (exemple sur le serveur centralisé Aster4).

- spécifier dans ASTK le nombre de processeurs et de nœuds de calcul à exploiter ; il faut utiliser au moins autant de processeurs que de sous-bandes fréquentielles non vides, et on conseille d'utiliser un nombre de processeurs multiple du nombre de sous-bandes non vides (par exemple : s'il y a 5 sous-bandes non vides, utiliser 10 ou 15 ou … processeurs) ;

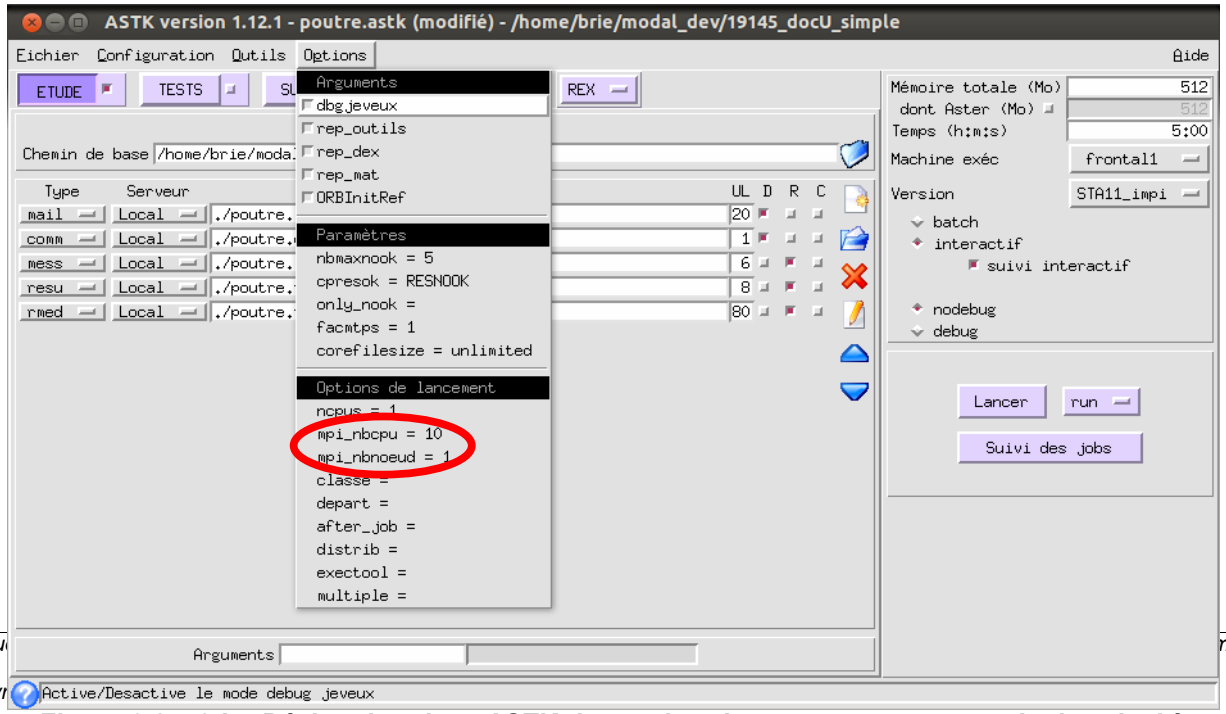

Figure 3.2.4.2-b : Déclaration dans ASTK du nombre de processeurs et nœuds de calcul à exploiter.

•Dans le fichier de commande *Code_Aster* :
- pour utiliser le mode basique avec INFO_MODE ou MACRO_MODE_MECA (parallélisation des sous-bandes uniquement), il n'y a rien à faire : les mot-clés par défaut activent la parallélisation des sous-bandes ;
- pour utiliser le mode avancé (parallélisation des sous-bandes pour INFO_MODE et MACRO_MODE_MECA, et du solveur linéaire pour tous les opérateurs modaux) : utiliser le solveur linéaire MUMPS (mot-clé facteur SOLVEUR, opérande METHODE='MUMPS' ; on recommande également de paramétrer les opérandes RENUM='QAMD' et GESTION_MEMOIRE='IN_CORE').

Exemple :

identique au paragraphe 3.2.4.1 en parallélisant à la fois les calculs sur les sous-bandes et le solveur linéaire :

```
modes = MACRO_MODE_MECA( MATR_RIGI = matr_k,
                         MATR_MASS = matr_m,
                         CALC_FREQ = F( FREQ = (20., 100., 200., 300.) ),
                         NIVEAU_PARALLELISME = 'COMPLET',
                         SOLVEUR = F( METHODE = 'MUMPS',
                                      RENUM = 'QAMD',
                                      GESTION_MEMOIRE = 'IN_CORE' ),
                       );
```

Remarque :

Pour des performances optimales, il est conseillé d'avoir des sous-bandes les plus équilibrées possibles (i.e. : avec un nombre de modes recherchés par sous-bande relativement uniforme).

La mise en œuvre du parallélisme est présentée de manière plus détaillée dans la documentation générique [U2.08.06] et les documentations d'utilisation de INFO_MODE [U4.52.01] et de MACRO_MODE_MECA [U4.52.02].

3.2.4.3 Réduction de modèle : calcul par sous-structuration

Lorsque le modèle numérique comporte un nombre élevé de degrés de liberté ou que la structure étudiée est un assemblage de composants maillés séparément, on peut utiliser des méthodes de réduction de modèle par sous-structuration, qui reposent sur un partitionnement géométrique de la structure globale. Sur des grands modèles, ces méthodes présentent de meilleures performances CPU qu'un calcul direct.

•Sous-structuration dynamique

Cette méthode a un champ d'application très général. La documentation [U2.07.05] détaille sa mise en œuvre.

•Sous-structuration cyclique

Cette méthode a un champ d'application beaucoup plus restrictif que la précédente : elle permet de traiter uniquement des structures à répétitivité cyclique (par exemple : roue aubagée, …). Le cas-test SDLV301 donne un exemple de mise en œuvre.

4 Paramètres contenus dans un résultat de calcul modal

L'exécution de l'un des opérateurs de calcul modal s'accompagne de l'impression automatique de certains paramètres dans le fichier RESULTAT :

```
--------------------------------------------------------------------
LE NOMBRE DE DDL
```

```
   TOTAL EST:                       234

   DE LAGRANGE EST:                 120

LE NOMBRE DE DDL ACTIFS EST:         54
--------------------------------------------------------------------------------
L'OPTION CHOISIE EST: CENTRE

LA VALEUR DE DECALAGE EN FREQUENCE EST :  5.00000E+01
--------------------------------------------------------------------------------

 INFORMATIONS SUR LE CALCUL DEMANDE:
 NOMBRE DE MODES RECHERCHES     :                      1

 LA DIMENSION DE L'ESPACE REDUIT EST :                        0
 ELLE EST INFERIEURE AU NOMBRE DE MODES, ON LA PREND EGALE A
                    4



     ============================================
     =       METHODE DE SORENSEN (CODE ARPACK)  =
     =       VERSION :  2.4                     =
     =          DATE :  07/31/96                =
     ============================================
     NOMBRE DE REDEMARRAGES                     =     2
     NOMBRE DE PRODUITS OP*X                    =     7
     NOMBRE DE PRODUITS B*X                     =    20
     NOMBRE DE REORTHOGONALISATIONS  (ETAPE 1)  =     6
     NOMBRE DE REORTHOGONALISATIONS  (ETAPE 2)  =     0
     NOMBRE DE REDEMARRAGES DU A UN V0 NUL      =     0

--------------------------------------------------------------------------------

   LES FREQUENCES CALCULEES INF. ET SUP. SONT:
      FREQ_INF :  5.22037E+01
      FREQ_SUP :  5.22037E+01


--------------------------------------------------------------------------------
       CALCUL MODAL:  METHODE D'ITERATION SIMULTANEE
                      METHODE DE SORENSEN
                                          fréquences propres
    NUMERO     FREQUENCE (HZ)     NORME D'ERREUR
         2       5.22037E+01       7.02498E-10
         3       6.74211E+01       9.12843E-10
 NORME D'ERREUR MOYENNE:  0.80767E-09
  position du mode dans le spectre global

--------------------------------------------------------------------------------

          VERIFICATION A POSTERIORI DES MODES

   DANS L'INTERVALLE ( 5.20730E+01, 5.23340E+01)
   IL Y A BIEN    1 FREQUENCE(S)
--------------------------------------------------------------------------------
```

Si les modes calculés sont complexes, il y a en plus une colonne donnant les amortissements modaux :

```
 ---------------------------------------------------------------------
 LE NOMBRE DE DDL

    TOTAL EST:                          74

    DE LAGRANGE EST:                    44

 LE NOMBRE DE DDL ACTIFS EST:            8
 ---------------------------------------------------------------------
  INFORMATIONS SUR LE CALCUL DEMANDE:
  NOMBRE DE MODES RECHERCHES     :                      5


   le problème traité étant quadratique, on double l'espace de recherche

       Méthode QZ dans MODE_ITER_SIMULT: On trouve un nombre de valeurs
 propres
     17 différent du nombre de ddls physiques actifs 8 !

  votre problème est fortement amorti.
  valeur(s) propre(s) réelle(s)                  :  14
  valeur(s) propre(s) complexe(s) avec conjuguée :  10
  valeur(s) propre(s) complexe(s) sans conjuguée :  0
        CALCUL MODAL:  METHODE GLOBALE DE TYPE QR
                       ALGORITHME QZ_SIMPLE
```

fréquences propres (amorties)

NUMERO	FREQUENCE (HZ)	AMORTISSEMENT	NORME D'ERREUR
1	5.52718E+00	8.68241E-03	4.00918E-13
2	1.08852E+01	1.71010E-02	7.31808E-14
3	1.59105E+01	2.50000E-02	5.40182E-14
4	2.04500E+01	3.21394E-02	4.03817E-14
5	2.43661E+01	3.83022E-02	3.48265E-14

```
 NORME D'ERREUR MOYENNE:  0.12067E-12
```

amortissements modaux

```
  Attention : pour l'instant, il n'y a pas de vérification de type STURM
 (comptage du bon nombre des valeurs propres calculées)
  lorsqu'on est dans le plan complexe :
        problème modal généralisé avec MATR_RIGI complexe,
     ou problème modal généralisé avec matrice(s) non symétrique(s),
     ou problème modal quadratique (préence du mot-clé MATR_AMOR).

 ---------------------------------------------------------------------

           VERIFICATION A POSTERIORI DES MODES

 ---------------------------------------------------------------------
```

En outre, la structure de données informatique produite lors d'un calcul modal peut contenir les paramètres suivants :

Intitulé du paramètre dans *Code_Aster*	Définition
FREQ	Fréquence propre (amortie, le cas échéant)
AMOR_GENE	Amortissement modal généralisé
AMOR_REDUIT	Amortissement modal réduit
FACT_PARTICI_D* (* = X ou Y ou Z)	Facteur de participation du mode dans la direction D*
MASS_EFFE_D* (* = X ou Y ou Z)	Masse modale effective dans la direction D*
MASS_EFFE_UN_D* (* = X ou Y ou Z)	Masse modale effective unitaire dans la direction D*
MASS_GENE	Masse généralisée du mode
OMEGA2	Pulsation propre (amortie, le cas échéant) au carré
RIGI_GENE	Raideur généralisée du mode

Tableau 4.1 : liste des paramètres modaux.

Ces paramètres sont définis mathématiquement dans la documentation de référence [R5.01.03]. L'utilisateur y a accès en imprimant le contenu de la structure de donnée avec l'opérateur IMPR_RESU au FORMAT='RESULTAT' avec l'option TOUT_PARA='OUI'.

5 Post-traitements des modes propres

5.1 Visualisation

Les déformées modales calculées par l'une des méthodes décrites précédemment peuvent être exportées dans différents formats afin d'être visualisées dans des plates-formes de calcul mécanique : format MED pour la plate-forme Salomé, format UNV, …

L'utilisateur peut ainsi caractériser graphiquement les modes calculés : mode de flexion ? mode dans un plan donné ? mode local ? etc.

Exemple :

impression au format MED.

```
modes = ...          # calcul par l'une des méthodes précédemment décrites

IMPR_RESU( FORMAT = 'MED',
           RESU=_F( RESULTAT = modes ) );
```

On peut alors ouvrir le fichier créé dans la plateforme Salome pour visualiser la déformée modale, l'animer, …

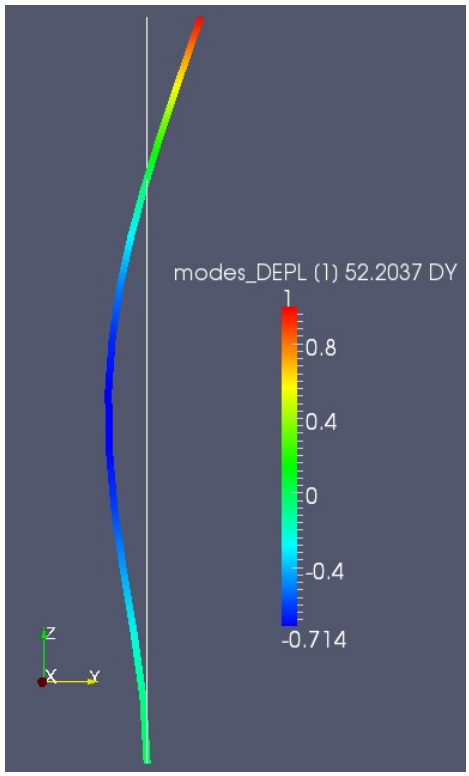

Figure 5.1-a : Visualisation d'un mode dans Salomé (module ParaVis) : ici le mode de flexion d'ordre 2 d'une poutre.

5.2 Normalisation des modes

Les déformées modales sont définies à un facteur multiplicatif près (cf formulation du problème modal au paragraphe 1).

Par défaut, les modes calculés par les opérateurs CALC_MODAL, MODE_ITER_SIMULT et MODE_ITER_INV sont normés de manière à ce que la plus grande composante physique soit égale à 1. L'utilisateur peut modifier cette normalisation grâce à l'opérateur NORM_MODE [U4.52.11], qui calcule également ou met à jour les paramètres modaux suivants, qui dépendent de la normalisation choisie : FACT_PARTICI_D*, MASS_GENE et RIGI_GENE. Il enrichit également la structure de donnée avec les paramètres MASS_EFFE_UN_D* (qui sont eux indépendants de la normalisation). Ces paramètres (définis au paragraphe 4) peuvent être utiles notamment pour éliminer d'une base modale certains modes non désirés (cf paragraphe 5.3).

Exemple :

norme par rapport à la masse.

```
modes = ...          # calcul par l'une des méthodes précédemment décrites

modes = NORM_MODE( reuse = modes,
                   MODE = modes,
                   NORME = 'MASS_GENE' );
```

Remarque :

si l'utilisateur calcule les modes avec l'opérateur MACRO_MODE_MECA , il peut choisir la norme directement à l'intérieur de cet opérateur, avec le mot-clé facteur NORM_MODE .

5.3 Filtrage des modes selon un critère

Dans la perspective d'un calcul de réponse transitoire par exemple, l'utilisateur peut choisir de conserver dans sa base modale de projection, seulement certains modes jugés importants dans la réponse dynamique ou remplissant un critère donné. Cela est fait grâce à l'opérateur EXTR_MODE [U4.52.12] qui permet de filtrer les modes selon différentes options : à partir de leur numéro dans le spectre global, de leur masse généralisée, etc.

Exemple :

élimination des modes dont la masse effective unitaire dans la direction DX est inférieure à 5 %, et affichage dans le fichier RESULTAT du cumul des masses effectives unitaires des modes conservés.

```
modes = ...         # calcul par l'une des méthodes précédemment décrites

modes_f = EXTR_MODE( FILTRE_MODE =_F( MODE = modes,
                                      CRIT_EXTR = 'MASS_EFFE_UN',
                                      SEUIL_X = 0.05 ),
                     IMPRESSION =_F( CRIT_EXTR = 'MASS_EFFE_UN',
                                     CUMUL = 'OUI'),
                     );
```

Remarque :

si l'utilisateur calcule les modes avec l'opérateur MACRO_MODE_MECA , il peut réaliser ce filtre directement à l'intérieur de cet opérateur, avec le mot-clé facteur FILTRE_MODE .