Macro-commande MACRO_MODE_MECA

1 But

Cette macro-commande permet de lancer une **succession de calculs de modes propres réels sur un ensemble de sous-bandes fréquentielles contigües**.
Les actions suivantes sont réalisées: obtention des modes par itérations simultanées dans les sous-bandes spécifiées, application d'une norme, filtrage selon un critère de valeur de paramètre modal supérieure à un certain seuil, et enfin, concaténation des structures de données calculées en une seule.

En terme d'opérateur *Code_Aster*, cela se traduit par une étape de pré-estimation (*via* INFO_MODE [U4.52.01]) du nombre de fréquences présentes dans chacune des sous-bandes. On va ensuite calculer de manière effective ces modes par la commande MODE_ITER_SIMULT [U4.52.03].
En post-traitement, on contrôle par défaut le bon nombre de fréquences calculées dans la bande globale *via* un dernier INFO_MODE. Puis les modes associés sont normés (par NORM_MODE [U4.52.11]), filtrés et concaténés (par EXTR_MODE[U4.52.12]).

Par rapport à un simple appel à MODE_ITER_SIMULT , **cette macro permet d'optimiser les coûts calcul** et la **précision des résultats** en traitant des intervalles de taille plus réduite.
En mode parallèle, on profite pleinement de cette décomposition en calculs modaux quasi-indépendants (cf. §3.7). Par exemple, avec un calcul découpé en 10 sous-bandes équilibrées via une calibration préalable avec INFO_MODE, on peut **accélérer le temps de calcul d'un facteur 10 à 20 en utilisant une quarantaine de processeurs**. Le pic mémoire peut aussi baisser de quelques dizaines de pourcents.

Idéalement les intervalles ne devraient comporter que quelques dizaines de modes et être équilibrés. Le chiffre optimal dépend du solveur modal, de son paramétrage et de l'étude. Souvent une quarantaine de modes par sous-bande est un bon chiffre en séquentiel. On peut réduire leur taille en parallèle suivant le nombre de processeurs disponibles.

Cet opérateur produit un concept mode_meca .
Dans une première approche on peut se contenter de renseigner les paramètres: MATR_* et FREQ.

Table des Matières

2 Syntaxe

```
mod_meca =   MACRO_MODE_MECA  (

# Caractéristiques du calcul (partie 1/2)
      ♦  MATR_RIGI =  matra                              [matr_asse_DEPL_R]
      ♦  MATR_MASS =  matra                              [matr_asse_DEPL_R]
```

Paramétrage des opérateurs MODE_ITER_SIMULT et INFO_MODE

```
# Prétraitement des modes rigides (seulement si METHODE='TRI_DIAG' )
      ◊  OPTION =     /  'MODE_RIGIDE'                   [DEFAUT]
                      /  'SANS'

# Choix de la méthode
      ◊  METHODE =    /  'TRI_DIAG'
                      /  'JACOBI'
                      /  'SORENSEN'                      [DEFAUT]

# Caractéristiques du calcul (partie 2/2)
      ♦  CALC_FREQ = _F  (
# Le nombre de fréquences délimitant les intervalles de calcul est noté nb_freq.
                      ♦  FREQ      =  l_f                [l_R]

# Paramétrage commun des méthodes

   # Caractéristiques de l'espace de projection
      ◊  DIM_SOUS_ESPACE=dse                             [I]
      ◊  COEF_DIM_ESPACE=mse                             [I]
         EXCLUS('DIM_SOUS_ESPACE','COEF_DIM_ESPACE')

   # Pour pré et post-traitements
      ◊  PREC_SHIFT=/0.05                                [DEFAUT]
                   /p_shift                              [R]
      ◊  NMAX_ITER_SHIFT=/3                              [DEFAUT]
                         /n_shift                        [I]
      ◊  SEUIL_FREQ=/1.E-2                               [DEFAUT]
                   /f_seuil                              [R]

   # Comportement en cas de bande vide
   ◊  STOP_BANDE_VIDE=/'NON'                             [DEFAUT]
                      /'OUI'

# Paramétrage interne des méthodes

   # Si METHODE='SORENSEN'
                      ◊  PREC_SOREN=/0                   [DEFAUT]
                                   /pso                  [R]
                      ◊  NMAX_ITER_SOREN= /20            [DEFAUT]
                                         /nso            [I]
                      ◊  PARA_ORTHO_SOREN=/0.717         [DEFAUT]
                                         /porso          [I]
   # Si METHODE='TRI_DIAG'
                      ◊  PREC_ORTHO=/1.E-12              [DEFAUT]
                                   /po                   [R]
```

```
                        ◊ NMAX_ITER_ORTHO=/5                    [DEFAUT]
                                         /nio                   [I]
                        ◊ PREC_LANCZOS=/1.E-8                   [DEFAUT]
                                       /pl                      [R]
                        ◊ NMAX_ITER_QR=/30                      [DEFAUT]
                                       /nim                     [I]
# Si METHODE='JACOBI'
                        ◊ PREC_BATHE=/1.E-10                    [DEFAUT]
                                     /pbat                      [R]
                        ◊ NMAX_ITER_BATHE=/40                   [DEFAUT]
                                         /nbat                  [I]
                        ◊ PREC_JACOBI=/1.E-2                    [DEFAUT]
                                      /pjaco                    [R]
                        ◊ NMAX_ITER_JACOBI=/12                  [DEFAUT]
                                         /njaco                 [I]
                        )
```

 # Pour vérifications finales

```
        ◊ VERI_MODE = _F   (
                        ◊ STOP_ERREUR =    /   'OUI'            [DEFAUT]
                                           /   'NON'
                        ◊ SEUIL =      /   rseuil               [R]
                                       /   1.E-6                [DEFAUT]
                        ◊ STURM =      /   'OUI'                [DEFAUT]
                                       /   'NON'                [l_Kn]
                        ◊ PREC_SHIFT = /   pshif                [R]
                                       /   0.005                [DEFAUT]
                        )
```

Paramétrage de l'opérateur NORM_MODE

```
        ♦ NORM_MODE = _F   (
                     ◊ /   NORME =    /   'EUCL_TRAN'
                                      /   'MASS_GENE'
                                      /   'RIGI_GENE'
                                      /   'TRAN'
                                      /   'TRAN_ROTA'           [DEFAUT]
                                      /   'EUCL'
                        ◊   INFO =    /   1                     [DEFAUT]
                                      /   2
                            )
```

Paramétrage de l'opérateur EXTR_MODE

```
        ◊   FILTRE_MODE = _F  (
                        ◊   CRIT_EXTR =  /   'MASS_EFFE_UN'     [DEFAUT]
                                         /   'MASS_GENE'
                        ◊   SEUIL =      /   0.001              [DEFAUT]
                                         /   rseuil             [R]
                            )
```

Paramétrage d'impression

```
        ◊   IMPRESSION = _F  (
                        ◊   CUMUL =   /   'OUI'                 [DEFAUT]
```

```
                    /   'NON'
  ◊  CRIT_EXTR =    /   'MASS_EFFE_UN'          [DEFAUT]
                    /   'MASS_GENE'
  ◊  TOUT_PARA =    /   'OUI'                   [DEFAUT]
                    /   'NON'
      )
```

 # Solveur linéaire et parallélisme

◊ SOLVEUR=_F (Pour plus de détails voir le document [U4.50.01]).

En parallèle, on conseille particulièrement le paramétrage METHODE='MUMPS'+RENUM='QAMD' .

```
  ◊  NIVEAU_PARALLELISME =          /   'COMPLET'       [DEFAUT]
                                     /   'PARTIEL'
```

Activé uniquement en mode parallèle (nb_proc>1).

L'option 'COMPLET' fonctionne quelque soit le solveur linéaire direct si nb_proc=nb_freq-1. Avec l'option 'PARTIEL', seul SOLVEUR=_F(METHODE='MUMPS') est licite.

 # Divers

```
  ◊  INFO =         /   1                       [DEFAUT]
                    /   2
                    );
```

3 Opérandes

3.1 Opérandes MATR_RIGI/MATR_MASS/INFO/METHODE/OPTION

Ils ont la même signification que dans la commande MODE_ITER_SIMULT[U4.52.03].

3.2 Mot clé CALC_FREQ

Joue le même rôle que dans la commande MODE_ITER_SIMULT[U4.52.03], a les mêmes mots-clés internes avec les mêmes valeurs par défaut, à l'exception des quelques mots clés suivants.

3.2.1 Opérande FREQ

♦ FREQ = l_f

Liste de fréquences (en Hertz) définissant les sous-bandes que l'on veut étudier $l\_f=(f_i)_i$ (on note nb_freq le nombre de fréquences de cette liste). On recherche alors les modes dans les sous-bandes $[\lambda_i, \lambda_{i+1}]$ avec $\lambda_*=(2\pi f_*)^2$ et $i=1..nb\_f$.
Cette liste doit comporter au moins deux valeurs. Ces valeurs doivent être rangées par ordre strictement croissant et toutes positives.

Remarques:

- *Chaque fréquence n'est traitée qu'une seule fois: en tant que borne inférieure de la première sous-bande pour la première de la liste, en tant que borne supérieure des sous-bandes qui suivent pour les autres fréquences. En particulier, si cette fréquence est jugée trop proche d'une valeur propre, on la décale (cf. [U4.52.01] et [R5.01.04]).*
- *Le décalage éventuel d'une borne de fréquence ne s'opère plus qu'une seule fois dans l' INFO_MODE initial. Il n'y a donc plus de risque de chevauchement d'intervalles décalés comme jusqu'en v10. On ne risque donc plus de calculer par erreur deux fois le même mode.*

◊ STOP_BANDE_VIDE=/**'NON'** **[DEFAUT]**
/'OUI'

Permet d'indiquer à chaque occurrence de MODE_ITER_SIMULT s'il elle doit s'arrêter ('OUI') ou continuer ('NON') dans le cas où la sous-bande $[\lambda_i, \lambda_{i+1}]$ ne comporterait pas de fréquence. De même, ce paramètre décide du comportement global de la macro en cas de bande globale vide.
Notez que, de part le fonctionnement particulier de la macro, la valeur par défaut de ce paramètre est opposée à celle retenue pour MODE_ITER_SIMULT.

3.3 Mot clé VERI_MODE

Les opérandes internes ont la même signification que dans le mot clé de même nom dans la commande MODE_ITER_SIMULT. Sauf en ce qui concerne le test de Sturm, pour lequel le fonctionnement particulier de la macro impose d'autres valeurs.

◊ STURM= /**'GLOBAL'** **[DEFAUT]**
/'LOCAL'
/'NON'

Vérification dite de STURM permettant de s'assurer que l'algorithme utilisé dans l'opérateur a déterminé le nombre exact de valeurs propres, sous-bande par sous-bande ('LOCAL') ou

uniquement dans la bande globale[1] ('GLOBAL') (cf. [U4.52.01][R5.01.04]). La deuxième variante est la plupart du temps amplement suffisante et beaucoup moins coûteuse que la première.
Cependant, lorsque les bornes fournies au test de Sturm sont proches d'une valeur propre, il faut les décaler (pour préserver la robustesse du processus). Parfois ce décalage est trop prononcé et il va donc conduire le test de Sturm à englober un intervalle trop grand comportant des fréquences non calculées (et non souhaitées !).
Le test va alors alerter l'utilisateur parfois inutilement. Après s'être assuré qu'il ne s'agissait pas de fréquences multiples ratées proches des bornes de la bande, on peut alors le débrancher ('NON') ou réduire les paramètres de décalage (passer de PREC_SHIFT=5% à 2% par exemple).

Par exemple, on teste l'intervalle $[100,500]$ et 499.5 et 520 sont des valeurs propres du problème. Du fait de la proximité de la valeur propre 499.5 de la borne maximum 500, le test de Sturm va devoir décaler cette dernière. Par défaut elle va prendre la valeur 525. Cette nouvelle bande de test $[100,525]$ est maintenant trop importante car elle englobe la valeur 520: le test va conclure, faussement, qu'il y' a un problème en comptant une fréquence en trop !
A contrario, si 500.1 avait été une valeur propre, le test de Sturm aurait sans doute bien fait d'alerter l'utilisateur !

Remarque:

- *En mode parallèle standard (NIVEAU_PARALLELISME='COMPLET') , il n'y a pas de possibilité de test de Sturm local. STURM='GLOBAL' ou ' LOCAL' effectuent le même traitement: ils vérifient la validité du test de Sturm sur l'ensemble des sous-bandes de calcul.*
- *Ce test de post-vérification s'effectue en plus d'autres tests (non débrayables et indispensables):*
  Tests de convergence internes [2] au solveur modal (' SORENSEN ', ' TRI_DIAG ' et ' JACOBI') modulable via les mots-clés PREC_* .
  Vérification des résidus (cf. [R5.01.01/02] algorithme n°2/n°1) de chaque mode calculé (cf. mot-clés SEUIL_FREQ et SEUIL).
  On s'assure enfin que les fréquences exhumées pour chaque sous-bande appartiennent bien à l'intervalle choisi (à VERI_MODE/PREC_SHIFT % près).

3.4 Mot clé NORM_MODE

Sert à définir les arguments pour la normalisation des modes. Tous les modes sont normés de la même façon. Les arguments sont les mêmes que pour la commande NORM_MODE [U4.52.11]

3.5 Mot clé FILTRE_MODE

S'il est présent, sert à introduire les arguments de filtrage des modes à l'intérieur des mots clés FILTRE_MODE (une occurrence par sous-intervalle) de la commande EXTR_MODE[U4.52.12] produisant le résultat final. Tous les modes sont filtrés avec le même critère.

S'il est absent, l'appel à la commande EXTR_MODE produit le résultat final par concaténation sans filtrage des modes propres calculés dans chaque sous-intervalle. On a alors nb_f-1 mots clés FILTRE_MODE ayant pour argument TOUT_ORDRE='OUI'.

3.6 Mot clé IMPRESSION

1 Cf. Exemple du §4.
2 Ces tests sont exprimés dans le contexte du « problème de travail » fournit pour chaque sous-bande au solveur modal. Souvent ce problème transformé est différent du problème initial. La bonne convergence de cette étape n'assure donc pas à 100% celle du problème initial.

Permet d'afficher éventuellement le cumul de valeurs d'un paramètre modal choisi, pour les modes propres calculés du résultat final. Les mots clés internes ont la même signification que dans la commande EXTR_MODE[U4.52.12].

Le paramètre modal choisi peut ne pas être le même que celui qui a servi éventuellement à filtrer les modes calculés.

Le mot clé TOUT_PARA permet d'afficher après chaque calcul modal et normalisation, la valeur de tous les paramètres modaux (fréquence, masses effectives, ...).

3.7 Mot-clé facteur SOLVEUR

◊ SOLVEUR=_F(),

On a accès à tous les paramètres des solveurs linéaires directs (METHODE='LDLT'/'MULT_FRONT'/'MUMPS') sauf ceux explicitement liés à l'étape finale de descente-remontée. Cette restriction ne concernent que les deux paramètres suivant du solveur MUMPS: POSTTRAITEMENTS et RESI_RELA.

En parallèle, on conseille particulièrement le paramétrage [3] METHODE='MUMPS'+RENUM='QAMD'.

Pour plus de détails sur les solveurs, on pourra consulter le document [U4.50.01].

3.8 Opérande NIVEAU_PARALLELISME

◊ NIVEAU_PARALLELISME=/**'COMPLET'** **[DEFAUT]**
/'PARTIEL'

L'usage de MACRO_MODE_MECA est à privilégier par rapport à une combinaison MODE_ITER_SIMULT+NORM_MODE+EXTR_MODE, lorsqu'on traite des problèmes **de tailles moyennes ou grandes** (> 0.5M ddls) et/ou que l'on cherche une **bonne partie de leurs spectres** (> 50 modes).
On découpe alors le calcul en plusieurs sous-bandes fréquentielles (cf. opérande FREQ). Sur chacune de ces sous-bandes, un solveur modal effectue la recherche de modes associée. Pour ce faire, ce solveur modal utilise intensivement un solveur linéaire.
Ces deux briques de calcul (solveur modal et solveur linéaire) sont les **étapes dimensionnantes** du calcul en terme de consommation mémoire et temps. C'est sur elles qu'il faut mettre l'accent si on veut réduire significativement les coûts calcul de cet opérateur.

Or, l'organisation du calcul modal sur des sous-bandes distinctes offre ici un cadre idéal de parallélisme: **distribution de gros calculs presque indépendants**[4]. Son parallélisme permet de gagner beaucoup en temps mais au prix d'un surcoût en mémoire[5].
Si on dispose d'un nombre de processeurs suffisant (> au nombre de sous-bandes non vides), on peut alors enclencher un **deuxième niveau de parallélisme *via* le solveur linéaire** (si on a choisit METHODE='MUMPS'). Celui-ci permettra de continuer à gagner en temps mais surtout, il permettra de compenser le surcoût mémoire du premier niveau voire de diminuer notablement le pic mémoire séquentiel.

3 Afin de réduire au minimum le coût en temps de la phase d'analyse (séquentielle) de MUMPS. Ce paramétrage se fait cependant au détriment de la consommation mémoire. Mais ce surcoût s'avère rapidement compensé par la distribution des données sur les processeurs qu'implique le parallélisme.
4 Aux coûteuses communications de vecteurs propres près.
5 Du fait des buffers MPI requis par les communications de vecteurs propres en fin de MODE_ITER_SIMULT.

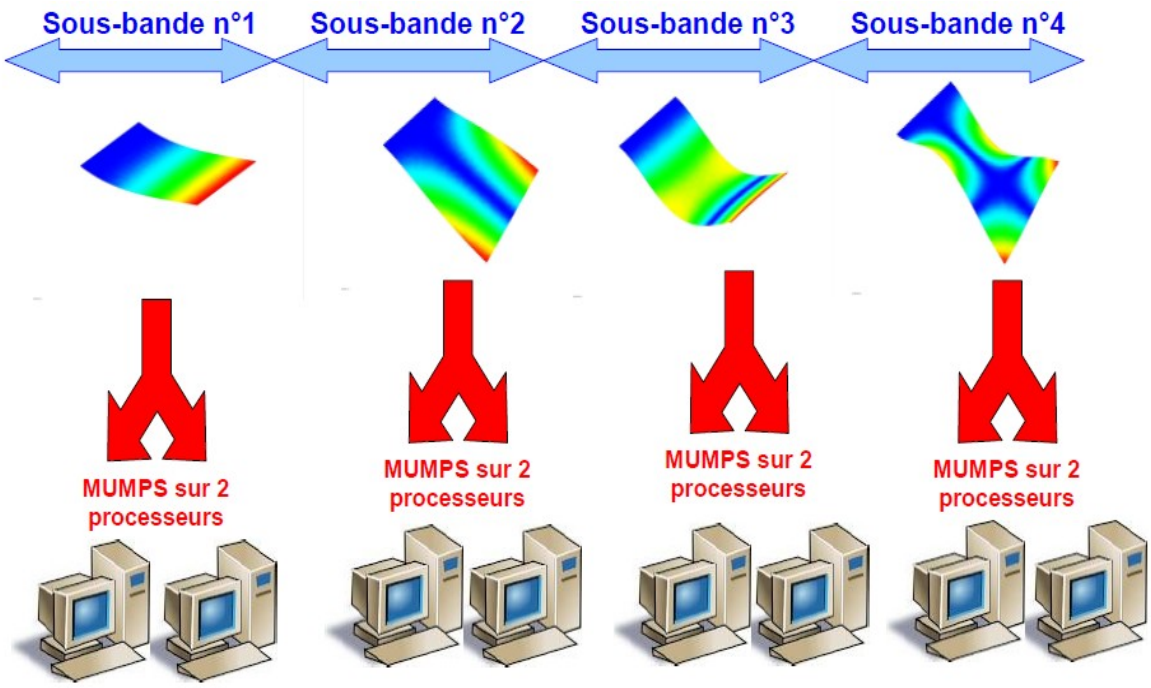
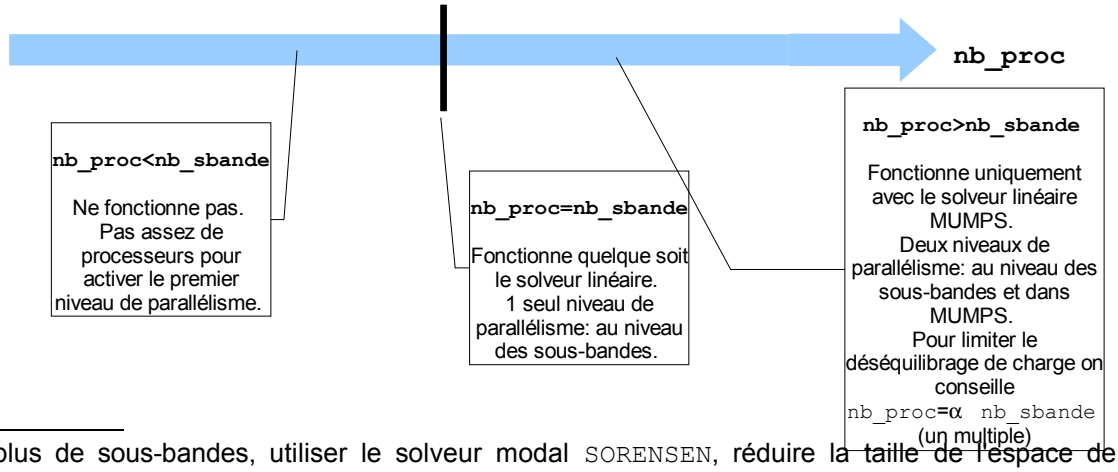


Figure 3.8-1. Exemple de distribution des calculs de MACRO_MODE_MECA sur 8 processeurs avec un découpage en 4 sous-bandes fréquentielles.

Ce double niveau de parallélisme (activé par défaut *via* le mot-clé NIVEAU_PARALLELISME='COMPLET') permet alors de tirer profit, au mieux, des deux aspects.
Lorsqu'on souhaite véritablement gagner en pic mémoire parce que le calcul ne passe pas sur la machine et que l'on a essayé, sans succès, tous les autres bras de levier[6], on peut choisir sciemment de limiter le parallélisme uniquement au niveau du solveur linéaire[7]: NIVEAU_PARALLELISME='PARTIEL'. Cela fonctionne uniquement avec le solveur linéaire parallèle MUMPS.

Les règles fonctionnelles sont les suivantes, en notant nbproc le nombre de processeurs paramétré (onglet option/mpi_nbcpu d'Astk) et nb_sbande le nombre de sous-bandes non vides (=nb_freq-1):

•Avec NIVEAU_PARALLELISME='COMPLET' (**défaut**): très gros gain en temps/amélioration ou détérioration moyenne du pic mémoire RAM.



6 Découper en plus de sous-bandes, utiliser le solveur modal SORENSEN, réduire la taille de l'espace de projection *via* COEF_DIM_ESPACE, utiliser le solveur linéaire MUMPS en OUT_OF_CORE et/ou avec METIS...
7 C'est ce type de parallélisme qui est déployé dans le reste du code.

Figure 3.8-2. Périmètre d'utilisation avec NIVEAU_PARALLELISME='COMPLET'.

•Avec NIVEAU_PARALLELISME='PARTIEL' : gain modéré en temps/gain important sur le pic mémoire RAM.

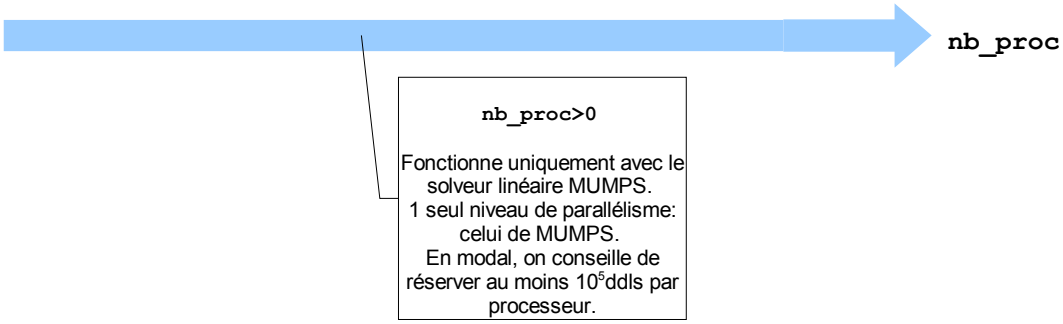


Figure 3.8-3. Périmètre d'utilisation avec NIVEAU_PARALLELISME='PARTIEL'.

Pour un **usage optimal** de cet activation du parallélisme, il est donc conseillé de :

•Construire des sous-bandes de calcul relativement équilibrées. Pour ce faire, on peut donc, au préalable, calibrer le spectre étudié *via* un ou plusieurs appel à INFO_MODE[U4.52.01]. Si possible en mode parallèle. Puis lancer le calcul MACRO_MODE_MECA parallèle en fonction du nombre de sous-bandes choisies et du nombre de processeurs disponibles.

•De prendre des sous-bandes plus fines qu'en séquentiel, entre 10 et 20 modes au lieu de 40 à 80 modes en séquentiel. La qualité des modes et la robustesse du calcul s'en trouvera accrue. Le pic mémoire en sera diminué. Il reste cependant à avoir suffisamment de processeurs disponibles (et avec assez de mémoire).

•Sélectionner un nombre de processeurs qui est un multiple du nombre de sous-bandes (non vides). Ainsi, on réduit les déséquilibrages de charges qui nuisent aux performances.

Pour réduire le pic mémoire d'un calcul, on dispose de plusieurs bras de levier: réduire la taille des sous-bandes, utiliser le solveur linéaire MUMPS (éventuellement en OUT_OF_CORE[U4.50.01]) et/ou ne paralléliser que cette brique de calcul (NIVEAU_PARALLELISME='PARTIEL').

Pour utiliser efficacement **MACRO_MODE_MECA en parallèle** , on propose donc de procéder en **trois étapes:**

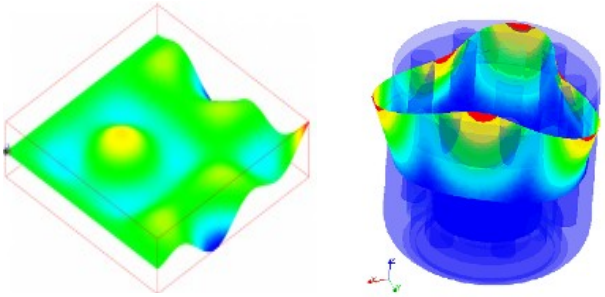
•**Pré-calibrations modales** préalables *via* INFO_MODE. Si possible, en mode parallèle (Gains potentiels en temps x70 sur une centaines de processeurs. Gain en pic mémoire RAM jusqu'à x2).

•Examiner les résultats produits et décomposer le calcul en sous-bandes de taille modeste (par ex. 20 modes) et équilibrée, en fonction du nombre de processeurs disponibles.

•Lancer en mode POURSUITE , le calcul MACRO_MODE_MECA parallèle proprement dit.

Cas-test perf016a (N=4.0M, 50 modes) découpage en **8 sous-bandes**	**Temps elapsed**	**Pic mémoire RAM**
1 processeur	5524s	16.9Go
8 processeurs	1002s	19.5Go
32 processeurs	643s	13.4Go
découpage en **4 sous-bandes**		
1 processeur	3569s	17.2Go

4 processeurs	1121s	19.5Go
16 processeurs	663s	12.9Go



Etude sismique (N=0.7M, 450 modes) découpage en **20 sous-bandes**	**Temps elapsed**	**Pic mémoire RAM**
1 processeur	5200s	10.5Go
20 processeurs	407s	12.1Go
80 processeurs	270s	9.4Go
découpage en **5 sous-bandes**		
1 processeur	4660s	8.2Go
5 processeurs	1097s	11.8Go
20 processeurs	925s	9.5Go

Tableaux 3.8-1. Quelques résultats de tests de MACRO_MODE_MECA parallèle avec les paramètres par défaut (+ SOLVEUR=MUMPS en IN_CORE et RENUM='QAMD'). Code_Aster v11.3.11 sur la machine IVANOE (1 ou 2 processus MPI par noeud).

Remarques:

- *En mode NIVEAU_PARALLELISME='COMPLET' , si le nombre de processeurs n'est pas un multiple du nombre de sous-bandes (non vides), on distribue le reliquat de processeurs en privilégiant les premières sous-bandes. Un message avertit l'utilisateur du potentiel déséquilibrage de charge et du caractère sous-optimal du calcul.*
- *En mode NIVEAU_PARALLELISME='COMPLET', on a désactivé le parallélisme des calculs élémentaires et des assemblages qui peuvent s'opérer dans NORM_MODE . Leur coût est de toute manière marginal. Cette désactivation est temporaire et juste limité à MACRO_MODE_MECA .*
- *En mode NIVEAU_PARALLELISME='COMPLET', on communique tous les vecteurs propres exhumés en fin de MODE_ITER_SIMULT. Donc la distinction[8] entre les valeurs STURM='LOCAL' ou ' GLOBAL' n'a plus lieu d'être fonctionnellement. Ce n'est pas grave car le mode par défaut à privilégier est le mode ' GLOBAL'.*

Pour la mise en œuvre pratique du parallélisme, on se reportera au documents générique [U2.08.06] sur le parallélisme, et au paragraphe dédié de [U2.06.01] sur le calcul modal.

8 La distinction entre les deux modes est juste ici d'ordre informatique: dans le cas 'GLOBAL', le test de Sturm est mis en oeuvre au niveau du fichier PYTHON de la macro, alors que dans le cas 'LOCAL', il est opéré dans le F77 de MODE_ITER_SIMULT.

4 Exemple

Soit la séquence suivante :

```
mode=MACRO_MODE_MECA  (
      MATR_RIGI=rigi, MATR_MASS=masse,

      CALC_FREQ=_F(FREQ=(1.,3.,5.),)
      VERI_MODE=_F(),

      NORM_MODE=_F(NORME='TRAN_ROTA',),

      FILTRE_MODE=_F(CRIT_EXTR='MASS_EFFE_UN'),

      IMPRESSION=_F(CUMUL='OUI',
                    CRIT_EXTR='MASS_EFFE_UN')
                          );
```

On va donc chercher tous les modes compris dans la bande globale $[1.,5.]$ en la découpant en deux sous-bandes fréquentielles: $[1.,3.]$ et $[3.,5.]$.
Une fois interprétée, la macro-commande consiste à l'enchaînement de commandes usuelles décrit ci-dessous.

4.1 Étape n°1

Détermination du nombre de fréquences dans chaque sous-bandes

```
      table1=INFO_MODE(MATR_RIGI=rigi, MATR_MASS=masse,
                       FREQ=(1.,3.,5.))
```

Calcul du nombre de fréquences théoriques de la bande globale[9]: nbmodeth
Si la bande globale est vide: ALARME ou ERREUR_FATALE suivant la valeur de CALC_FREQ/STOP_BANDE_VIDE.

4.2 Étape n°2

Calcul et normalisation des modes dans chaque sous-bandes
Pour économiser les coûts calcul, on réutilise la table générée précédemment[10] et, par défaut, on ne fait pas localement à chaque sous-bande le test de Sturm de post-vérification.
Si la sous-bande locale est vide: ALARME ou ERREUR_FATALE suivant la valeur de CALC_FREQ/STOP_BANDE_VIDE.

```
      mode_1=MODE_ITER_SIMULT( MATR_RIGI=rigi,MATR_MASS=masse,
                CALC_FREQ=_F(OPTION='BANDE',
                             FREQ=(1.,3.),TABLE_FREQ=table1),),
                             VERI_MODE(STURM='NON'));
      mode_1=NORM_MODE(MODE=mode_1,reuse=mode1,
                NORME='TRAN_ROTA',);
      mode_2=MODE_ITER_SIMULT( MATR_RIGI=rigi,MATR_MASS=masse,
                CALC_FREQ=_F(OPTION='BANDE',
                             FREQ=(3.,5.),TABLE_FREQ=table1),),
                             VERI_MODE(STURM='NON'));
      mode_2=NORM_MODE(MODE=mode_2,reuse=mode2,
```

9 On somme juste les nombres de fréquences calculés précédemment et stockés dans table1.
10 Pour ne pas refaire le test de Sturm de prétraitement propre à chaque sous-bande.

```
                    NORME='TRAN_ROTA',);
```

Vérification par un test de Sturm global du bon nombre de fréquences calculées
Détermination de la plus petite (resp. grande) fréquence de la première (resp. dernière) sous-bande non vide: freq_ini (resp. freq_fin).
Calcul du nombre de fréquences comprises dans l'intervalle: [freq_ini,freq_fin] : nbmodeef .

```
      table2=INFO_MODE(MATR_RIGI=rigi, MATR_MASS=masse,
                       FREQ=(freq_ini,freq_fin))
```

Si ce nombre de modes est différent du nombre de modes prévu initialement: ERREUR_FATALE .

4.3 Étape n°3

Filtrage, concaténation et impression des modes calculés.

```
      mode=EXTR_MODE(FILTRE_MODE=_F(MODE=mode_1,
                                    CRIT_EXTR='MASS_EFFE_UN'),
                     FILTRE_MODE=_F(MODE=mode_2,
                                    CRIT_EXTR='MASS_EFFE_UN'),
                     IMPRESSION=_F(CUMUL='OUI',
                                    CRIT_EXTR='MASS_EFFE_UN'),
                     ) ;
```