Procédure de dénombrement de valeurs propres

Résumé

Que cela soit pour étudier les **vibrations d'une structure** ou rechercher ses **modes de flambement**, le mécanicien doit souvent résoudre un problème modal : soit généralisé (GEP) [R5.01.01], soit quadratique (QEP)[R5.01.02]. Pour ce faire, *Code_Aster* propose plusieurs algorithmes et méthodologies au travers de l'opérateur CALC_MODES, sans compter les opérateurs de support : INFO_MODE et NORME_MODE.

Souvent ces opérateurs utilisent en pré ou en post-traitement une procédure de dénombrement. Car certaines de leurs fonctionnalités requièrent le nombre de valeurs propres incluses à l'intérieur d'un intervalle (si elles sont réelles) ou d'un disque (si elles sont complexes). Dans le premier cas de figure, concernant uniquement les **GEPs symétriques réels**, on dispose de la traditionnelle méthode de Sturm. Dans le second cas, regardant les **GEPs quelconques** et les **QEPs**, la situation est beaucoup moins favorable. Ce problème numérique fait encore l'objet de recherches actives et aucune solution complètement satisfaisante n'a encore émergé.
Après avoir testé plusieurs variantes de ces méthodes, nous en avons finalement industrialisée une dans le code : la méthode **APM variante** $\mathbf{LDLT}$. Elle n'est pour l'instant disponible que dans INFO_MODE [U4.52.01] (METHODE='APM'), en complément de la méthode de Sturm (METHODE='STURM'). Cette méthode fait encore l'objet de recherches. Elle est à réserver aux problèmes simplifiés de petite taille (< à 10^4 degrés de liberté).

Dans le cas le plus fréquent de GEP symétriques réels, **il est fortement conseillé de pré-calibrer son calcul modal *via* des estimations INFO_MODE préalables** [U4.52.01]. Grâce à leurs **deux niveaux de parallélisme**[1], ces INFO_MODE peuvent être « quasi-gratuits » ! Sur une centaine de processeurs, on peut obtenir des accélérations en temps de l'ordre de 70 et des réductions de pic mémoire de l'ordre de 2.

Dans la première partie de ce document nous résumons la problématique du dénombrement de valeurs propres. Puis, avant de détailler les différentes solutions existantes, nous précisons leurs principaux ingrédients numériques. Un chapitre spécifique détaille la mise en œuvre du parallélisme multi-niveaux. Enfin, nous concluons par un résumé des algorithmes effectivement disponibles dans le code et de leurs paramétrages.

1 Tout comme CALC_MODES avec option 'BANDE' découpée en sous-bandes.

Table des Matières

1 Introduction

1.1 Contexte

Que cela soit pour étudier **les vibrations d'une structure** , éventuellement amortie et/ou tournante, ou rechercher ses modes de flambement, le mécanicien doit souvent résoudre **un problème modal** . Pour ce faire, *Code_Aster* propose différents opérateurs [Boi10] [2] qui traitent deux types de problèmes modaux : les **Généralisés** (GEP pour 'Generalized Eigenvalue Problem') et les **Quadratiques** (QEP pour 'Quadratic Eigenvalue Problem'):

$$\text{Trouver } (\lambda, \mathbf{u}) \text{ tel que : } \begin{array}{ll} (\mathbf{A}-\lambda\mathbf{B})\mathbf{u}=0 & (GEP) \\ (\mathbf{A}+\lambda\mathbf{B}+\lambda^2\mathbf{C})\mathbf{u}=0 & (QEP) \end{array} \quad (1.1)$$

Suivant la problématique, les matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$ précitées sont des combinaisons linéaires des différentes matrices mécaniques usuelles:

- Masse $\mathbf{M}$,
- Rigidité $\mathbf{K}$,
- Rigidité géométrique $\mathbf{K}_g$,
- Amortissement induit par des forces dissipatives $\mathbf{E}_{visq}$ ou par la structure $\mathbf{E}_{hyst}$,
- Effet gyroscopique $\mathbf{G}$.

On s'est volontairement placé dans le cadre le plus général, mais souvent, on peut néanmoins faire l'amalgame :

$$\mathbf{A} \leftarrow \mathbf{K}, \quad \mathbf{B} \leftarrow \mathbf{M} \text{ et } \mathbf{C} \leftarrow \mathbf{E}_{visc} + \xi \mathbf{G} \quad (1.2)$$

avec ξ un paramètre réel représentant la vitesse de rotation (en cas d'effet gyroscopique).

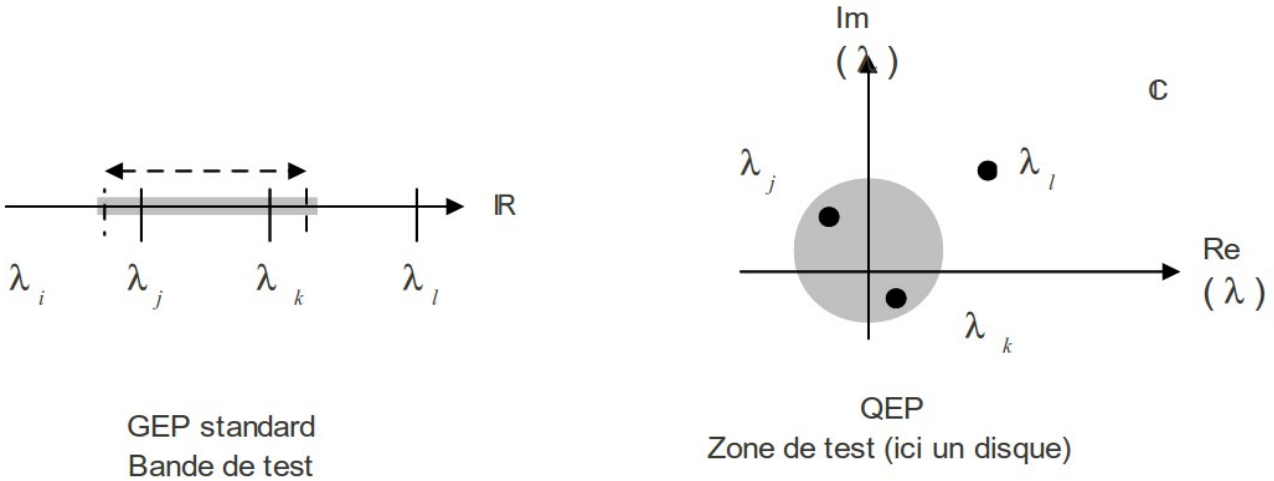


Figure 1.1 Deux problématiques distinctes: dénombrement des valeurs propres contenues strictement dans un segment de l'axe réel et dans une portion finie du plan complexe.

Ces matrices sont souvent réelles (sauf en présence d'amortissement hystérétique), mais pas toujours symétriques (par exemple en présence d'effets gyroscopiques) et rarement définies positives (à cause des Lagranges notamment). **Cet écart à la norme des problèmes modaux usuels** (SEP symétrique et définis positif) et **cette variabilité des problématiques complique indubitablement leurs traitements algorithmiques dans le code.**

2 INFO_MODE et CALC_MODES.

1.2 Dénombrement et calcul modal dans *Code_Aster*

En effet, **la résolution d'un problème modal, tant standard, généralisé que quadratique reste un problème difficile**. Même si, dans la littérature, certaines catégories de problèmes modaux semblent clairement résolues depuis longtemps (par exemple les GEPs standards). Il n'en reste pas moins que la mise en oeuvre numérique reste difficile.

On rajoute donc systématiquement[3], en vérification d'un calcul modal, une procédure de dénombrement de valeurs propres. **Pour les GEPs standards (symétriques réels) ce dénombrement s'effectue dans un segment de l'axe réel** (cf. figure 1.1a). Par exemple, on vérifie que, dans une bande fréquentielle donnée, le nombre de valeurs propres «théoriques» correspond bien au nombre de valeurs propres identifiées par le solveur modal. Pour ce faire, on utilise un algorithme ancien et classique (test de Sturm, cf. §2.2/3.2), qui bien que couteux[4] reste assez fiable.

Mais l'extension de ce critère aux QEPs et aux GEPs atypiques (non symétriques ou complexes), pour laquelle **on cherche à estimer le nombre de valeurs propres contenues dans une zone finie du plan complexe** (par exemple un disque centré sur la valeur qui intéresse l'utilisateur [5] , cf. figure 1.1b), n'est sérieusement envisageable que depuis quelques années (cf. tableau ci dessous).

Le passage d'un dénombrement sur l'axe réel à un dénombrement dans le plan complexe n'est en effet pas trivial. Les algorithmes utilisées partagent des outils communs (cf. § 2), mais leur complexité, leur risque de défaillance et leur coût calcul sont considérablement amplifiés par ce changement de dimension.

Dans les fonctionnalité de calcul modal de *Code_Aster*, on rajoute ce test de comptage de valeurs propres aux vérifications sur les résidus modaux[6]

$$\begin{aligned} (GEP) \quad & \frac{\|(\mathbf{A}-\lambda\mathbf{B})\mathbf{u}\|}{\|\mathbf{A}\mathbf{u}\|} < \text{SEUIL} \\ (QEP) \quad & \frac{\|(\mathbf{A}+\lambda\mathbf{B}+\lambda^2\mathbf{C})\mathbf{u}\|}{\|\mathbf{A}\mathbf{u}\|} < \text{SEUIL} \end{aligned} \tag{1.3}$$

Le dénombrement des valeurs propres est aussi indispensable à quatre autres fonctionnalités des opérateurs modaux:

- **Opérateur `INFO_MODE`:** Cet opérateur fournit une estimation préalable du nombre de valeurs propres contenues dans une bande fréquentielle donnée afin de jauger un futur calcul de dynamique.
- **Opérateur `CALC_MODES + OPTION='BANDE'`:** La même problématique mais cette fois pour paramétrer les principaux solveurs modaux. On doit en effet leur fixer un nombre de fréquences maximum à calculer, alors que l'utilisateur peut fournir uniquement la bande fréquentielle qui l'intéresse. A partir de cette bande, on doit donc estimer le nombre requis.
- **Opérateur `CALC_MODES + OPTION='BANDE'` avec découpage en plusieurs sous-bandes[7]:** Lorsqu'on cherche à calculer une partie conséquente du spectre (>40 valeurs propres), il est beaucoup plus efficace et fiable de décomposer son calcul global en plusieurs sous-calculs indépendants. Cette décomposition est basée sur une partition de la zone de recherche. Pour organiser cette partition, on se base notamment sur cette procédure de comptage.

3 Si le problème modal est dans le périmètre de la méthode actuelle (GEP standard) et si l'utilisateur ne l'a pas intentionnellement débranchée (`VERI_MODE/STURM='OUI'`).

4 Coût (en temps et en mémoire) de deux factorisations LDL^T successives.

5 Par exemple la valeur du shift: la valeur `FREQ` donnée par l'utilisateur avec l'option '`CENTRE`', sinon la valeur 0 pour les autres options.

6 En amont, se rajoutent aussi les vérifications et les critères d'arrêts propres aux solveurs modaux. Ces derniers s'effectuent généralement sur un problème modal modifié (dit de travail); D'où le besoin en sortie de ces «boîtes à outils génériques», après remise à l'échelle des solutions et quelques filtrages et tris, de post-vérifications globales dans le cadre fonctionnel initial.

7 La parallélisation de cet opérateur nécessite, idéalement, que ces partitions soient équilibrées en nombre de valeurs propres.

- **Opérateur CALC_MODES + OPTION='PROCHE' ou 'AJUSTE' ou 'SEPARE' :** C'est un opérateur qui sert à calculer seulement quelques modes du spectre et dont l'heuristique initiale[8] (activée avec OPTION='AJUSTE'/'SEPARE') est basée sur un critère de comptage.

1.3 Périmètre d'utilisation

En pratique, **ces cinq besoins de dénombrement sont couverts par la méthode de Sturm dans le cadre des GEPs standards** . Lorsqu'on passe à des GEPs atypiques ou à des QEPs, cette fonctionnalité n'est plus disponible. Cette restriction du périmètre d'utilisation est protégée par des alarmes dédiées et elle est mentionnée dans les documentations Utilisateur.

Seul l'opérateur INFO_MODE [U4.52.01] **bénéficie depuis peu** [Boi11] d**e cette extension du périmètre au plan complexe** (*via* la METHODE='APM'). Après avoir maquetté et testé plusieurs familles de méthode de comptage (cf. §3.4/3.5), seule la «méthode APM variante LDLT» semble suffisamment mature (cf. §3.3), malgré son coût calcul, pour être industrialisée. Au gré des besoins, de **nombreuses améliorations fonctionnelles pourront d'ailleurs lui être apportée :**

- Divers contours optimisés (demi-cercle, quart de cercle…),
- Contour utilisateur,
- Méthode de suivi de contour pour fiabiliser/optimiser la discrétisation [KP11],
- Optimisation des calculs de déterminant [KP12],
- Gain en performance par mutualisation de calculs d'argument dans l'heuristique de comptage,
- Distribution parallèle de ces calculs indépendants [NK12],
- A proximité des valeurs propres, processus de recalcul du déterminant similaire à celui du test de Sturm (cf. paramètres PREC_SHIFT et NMAX_ITER_SHIFT)….

1.4 Synopsis

On récapitule ci-dessous parmi les méthodes de comptage existantes, celles qui ont été seulement évaluées et celles qui sont effectivement disponibles dans *Code_Aster*.

Famille de méthode	**Sturm Standard**	**APM ('Argument Principle-based Method')**	**Formule de Quadrature**	**Sturm Modifié ('Modified Sturm Sequence Method')**
Périmètre d'utilisation	Axe réel (GEP symétrique réel)	Plan complexe (GEP et QEP quelconques)		
Stratégie explorée par les équipes	Sans objet	INRIA-Rennes (APM+LDLT), HYUNDAI (APM+LDLT/ Rombouts).	INRIA-Rennes.	HYUNDAI.
Stratégie maquettée et testée dans Code_Aster	Oui	Oui (les deux variantes)	Non	Oui
Stratégie retenue dans une fonctionnalité de *Code_Aster*	INFO_MODE, CALC_MODES avec option 'BANDE', CALC_MODES + post-vérification, CALC_MODES avec option 'BANDE' découpée en sous-bandes, CALC_MODES avec option 'AJUSTE' ou 'SEPARE'	APM+LDLT dans INFO_MODE.	Non	

8 Qui participe à l'algorithme des puissances inverses.

Famille de méthode	Sturm Standard	APM ('Argument Principle-based Method')	Formule de Quadrature	Sturm Modifié ('Modified Sturm Sequence Method')
Paramétrage	§4.1 SEUIL_FREQ, PREC_SHIFT, NMAX_ITER_SHIFT.	§4.1 NBPOINT_CONTOUR, NMAX_ITER_CONTOUR.	Non	
Paragraphes concernés	§2.2	§2.1/2.3/3.3/4	§3.4	§2.4/2.5/2.6/3.5
Avantages	Relativement peu couteuse (deux factorisations LDLT).	Pas de restriction du périmètre d'utilisation (GEP/QEP qcq); Souplesse d'utilisation (contour optimisé ou utilisateur…); Evolutivité (plusieurs variantes…); Pic mémoire identique à celui de la méthode de Sturm standard.	Pas de restriction du périmètre d'utilisation (GEP/QEP qcq); Contrôle intrinsèque à la méthode du niveau de discrétisation ; Pic mémoire identique à celui de la méthode de Sturm standard.	Faible coûts calcul (pas de calcul de déterminant .
Inconvénients	Périmètre réduit à l'axe réel (GEP symétrique réel). A adapter pour prendre en compte les spécificités des matrices *Aster*: dualisation et variables de Lagrange.	Contrôle de la discrétisation; Évaluation fiable de $Arg(P(\lambda))$; Procédure de comptage du nombre de tours; Coûts en temps calcul[9] ($100\,\vartheta(N^{\alpha}\xi^{\beta})$); Méthode plus adaptée à un contrôle *a posteriori*.	Prise en compte d'un contour quelconque; Évaluation fiable de $P(\lambda)/P'(\lambda)$; Coûts en temps calcul ($100\,\vartheta(N^{\alpha}\xi^{\beta})$).	Limité au SEP; Limité au disque centré à l'origine; Manipulation très instable des coefficients polynomiaux; Coûts mémoire très importants (en $\vartheta(N^2)$).

Tableau 1.1 Récapitulatif des procédure de dénombrement.

Avant d'aborder les pistes numériques testées voire industrialisées dans le code ainsi que leur paramétrage / cadre d'utilisation, il est nécessaire de passer en revue quelques ingrédients numériques et mathématiques.

9 Avec N la taille du problème, ξ sa largeur de bande et α/β deux entiers <2 .

2 Pré-requis mathématiques et numériques

2.1 Ingrédient n°1: Le polynôme caractéristique

Les problèmes modaux GEP et QEP (1.1) peuvent se reformuler sous la forme d'une recherche de racines de polynômes particuliers, appelé **polynômes caractéristiques**

Trouver λ tel que :

$$\begin{aligned} P_{GEP}(\lambda) &:= dét(\mathbf{A}-\lambda\mathbf{B}) := a_0+a_1\lambda+...+a_n\lambda^n=0 \quad (GEP) \\ P_{QEP}(\lambda) &:= dét(\mathbf{A}+\lambda\mathbf{B}+\lambda^2\mathbf{C}) := a_0+a_1\lambda+...+a_{2n}\lambda^{2n}=0 \quad (QEP) \end{aligned} \tag{2.1.1}$$

avec n la taille des matrices considérées. Ces polynômes sont caractéristiques des problèmes modaux concernés. Ils contiennent via leurs coefficients des informations importantes : les valeurs propres, les déterminants et les traces des matrices considérées.

Les zéros de ces polynômes caractéristiques sont les valeurs propres recherchées. Ses racines sont distinctes ou non, elles peuvent être réelles (GEPs standards) ou complexes (dans le cas des QEPs et des GEPs atypiques). Contrairement aux GEPs, les QEPs comportent plus de racines que la taille du problème $(2n)$ et celles-ci peuvent être infinies.

En général **on ne se sert pas de cette formulation polynomiale pour calculer les modes propres** des problèmes (1.1). Outre le calcul des coefficients polynomiaux (cf. §2.4), se pose le problème aigu de leur manipulation. Celle-ci est souvent coûteuse et très instable[10] (cf. §2.6). On lui **préfère la représentation matricielle sous la forme d'une factorisation** $\mathbf{LDL^T}$, ainsi le calcul est plus stable:

$$\begin{aligned} P_{GEP}(\lambda) &:= det(\mathbf{A}-\lambda\mathbf{B}) = \sigma\, det(\mathbf{LDL^T}) = \sigma \prod_{j=1}^{n} \mathbf{D_{jj}} \quad (GEP) \\ P_{QEP}(\lambda) &:= det(\mathbf{A}+\lambda\mathbf{B}+\lambda^2\mathbf{C}) = \sigma\, det(\mathbf{LDL^T}) = \sigma \prod_{j=1}^{n} \mathbf{D_{jj}} \quad (QEP) \end{aligned} \tag{2.1.2a}$$

En considérant, pour simplifier et comme c'est souvent le cas dans *Code_Aster*, que les matrices sont symétriques. On note σ (entier valant ± 1) la signature de la matrice de permutation $\mathbf{P}$ assurant la décomposition recherchée

$$\begin{aligned} P(\mathbf{A}-\lambda\mathbf{B}) &:= \mathbf{LDL^T} \quad (GEP) \\ P(\mathbf{A}+\lambda\mathbf{B}+\lambda^2\mathbf{C}) &:= \mathbf{LDL^T} \quad (QEP) \end{aligned} \tag{2.1.2b}$$

Lorsqu'on passe en non symétrique, cela ne change rien au procédé. On parle alors de décomposition $\mathbf{LU}$ et les termes diagonaux manipulés seront ceux de $\mathbf{U}$ plutôt que ceux de $\mathbf{D}$.
Ainsi, après avoir factorisé sous forme $\mathbf{LDL^T}$, les matrices de travail $\mathbf{A}-\lambda\mathbf{B}$ (pour les GEPs) ou $\mathbf{A}+\lambda B+\lambda^2 C$ (pour les QEPs), le calcul du déterminant[11] devient juste un produit des n termes diagonaux de $\mathbf{D}$.

Remarques:

- Dans Code_Aster, la terminologie retenue désigne souvent ces matrices de travail sous le vocable de «matrices dynamiques».

10 Erreurs d'arrondis et overflows dues aux manipulations d'un grand nombre de termes d'ordres de grandeurs très différents (cf. paragraphes sur la méthode de Rombouts). Le polynôme caractéristique est connu pour être un polynôme très difficile à appréhender. Et si cette méthode apporte un gain par rapport à une approche de simple calcul de déterminant, elle reste instable et très coûteuse sur beaucoup de problèmes, même de petite taille.

11 Puisque la matrice $\mathbf{L}$ est triangulaire inférieure à diagonal unité et que $\mathbf{D}$ est une matrice diagonale.

- Dans les formules ci-dessus, la signature σ de la matrice de permutation intervient. Cet entier correspond au nombre de permutations élémentaires[12] constituant cette permutation. Il est égale à 1 si ce nombre est pair, égale à -1 sinon. En pratique, on ne le calcule pas explicitement, il est induit dans les pré-traitements[13] du solveur direct.
- *Dès que l'une des matrices comporte des termes complexes (par exemple en présence d'amortissement hystérétique) ou dès qu'on cherche à calculer $P(\lambda)$ avec un λ complexe, - cela sera notre cas dans certaines méthodes de dénombrement -, le déterminant est un produit de nombres complexes, donc potentiellement un nombre complexe. Dans les méthodes de dénombrement qui vont suivre, on ne sera intéressé en fait que par son argument.*

Par contre, même dans ce cadre matriciel couramment utilisé dans les codes, **l'évaluation numérique du déterminant peut s'avérer problématique**. Cette fonction de λ peut subir de fortes variations suivant que l'on est plus ou moins proche d'une valeur propre. D'autre part, puisqu'il faut faire un produit de n termes d'ordre de grandeurs très différents, se pose des problèmes de dépassement de capacité, d'élimination et d'absorption. Donc, pour essayer de limiter ces problèmes, souvent on ne calcule pas directement le déterminant comme un produit d'éléments diagonaux

$$P(\lambda)=\sigma\prod_{j=1}^{n}\mathbf{D_{jj}} \tag{2.1.3}$$

mais plutôt sa version normalisée

$$\frac{P(\lambda)}{|P(\lambda)|}=\sigma\prod_{j=1}^{n}\frac{\mathbf{D_{jj}}}{|\mathbf{D_{jj}}|} \tag{2.1.4}$$

avec, dans un vecteur auxiliaire, les modules de normalisation $\left(|\mathbf{D_{jj}}|\right)_{j=1}^{n}$ permettant de le reconstituer le cas échéant. Ce déterminant n'est alors connu que de manière implicite. L'information recherchée étant souvent un changement de signe[14] ou une forte évolution de sa valeur (augmentation ou diminution), cette connaissance parcellaire n'est pas forcément préjudiciable.
Par exemple, dans le cadre de nos procédures de comptage, **nous ne sommes intéressés que par l'argument θ du déterminant**

$$P(\lambda)=\sigma\prod_{j=1}^{n}\chi_j e^{i\varphi_j}:=\rho e^{i\theta} \tag{2.1.5}$$

Comme le fait de normaliser un produit de complexes ne change pas l'argument du résultat[15], on peut calculer sans trop de risque numérique cet argument θ en se référant uniquement à la version normalisée (2.1.4).

Remarques:

- *D'un point de vue logicielle, nous avons apporté un soin particulier à ces estimations de déterminants et à l'évaluation de leurs arguments. C'est en effet l'ingrédient principal de beaucoup des méthodes étudiées ici et elles sont très sensibles à sa qualité. La moindre fausse information les fait diverger. Ainsi nous avons essayer de gérer méticuleusement les cas extrêmes (termes quasi-nuls, argument proche des bornes 0 et π , évaluation via la routine `ATAN2` …).*
- *Dans Code_Aster, il existe déjà une routine de calcul de déterminant (`MTDETE`). Elle est dédiée au déterminant réel et elle gère les problèmes d'overflow/underflow en dissociant la partie mantisse de la partie exposant du déterminant produit. Suite à l'adoption de l'astuce précitée, nous n'avons pas eu besoin de l'utiliser. D'autre part notre besoin (simplement l'argument du déterminant) est trop éloigné de ses spécifications (le calcul «exact» du déterminant).*

12 Permutation élémentaire: permutation $\mathbf{P_{ij}}$ inter-changeant seulement deux lignes i et j de la matrice (parfois appelée aussi transposition $\mathbf{T_{ij}}$).

13 Étape de renumérotation (METIS, AMD...) voire permutation ligne/colonne avec MUMPS.

14 On pourrait peut-être se servir de ce type de critère pour les études de flambement de *Code_Aster*. La nouvelle fonctionnalité de MUMPS de calcul de déterminant de la dernière version (v4.10) pourrait ainsi s'avérer très utile en parallèle.

15 C'est-à-dire, pour reprendre un formalisme géométrique, l'orientation du vecteur image V dans le plan complexe associé à l'affixe $Z:=P(\lambda)$ est identique à celle de $V/\|V\|$ d'affixe $z:=P(\lambda)/|P(\lambda)|$.

- *D'autres auteurs, E.Kamgnia et B.Philippe[KP11], ont proposé récemment de manipuler le déterminant via un triplet comportant un réel strictement positif, un complexe de module unité et un entier positif* (ρ, K, n) *vérifiant:*

$$P(\lambda)=\rho K^n$$
$$\rho := \sigma \prod_{j=1}^{n} \frac{\mathbf{D_{jj}}}{|\mathbf{D_{jj}}|}$$
$$K := \left(\prod_{j=1}^{n} \mathbf{D_{jj}} \right)^{1/n} \quad (2.1.6)$$

- Ce procédé pourrait être testé dans Code_Aster car il facile la gestion des problèmes de dépassement de capacité.

2.2 Ingrédient n°2: Le théorème des suites de Sturm

Lorsque le **spectre d'un (GEP) est uniquement contenu sur l'axe réel** (matrices $\mathbf{A}$ et $\mathbf{B}$ symétriques réelles), on dispose depuis une quarantaine d'années d'une méthode efficace, appelée «**méthode des suites de Sturm**» ([Boi10c] §3, [Hau80]).

Cette méthode est due au mathématicien suisse Charles Sturm (1829). Son objectif est de calculer le nombre de racines distinctes d'un polynôme $P(x)$ à coefficients réels dans un intervalle $[a,b]$ de l'axe réel (avec a et b qui ne sont pas des racines de ce polynôme). Elle propose un processus pour construire deux suites finies de polynômes $(P_n(x))_{n=1}^{k}$ et $(Q_n(x))_{n=1}^{k}$, associé au polynôme initial *via* le processus suivant

$$\begin{aligned} P_0 &= P \\ P_1 &= P' \\ P_0 &= P_1 Q_1 - P_2 \\ P_1 &= P_2 Q_2 - P_3 \\ &\dots \end{aligned} \quad (2.2.1)$$

Ce processus constructif, une fois initialisé, suit un schéma du type «algorithme d'Euclide». On poursuit ce processus jusqu'à un rang noté k, puis on estime les séquences finies suivantes:

$$\begin{aligned} &P_0(a), P_1(a), \dots, P_{k-1}(a), P_k(a) \\ &P_0(b), P_1(b), \dots, P_{k-1}(b), P_k(b) \end{aligned} \quad (2.2.2)$$

Soit $\sigma(a)$ (respectivement $\sigma(b)$) le nombre de changements de signe de ces séquences, le nombre de racines recherché est alors égal à l'écart de ces deux chiffres

$$\underset{]a,b[}{card}\{x / P(x)=0\} = \sigma(b) - \sigma(a) \quad (2.2.3)$$

$$\mathbf{A}\mathbf{u}_i = \lambda_i \mathbf{B}\mathbf{u}_i$$

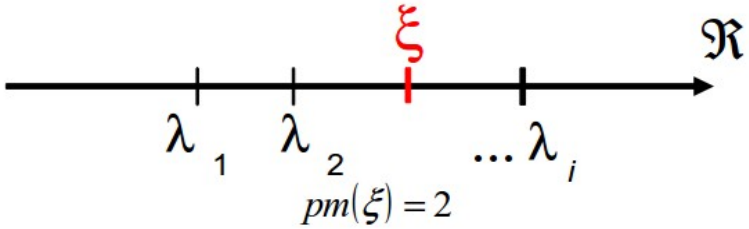


Figure 2.1 Test de Sturm de Code_Aster : *principe des suites de Sturm appliqué au GEP standard.*

En appliquant ce résultat à un polynôme particulier, le polynôme caractéristique d'un GEP standard (cf. figure 2.1), on trouve un résultat tout à fait intéressant et «non trivial»: la factorisation $\mathbf{LDL}^{\mathbf{T}}$ de la matrice de travail $\mathbf{A}-\lambda\mathbf{B}$ «mime» en fait ce processus constructif du type «Sturm-Euclide» et **il suffit de compter les coefficients strictement négatifs de la matrice $\mathbf{D}$ pour obtenir le nombre de valeurs propres strictement inférieures à** λ. C'est ce qu'on appelle la position modal de ξ (noté $pm(\xi)$ dans la documentation *Aster*). Soit, en redéfinissant dans ce cadre modal les changements de signe précédent

$$\sigma(\xi)=card\{\mathbf{D_{ii}}<0/\mathbf{A}-\xi\mathbf{B}=\mathbf{LDL}^{\mathbf{T}}\} \tag{2.2.4}$$

on a donc

$$\begin{aligned} pm(\xi)&=\sigma(\xi)\\ N_{GEP}^{]a,b[}&=\sigma(b)-\sigma(a)\end{aligned} \tag{2.2.5}$$

Ce test de Sturm a été étendu aux formes de GEPs standards plus générales rencontrées dans *Code_Aster* (cf. §3.2). Et ce, afin de tenir compte des Lagranges et des formes dualisées des matrices associées.

Lorsque les valeurs propres appartiennent au plan complexe ce test n'est plus suffisant. Il faut alors recourir à la formule intégrale de Cauchy.

2.3 Ingrédient n°3: La formule intégrale de Cauchy

Avant d'aborder cette notion, il faut définir la zone du plan complexe à laquelle on va s'intéresser. Cette zone sera choisie par l'utilisateur sous la forme d'une forme géométrique simple (disque, carré…). Elle sera déterminée, soit explicitement, en pré-traitement pour une estimation préalable à un calcul modal, soit implicitement, en post-traitement de vérification d'un calcul modal. *A priori*, cette forme peut être plus compliquée (demi-disque, réunion de formes simples, contour utilisateur…) voire construite en temps réel (méthode de suivi de contour cf. [BP99]). Pour simplifier l'exposé, on se limitera parfois et sans perte de généralité, au cas le plus simple (et effectivement codé dans *Code_Aster*) du disque de rayon R_c centré en Ω_c.

Le point essentiel est de **manipuler cette zone du plan complexe *via* sa frontière** Γ. Celle-ci doit être une **courbe de Jordan**. C'est-à-dire, qu'elle doit être paramétrisable par une fonction $\Gamma:=[\alpha,\beta]\rightarrow C, T\rightarrow z(t)$ et elle ne doit comporter qu'un point multiple aux bornes de l'intervalle: $z(\alpha)=z(\beta)$.

On a maintenant tous les ingrédients pour aborder l'objet de ce paragraphe, la formule intégrale de Cauchy (cf. portrait ci-contre). On considère ainsi, d'une part une fonction analytique (ou holomorphe) $f:C\rightarrow C, z\rightarrow f(z)$, et d'autre part une courbe de Jordan Γ. On rajoute la condition supplémentaire que f ne s'annule pas sur cette courbe. Alors on a le corollaire[16] suivant du «théorème des résidus». Ce corollaire est parfois appelé «**Formule de Cauchy**»:

$$\frac{1}{2i\pi}\int_{\Gamma}\frac{f'(z)}{f(z)}dz= \text{ somme des multiplicités des racines de } f \tag{2.3.1}$$

En appliquant ce corollaire aux polynômes caractéristiques (4.1.1), pour toute courbe de Jordan ne passant pas par le spectre des problèmes modaux considérés $\Gamma\in C$ - $\Delta_{GEP}(\mathbf{A},\mathbf{B})$ (ou $\Delta_{QEP}(\mathbf{A},\mathbf{B},\mathbf{C})$), on a

$$\begin{aligned} \frac{1}{2i\pi}\int_{\Gamma}\frac{P'_{GEP}(z)}{P_{GEP}(z)}dz&=N_{GEP}^{\Gamma}\\ \frac{1}{2i\pi}\int_{\Gamma}\frac{P'_{QEP}(z)}{P_{QEP}(z)}dz&=N_{QEP}^{\Gamma}\end{aligned} \tag{2.3.2}$$

16 En l'appliquant ce dernier à la fonction f'/f.

avec N_{GEP}^{Γ} (respectivement N_{QEP}^{Γ}) le nombre[17] de valeurs propres du GEP (respectivement du QEP) strictement incluses à l'intérieur du contour Γ. Dorénavant, pour simplifier les notations, on notera de manière indifférenciée $P(z)$ le polynôme caractéristique d'un GEP ou d'un QEP.
En remplaçant, dans ces derniers résultats, la courbe de Jordan par sa paramétrisation $z(t)$ et en faisant intervenir la notion de logarithme complexe (appliqué à la fonction $\phi : t \xrightarrow{\phi(t)} (P \circ z)(t)$), il apparaît le résultat :

$$N^{\Gamma} = \frac{1}{2i\pi}\int_{\Gamma}\frac{P'(z)}{P(z)}dz = \frac{\Delta\theta}{2\pi} \qquad (2.3.3)$$

avec $\Delta\theta$ la variation de l'argument de P en suivant le contour Γ.

Eléments de Preuve

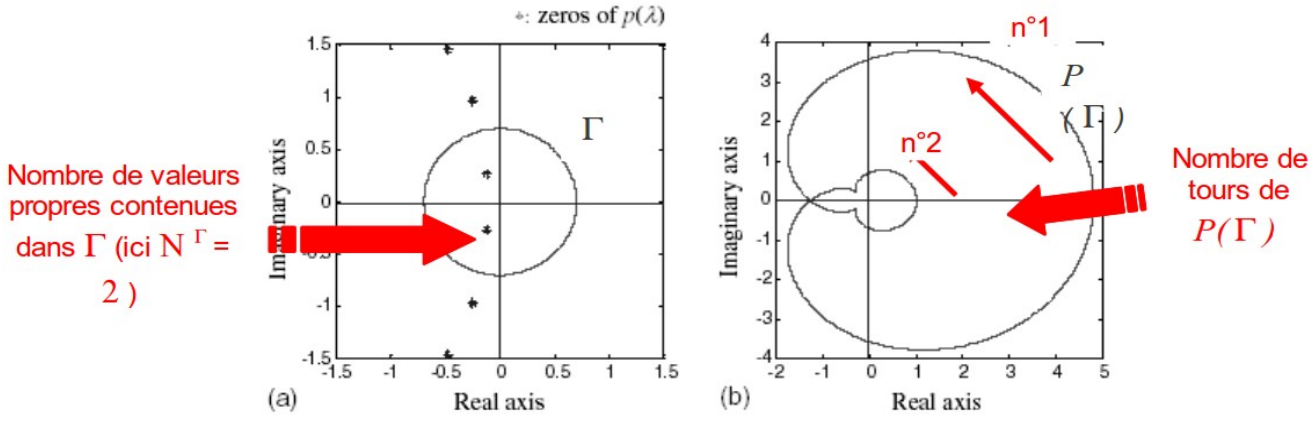
$$\frac{1}{2i\pi}\int_{\Gamma}\frac{P'(z)}{P(z)}dz = \frac{1}{2i\pi}\int_{\alpha}^{\beta}\frac{\frac{d}{dt}(P\circ z)(t)}{(P\circ z)(t)}dt = \frac{1}{2i\pi}\left[\Re e\left[\int_{\alpha}^{\beta}\frac{\phi'(t)}{\phi(t)}dt\right] + i\,\Im m\left[\int_{\alpha}^{\beta}\frac{\phi'(t)}{\phi(t)}dt\right]\right]$$

$$\Rightarrow \quad \frac{1}{2i\pi}\int_{\Gamma}\frac{P'(z)}{P(z)}dz = \frac{1}{2i\pi}\left[(\ln(\phi(\beta)) - \ln(\phi(\alpha))) + i(\arg(\phi(\beta)) - \arg(\phi(\alpha)) + 2k\pi)\right]$$

or le contour est de Jordan donc

$$\Rightarrow N^{\Gamma} = \frac{1}{2i\pi}\int_{\Gamma}\frac{P'(z)}{P(z)}dz = \frac{1}{2i\pi}[i(2k\pi)] = k$$

C'est-à-dire que, lorsque z parcourt le contour Γ, son image $P(z)$ décrit une courbe fermée qui entoure N^{Γ} fois l'origine du plan complexe. **Donc il «suffit» de compter ce nombre de tours pour connaître le nombre de valeurs propres recherchées.**



17 Multiplicités algébriques comprises. C'est-à-dire que, si une valeur propre non déficiente a une multiplicité d'ordre k, elle compte pour k dans ce chiffre.

Figure 2.2 Principe des méthodes de type 'Argument Principal' (extrait de [JKL01]). Ici un disque de contrôle de frontière Γ . Il englobe strictement deux valeurs propres. On vérifie bien que $P(\Gamma)$ décrit une courbe fermée effectuant aussi deux tours (dans le même sens) autour de l'origine.

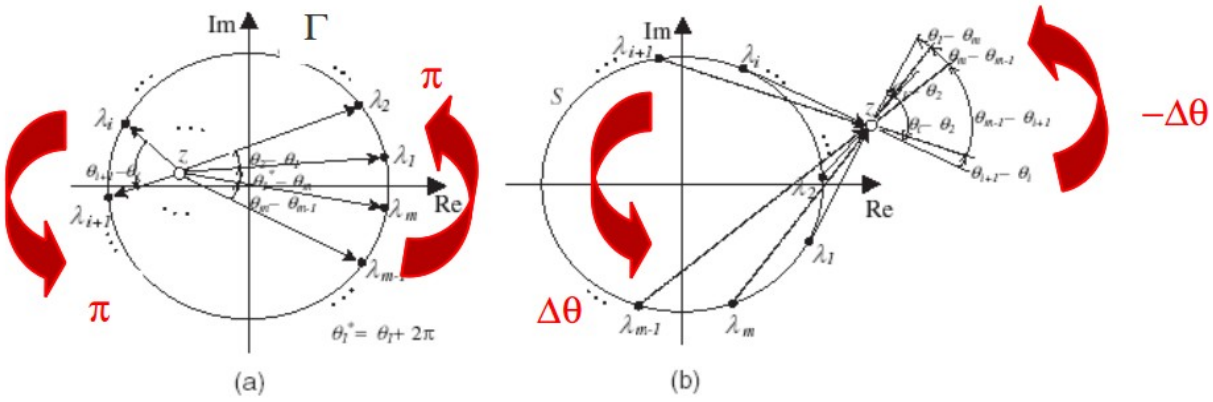


Figure 2.3 Fonctionnement des méthodes de type 'Argument Principal' (extrait de [JHKL03]) décortiqué sur un problème modèle. Il comporte une seule valeur propre simple z et un contour Γ de type cercle. Dans le cas (a), le contour entoure cette valeur propre, dans le cas (b), elle lui est extérieure.

Théoriquement, l'idée est séduisante. Hormis quelques cas particuliers pour lesquels on connaît explicitement le polynôme caractéristique et pour lesquels on sait l'intégrer analytiquement, dans le cas général, il faut procéder numériquement. On commence par discrétiser Γ en k points dits «d'observation»: $(\lambda_j)_{j=1}^k$. En chacun de ces points, on estime θ_j l'argument du complexe $P(\lambda_j)=\rho_j e^{i\theta_j}$ et la somme des écarts des arguments va nous fournir l'entier recherché:

$$N^\Gamma = \frac{\sum_{j=1}^{k}(\theta_{j+1}-\theta_j)}{2\pi} \qquad (2.3.4)$$

Illustrons ce résultat sur le problème modèle de la figure 4.3. On peut écrire en chaque point d'observation λ_j : $P(\lambda_j)=constante\cdot(\lambda_j-z)=r_j e^{i\theta_j}$.
Graphiquement, le cumul algébrique des écarts d'arguments nous donne sur la figure (a)

$$\sum_j(\theta_{j+1}-\theta_j)=2\pi\ ,$$

alors qu'il s'annule sur la figure (b)

$$\sum_j(\theta_{j+1}-\theta_j)=0\ .$$

On retrouve donc bien les $N_a^S=1$ et $N_b^S=0$ prédits par le corollaire de Cauchy.
Donc seuls les arguments des nombres complexes sont utiles ici, puisqu'on cherche uniquement à estimer un nombre de tours. D'où la dénomination **«Méthode de l'Argument Principal» (APM pour 'Argument Principle-based counting Method'),** souvent retenue pour les algorithmes de calcul basés sur cette formule.

Remarques:

- Le nombre de tours est compté de manière algébrique, il faut ne pas oublier de retrancher les tours rétrogrades.
- En pratique, une valeur propre peut être située très près de la frontière : $z\approx\lambda_j$. Dans ce cas, elle devrait compter pour une portion correspondant au ratio angulaire $\frac{mes(\lambda_{j-1}\lambda_j,\lambda_j\lambda_{j+1})}{2\pi}$. Par exemple, si les trois points de discrétisation λ_{j-1}, λ_j et λ_{j+1} se suivent sur une droite et si la valeur propre $z\approx\lambda_j$ est

simple, on ne comptabilisera qu'un demi-tour. Donc en toute rigueur l'entier recherché peut être un réel. Ce cas de figure n'a pas été pris en compte dans l'implantation dans Code_Aster.

- Deux autres problèmes récurrents : l'estimation rigoureuse d'au moins l'argument de $P(\lambda_j)$, ainsi qu'un algorithme de suivi de contour rigoureux (entre Γ et $P(\Gamma)$) pour être sûr de ne pas rater la discrétisation d'un tour. Des éléments de réponse sur ces points sont présentés dans les paragraphes suivant.

2.4 Ingrédient n°4: Méthode de calcul du polynôme caractéristique

L'évaluation du polynôme caractéristique est un ingrédient essentiel des algorithmes de comptage. Pour ce faire, on a vu qu'on pouvait avoir recours à une factorisation $\mathbf{LDL^T}$. Malheureusement, celle-ci est très coûteuse en temps[18] et en mémoire. Pour optimiser ces coûts, on peut donc chercher à calculer une fois pour toute, en début de processus, la décomposition du polynôme

$$\begin{aligned} P_{GEP}(\lambda) &:= det(\mathbf{A}-\lambda\mathbf{B}) := a_0+a_1\lambda+a_2\lambda^2+...+a_n\lambda^n \quad (GEP) \\ P_{QEP}(\lambda) &:= det(\mathbf{A}+\lambda\mathbf{B}+\lambda^2\mathbf{C}) := a_0+a_1\lambda+a_2\lambda^2+...+a_{2n}\lambda^{2n} \quad (QEP) \end{aligned} \tag{2.4.1}$$

Une fois connue la suite finie des nombres complexes $(a_i)_i$, l'évaluation du polynôme en un point de discrétisation du contour n'a plus qu'une complexité en temps et en espace de l'ordre de n.

Remarque:

- *Ce scénario est théoriquement séduisant. Malheureusement, en pratique, même sur des problèmes modèles de très petites tailles, on verra que l'évaluation de ce type de polynôme est souvent source d'erreurs voire problématique (overflow). Et ceci malgré le recours à des algorithmes adaptés de type Horner et/ou via des outils informatiques particuliers (cf. §2.6, multiples précisions, algorithmes compensés….).*

Le premier algorithme permettant d'évaluer les monômes du polynôme caractéristique d'un SEP est dû à l'astronome français Leverrier (1840). Il a été formalisé en notation matricielle par le mathématicien russe Faddeev (1949), d'où la dénomination d'algorithme de «**Faddeev-Leverrier**» souvent retenue pour cette méthode. En un mot, partant d'une matrice carrée $\mathbf{M}$ de taille n

$$\mathbf{M}\,\mathbf{u}=\lambda\,\mathbf{u} \quad (SEP) \tag{2.4.2}$$

elle propose de construire une récurrence matricielle $(\mathbf{M_i})_i$ sous la forme

$$\begin{aligned} \mathbf{M_0} &= \mathbf{M} \\ &\dots \\ \mathbf{M_k} &= \mathbf{M_0}\left(\mathbf{M_{k-1}}-\frac{1}{k}tr(\mathbf{M_{k-1}})\,I_n\right) \end{aligned} \tag{2.4.3}$$

Le polynôme caractéristique peut alors s'écrire sous la forme

$$P_M(\lambda) := det(\mathbf{M}-\lambda\mathbf{I_n}) = \underbrace{1}_{a_n}\cdot\lambda^n - \sum_{k=1}^{n}\underbrace{\frac{1}{k}tr(\mathbf{M_k})}_{a_{n-k}}\lambda^{n-k} \tag{2.4.4}$$

d'où les coefficients $(a_i)_i$ recherchés.

Ce cas de figure s'applique à nos GEPs en posant, si $\mathbf{B}$ est inversible, $\mathbf{M} := \mathbf{B^{-1}A}$.
Cependant cette méthode peut s'avérer instable et très coûteuse. Depuis une trentaine d'années plusieurs **variantes** ont été proposées (Wang & Chen 1982; Ormand, Dean et al 1994), mais la plus aboutie semble être celle **de S.Rombouts & K.Heyde**[RH97].

Elle consiste, tout d'abord, à transformer la matrice initiale $\mathbf{M}$ sous une forme de Hessenberg supérieure[19] : $\bar{\mathbf{M}}$. Comme il s'agit d'une transformation semblable, les deux matrices conservent le même spectre. Puis on s'intéresse à la matrice $\bar{\mathbf{M}}+\lambda\,\mathbf{I_n}$. L'idée de la méthode est alors de renverser les représentations «polynôme/matrice» et de considérer ce polynôme de matrices comme une matrice de polynômes en λ. On obtient alors un processus constructif des coefficients de ce polynôme en le factorisant «formellement» sous forme $\mathbf{LDL^T}$ (avec une factorisation de polynômes et non plus de nombres complexes). En résumé, cela conduit à l'algorithme suivant:

18 En $n^3/3$ pour les matrices pleines, en $n^{\alpha}(largueur\,de\,bande)^{\beta}$ pour les matrices creuses (avec α, β réels inférieurs ou égaux à 2).

19 Matrice triangulaire supérieure avec juste, dans la partie inférieure, la première sous-diagonale.

- Réduire la matrice $\mathbf{M}$ sous forme de Hessenberg supérieure $\Rightarrow \bar{\mathbf{M}}$,
- Initialiser à zéro une nouvelle matrice pleine $\mathbf{B}$ de taille $2n$ $\Rightarrow \mathbf{B}=\mathbf{0}_{2n}$,
- Calculer les termes de $\mathbf{B}$ à partir de ceux de $\bar{\mathbf{M}}$ *via* le processus suivant $\Rightarrow \mathbf{B}$,

Do $j=2n,1,-1$
  Do $i=1,j$
    Do $k=2n-j,1,-1$
      $\mathbf{B}_{k+1,i}=\overline{\mathbf{M}}_{i,j}\mathbf{B}_{k,j+1}-\overline{\mathbf{M}}_{j+1,j}\mathbf{B}_{k,i}$
    Enddo
    $\mathbf{B}_{1,i}=\overline{\mathbf{M}}_{i,j}$
  Enddo
  Do $k=1,2n-j$
    $\mathbf{B}_{k,j}=\mathbf{B}_{k,j}+\mathbf{B}_{k,j+1}$
  Enddo
Enddo

- On revient aux coefficients du polynôme *via* la première colonne de $\mathbf{B}$ $\Rightarrow (a_i)_i$.

$$a_i=(-1)^{2n-i}\mathbf{B}_{2n-i,1}$$

Algorithme 1. Méthode de Rombouts-Heyde de calcul des coefficients du polynôme caractéristique de la matrice carrée $\mathbf{M}$.

Remarque:

- *En plus les problèmes numériques qu'implique la manipulation de termes d'ordres de grandeur très différents, cette méthode présente un inconvénient majeur : la transformation de Hessenberg (souvent dense) de $\mathbf{M}$ et l'allocation d'une matrice pleine dimensionnée à deux fois la taille du problème. Cette complexité mémoire en n^2 bride, pour l'instant, l'usage éventuelle de cette méthode aux problèmes de petite taille ($<10^4$ degrés de liberté) [20] . Toutefois si besoin est, cet inconvénient peut être sans doute partiellement levé en redistribuant différemment la construction des termes de $\mathbf{B}$ et en construisant une factorisation de Hessenberg 'sparse'.*

20 On a déjà ce type de limitation avec la méthode QZ implantée dans CALC_MODES.

2.5 Ingrédient n°5: Méthode de dénombrement des racines d'un polynôme réel

Une fois que l'on connaît les coefficients d'un polynôme, il existe de nombreuses voies pour en déterminer le nombre de zéros: la fastidieuse méthode de Bezout (1762), celle utilisée par Ph. Saux-Picart (1993) basée sur les transformations de Schur-Cohn, celle de Gantmacher (1959) qui utilise un critère de Routh-Hurwitz[21]…

Une des plus abouties semble être la méthode de Gleyse & Moflih[GM99] (INSA Rouen, 1998). Elle est aussi basée sur la matrice de Schur-Cohn $\mathbf{T}$ du polynôme réel considéré

$$P(\lambda) := a_0 + a_1\lambda + a_2\lambda^2 + \cdots + a_n\lambda^n \quad \Rightarrow \quad \mathbf{T} := \begin{matrix} & & & & \mathbf{T}_2 \cdots \\ \mathbf{T}_1 & & & & \\ \end{matrix} \begin{bmatrix} a_n^2 - a_0^2 & a_n a_{n-1} - a_0 a_1 & a_n a_{n-2} - a_0 a_2 & \cdots \\ sym & (a_n^2 - a_0^2 + & (a_{n-1} a_{n-2} - a_1 a_2 + & \cdots \\ & a_{n-1}^2 - a_1^2) & a_n a_{n-1} - a_0 a_1) & \\ sym & sym & (a_n^2 - a_0^2 + & \cdots \\ & & a_{n-1}^2 - a_1^2 + & \\ & & a_{n-2}^2 - a_2^2) & \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{2.5.1}$$

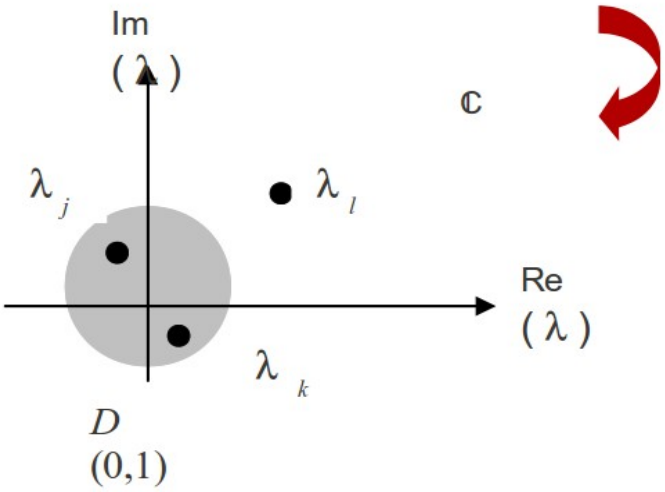
$$\mathbf{T}_{ij} := \sum_{h=1}^{\min(i,j)} \left(a_{n-i+h} a_{n-j+h} - a_{i-h} a_{j-h} \right) \quad (i, j = 1,2 \cdots n)$$

En calculant les mineurs $(d_i)_i$ **de cette matrice** un peu «exotique», dite de **Schur-Cohn**,

$$d_i := det(T_i) \text{ avec } \mathbf{T}_i := \mathbf{T}(1:i, 1:i) \tag{2.5.2}$$

le théorème de Gleyse & Moflih nous assure la connaissance du nombre de racines du polynôme strictement à l'intérieur du disque unité centré à l'origine $N_P^{D(0,1)}$.

En notant $V[1, d_1, d_2, \ldots, d_n]$ le nombre de changements de signe de la séquence finie de nombres réels $1, d_1, d_2, \ldots, d_n$, ce nombre de racines résulte en effet de la formule



21 Ce critère est utilisé pour la stabilité de système dynamique: un polynôme est dit de Hurwitz (mathématicien allemand) si toutes ses racines sont à partie réelle strictement négative.

$$N_P^{D(0,1)} = n - V[1, d_1, d_2, \ldots, d_n] \qquad (2.5.3)$$

Figure 2.4. Formule de Gleyse-Moflih donnant le nombre de racines d'un polynôme réel à l'intérieur du cercle unité centré à l'origine.

En appliquant maintenant ce résultat au cas particulier du polynôme caractéristique du problème modal standard (SEP)

$$\mathbf{Mu} = \lambda \mathbf{u}$$

J.JO et al[JHKL03] **ont publié un critère encore plus simple à mettre en œuvre**. Partant du polynôme caractéristique du SEP, on peut construire sa matrice de Schur-Cohn selon la formule (2.5.1). Puis on la factorise sous la forme $\mathbf{LDL}^{\mathbf{T}}$ et on compte le nombre de termes (strictement[22]) positifs de $\mathbf{D}$. On note par exemple $\sigma(\mathbf{M})$ ce nombre de termes positifs

$$\sigma(\mathbf{M}) = card\{\mathbf{D_{ii}} > 0 / \mathbf{T} = \mathbf{LDL}^{\mathbf{T}}, \mathbf{T} = SC(\mathbf{M})\} \qquad (2.5.4)$$

Alors, le **nombre de racines recherchées** est égale à ce chiffre

$$N_{SEP}^{D(0,1)} = \sigma(\mathbf{M}) \qquad (2.5.5)$$

D'où, pour résumer, le déroulement suivant:

- Construction de la matrice de Schur-Cohn associée au polynôme suivant (2.5.1) $\Rightarrow \mathbf{T}$,
- Factorisation $\mathbf{LDL}^{\mathbf{T}}$ de cette matrice $\mathbf{T} \Rightarrow \mathbf{D}$,
- Comptage du nombre de termes strictement positifs de $\mathbf{D}$ suivant (2.5.5) $\Rightarrow N_{SEP}^{D(0,1)}$.

Algorithme 2 Méthode de Gleyse-Moflih de calcul du nombre de racines d'un polynôme réel appartenant strictement au disque de rayon unité et centré à l'origine.

Remarques:

- Ce résultat est à la fois très rassurant, car il est similaire au critère de Sturm exhumé dans le cas généralisé standard (cf. §2.2) et très déroutant, car les développements théoriques des deux démonstrations n'ont rien à voir !
- Hormis le fait que ce critère requiert la connaissance exacte des coefficients du polynôme caractéristique (donné par exemple par la méthode de Rombouts vue précédemment), il souffre de plusieurs limitations: il ne concerne que les disques centrés à l'origine et de rayon unité, il nécessite une transformation du problème modal initial en un problème standard et, enfin, il nécessite la construction d'une matrice de Schur-Cohn dense et dimensionnée à la taille du problème.
- La restriction au cercle unité peut être facilement levée comme on le verra par la suite. Lorsqu'on s'intéresse à un cercle $D(0,R)$, il suffit d'effectuer un changement de variable dans le développement polynomiale. Malheureusement, le rayon R doit rester très raisonnable ($R<10$ par exemple) sous peine de faire «exploser» les valeurs des plus grands coefficients polynômiaux.
- *Il faut s'assurer, dès le début, que les coefficients polynomiaux du SEP restent réels. Cela peut restreindre la classe d'utilisation de cette méthode (notamment avec un GEP à amortissement hystérétique).*

2.6 Ingrédient n°6: Évaluation polynomiale fiable

Développer des algorithmes numériques fiables pour l'évaluation polynomiale reste un challenge. Surtout lorsqu'on cherche une stratégie avec un coût calcul modéré.

22 Il ne peut pas être nul sinon la matrice est non factorisable car singulière.

$$P_{GEP}(\lambda) \neq a_0 + a_1\lambda + a_2\lambda^2 + ... + a_n\lambda^n \quad (GEP)$$
$$P_{QEP}(\lambda) \neq a_0 + a_1\lambda + a_2\lambda^2 + ... + a_{2n}\lambda^{2n} \quad (QEP) \qquad (2.6.1)$$

Une réponse classique pour palier ce problème est d'augmenter la précision des calculs[23]. Diverses solutions existent:

- Passer les calculs et les variables concernés en **quadruple précision**. Malheureusement cette solution ne suffit pas toujours et elle n'est pas portable.
- Utiliser une **librairie émulant l'arithmétique exacte** sur l'ensemble des entiers et/ou des rationnels. Par exemple la librairie GMP[GMP]. C'est très coûteux en temps et suppose une dépendance logicielle supplémentaire.
- Utiliser **une arithmétique multiprécisions *via* une bibliothèque externe**: MP de P.Brent (1978), ARPREC de H.Bailey (2005) et surtout celle proposée par l'INRIA, MPFR (2000) [MPFR].
- Recourir à des **expansions d'arithmétiques fixes** telles que les doubles-doubles et les quad-doubles qui simulent, respectivement, une précision deux fois ou quatre fois supérieure à la double. Pour ce faire, elles redéfinissent l'arithmétique flottante en somme (non évaluée) de deux ou quatre flottants. Plusieurs bibliothèques proposent ce type d'approche: par exemple, QD de H.Bailey[QD] ou XBLAS de W.Demmel[XB] (pour 'eXtended and mixed precision BLAS').
- Ré-écrire les portions d'algorithmes «critiques» de manière à compenser (jusqu'à un certain seuil) les erreurs d'arrondi. En effet, à l'aide d'algorithmes connus sous le vocable de «**Transformations Exactes**» (ou '**Error-Free Transformation**'), il est possible de calculer l'erreur d'arrondi générée par chaque opération flottante. Ces «**algorithmes compensés**» ont donc pour principe d'adjoindre, à chaque calcul élémentaire effectué (somme, produit…), une estimation fine de l'erreur d'arrondi associée. L'objectif étant d'utiliser ces dernières pour corriger («compenser») le résultat final.

$x = a + b \Rightarrow$
$\text{function}[x,y] = \text{TwoSum}(a,b)$
$x = a + b$
$z = x - a$
$y = (a - (x - z)) + (b - z)$

$x = a \times b \Rightarrow$
$\text{function}[x,y] = \text{TwoProd}(a,b)$
$x = a \times b$
$[a_h, a_l] = \text{Split}(a)$
$[b_h, b_l] = \text{Split}(b)$
$y = (a_l \times b_l) - (((x - a_h \times b_h) - a_l \times b_h) - a_h \times b_l)$

$\text{function}[c_h, c_l] = \text{Split}(c)$
$z = c \times (2^r + 1)$ (souvent on prend $r = 27$)
$c_h = z - (z - c)$
$c_l = c - c_h$

Avec Split, un algorithme de découpe auxiliaire d'un nombre flottant en deux parties (Dekker 1971)

Algorithme 3 Opérateurs de somme et de produit compensés.

C'est cette dernière stratégie qui semble gagner du terrain ces dernières années (éventuellement couplée à un outil qui permet d'analyser le code et de détecter les zones sensibles, cf. outils PRECISE ou CADNA). Elle est en effet **autoportante et ne nécessite pas le recours à une librairie externe**. Elle peut s'avérer aussi **plus performante** à précision fixée. Au prix de l'enrichissement des sources du code étudié de quelques routines dédiées, cette technique se propose de gérer les erreurs d'arrondi aussi finement et plus rapidement que les produits externes pré-cités.

23 Par rapport à la double précision telle que définie par la norme IEEE-754.

De nombreux algorithmes compensés existent de longue date dans la littérature : Kahan 1965, Pichat 1972, Priest 1992, Ogita et al 2005, Graillat/Langlois/Louvet et al 2003. Pour plus de renseignements le lecteur pourra consulter par exemple, les documents suivants [GLL09] ou [ROO08].

Le principe consiste donc à substituer, uniquement dans les zones «critiques du point de vue propagation d'erreurs d'arrondi», aux opérateurs d'addition et de multiplication classiques, des appels du type :
Dans ces nouvelles routines élémentaires, qui portent souvent des appellations normalisées[24], on manipule donc en même temps deux nombres flottants: le résultat classique entaché d'erreurs d'arrondi (ici noté x) et sa compensation (ici notée y).
Ainsi, de proche en proche, on parvient à surcharger le classique algorithme de Ruffini-Horner[25] d'évaluation de polynôme

$$\begin{aligned}&\text{function } x = \text{Horner}(P, \lambda)\\ &\quad x = a_n\\ &\quad for\ i = n - 1 : -1 : 0\\ &\qquad x = (x \times \lambda) + a_i\\ &\quad end\end{aligned}$$

Algorithme 4 Algorithme de Ruffini-Horner standard pour évaluer un polynôme.

Donc les auteurs fournissent (en particuliers les chercheurs français de l'équipe «LIP6/INRIA/Université de Perpignan») toute une famille de routines permettant, de proche en proche, de remonter jusqu'à Horner : `EFTHorner`, `CompHorner`… **Le résultat de l'évaluation «compensée» s'écrit alors**

$$P(\lambda) \simeq Horner(P, \lambda) + CompHorner(P, \lambda) \qquad (2.6.2)$$

Sa précision est le double de la précision usuelle (si le point d'évaluation λ est loin des racines du polynôme) pour un surcoût faible[26] (surtout dans nos applications de dénombrement de valeurs propres où cette étape d'évaluation est marginale).

Pour améliorer cette stratégie, il est possible de la **pratiquer de manière récursive sur K niveaux** (afin d'augmenter la précision d'autant). C'est l'algorithme `CompHornerK` [LL08].

Une autre évolution, qui est cruciale pour notre application, est **l'adaptation de ces techniques aux polynômes complexes**. Des publications récentes évoquent le sujet (par exemple [GM11]) mais n'intègrent pas les travaux récents sur les flottants. Cette technique est enrichie de nouvelles fonctions `TwoSumCplx` et `TwoProdCplx` qui s'appuient bien sûr sur leurs équivalents réels pour traiter, séparément, la partie réelle, la partie imaginaire et leur produit. Même si leur déroulement est (beaucoup) plus laborieux, elles débouchent elles-aussi sur un `CompHornerCplx` deux fois plus précis que la version standard (algorithme n°4). Sa version récursive n'a cependant pas encore été développée, ce qui limite un peu son applicabilité.

Nous avons testé sur des problèmes modèles ces différentes approches. **Nos résultats ont été souvent bons mais jamais suffisants pour faire fonctionner correctement nos calculs de dénombrement utilisant l'approche polynomiale** (APM+Rombouts et «Sturm étendu»). Cependant, l'exemple ci-dessous (cf. figure 2.5) extrait d'une publication de l'équipe française, illustre la nécessité et l'intérêt de l'approche. Il traite d'un polynôme canonique $P(\lambda)$, de très faible taille et dont les monômes sont de faible amplitude par rapport à celui du polynôme caractéristique.

24 Au même type, par exemple, que les appels BLAS ou LAPACK, dont elles s'inspirent.
25 Mathématicien anglais du 19ième siècle. Méthode aussi connue sous le nom de Ruffini ou Ruffini-Horner.
26 Surcoût d'un facteur 11 en théorie mais seulement d'un facteur 2 ou 3 en pratique. Ce type de méthode bénéficie en fait «gratuitement» du parallélisme d'instructions mise en œuvre automatiquement par les compilateurs actuels et des effets caches de la hiérarchie mémoire.

Il est évalué près de ses racines (de bas en haut : $[0.68,1.15]$, $[0.74995,0.75005]$ et $[0.9935,1.0065]$) en utilisant bien sûr son développement polynomial (les coefficients réels a_i pré-calculés).

$$P(\lambda):=(0.75-\lambda)^5(1-\lambda)^{11}=a_0+a_1\lambda+...+a_{16}\lambda^{16}$$

Et pourtant, l'algorithme de Horner «souffre» énormément dans les zones proches des racines (fonction `Horner`, colonne de gauche). Le polynôme paraît n'être même plus dérivable[27] ! L'algorithme de Horner compensé (fonction `CompHorner`, colonne centrale) améliore grandement l'évaluation. Mais c'est surtout la version récursive (fonction `CompHornerK` paramétrée ici avec $K=3$, colonne de droite), malgré son surcoût certain, qui fournit un résultat véritablement robuste (pas d'oscillation), donc exploitable par un processus extérieur.

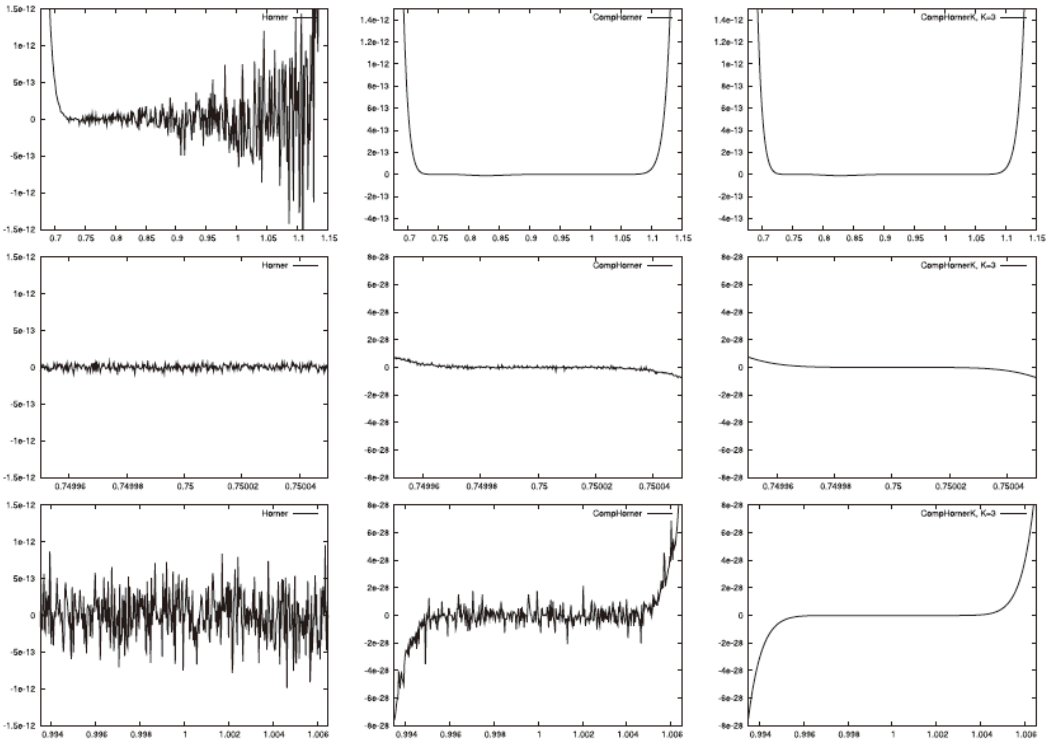


Figure 2.5 Évaluation du polynôme $P(\lambda):=(0.75-\lambda)^5(1-\lambda)^{11}$ dans trois voisinages de ses racines (intervalles $[0.68,1.15]$, $[0.74995,0.75005]$ et $[0.9935,1.0065]$) avec trois méthodes : Horner, Horner compensé et sa version récursive à trois niveaux (extrait de [GLL08]).

Remarque:

- *Ce type de problématique nous paraît complètement sous-estimée dans nos codes (hormis les initiatives récentes autour du produit CADNA cf. travaux de C.Denis/S.Montant [28] EDF R&D/LIP6). Sans doute certaines évaluations de polynômes pourraient bénéficier de ce type de méthodologie: par exemple, lors de l'évaluation de certaines lois de comportement de Code_Aster. Certains logiciels, tel MUMPS, ont déjà pris conscience de ce type de problématique.*

Fort de tout ces ingrédients mathématiques, nous allons maintenant étudier (et mettre en œuvre) différentes stratégies permettant de réaliser notre objectif: compter le nombre de valeurs propres d'un GEP (standard ou atypique) ou d'un QEP appartenant strictement, soit à l'axe réel (cf. §3.2) à un domaine fermé du plan complexe (cf. §3.3/3.4/3.5).

27 On imagine donc le problème que poserait l'évaluation de la dérivée (par différences finies) de ce polynôme pour mettre en œuvre une méthode de dénombrement par quadrature (cf. §5.3).

28 Cf. Symposium SMAI 2011 «Étude de la qualité numérique de codes de calcul industriel».

3 Algorithmes de dénombrement : aspects numériques et choix d'implémentation

3.1 Bibliographie

Concernant l'axe réel (GEPs standards), cette fonctionnalité est disponible de longue date dans le code où elle est connue sous le nom de test de Sturm (cf. §2.2/3.2). Elle est d'ailleurs utilisée par tous les codes commerciaux du domaine (ANSYS, NASTRAN, SAMCEF, LSDYNA…).

Par contre, concernant le dénombrement dans une zone fixée du plan complexe (GEPs atypiques et QEPs), hormis quelques résultats analytiques globaux (type théorème de Gershgorin[HSZ08][Var04]), **il existe peu de papiers proposant des processus algorithmiques adaptés**. D'ailleurs, ces résultats théoriques fournissent une description analytique[29] de la courbe fermée englobant tous le spectre du problème, alors que ce qu'on cherche ici est l'approche duale: considérant une courbe fermée donnée, on souhaite connaître le nombre de valeurs propres qu'elle englobe.

Après une rapide étude bibliographique nous n'avons finalement retenu que trois pistes:

Pour le dénombrement sur l'axe réel:
- La méthode de Sturm standard[Hau80]. D'un point de vue théorique, cette méthode ne fait plus l'objet de travaux depuis une vingtaine d'années. Il a fallu néanmoins l'adapter numériquement (cf. corollaire 3) aux matrices dualisées et aux Lagranges *Aster*.

Pour le dénombrement dans le plan complexe:
- Des travaux très pointus issus de la communauté numéricienne française qui travaille sur les solveurs modaux préconditionnés (algorithme de type Jacobi-Davidson) et les calculs de pseudo-spectre pour les problèmes modaux difficiles. Il s'agit notamment des travaux de **B.Philippe et al**[BP99][KP11] (**INRIA-Rennes**).
- Des heuristiques plus empiriques et plus proches des préoccupations de l'ingénieur, proposées par des chercheurs travaillant pour le constructeur automobile **HYUNDAI**[JKL01/03/08].

Ces deux dernières équipes ne semblent d'ailleurs pas se connaître et leurs études bibliographiques sont assez brèves. Concernant le dénombrement dans le plan complexe, elles soulignent toutes les deux, **la nouveauté et la difficulté de l'approche.** Notamment en terme de robustesse et de coût calcul. A notre connaissance, **aucun code généraliste en mécanique des structures** ne propose ce genre de fonctionnalité.

Pour résumer, on peut regrouper les approches proposées en quatre familles. On rappelle que seules les deux premières sont effectivement disponibles dans *Code_Aster* (cf. §4.1).

Pour le dénombrement sur l'axe réel:
- La méthode de Sturm standard (cf. §2.2/3.2).

Pour le dénombrement dans le plan complexe:
- Les méthodes de type **«Argument Principal»** ('Argument Principle-based Method' ou APM, cf. §3.3) qui se basent sur la formule intégrale de Cauchy appliquée au polynôme caractéristique (cf. §2.3). Cette formule permet de calculer le nombre de valeurs propres en se basant sur le comptage du nombre de tours, autour de l'origine, d'une courbe discrétisée (donc connue plus ou moins finement). Son ingrédient principal est l'estimation du polynôme caractéristique en chaque point de discrétisation du contour $P(\lambda_j)$.
  Pour ce faire on a vu précédemment qu'on dispose de deux variantes: l'une basée sur la factorisation $\mathbf{LDL}^{\mathrm{T}}$ du problème (cf. §2.1, formule (2.1.5)) et l'autre sur la décomposition de Rombouts dudit polynôme (cf. §2.4 algorithme n°1).
- Les méthodes de type «**Formules de Quadrature**» (cf. §3.4) qui cherche à estimer l'intégrale de Cauchy précédente, non pas en comptant des tours, mais en approximant directement l'intégrale.

29 Sous forme d'union de courbes fermées type disque, ellipse...

- Des méthodes plus généralistes de type «**Recherche de Zéros de Polynômes**» («Sturm modifié» ou 'Modified Sturm Sequence Method' cf. §3.5) qui cherche tout d'abord à trouver les coefficients du polynôme caractéristique, afin de lui appliquer un algorithme de recherche de racine. Cette méthode est la combinaison de l'algorithme de Rombouts-Heyde (cf. §2.4 algorithme n°1), pour l'exhumation des coefficients et de la méthode de Gleyse-Moflih (cf. §2.5 algorithme n°2), pour la recherche des racines.

La première famille est disponible de longue date dans le code pour tous les usages se déroulant sur l'axe réel, c'est-à-dire impliquant seulement les GEPs symétriques réels.
Concernant les autres types de problèmes modaux, réquérant forcément un dénombrement dans le plan complexe, nous avons maquetté [Boi11] sur des cas canoniques et des cas-tests *Aster* les deux variantes de type APM, ainsi que la quatrième méthode. De cette campagne de test, il ressort que seule **la variante APM+LDLT semble suffisamment mature (malgré son coût calcul) pour être opérationnelle** dans *Code_Aster*[30]**.**

Remarques:
- Tsai et Chen(1993) ont proposé il y a une quinzaine d'années une extension de la méthode de Sturm mais elle est très difficile à mettre en œuvre sur des QEPs quelconques.
- Hormis les besoins de l'analyse spectrale de certains phénomènes physiques (mécanique vibratoire, hydrodynamique, électromagnétisme…), ce type de technique peut aussi servir à mieux calibrer les préconditionneurs utilisés dans certains algorithmes itératifs: solveurs linéaires, solveurs modaux…

Ces différentes approches sont synthétisées dans le tableau ci-dessous:

Famille de méthode	**Sturm Standard**	**APM ('Argument Principle-based Method')**	**Formule de Quadrature**	**Sturm Modifié ('Modified Sturm Sequence Method')**
Périmètre d'utilisation	Axe réel (GEP symétrique réel)	Plan complexe (GEP et QEP quelconques)		
Stratégie explorée par les équipes	Sans objet	INRIA-Rennes (APM+LDLT), HYUNDAI (APM+LDLT/ Rombouts).	INRIA-Rennes.	HYUNDAI.
Stratégie maquettée et testée dans *Code_Aster*	Oui	Oui (les deux variantes)	Non	Oui
Stratégie retenue dans une fonctionnalité de *Code_Aster*	INFO_MODE, CALC_MODES avec option 'BANDE', CALC_MODES + post-vérification, CALC_MODES avec option 'BANDE' découpée en sous-bandes, CALC_MODES avec option 'AJUSTE' ou 'SEPARE'	APM+LDLT dans INFO_MODE.	Non	
Paragraphes concernés	§2.2	§2.1/2.3/3.3/4	§3.4	§2.4/2.5/2.6/3.5
Avantages	Relativement peu couteuse (deux factorisations LDLT).	Pas de restriction du périmètre d'utilisation (GEP/QEP qcq); Souplesse d'utilisation (contour optimisé ou utilisateur…); Évolutivité (plusieurs	Pas de restriction du périmètre d'utilisation (GEP/QEP qcq); Contrôle intrinsèque à la méthode du niveau de discrétisation; Pic mémoire	Faible coûts calcul (pas de calcul de déterminant $\vartheta(100N)$).

30 On se limitera au début à l'opérateur INFO_MODE.

Famille de méthode	Sturm Standard	APM ('Argument Principle-based Method')	Formule de Quadrature	Sturm Modifié ('Modified Sturm Sequence Method')
		variantes…); Pic mémoire identique à celui de la méthode de Sturm standard.	identique à celui de la méthode de Sturm standard.	
Inconvénients	Périmètre réduit à l'axe réel (GEP symétrique réel). A adapter pour prendre en compte les spécificités des matrices *Aster*: dualisation et variables de Lagrange.	Contrôle de la discrétisation; Évaluation fiable de $Arg(P(\lambda))$; Procédure de comptage du nombre de tours; Coûts en temps calcul[31] ($100\vartheta(N^{\alpha}\xi^{\beta})$); Méthode plus adaptée à un contrôle *a posteriori*.	Prise en compte d'un contour quelconque; Évaluation fiable de $P(\lambda)/P'(\lambda)$; Coûts en temps calcul ($100\vartheta(N^{\alpha}\xi^{\beta})$).	Limité au SEP; Limité au disque centré à l'origine; Manipulation très instable des coefficients polynomiaux; Coûts mémoire très importants (en $\vartheta(N^2)$).

Tableau 3.1 Récapitulatif des procédure de dénombrement.

Nous allons maintenant détailler le fonctionnement des quatre classes de méthodes et leurs liens avec les contingences des calculs modaux *Aster*. Lorsque cela s'avère nécessaire, on fait aussi la jointure avec le paramétrage pré-existant dans les opérateurs modaux.

31 Avec N la taille du problème, ξ sa largeur de bande et α/β deux entiers <2 .

3.2 Méthode de Sturm standard

Dans le cas, le plus courant, des GEPs standards (matrices A et $\mathbf{B}$ réelles et symétriques), les valeurs propres appartiennent à l'axe réel. Le problème du comptage de valeurs propres s'en trouve grandement **simplifié.** On n'a pas à traiter de régionnement du plan complexe et l'on peut s'appuyer sur le corollaire de la loi d'inertie de Sylvester suivant.

Corollaire 1
Soient $\mathbf{A}$ et $\mathbf{B}$ deux matrices réelles symétriques, $\mathbf{B}$ étant de plus définie positive. Le nombre de valeurs propres, strictement inférieure à σ, du problème généralisé $\mathbf{A}\mathbf{u}=\lambda\mathbf{B}\mathbf{u}$ est alors égal au nombre de coefficients diagonaux strictement négatifs de la matrice $\mathbf{D}$ telle que $\mathbf{P}(\mathbf{A}-\sigma\mathbf{B})=\mathbf{L}\mathbf{D}\mathbf{L}^{\mathbf{T}}$ (avec **P** matrice de permutation).

Preuve:
Cf. paragraphe n°1 de l'article de Y. Haugazeau [Hau80].

Par la suite, on appellera position modale de σ et on la notera $pm(\sigma)$, ce nombre de coefficients diagonaux strictement négatifs.

Remarques:
- *Ce corollaire s'étend aux matrices hermitiennes.*
- *Les valeurs propres multiples éventuelles sont comptées avec leur multiplicité.*

Ce corollaire permet donc de déterminer facilement le nombre de valeurs propres contenues dans un intervalle $[\sigma,\mu]$ et la position modale de ces valeurs propres dans le spectre. Il suffit d'effectuer deux décompositions $\mathbf{L}\mathbf{D}\mathbf{L}^{\mathbf{T}}$, celle de $(\mathbf{A}-\sigma\mathbf{B})$ et celle de $(\mathbf{A}-\mu\mathbf{B})$ et de comptabiliser la différence du nombre de termes strictement négatifs entre les deux matrices diagonales. Dans le jargon du code, on désigne[32] ce test sous le vocable de **"Test de Sturm"**.
Il doit cependant être étendu aux formes bien particulières des matrices rencontrées dans *Code_Aster*, et notamment, il faut pouvoir prendre en compte le flambement avec ou sans Lagrange. Pour ce faire, on a **élargi le critère à une matrice $\mathbf{B}$ quelconque** et on l'a **généralisé** en prenant en compte les **matrices dualisées**.

Rappelons que l'on définit habituellement la **signature** d'une matrice (et celle de la forme quadratique associée) comme le triplet d'entiers naturels (r,s,t) où r désigne le nombre de valeurs propres >0, s le nombre de valeurs propres nulles et t celles <0. Ce dernier entier est donc ce que l'on note dans notre cas de figure.

$$pm(\sigma)=t \tag{3.2.1}$$

Propriété 2
Soient $\mathbf{A}$ et $\mathbf{B}$ les deux matrices réelles symétriques (d'ordre n) liées au problème modal généralisé (S) : $\mathbf{A}\mathbf{u}=\lambda\mathbf{B}\mathbf{u}$. Notons $\tilde{\mathbf{A}}$ et $\tilde{\mathbf{B}}$, leurs matrices associées résultant de la double dualisation de p Lagranges permettant de vérifier $\mathbf{C}\mathbf{u}=0$, avec $\mathbf{C}$ matrice réelle de taille $p\times n$.
Alors, $\forall\sigma\in\mathbb{R}$, la signature de $\tilde{\mathbf{A}}-\sigma\tilde{\mathbf{B}}$ s'écrit, en notant $card_{\lambda}[a,b]$ le nombre de valeurs propres du problème généralisé incluses dans l'intervalle $[a,b]$:
si $\mathbf{B}$ est indéfinie et $\mathbf{A}$ est définie positive
alors $card_{\lambda}\{0\}=0$ et

$$\textbf{si } \sigma<0 : \quad \begin{cases} r=card_{\lambda}]-\infty,\sigma[+card_{\lambda}[0,+\infty[\\ s=p+card_{\lambda}\{\sigma\} \\ t=pm(\sigma)=card_{\lambda}]\sigma,0]+p \end{cases} \tag{3.2.2}$$

32 *Stricto sensu*, la dénomination «test de Sturm» est un peu impropre. Ici l'algorithme de Sturm n'est qu'un outil (plus générique) appliqué à une recherche de zéros particulière.

$$\textbf{si } \sigma=0 : \quad \begin{cases} r=card_\lambda]-\infty,+\infty[\\ s=p \\ t=pm(\sigma)=p \end{cases} \tag{3.2.3}$$

$$\textbf{si } \sigma>0 : \quad \begin{cases} r=card_\lambda]-\infty,0[+card_\lambda[\sigma,+\infty[\\ s=card_\lambda\{\sigma\}+p \\ t=pm(\sigma)=card_\lambda[0,\sigma,[+p \end{cases} \tag{3.2.4}$$

si $\mathbf{B}$ est définie positive et si $\mathbf{A}$ est semi-définie positive
alors $card_\lambda]-\infty,0[=0$ et

$$\textbf{si } \sigma=0 : \quad \begin{cases} r=card_\lambda]0,+\infty[\\ s=p+card_\lambda\{0\} \\ t=pm(\sigma)=p \end{cases} \tag{3.2.4}$$

$$\textbf{si } \sigma>0 : \quad \begin{cases} card_\lambda]\sigma,+\infty[\\ s=card_\lambda\{\sigma\}+p \\ t=pm(\sigma)=card_\lambda[0,\sigma,[+p \end{cases} \tag{3.5-5}$$

Preuve:
Cf. Annexe n°1.

D'après le tableau ci-dessous regroupant tous les cas de figure des GEPs standards de *Code_Aster* (cf. [R5.01.01] §3.3),

	Structure libre	Lagranges [33]	Flambement	Fluide-structure
$\mathbf{A}(\mathbf{K})$	≥ 0 et S	<0 ou >0	>0	≥ 0 et S
$\mathbf{B}$ (resp. $\mathbf{M}$ ou $\mathbf{K_g}$)	>0	≥ 0 et S	≤ 0 ou ≥ 0	>0

Tableau 3.2. Propriétés des matrices utilisées dans les GEPs standards du code (on note >0 pour définie positif, ≥ 0 / ≤ 0 pour semi-définie positif/négative et S pour singulier).

on constate que la propriété 2 précédente s'applique à tous les couples de matrices manipulés par *Code_Aster.* On peut donc construire le corollaire suivant.

Corollaire 3 (Sturm étendu théorique)
Dans les configurations matricielles du *Code_Aster*, le nombre de modes propres du problème généralisé (S) dont le vecteur propre vérifie les conditions limites linéaires et dont la valeur propre est contenue dans l'intervalle $]\sigma,\mu[$ est:

Si $\sigma\mu\geq 0$ $\quad \text{card}_\lambda]\sigma,\mu[=pm(\mu)-pm(\sigma)$

Sinon $\quad \text{card}_\lambda]\sigma,\mu[=pm(\mu)+pm(\sigma)-2p$

Si aucune dualisation de Lagranges n'est requise, on pose $p=0$.

Preuve:
Elle est immédiate en combinant les résultats du tableau à ceux de la propriété 2, et, en remarquant que les configurations modales généralisées du *Code_Aster* ne peuvent être que de deux types:

- Flambement: $\mathbf{A}$ est définie positive et $\mathbf{B}$ est indéfinie, le spectre peut être négatif et on utilise les relations de type (1).

33 Cette colonne concerne bien sûr les propriétés des matrices dualisées constituées à partir des matrices initiales.

- Dynamique: $\mathbf{A}$ est semi-définie positive et $\mathbf{B}$ est définie positive, le spectre est positif et on peut utiliser indifféremment les relations de type (1) ou (2).

Bien sûr, si aucune dualisation de Lagranges n'est requise, le même raisonnement s'applique en posant $p=0$, $\tilde{\mathbf{A}}=\mathbf{A}$, $\tilde{\mathbf{B}}=\mathbf{B}$, $\tilde{\mathbf{u}}=\mathbf{u}$.

Pour finir, le principe d'inertie de Sylvester nous assure que la signature de la factorisation $(\tilde{\mathbf{A}}-\sigma\tilde{\mathbf{B}})=\tilde{\mathbf{L}}\tilde{\mathbf{D}}\tilde{\mathbf{L}}^T$ est identique à celle de la matrice shiftée. Malgré cette transformation, les positions modales demeurent donc bien des informations pérennes.

Cependant ce corollaire est tributaire de l'arithmétique exacte, en pratique il faut l'adapter et contrôler son application. Ce test de Sturm est confronté, en arithmétique finie, à deux problèmes concomitants:

- La factorisation de la matrice shiftée lorsque σ est très proche d'une valeur propre,
- Le décompte des termes strictement négatifs de $\tilde{\mathbf{D}}$ pour évaluer la position modale $pm(\sigma)$.

Le premier point nécessite, au préalable, la détermination d'un critère d'appartenance du shift au spectre du problème. Dans *Code_Aster* celui ci est *grosso-modo* (cf. [U2.08.03] §7.1) fondé sur la perte de décimales lors de la factorisation de la matrice dynamique $(\tilde{\mathbf{A}}-\sigma\tilde{\mathbf{B}})=\tilde{\mathbf{L}}\tilde{\mathbf{D}}\tilde{\mathbf{L}}^T$. Plus précisément, σ est considéré comme étant une valeur propre, si lors de la factorisation on perd plus de NPREC[34] décimales. Il faut alors modifier la valeur de σ en la décalant dichotomiquement d'un certain pourcentage (paramétré par PREC_SHIFT) suivant l'algorithme:

Pour $i=1$, NMAX_ITER_SHIFT
  Si perte de plus de NPREC décimales alors
    Si $|\sigma|<$SEUIL_FREQ alors
      $\sigma \Leftarrow \pm$SEUIL_FREQ [35],
      Exit,
    Sinon
      $\sigma \Leftarrow \sigma \pm sign(\sigma)\,max(\text{SEUIL\_FREQ}, 2^{(i-1)}.\text{PREC\_SHIFT}.\sigma)$,
  Sinon
    Exit,
Fin boucle.

Algorithme 5. Procédure de décalage du shift lors de la construction de la matrice dynamique et/ou du calcul du critère de Sturm.

Ce décalage est potentiellement très coûteux en temps puisqu'il coute à chaque fois une factorisation matricielle. Ce surcoût peut donc être de plusieurs dizaines de pourcents du temps total de l'opérateur. Il faut donc veiller à limiter ce type de décalage correctif. Par exemple en:

- N'augmentant pas la valeur par défaut de NMAX_ITER_SHIFT,
- Choisissant une autre option de calcul afin de décaler « fonctionnellement » ce shift (cf. [Boi10c] §3.7) afin qu'il ne coïncide pas « numériquement » avec une valeur propre du problème,
- Voire, débrancher ponctuellement, le test de Sturm de post-vérification (VERI_MODE/STURM='NON').

Si au bout de NMAX_ITER_SHIFT tentatives, la matrice n'est toujours pas numériquement inversible:

- Soit on continue tout de même le calcul avec la dernière valeur shiftée (après émission d'une ALARME). C'est le cas le plus courant où l'on ne cherche que la position modale $pm(\sigma)$ associée à la matrice dynamique. On est alors près à prendre le risque de manipuler une position modale faussée.
- Soit en s'arrête en ERREUR_FATALE, car en plus du test de Sturm on cherche aussi à factoriser la matrice dynamique. Et dans ce cas, la matrice factorisée « quasi-singulière » risque de perturber grandement le calcul de modal qui va suivre (elle sert à construire l'opérateur de travail du solveur modal).

34 Paramètre du mot-clé SOLVEUR[U4.50.01].
35 Le signe dépend de l'option de calcul et de la borne (inférieure ou supérieure).

L'encadré ci-dessous montre la trace de l'algorithme 5 de décalage dans le fichier message.

```
 i =1 | ON MODIFIE LA VALEUR DE DECALAGE DE : 5.00 %
      | LA VALEUR DE DECALAGE DEVIENT :            5.96840E+04
 i =2 | ON MODIFIE LA VALEUR DE DECALAGE DE : 5.00 %
      | LA VALEUR DE DECALAGE DEVIENT :            5.66998E+04
      | ....
```

PREC_SHIFT

nouvelle valeur du shift

Exemple 1. Impressions dans le fichier de message lors d'une procédure de décalage du shift

Dans l'algorithme 5, si $|\sigma| \leq \text{SEUIL\_FREQ}$ alors on impose $\sigma = \pm \text{SEUIL\_FREQ}$. (suivant l'option de calcul et la borne considérée) Ce paramètre SEUIL_FREQ correspond à une valeur seuil en dessous de laquelle on considère qu'on a une valeur propre numériquement nulle (en mécanique classique, cela correspond à un mode de corps rigide). Cette imposition $\sigma = \pm \text{SEUIL\_FREQ}$ permet ainsi de dissocier ce type de modes du reste du spectre et d'éviter des instabilités numériques lors du test de Sturm. Bien sûr, si ce seuillage a lieu, il n'est plus alors question de décaler le shift. Bouger de quelques pourcents une valeur considérée comme nulle n'aurait plus aucun sens !.

Remarques:

- *Les mot-clés précédents sont accessibles à partir du mot-clé facteur* `CALC_FREQ` *et* `VERI_MODE` *des opérateurs modaux.*
- *L'algorithme 5 est étroitement lié au critère de singularité appliqué à la matrice dynamique. Ce critère dépend du paramètre de singularité (*`SOLVEUR/NPREC`*) mais aussi du type de solveur linéaire (*`SOLVEUR/METHODE='MULT_FRONT'`* ou *`'MUMPS'`*) et du paramétrage de celui-ci (notamment pour* `MULT_FRONT` *le paramètre* `RENUM` *et pour* `MUMPS`*,* `RENUM`*,* `PRE/POSTTRAITEMENTS` *et* `ELIM_LAGR2`*). Parfois, un critère de Sturm va donc décaler plusieurs fois la matrice dynamique alors qu'en relachant un peu le paramètre* `NPREC` *(en passant par exemple de 8, la valeur par défaut, à 9) ou en changeant de solveur linéaire, on aurait pu éviter ce coûteux et alarmant correctif. Malheureusement ce type « d'effet seuil » peut difficilement être évité (sans surcoût calcul prohibitif). Il traduit plus une sensibilité du processus aux différents critères numériques qu'une véritable singularité du problème.*
- *Y.Haugazeau [Hau80] propose de traiter plus généralement ces problèmes de pivots numériquement "petits" en construisant une matrice, unitairement semblable*[36] *dont la factorisation présenterait moins d'instabilité.*

En prenant en compte ces éléments et sachant que, numériquement, le décompte des pivots strictement négatifs de la matrice diagonale englobe en fait aussi les éléments (théoriquement) nuls de la signature, on peut réécrire le corollaire précédent. C'est ce critère qui est effectivement codé dans *Code_Aster*.

Corollaire 3bis (Sturm étendu numérique)
Suivant les hypothèses du corollaire 3, on a le critère numérique de comptabilité des modes suivant:

Si $\sigma\,\mu \geq 0$ $\quad \text{card}_\lambda[\sigma, \mu] = pm(\mu) - pm(\sigma)$

si $\sigma\,\mu < 0$ $\quad \text{card}_\lambda[\sigma, \mu] = pm(\mu) + pm(\sigma) - 4\,p$

Preuve (heuristique):
On applique le fait que numériquement l'opérateur de factorisation fournit la "position modale numérique" $pm(\sigma) = s + t$ à la propriété 2 et au corollaire 3. D'autre part, l'implantation du critère ne permet ni la nullité du produit, ni l'estimation de $\text{card}_\lambda\{\sigma\}$.

36 C'est-à-dire semblable, *via* une matrice de passage unitaire, à la matrice shiftée.

3.3 Méthode de type «Argument Principal»

En reprenant les éléments déjà cités (cf. §2.3/3.1), on peut schématiser le fonctionnement de cette méthode sous la forme suivante:

- Choisir la zone du plan complexe et ses dimensions caractéristiques suivant le type de problème modal, le cadre d'utilisation et les besoins de l'utilisateur.

$$\Rightarrow \Gamma$$

- Choisir la discrétisation de la frontière de cette zone (compromis coût calcul/besoins utilisateurs/éventuellement information spectrale préalable)

$$\Rightarrow (\lambda_j)_{j=1}^{k}$$

Si on utilise la variante APM+Rombouts, détermination des coefficients de P *via* l'algorithme n°1.

- En chacun des points de discrétisation, calculer $(P(\lambda_j))_{j=1}^{k}$ puis en extraire l'argument.

$$\Rightarrow (\theta_j)_{j=1}^{k}.$$

Avec la version APM+LDLT ce calcul s'effectue *via* une factorisation $\mathbf{LDL}^{\mathrm{T}}$ (cf. §2.1)

$$\Rightarrow \frac{P(\lambda_j)}{|P(\lambda_j)|}=\prod_{1=1}^{n}\frac{\mathbf{D_{II}}}{|\mathbf{D_{II}}|}$$

Avec la variante APM+Rombouts, on calcule ces valeurs *via* sa décomposition en monômes (+ algorithme de Horner standard ou compensé cf. §2.6)

$$\Rightarrow P(\lambda_j)=a_0+\lambda_j(a_1+a_2(\lambda_j+...+a_n\lambda_j))$$

- Comptage du nombre de tours par un algorithme adapté (éventuellement couplé à une méthode de suivi de frontière). Rajout de points de discrétisation supplémentaires si nécessaire.

$$\Rightarrow \Delta\theta=\sum_{j=1}^{k}(\theta_{j+1}-\theta_j)$$

- Estimation du nombre de valeurs propres: en tenant éventuellement compte des valeurs conjugués et des portions de nombres de tours.

$$\Rightarrow N^{\Gamma}=\frac{\Delta\theta}{2\pi}$$

Algorithme 6 Principe de la méthode APM.

Nous allons maintenant discuter certaines étapes en croisant les éléments piochés dans la littérature et notre petite expérience sur des cas-tests modaux *Aster*.

3.3.1 Choix du domaine de comptage

Les méthodes de type «Argument Principal» partent donc de la discrétisation de la frontière du domaine

$$\Gamma\approx(\lambda_j)_{j=1,k}$$

Le cas le plus simple est celui du disque de rayon R_c centré en Ω_c (cf. figure 3.1ab). Mais pour économiser des ressources calculs on peut s'appuyer sur certaines propriétés du problème modal pour réduire la facture des évaluations de $(P(\lambda_j))_{j=1,k}$. Ainsi lorsqu'on s'attend à des valeurs propres complexes conjuguées $(\lambda,\bar{\lambda})$, un demi-disque centré à l'origine (+ le segment de droite réel correspondant, cf. figure 3.1c) peut suffire. On peut opérer cette stratégie, par exemple, pour QEP sans amortissement hystérétique.
Par contre avec des valeurs propres en couple $(\lambda,-\bar{\lambda})$, c'est le «nœud-papillon» de quarts de cercle qui est optimal (+ les deux segments réel et imaginaire pure correspondant, cf. figure 3.1d).

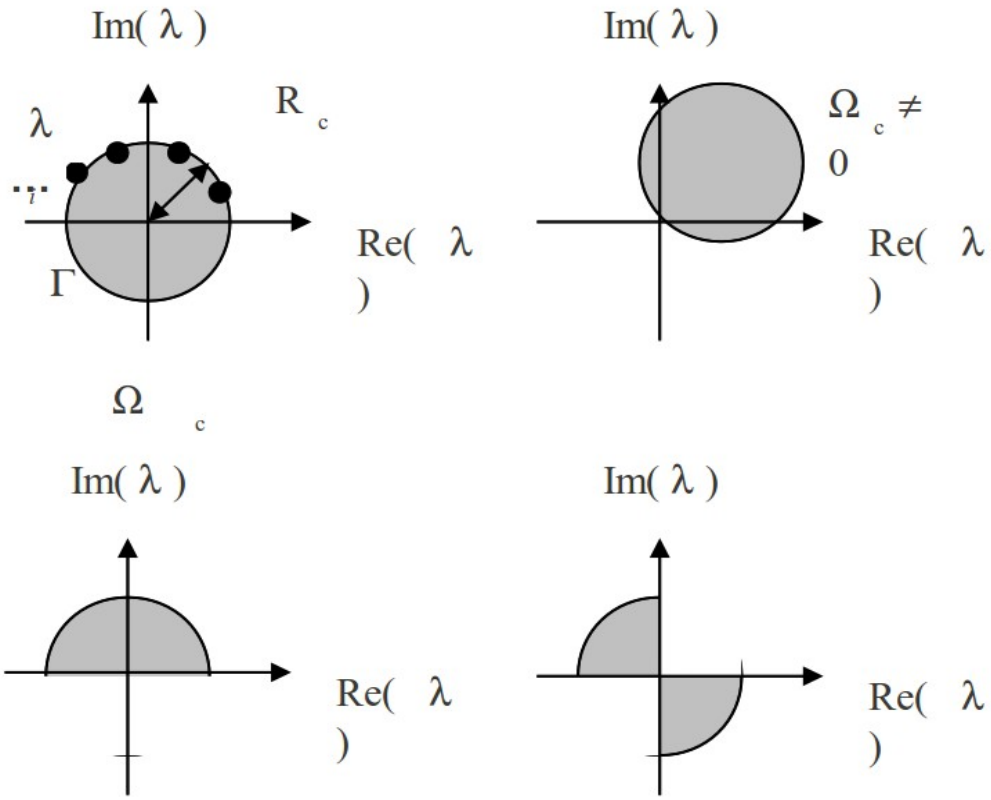


Figure 3.1abcd Différentes formes de domaine de contrôle pour la méthode APM.

De nombreux autres cas de figures optimisés sont envisageables, mais en pratique leur mise en œuvre fragilise l'ensemble. En effet, ils posent de nouveaux problèmes parmi lesquels:

- **Comment traiter les valeurs propres purement réelles ou imaginaires pures** qui accompagnent souvent ces cas de figures ? Comment les exclure proprement de l'évaluation sans risque de rater un couple particulier (par exemple très proche de l'axe réel) ou de les compter partiellement (cf. remarque sur N^{Γ} réel §2.3).

- Il faut passer obligatoirement par les **points singuliers de la frontière** (coins) pour ne pas risquer de rater une évolution drastique du polynôme caractéristique.

- Dans les **QEPs de *Code_Aster***, on commence par trier les valeurs propres calculées en trois groupes: purement réelles, imaginaires conjuguées et imaginaires dépareillées. Ces tris[37], moins simples qu'il y paraît, tiennent compte de nombreux critères (importance relative des parties réelles et imaginaires, critères de seuil…). Pour être cohérent avec l'ensemble du processus modal (et donc ne pas générer de fausses alarmes), l'implémentation de ces domaines optimisés doit tenir compte de ces critères de tri.

- Un domaine optimisé est **forcément centré sur l'axe réel** voire à l'origine. Ce qui exclut d'emblée des solutions parfois plus efficaces basées sur un disque centré en un complexe quelconque.

Remarques:
- Différentes formes de domaine ont été maquettées lors de cette étude. Nous n'avons finalement retenu pour l'industrialisation que la forme générique du disque. Dans la poursuite de ce chantier on prévoit de permettre la prise en compte de «contour utilisateur» particulier (fourni de manière discrétisé) voire le demi-

37 Dans les routines `wp*vect`.

disque ou d'autres formes optimisées (en retravaillant notamment les procédures de tris post-modaux dans Aster).

- Dans une estimation a priori il faut bien sûr fixer la forme du domaine, par contre dans une procédure de contrôle a posteriori, on peut prendre en compte un domaine de forme quelconque entourant au plus près le «nuage» des valeurs propres identifiées. L'équipe de l'IRISA[BP99] a travaillé sur ce type d'approche. Elle a même proposé une «méthode de continuation» qui construit la frontière au fur et à mesure de la détection des modes propres (cf. figure 3.2).
  Cette méthode de type «prédicteur-correcteur» est inspirée des techniques de calcul de pseudospectre de M.Brühl (1996). Cette philosophie de calcul conjoint des modes propres et des ingrédients requis pour leur vérification finale paraît d'ailleurs excellente et très optimale. Elle diminue sensiblement les surcoûts calcul [38] de la méthode de comptage et ces problèmes de robustesse. Cette approche se limite toutefois aux fonctionnalités de post-vérifications (1 des 5 usages pour Code_Aster cf. §1) et sa mise au point requiert un gros investissement en temps. D'autre part, leur approche est fortement intriquée dans un solveur modal particulier (une méthode de Jacobi-Davidson). Un travail algorithmique supplémentaire reste à finaliser pour rendre cette méthode indépendante des solveurs modaux (pour Code_Aster, il faudrait prendre en compte au moins IRAM et Lanczos).

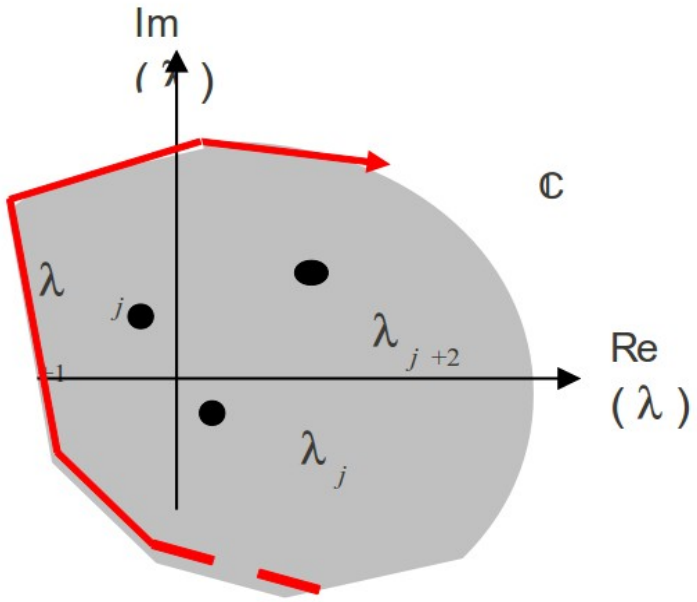


Figure 3.2 Méthode de continuation pour construire, «en temps réel», le domaine «optimisé» (contrôle a posteriori).

Dans la littérature, les tests sont basés sur le disque centré à l'origine $\Omega_c=0$ et ayant pour rayon $R_C=1.005\,\max_i|\lambda_i|$. Il est fixé arbitrairement en se référant au critère conseillé par K.J.Bathe (1996) pour le test de Sturm classique[39]

$$R_C=1.01\,\max_i|\lambda_i| \quad .$$

38 La frontière est potentiellement moins longue donc on diminue le nombre de points de contrôle. D'autre part, la connaissance du spectre en cours de calcul permet de «slalomer» entre les modes afin d'éviter de la faire passer trop près de l'un deux. C'est un peu ce qu'on fait avec le critère de Sturm dans *Code_Aster* lorsqu'on utilise la méthode de Lanczos (`CALC_MODES + SOLVEUR=_F(METHODE='TRI_DIAG)`). On choisit les bornes du segment de contrôle *a posteriori* à mi-chemin des derniers modes calculés et des modes suivant non retenus (cf. [Boi10c] §3.8)

39 Dans *Code_Aster*, ce critère, noté `PREC_SHIFT`, est paramétrable et il est fixé par défaut à 5%: $R_C=1.05\,\max_i|\lambda_i|$.

Concernant ce point, les choix opérés dans les deux fonctionnalités testées sont les suivants. On rappelle que seule la première a été effectivement versée dans le code :

- **Dans INFO_MODE**, l'utilisateur ne peut choisir pour l'instant qu'un domaine de type disque (mot-clé proposé TYPE_CONTOUR='CERCLE') dont il précise le centre Ω_c et le rayon R_c : mot-clés CENTRE/RAYON_CONTOUR. Pour être cohérent avec le paramétrage classique en modal, le rayon ne peut être inférieur à la valeur en deçà de laquelle on considère qu'une valeur propre est un mode rigide: SEUIL_FREQ.

- **Dans CALC_MODES avec OPTION parmi** ['BANDE','CENTRE','PLUS_*','TOUT'], pour le contrôle *a posteriori* du nombre de modes, l'utilisateur ne peut pas choisir son domaine. Celui-ci doit donc être paramétré automatiquement. Par défaut nous nous sommes limités au disque centré en $\Omega_c=0$ et de rayon

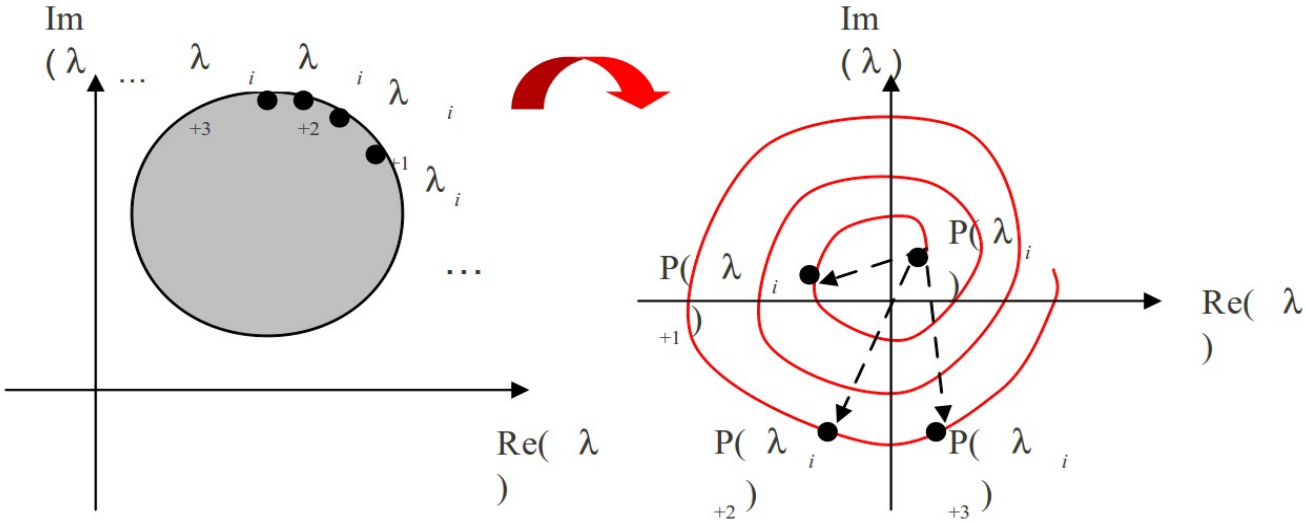
$$R_C=max\left(\text{SEUIL\_FREQ},\underset{i}{max}|\lambda_i|(1+\text{PREC\_SHIFT})\right) \quad (3.3.1)$$

avec SEUIL_FREQ et PREC_SHIFT deux paramètres modifiables par l'utilisateur et de valeurs par défaut, respectivement, 0.01 et 5%. Le premier spécifie la valeur en deçà duquel on considère une valeur propre comme «nulle» et le second précise la marge (en %) qu'on utilise pour distinguer deux modes très proches (en deçà ils sont multiples).

3.3.2 Choix de la discrétisation

Dans la littérature, comme dans nos tests, on s'est limité à une discrétisation uniforme. On pourrait bien sûr envisager des solutions plus «adaptatives» basées, par exemple, sur des critères d'évolution du polynôme caractéristique: si il évolue peu on relâche la discrétisation, sinon, au contraire, on conserve la largeur de pas prédéfinie voire on rajoute de nouveaux points d'observation λ_i.

Ce nombre de points de calcul est bien évidemment crucial. Il conditionne complètement la robustesse de la méthode: si les λ_i ne sont pas assez rapprochés, on risque de rater de l'information (cf. figure 3.3) lors du comptage des tours discrétisés $\theta_j=arg(P(\lambda_j))$. Par contre, cette discrétisation ne doit pas être trop fine, car surtout avec la méthode APM+LDLT, chaque évaluation est assez coûteuse en temps calcul[40].



40 Même si à terme on peut escompter des gains en temps de réponse en parallélisant ces opérations nativement indépendantes.

Figure 3.3 Problématique de la discrétisation du contour afin de capter tous les tours de la courbe fermée. Ici l'arc de cercle $\lambda_{i+1}\lambda_{i+2}$ n'est pas assez discrétisé, on rate un tour ! Donc la vérification sera faussée.

La figure 3.3 illustre ce problème de discrétisation. Tout d'abord ce n'est pas parce que la discrétisation initiale (en λ_i) est régulière que son image dans le plan $P(\lambda)$ l'est. D'autre part, il faudrait idéalement prévoir 2 ou 3 points par tour pour pouvoir les compter «sans risque». C'est pour cela que les auteurs coréens préconisent, lors d'un usage en post-traitement, de discrétiser le contour avec au moins 6 fois le nombre de valeurs propres à contrôler.
En prétraitement, ils ne proposent pas de chiffre et seule l'équipe de l'INRIA s'est attaquée au problème *via* son algorithme de suivi de frontière déjà mentionné. **On choisit de ne pas développer une stratégie aussi sophistiquée, mais de recourir à une heuristique (décrite plus loin) pour résoudre ces problèmes de précision.**

Concernant ce point, les choix opérés dans les deux fonctionnalités testées sont les suivant. On rappelle que seule la première a été effectivement versée dans le code:

- **Dans INFO_MODE,** l'utilisateur choisit la discrétisation qu'il souhaite avec le mot-clé NBPOINT_CONTOUR$=k$. Sa valeur par défaut est empiriquement fixée à 40. L'heuristique de comptage peut amender ce chiffre initial pour s'adapter au problème. Son propre paramétrage sera précisé plus loin.
- **Dans CALC_MODES avec OPTION parmi ['BANDE','CENTRE','PLUS_*','TOUT'],** pour le contrôle *a posteriori* du nombre de modes, ce paramétrage est fait par défaut *via* la formule

$$k=\text{NBPOINT\_CONTOUR}=max(6\underset{i}{card}\{\lambda_i\},20) \tag{3.3.2}$$

Couplé à l'heuristique de comptage, ce paramétrage semble bien fonctionner sur les QEPs de la base de cas-tests.

3.3.3 Calcul du polynôme caractéristique

Dans la littérature, l'équipe coréenne a testée les deux variantes,_ APM+LDLT et APM+Rombouts _, tandis que celle de l'INRIA n'a testé que la première. Nous avons aussi testé les deux variantes. **Celle à base de décomposition de Rombouts fonctionne parfois**, mais comme le signalaient déjà les auteurs, **elle s'avère très instable numériquement.** L'évaluation du polynôme caractéristique *via* ses coefficients est constamment «au bord de l'overflow» ou elle est très faussée par des sommations de termes d'ordre de grandeurs trop différents.
Les algorithmes compensés et la quadruple précision n'y ont rien changé (cf. §2.6). Le recours à une version compensée du traditionnel algorithme de Horner (cf. algorithmes n°3/4) n'a pas suffit à empêcher ces problèmes d'overflow récurrents.

En conclusion, seule la stratégie la plus fiable est accessible à l'utilisateur (malgré son coût):

- **Dans INFO_MODE,** *via* la nouvelle valeur TYPE_MODE='COMPLEXE'+METHODE='APM'.

Remarques:
- Lors de l'évaluation des polynômes caractéristiques des QEP, pour limiter la propagation des erreurs d'arrondis nous avons testé, un calcul en trois étapes de type Horner:

$$\begin{aligned}\mathbf{X}&\leftarrow\lambda\mathbf{C}+\mathbf{B}\\ \mathbf{Y}&\leftarrow\mathbf{A}+\lambda\mathbf{C}\\ P(\lambda)&=det\,\mathbf{Y}\end{aligned} \tag{3.3.3}$$

- *Outre le fait qu'il s'avère un plus coûteux en mémoire [41] , cette précaution n'a pas procuré de gain tangible sur les cas de figures testés. Nous l'avons donc débranchée.*

41 Il faut générer une matrice creuse auxiliaire pour stocker le résultat intermédiaire. Cela ne dimensionne pas le pic mémoire du calcul qui est dû à la taille de la factorisée. Cela ne fait que le décaler.

3.3.4 Heuristique de comptage du nombre de tours

Le fait de transcrire un **problème d'algèbre linéaire** aussi difficile en un simple **problème géométrique** peut sembler être une grande victoire **. En fait, le comptage des tours,** dans le plan $P(\lambda)$, de cette sorte de «coquille d'escargot» **est loin d'être trivial !**
On a déjà souligné les problèmes de discrétisation, le fait que les tours pouvaient s'inverser ou que la forme peut être plus décentrée (type lemniscate[42] ou pétale de fleur). Et c'est sans compter les calculs faussés par une trop grande proximité avec une racine. Les exemples de la figure 3.4 illustrent le problème.

Les contours théoriques effectués sur un même calcul modal du cas-test *Aster* SDLL123a illustrent cette problématique. Entre les deux contours, seul le diamètre du domaine de contrôle initial change. Pourtant les formes sont très différentes: dans le premier, les rayons sont concentriques tous dans le même sens avec quelques ratés près du bord droit dus à la proximité d'une valeur propre; tandis que dans le second, le contour change de sens en cours de trajet. On peut conclure cet exemple par un petit test. Combien de tours fallait il compter dans les deux cas ? Réponse[43].

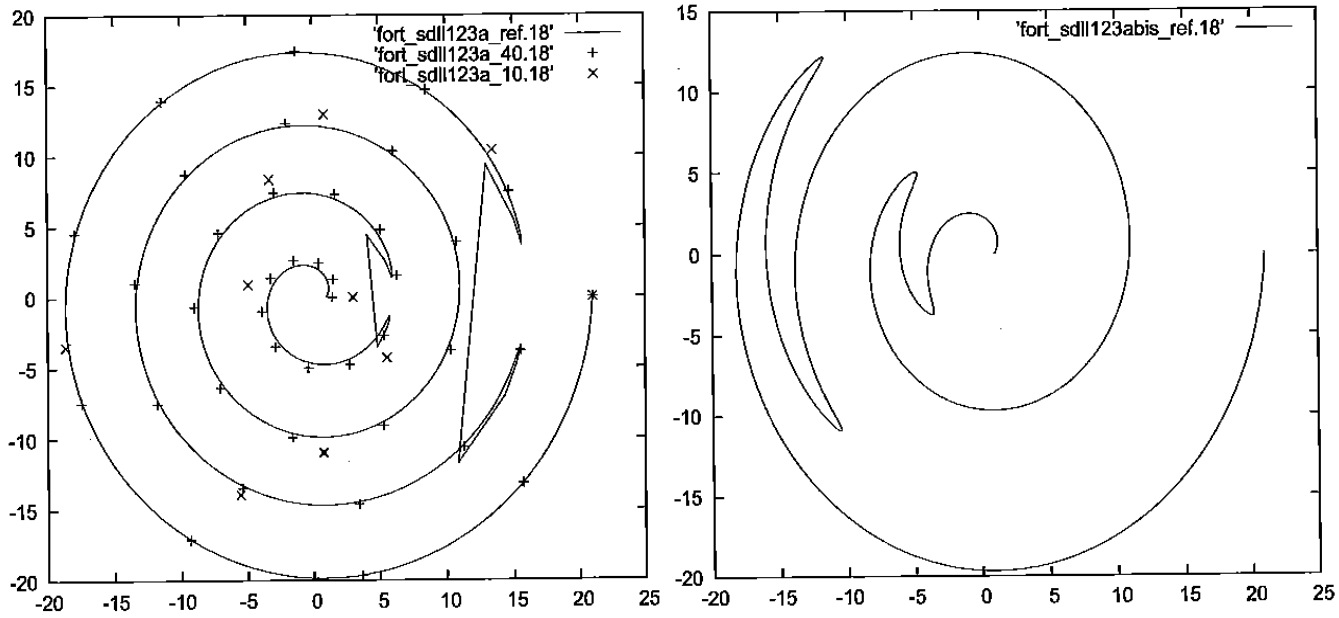


Figure 3.4ab Contours dans le plan $P(\lambda)$ sur le même calcul modal Aster: QEP du cas-test sdll123a.

Seule l'information angulaire est ici pertinente, les modules des points ont été fixés arbitrairement de manière à rendre la courbe lisible.

Des deux équipes qui ont travaillé sur le sujet, seule l'équipe INRIA propose un algorithme de comptage. L'équipe coréenne ne présente pas d'heuristique particulière et semble presque se placer, au vu des résultats présentés dans ses papiers, dans une démarche «semi-automatique» qui requiert à la fois interprétation de l'utilisateur et plusieurs runs type «essai et erreur». **Car la combinatoire des cas de figures est importante, et, sans démarche en amont de type «suivi de contour», ce type d'heuristique paraît peu fiable.**
La principale difficulté vient, qu'en fait, on n'estime pas directement les arguments θ_j du polynôme caractéristique aux points de contrôle, mais leurs valeurs principales $\tilde{\theta}_j$ ($\in[0,2\pi]$). On a donc une information *modulo* 2π du type

$$\theta_j = arg(P(\lambda_j)) = \tilde{\theta}_j + 2n_j\pi \qquad (3.3.4)$$

42 Courbe plane ovalisée ayant une forme de 8.
43 Réponses: 4 et 2.

où n_j est un entier. Ce qui est gênant… lorsque ce sont justement les écarts de nombre de tours qu'on cherche à évaluer !

Certains misent donc sur la chance et considèrent qu'en discrétisant suffisamment le contour initial, on devrait obtenir empiriquement au moins 2 ou 3 évaluations θ_j par tours de $z \to P(z)$ et donc ne pas rater de tour ! C'est-à-dire, qu'au pire, $n_{j+1} = n_j + 1$.

Les chercheurs de l'INRIA sont plus prudents et ils proposent, comme on l'a déjà mentionné, une méthode de suivi de contour en amont pour s'assurer que l'écart d'argument ne va pas déraper ! Il propose un critère du type

$$|\theta_{j+1} - \theta_j| < \pi \tag{3.3.5}$$

Car sur ce critère ils battissent un algorithme de dénombrement exhaustif et fiable. En effet, si la discrétisation des points de contrôle vérifie cette contrainte, on sait alors déduire, de l'estimation de $\tilde{\theta}_j$, $\tilde{\theta}_{j+1}$ et n_j, le n_{j+1} recherché

$$\begin{aligned}
&\theta_j = \arg(P(\lambda_j)) = \tilde{\theta}_j + 2n_j\pi \\
&\theta_{j+1} = \arg(P(\lambda_{j+1})) = \tilde{\theta}_{j+1} + 2n_{j+1}\pi \\
&\text{si } \tilde{\theta}_j < \pi \text{ alors } n_{j+1} = n_j \text{ , sauf si } \tilde{\theta}_{j+1} > \tilde{\theta}_j + \pi \text{ alors } n_{j+1} = n_j - 1 \\
&\text{si } \tilde{\theta}_j > \pi \text{ alors } n_{j+1} = n_j \text{ , sauf si } \tilde{\theta}_{j+1} < \tilde{\theta}_j - \pi \text{ alors } n_{j+1} = n_j + 1
\end{aligned} \tag{3.3.6}$$

Car c'est la somme des écarts de ce nombre de tours n_j qui est l'entier recherché.
En effet, si on reporte la décomposition en argument principal dans la formule de Cauchy (2.3.3), tous ces arguments principaux s'éliminent deux à deux ; Y compris ceux des extrémités, car elles constituent un point double (par hypothèse).

$$N^\Gamma = \frac{\Delta\theta}{2\pi} = \frac{1}{2\pi}(\tilde{\theta}_k + 2n_k\pi - \tilde{\theta}_{k-1} - 2n_{k-1}\pi + \tilde{\theta}_{k-2} + 2n_{k-2}\pi - \ldots - \tilde{\theta}_1 - 2n_1\pi) = (n_k - n_{k-1} + n_{k-2} - \ldots - n_1) \tag{3.3.7}$$

D'où l'algorithme proposé par l'INRIA (légèrement modifié/corrigé par nos soins) :

- Si $|\tilde{\theta}_j - 2\pi| < \varepsilon$ alors $\tilde{\theta}_j = \varepsilon$ (correction[44] si on est très proche des bornes),
- Si $\tilde{\theta}_j < \pi$ et alors $\left[(\tilde{\theta}_{j+1} > \tilde{\theta}_j + \pi) \text{ ou } (|\tilde{\theta}_{j+1}| \le \varepsilon) \text{ et } (j > 2)\right]$
  $n_{j+1} = n_j - 1$ (on recule d'un tour)
- Si $\tilde{\theta}_j \ge \pi$ et $\left[(\tilde{\theta}_{j+1} \le \tilde{\theta}_j + \pi) \text{ ou } (|\tilde{\theta}_{j+1}| \le \varepsilon)\right]$ alors
  $n_{j+1} = n_j + 1$ (on avance d'un tour)

Algorithme 6 Algorithme de comptage de nombre de tours, étant connue une suite discrétisée d'arguments $(\theta_j)_{j=1}^k$ censée vérifier le critère (3.3.5).

On retrouve dans l'algorithme la valeur $\varepsilon =$ SEUIL_FREQ déjà rencontrée pour paramétrer le «zéro modal» .

Concernant ce point, **dans les sources restituables dans *Code_Aster*** nous n'avons pas implémenté de méthode de suivi de frontière. **Nous n'avons donc aucun critère de sauvegarde du type (3.3.5)** nous assurant que l'algorithme n°4 fonctionne à tous les coups ! Pour néanmoins **proposer une solution fonctionnelle, nous avons mis en place une heuristique, certes naïve et coûteuse, mais validée et plutôt robuste**. Du moins si la discrétisation initiale sur laquelle elle se base est assez précise.

44 On fait la même chose avec les modes rigides dans CALC_MODES avec OPTION parmi ['BANDE','CENTRE','PLUS_*','TOUT']. Si ils sont inférieurs à une valeur paramétrée par SEUIL_FREQ, on leur impose d'être égale à l'opposé de cette valeur. Cela permet d'éviter des instabilités numériques et de moduler la notion de valeur propre «nulle» suivant le contexte.

Elle se décompose comme suit. Dans une boucle, on va effectuer trois discrétisations emboîtées[45] de points de contrôle :

1) La première avec deux fois moins de points de contrôle que spécifié par l'utilisateur (dans INFO_MODE) ou induit par le résultat du calcul (en post-traitement de CALC_MODES avec OPTION parmi ['BANDE','CENTRE','PLUS_*','TOUT']) ⇒ $k_1=$ NBPOINT_CONTOUR/2
2) La seconde avec la discrétisation prévue ⇒ $k_2=$ NBPOINT_CONTOUR,
3) La troisième deux fois plus fine que celle prévue ⇒ $k_3=2\times$ NBPOINT_CONTOUR.

Pour chacune de ces discrétisations on calcule l'estimation associée du nombre de tours (grâce notamment à l'algorithme précédent) ⇒ N_1^Γ, N_2^Γ et N_3^Γ.
Si ces trois chiffres sont licites (entiers positifs ou nuls) et égaux, on considère que le processus a convergé et que son résultat est $N^\Gamma=N_3^\Gamma$.
Si ces trois chiffres sont différents, on construit une nouvelle discrétisation suivant le même procédé dichotomique

$$k_4=2\times k_3 \tag{3.3.8}$$

On refait une estimation du nombre de tours pour cette nouvelle discrétisation, la plus fine. Et on se repose la question précédente. On itère ce processus un nombre de fois donné paramétrable par le mot-clé NMAX_ITER_CONTOUR. Après moult expérimentations, on l'a initialisé à 3 (dans INFO_MODE et CALC_MODES avec OPTION parmi ['BANDE','CENTRE','PLUS_*','TOUT']).

D'où simplement l'algorithme:

- Phase d'initialisation:
  Construction des discrétisations emboîtées ⇒
  $k_1=$ NBPOINT_CONTOUR/2 ,
  $k_2=$ NBPOINT_CONTOUR ,
  $k_3=2\times$ NBPOINT_CONTOUR.
  Calcul des nombres de tours associés (suivant algorithme n°4)
  ⇒ N_1^Γ, N_2^Γ et N_3^Γ
- Boucle de l'heuristique $l=1,$ NMAX_ITER_CONTOUR

  Si $N_l^\Gamma=N_{l+1}^\Gamma=N_{l+2}^\Gamma$ alors convergence ⇒ $N^\Gamma=N_{l+2}^\Gamma$
  Sinon
  Si $l=$ NMAX_ITER_CONTOUR on s'arrête en ERREUR_FATALE .
  Construction de la prochaine discrétisations ⇒ $k_{l+3}=2\,k_{l+2}$
  Calcul des nombres de tours associés (suivant algorithme n°4)
  ⇒ N_{l+3}^Γ
  Fin Boucle

Algorithme 7 : Algorithme complet de comptage de nombre de tours implanté dans Code_Aster.

45 D'autres choix étaient possibles: discrétisation décalée, discrétisation adaptative…

3.4 Méthode de type formule de quadrature

Ces méthodes s'appuient toujours sur la formule de Cauchy (2.3.3)

$$N^{\Gamma}=\frac{1}{2i\pi}\int_{\Gamma}\frac{P'(z)}{P(z)}dz=\frac{\Delta\theta}{2\pi} \tag{3.4.1}$$

Elles **cherchent à calculer** N^{Γ}, non pas comme pour les méthodes APM en s'appuyant sur les écarts d'arguments $\frac{\Delta\theta}{2\pi}$ du polynôme caractéristique, mais en **approximant l'intégrale par une formule de quadrature** (appliquée à la fonction $\varphi : t \overset{\varphi(t)}{\rightarrow} (P\circ z)(t)$),

$$\frac{1}{2i\pi}\int_{\Gamma}\frac{P'(z)}{P(z)}dz=\frac{1}{2i\pi}\int_{\alpha}^{\beta}\frac{\varphi'(t)}{\varphi(t)}dt\approx\frac{1}{2i\pi}\sum_{j=1}^{k}a_j\frac{\varphi'(t_j)}{\varphi(t_j)} \tag{3.4.2}$$

Par exemple, pour la **méthode des trapèzes à pas constant**, une fois déterminée la suite de points d'intégration $(t_j)_j$, le calcul s'écrit

$$N^{\Gamma}\approx h\left[\frac{1}{2}\varphi(t_1=\alpha)+\varphi(t_2)+\ldots+\varphi(t_{k-1})+\frac{1}{2}\varphi(t_k=\beta)\right] \tag{3.4.3}$$

De nombreux schémas de quadrature sont possibles. L'équipe de l'INRIA[BP99] a notamment testé:

- La méthode des trapèzes à pas constant,
- La méthode des trapèzes à pas variable,
- La méthode d'Adams (cf. portrait ci-contre).

Elles semblent donner autant satisfaction que la méthode APM+LDLT (avec suivi de frontière) mais elles souffrent des mêmes maux:

- Coût calcul des nombreux points d'intégration (typiquement des centaines) qui requièrent autant de factorisation $\mathbf{LDL}^{\mathbf{T}}$,
- Défaut de robustesse lorsque le contour passe «trop près» d'une valeur propre et/ou comporte des points singuliers (coin, arête vive),
- Meilleur comportement lorsqu'on injecte de l'information spectrale dans le processus (donc plus adapté à un contrôle *a posteriori*).

Par contre se rajoutent d'autres difficultés qui nous semblent rédhibitoires dans une première approche:

- Paramétrage pointus des schémas d'intégration qui impose une adaptation au cas par cas,
- Le surcoût et les problèmes de robustesse qu'impose le calcul de la dérivée de $P(z)$.

Ce dernier élément est un inconvénient majeur de la méthode. **L'évaluation fiable du polynôme caractéristique pose déjà assez de problèmes, nous n'avons pas souhaité affronter, en plus, ceux qu'implique le calcul de sa dérivée.**

Un autre élément a corroboré notre décision de ne pas sélectionner cette méthode dans une première approche. C'est le fait que l'équipe coréenne n'a pas retenu, ni même cité, cette option. Cette stratégie pourrait néanmoins faire l'objet d'une évaluation future par le biais d'une collaboration ou d'un stage.

3.5 Méthode de type Recherche de zéros d'un polynôme

Cette méthode, appelée **«Sturm modifié»** ou **'Modified Sturm Sequence Method'** par ses auteurs [JHHL03], procède en deux étapes:

- Détermination des coefficients du polynôme caractéristique *via* **l'algorithme de Rombouts-Heyde** (cf. §2.4 algorithme n°1),
- Application à ce polynôme particulier de l'**algorithme** de recherche de zéros de **Gleyse-Moflih** (cf. §2.5, algorithme n°2).

Cependant **cette combinaison**, si elle peut s'avérer probante pour tester quelques problèmes canoniques, **n'est pas suffisante pour traiter tous les problèmes modaux en mécanique des structures**. En particulier pour prendre en compte les Lagranges couramment utilisés dans les modélisations *Aster* ou pour compter les valeurs propres de module supérieure à l'unité.
Il faut donc lui rajouter aussi deux étapes. Celles-ci vont essayer de s'affranchir des limitations des deux algorithmes combinés:

- La transformation du **problème modal initial** (GEP ou QEP) en **un SEP de spectre identique** (au moins à l'intérieur du contour), pour pouvoir appliquer l'algorithme n°1.
- La **dilatation du contour initial** de l'algorithme n°2 (disque unité centré à l'origine) pour prendre en compte, au moins, les contours de type cercle centré à l'origine et de rayon quelconque.

Comme pour la variante APM+Rombouts, nous avons aussi essayé d'utiliser l'algorithmique compensée (cf. §2.6) pour prévenir les «crashs algorithmiques» dus principalement aux overflows lors des manipulations de monômes. Mais cela n'a pas suffit à résoudre tous les problèmes.
Les auteurs coréens avaient déjà souligné ce type de problème sans toutefois proposer explicitement de solution concrète.

3.5.1 Transformation sous forme d'un SEP

Premièrement, pour mettre en œuvre la première étape, il faut pouvoir transformer son problème modal initial (GEP ou QEP) en un SEP du type

$$\mathbf{M}\,\mathbf{u}=\lambda\,\mathbf{u}\quad(SEP) \tag{3.5.1}$$

Et bien sûr on souhaite que le spectre de ce nouveau problème modal de travail soit identique à celui du problème initial. **On ne peut donc pas utiliser toutes les astuces mises en œuvre lors des calculs modaux proprement dit pour transformer les problèmes initiaux en problèmes modaux standards**. Par exemple, il n'est pas question ici d'avoir recours à un shift σ pour pouvoir inverser le système et ainsi le transformer de GEP en SEP (technique dite de «shift and invert» cf. [Boic] §3.7):

$$\mathbf{A}\,\mathbf{u}=\lambda\,\mathbf{B}\,\mathbf{u}\quad(GEP)\Rightarrow\underbrace{(\mathbf{A}-\sigma\,\mathbf{B})^{-1}\mathbf{B}}_{M}\mathbf{u}=\underbrace{\frac{1}{\lambda-\sigma}}_{\mu}\mathbf{u}\quad(SEP) \tag{3.5.2}$$

Car nous sommes intéressés par le dénombrement des λ_i à l'intérieur d'un contour et non pas par celui des μ_i. Les techniques de linéarisation (permettant de passer d'un QEP à un GEP) et les transformations spectrales (pour passer d'un GEP au SEP final) sont donc à utiliser avec discernement pour cette problématique spécifique.

Cependant pour traiter **les GEPs de *Code_Aster*, il faut s'affranchir du fait que la matrice de masse $\mathbf{B}=\mathbf{M}$ n'est souvent pas inversible**. C'est dû aux blocages imposés par des dualisations et aux nouvelles variables de Lagranges associées. Les auteurs coréens qui promeuvent cette méthode n'avaient pas à affronter ce type de contingence. Leurs matrices de masse ne sont pas dualisées pour prendre en compte les Lagranges, elles sont donc inversibles.

Pour pouvoir construire un SEP qui ne change pas le spectre du GEP initial

$$\mathbf{A}\,\mathbf{u}=\lambda\,\mathbf{B}\,\mathbf{u}\quad(GEP)\Rightarrow\underbrace{\tilde{\mathbf{B}}^{-1}\tilde{\mathbf{A}}}_{M}\mathbf{u}=\lambda\,\mathbf{u}\quad(SEP) \tag{3.5.3}$$

on doit donc modifier lesdites matrices $\mathbf{A}/\mathbf{B}$ en $\tilde{\mathbf{A}}/\tilde{\mathbf{B}}$. Une modification que l'on a proposée consiste à «bluffer» le processus et à faire en sorte que les inconnues qui ne sont pas «dynamiquement actives» (degrés de liberté de lagranges et degrés de liberté physiques bloqués) ne participent pas au calcul de dénombrement de valeurs propres.

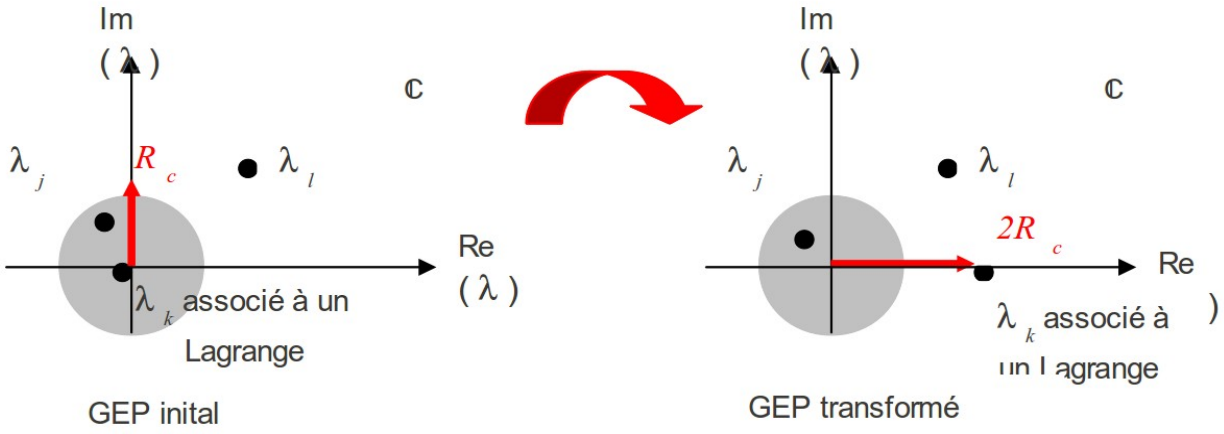


Figure 3.5 Transformation du GEP Aster initial en un GEP compatible avec l'algorithme de Rombouts-Heyde.

Pour mettre en œuvre cette stratégie, on peut utiliser l'astuce suivante:

- «Rendre diagonales» les lignes et colonnes de $\mathbf{A}$ et $\mathbf{B}$ correspondant à ces degrés de liberté i_0
- Imposer des valeurs diagonales $A_{i_0 i_0}$ et $B_{i_0 i_0}$ telles que les valeurs propres associées à ces degrés de liberté soient repoussées en dehors de la zone de contrôle.

$$(GEP)\Rightarrow \underbrace{\begin{pmatrix} \bar{A} & (0_{i_0\cdot})^T \\ 0_{i_0\cdot} & A_{i_0 i_0} \end{pmatrix}}_{\tilde{A}} \mathbf{u} = \lambda \underbrace{\begin{pmatrix} \bar{B} & (0_{i_0\cdot})^T \\ 0_{i_0\cdot} & B_{i_0 i_0} \end{pmatrix}}_{\tilde{B}} \mathbf{u} \quad (GEP\ modifié)\Rightarrow (SEP) \tag{3.5.4}$$

Au final, ce processus permet de construire les matrices $\tilde{\mathbf{A}}/\tilde{\mathbf{B}}$ recherchées: inversibles et ne modifiant pas le spectre (au moins dans la zone d'intérêt).

Dans nos tests, nous avons utilisé le contour le plus simple: un cercle de rayon R_c centré en $\Omega_c = 0$. Pour gérer ce problème, il a donc suffit d'imposer pour chacun des degrés de liberté «dynamiquement passifs» d'exciter la valeur propre $\lambda_{i_0} = 2R_C + i\cdot 0$ (*via* par exemple $A_{i_0 i_0} = 2R_C$ et $B_{i_0 i_0} = 1$, cf. figure 3.5). Bien sûr, n'importe laquelle des valeurs $\lambda_{i_0} = 2a_{i_0} + i b_{i_0}$ avec $|\lambda_{i_0}| = \sqrt{(a_{i_0})^2 + (b_{i_0})^2} > R_C + eps$ et $eps = 5\%$ de R_c par exemple, aurait fonctionné.

Cette stratégie un peu «bricolée» pose néanmoins différents problèmes:

- Les termes diagonaux substitués ont des ordres de grandeurs parfois très différents des autres termes matriciels. Cela peut poser des problèmes numériques dans la suite du processus. En particulier, avec le choix mentionné ci-dessus, **ils ne tiennent pas compte du coefficient de «mise à l'échelle»** ('scaling' en anglais) qu'on impose aux degrés de liberté de Lagranges en fin d'assemblage matriciel. Et il est difficile de maintenir l'objectif principal de filtrage des valeurs propres associées à ces ddls, tout en respectant les scalings des deux matrices.

- Si la situation des Lagranges de blocage simple (Dirichlet classique) semble bien comprise, **que faire pour ceux associés à des liaisons entre degrés de liberté physiques** (Dirichlet généralisé) ?

Remarque:

- *Dans les opérateurs modaux de Code_Aster, on rencontre ce type de problème lorsqu'on cherche à inverser la matrice de masse pour traiter les QEPs (`CALC_MODES + SOLVEUR_MODAL=_F(METHODE='TRI_DIAG' ou 'SORENSEN')`). On y répond en «régularisant» formellement la matrice, c'est-à-dire en annulant les composantes des vecteurs associées aux Lagranges lors des produits matrice-vecteur. Malheureusement cette astuce suffisante dans les algorithmes itératifs (qui ne requièrent que des produits matrice/vecteur) n'est pas transposable ici.*

3.5.2 Adaptation au cercle de centre l'origine et de rayon quelconque

La méthode de dénombrement de Gleyse-Moflih est très intéressante mais **elle souffre d'une limitation fonctionnelle : elle ne traite que le cercle unité centré à l'origine** $(Rc=1,\Omega_c=0)$. Sans rentrer dans ses arcanes numériques, la première extrapolation que l'on peut faire pour étendre son périmètre d'utilisation, consiste à lui permettre de traiter le même contour mais de rayon quelconque rayon R_c. Pour ce faire, il suffit de substituer dans la formule usuelle du polynôme caractéristique (ici une GEP)

$$P(\lambda):=a_0+a_1\lambda+a_2\lambda^2+...+a_n\lambda^n \tag{3.5.5}$$

à la variable λ, la nouvelle variable $\tilde{\lambda}$ telle que $\lambda:=R_c\tilde{\lambda}$. D'où la nouvelle formulation

$$P(\tilde{\lambda}):=\underbrace{a_0}_{\tilde{a}_0}+\underbrace{a_1 R_C}_{\tilde{a}_1}\tilde{\lambda}+\underbrace{a_2 R_C^2}_{\tilde{a}_2}\tilde{\lambda}^2+...+\underbrace{a_n (R_C)^n}_{\tilde{a}_n}\tilde{\lambda}^n \tag{3.5.6}$$

Ainsi, par ce changement de variable, le nouveau polynôme de travail $\tilde{P}(\tilde{\lambda})$ répond aux prérogatives de la méthode :

$$0<|\lambda|<R_C \Leftrightarrow 0<|\tilde{\lambda}|<1$$

Cependant, même si **ce scénario** fonctionne sur les petits cas modèles présentés dans les publications ($R_c<100$), il **s'avère ingérable, en pratique** , avec des contours de rayon R_c en 10^4 ou 10^5 (cf. cas-tests *Aster* `SDLL02` ou `SDLL123`). En effet, la manipulation des monômes de $P(\lambda)$ s'avère déjà souvent problématiques (dépassement de capacité, élimination ou absorption de termes d'ordres de grandeurs trop différentes cf. §2.6), donc celle de son *alter-ego* scalé $\tilde{P}(\tilde{\lambda})$ ne fait qu'amplifier ces perturbations.

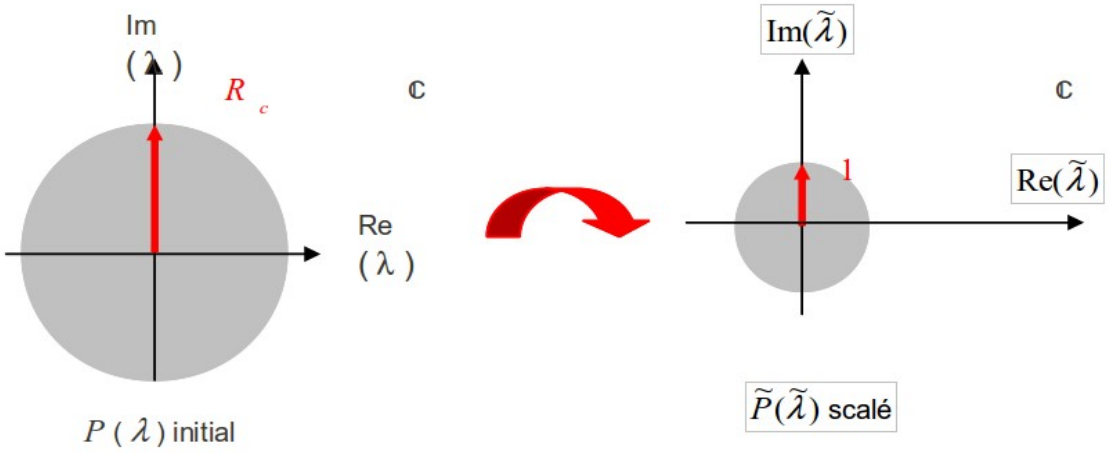


Figure 3.6 Adaptation au disque $D(0,R_c)$ **.**

Finalement, en réunissant tous ces éléments, on a le déroulement algorithmique suivant:

- Transformer le problème initial en un problème modal standard dit de «travail»
  ⇒ SEP.
- Calculer les coefficients de P via la méthode de Rombouts (algorithme n°1)
  ⇒ $(a_i)_i$.

- Déterminer le rayon du contour R_c et faire le changement de variable *ad hoc* cf. (3.5.6) $\Rightarrow (\tilde{a}_i)_i$.
- Appliquer la méthode de Gleyse-Moflih (algorithme n°2) $\Rightarrow N^{D(0,R_C)}$.

Algorithme 8 Principe de la méthode «Sturm modifié».

En dehors de ces aspects numériques, cette méthode dite de «Sturm modifié» souffre aussi de plusieurs limitations plus ou moins handicapantes:

- Être limité à un contour de type cercle,
- Ne pas pouvoir le centrer en un autre point du plan complexe,
- Devoir allouer, pour chacune des deux principales étapes de la méthode, deux matrices pleines de même taille que le polynôme caractéristique (n en GEP, $2n$ en QEP).

Fort de ces éléments sur les différentes méthodes testées voire industrialisées dans *Code_Aster*, nous allons maintenant récapituler leur paramétrage.

4 Parallélisme et calcul intensif

4.1 Premiers pas

Les procédures de dénombrement peuvent s'avérer coûteuses en mémoire et surtout en temps. C'est particulièrement vrai lorsqu'on cherche à compter des modes complexes (GEP non symétrique et/ou complexe ou QEP), mais c**ela peut être aussi le cas lorsqu'on souhaite calibrer un simple GEP standard** (symétrique réel) sur l'axe réel.

C'est en fait le cas de figure le plus courant, lorsqu'on cherche à mieux paramétrer et/ou d'optimiser[46] des calculs modaux ultérieurs. On fait alors **appel intensivement au test de Sturm pour évaluer le nombre de modes contenus dans plusieurs intervalles contiguës** (appelés aussi sous-bandes). Typiquement une dizaine (par exemple le nombre de sous-bandes est nb_sbande=10): $[freq_1, freq_2], [freq_2, freq_3], ...[freq_{10}, freq_{11}]$.
On a alors une dizaine de position modale à calculer $pm(freq_j)(j=1,11)$. Une pour chaque borne des intervalles considérés. Comme détaillé dans le chapitre 3, chacune de ces positions modales est calculée en factorisant la matrice dynamique associée au shift $\sigma_j=(2\pi\ freq_j)^2$ en dynamique (resp. $\sigma_j=freq_j$ en flambement).
On ne retient de ces factorisations que le nombre de termes négatifs de la diagonale (pour mettre en oeuvre le test de Sturm du corollaire 3). Avec, éventuellement, un décalage préalable du shift afin de ne manipuler que des matrices dynamiques « pas trop mal conditionnées » (cf. algorithme 5): $\tilde{\sigma}_j=\sigma_j+\epsilon$.

Donc finalement, sur cet exemple ci, on peut faire l'**analyse algorithmique** suivante:

- On a besoin de 10 (ou 11) calculs indépendants (donc facilement parallélisables) suivant que l'on adopte une vision par sous-bande (ou fréquentielle);
- Ces calculs ne produisent[47] globalement qu'un entier $pm(freq_j)$ et un réel $\tilde{\sigma}_j$;
- La factorisation numérique peut être elle-même optimisée en ne conservant en mémoire aucun des termes : $\mathbf{L}_{ij}$, $\mathbf{D}_{ij}$. Juste le nombre de $\mathbf{D}_{ij}<0$.

Ces éléments, détaillés sur ce petit exemple, vont guider l'optimisation des coûts calcul du test de Sturm, en séquentiel comme en parallèle. Concernant le premier point, on retient, suivant les cas de figure :

- soit une **vision par sous-bandes.** Elle est utilisée dans le test de Sturm d'INFO_MODE et dans celui de pré-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes ;
- soit une **vision fréquentielle.** Elle est utilisée dans les deux cas précédents si nb_sbande=1 ainsi que dans le test de Sturm de post-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes.

Quant à l'astuce du troisième point, elle n'est activée qu'avec le solveur linéaire MUMPS. Celui-ci étant le solveur linéaire a privilégier lorsqu'on souhaite faire du HPC dans *Code_Aster !*

Ces éléments ont donc permis récemment d'optimiser et de paralléliser ce type de calcul. Ainsi, lorsqu'on utilise la méthode de Sturm standard[48], celle-ci peut bénéficier d'accélérations notables. Jugez plutôt : sur le cas-test perf013c parallélisé sur une centaine de processeurs, les **gains en**

46 Pour optimiser les consommations en temps/mémoire et réduire/homogénéiser les résidus modaux, on conseille d'effectuer des calculs par paquets de modes (typiquement 40 ou 50). En parallélisme multi-niveaux avec CALC_MODES + option 'BANDE' découpée en sous-bandes, on cherche de plus à limiter les déséquilibrages de charge en paramétrant des sous-bandes de calcul homogènes en nombre de modes.

47 Sous-entendu, en mode parallèle, « communiquent ». Ce faible niveau de communication est très avantageux. La parallélisation des tests de Sturm peut donc profiter de speed-ups effectifs très proches des speed-ups théoriques.

48 Directement, *via* INFO_MODE, ou indirectement dans les étapes de pré et post-traitements de CALC_MODES avec option 'BANDE' découpée en sous-bandes.

temps sont proches d'un facteur 70 et ceux **en pic mémoire RAM sont proches de 2** (cf. tableau 4.1).
D'où des **tests de Sturm parallèles « quasi-gratuits » !** Pour peu que l'on adapte son paramétrage et que l'on dispose de nombreux processeurs. Et ce, avec un comportement fonctionnel et des précisions de résultats inchangés par rapport au mode séquentiel.

	V11.2 *1 proc*	**V11.3** 1 proc	**V11.4** 32 procs	**V11.4** *128 procs*
Temps elapsed	*3h40min*	1h48min	5min17	*3min*
Pic mémoire RAM	9.1Go	5.5Go	5.5Go	4.9Go

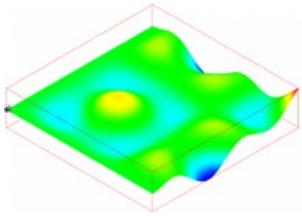


Tableau 4.1. Comparaison des performances d' `INFO_MODE` *sur le cas-test perf013c (plaque carrée en éléments de coques) : nombre de ddls=4.0M[49], 32 sous-bandes de recherche de taille 100Hz sur l'intervalle [0Hz,3200Hz]* ($[freq_i, freq_{i+1}]$ $freq_i=(i-1)\times 100.0\text{Hz}$ $(i=1,33)$). *Machine Ivanoe.*

4.2 Principe

Revenons sur les schémas de parallélisation. Jusqu'à présent, le parallélisme dans *Code_Aster* était principalement un **parallélisme en espace**. Il reposait sur une distribution du maillage entre les processeurs alimentant le parallélisme de deux étapes du calcul:

- la construction de matrice/second membre (calcul élémentaire/assemblage),
- la résolution du système linéaire associé (*via* les produits MUMPS ou PETSc).

En calcul modal, outre le fait que la parallélisation des calculs élémentaires/assemblages n'a pratiquement pas d'impact, celle du système linéaire procure des gains en temps souvent très moyens[50]. D'autre part, ces gains s'étiolent rapidement à partir d'une dizaine de processeurs[51].

Dans ce cas de figure très favorable des tests de Sturm, on se propose d'adjoindre à ce parallélisme en espace, un parallélisme plus efficace basé sur la distribution des sous-bandes (ou des fréquences) sur les processeurs. **Donc ici, l'union parallélisme en fréquence – parallélisme en espace fait la force !** Sous-entendu, on atteint de très bonnes accélérations sans révolutionner le code, ni modifier son comportement ou ses résultats.

Cette stratégie de cumul de deux niveaux de parallélisme permet:

- D'accélérer plus notablement les performances en tirant partie au mieux des efficacités parallèles de chacun des niveaux;
- De construire une stratégie dont l'efficacité est moins dépendante du cas de figure, du paramétrage numérique et de la plate-forme informatique;
- Et ce, sans réel impact sur les aspects fonctionnels, la robustesse et la précision des résultats.

Le découpage naturel des tests de Sturm en `nb_sbande` (ou `nb_freq`) **calculs indépendants procure donc un premier niveau de parallélisme**. Il est souvent très efficace en temps, mais il n'impacte pas les consommations mémoire.

49 M pour million.
50 MUMPS comporte trois étapes: la phase d'analyse, souvent peu coûteuse et séquentielle; la phase de factorisation numérique souvent la plus coûteuse mais elle bénéficie d'un parallélisme efficace (efficacité parallèle de l'ordre de 0.4); la phase de descente-remontée, qui peut s'avérer coûteuse en calcul modal et qui, en revanche, est peu efficace en parallèle (resp. 0.2).
51 Cela dépend de la taille du problème, de ses caractéristiques et du paramétrage numérique de MUMPS.

D'autre part, comme l'essentiel du coût de ces calculs est dû à la phase de factorisation numérique[52] du solveur linéaire, lorsqu'on parallélise cette étape *via* **MUMPS** (SOLVEUR=_F(METHODE='MUMPS')), on rajoute des gains notables en temps et en mémoire. C'est l**e deuxième niveau de parallélisme**.

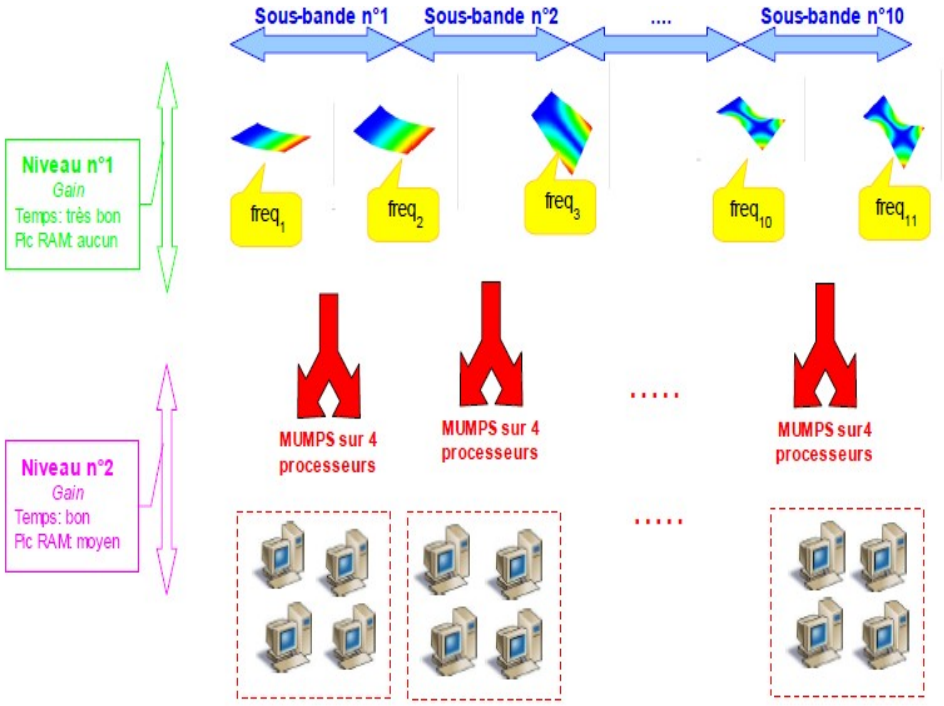


Figure 4-1. Deux niveaux de parallélisme.
INFO_MODE seul ou en pré-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes.
Exemple de distribution du test de Sturm sur nb_proc=40 processeurs avec un découpage en nb_sbande=10 sous-bandes (parallélisme dit « 10x4 »).
On utilise ici le solveur linéaire MUMPS et la paramétrage du parallélisme par défaut ('COMPLET').

Avec la valeur par défaut du paramétrage (mot-clé **NIVEAU_PARALLELISME='COMPLET'**), lorsque le calcul est parallèle[53], les deux niveaux de parallélisme se cumulent. Comme le second niveau est moins efficace que le premier, il ne se déclenche que si le nombre de processeurs le permet (nb_proc>nb_sbande) et que si le solveur linéaire sélectionné supporte du parallélisme MPI (SOLVEUR=_F(METHODE='MUMPS')).
Ce schéma de parallélisation est illustré à la figure 4.1. C'est un exemple de distribution du test de Sturm sur nb_proc=40 processeurs avec un découpage en 10 sous-bandes (nb_sbande=10): **parallélisme à 2 niveaux dit « 10x4 ».**

Si on sélectionne l'autre valeur du mot-clé (**NIVEAU_PARALLELISME='PARTIEL'**), on ne profite que du parallélisme du solveur linéaire (le second niveau). Cela peut avoir de l'intérêt pour tester des méthodes ou pour axer les gains sur la réduction du pic mémoire (après avoir essayé les autres bras de levier proposés dans le mot-clé SOLVEUR).
Ce schéma de parallélisation est illustré à la figure 4.2. C'est un exemple de distribution du test de Sturm sur nb_proc=40 processeurs avec un découpage en 10 sous-bandes: **parallélisme mono-niveau dit « 1x40 ».**

52 Etape du solveur linéaire qui tire le plus de profit du parallélisme.
53 On a sélectionné une version MPI du code et on a lancé l'étude sur plusieurs processeurs.

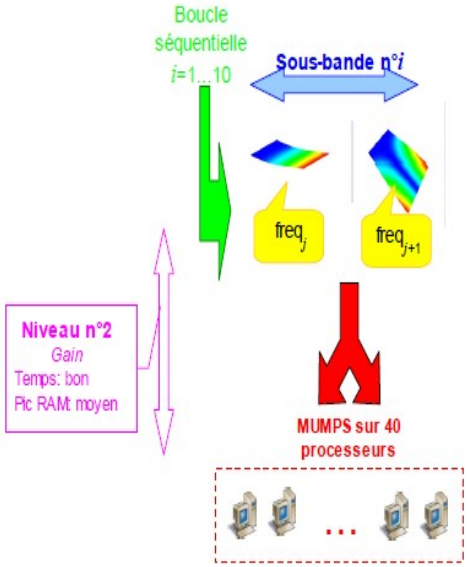


Figure 4-2. Un seul niveau de parallélisme.
INFO_MODE seul ou en pré-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes.
Exemple de distribution du test de Sturm sur nb_proc=40 processeurs avec un découpage en nb_sbande=10 sous-bandes (parallélisme « 1x40 »).
On utilise ici le solveur linéaire MUMPS et on change le paramétrage par défaut du parallélisme ('PARTIEL').

Le premier niveau de parallélisme peut être qualifié d'« embarrassingly » ou de « pleasingly » pour reprendre des dénominations classiques en HPC. Il est si facile à concevoir [54] qu'il aurait été embarrassant de ne pas l'avoir fait ! Il est d'autant plus favorable que ses communications sont très faibles: un entier, la position modale du shift considéré [55] $pm(\sigma)$, et, un réel, le shift éventuellement décalé $\tilde{\sigma}$.

Le second niveau de parallélisme est nettement plus difficile . Il comporte de nombreuses tâches interdépendantes, de granularité très variable et requièrant de coûteuses communications (morceaux de matrice/vecteur). De plus, certaines tâches sont gérées dynamiquement. C'est-à-dire qu'elles ne peuvent pas être prévues initialement. C'est donc un des schémas parallèles les plus difficiles. C'est pour cela que nous confions cette tâche au produit 'best-in-class' MUMPS.

Remarques:

- Lorsqu'on active le second niveau de parallélisme, il est conseillé de réserver, en modal, au moins 10^5 ddls par processeur afin d'alimenter suffisamment le solveur linéaire MUMPS.
- Plus précisément, le deuxième niveau de parallélisme est lié ici aux étapes d'analyse et de factorisation numérique de MUMPS. La première étape d'analyse est, pour l'instant, traitée séquentiellement dans le couplage Code_Aster-MUMPS. Pour optimiser les performances parallèles, il est donc préférable de choisir un paramétrage numérique du solveur linéaire rendant négligeable cette analyse par rapport à la factorisation. Souvent, en calcul modal, un renumeroteur « frustre » type AMD ou QAMD répond mieux à ce cahier des charges qu'un renumeroteur plus sophistiqué type METIS, PORD ou SCOTCH.

4.3 Détails fonctionnels

54 Moins aisé à mettre en oeuvre dans un code riche comme *Code_Aster.*
55 On parle indifféremment de la position modale d'un shift donné ou de celle de la fréquence (ou mode de flambement) associée.

Le parallélisme mis en oeuvre dans les tests de Sturm est géré différemment suivant les cas de figures. Ceux-ci sont de trois types :

- Ceux intégrés dans des opérateurs bénéficiant du parallélisme multi-niveaux directement pilotables par le paramétrage utilisateur: **INFO_MODE seul ou en prétraitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes.**

- Ceux bénéficiant du parallélisme multi-niveaux mais qui ne sont pas directement pilotables par le paramétrage utilisateur. Il s'agit des vérifications de **post-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes** (STURM='GLOBAL'/'LOCAL'). Plus le cas marginal, mais néanmoins informatiquement pris en compte, des configurations du premier point sur une seule sous-bande (nb_sbande=1).

- Les autres cas qui ne bénéficient éventuellement que du parallélisme de MUMPS (donc mono-niveau): **CALC_MODES avec OPTION='BANDE', CALC_MODES avec OPTION parmi ['BANDE','CENTRE','PLUS_*','TOUT'] et STURM='OUI', CALC_MODES avec OPTION parmi ['AJUSTE','SEPARE']**.

Dans les paragraphes qui suivent on détaille l'impact du parallélisme sur chacun de ces cas de figure. Pour plus de détails on pourra consulter les documentations Utilisateurs associées à ces opérateurs [U4.52.01/02/03/04].

4.3.1 INFO_MODE seul ou en pré-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes

D'un point de vue fonctionnel, plusieurs cas de figure sont à prendre en compte suivant le paramétrage du mot-clé NIVEAU_PARALLELISME, le solveur linéaire et le nombre de processeurs sélectionnés:

- **Avec le paramétrage du parallélisme par défaut ('COMPLET')**
  Si on utilise le solveur linéaire MUMPS, on peut activer 1 ou 2 niveaux de parallélisme dès qu'il y' a assez de données pour « nourrir » les processeurs: nb_proc>=nb_sbande (cf. figures 4.1/4.3a). Dans le cas contraire, le calcul s'arrête en erreur fatale et propose à l'utilisateur d'ajuster nb_proc ou nb_sbande en conséquence.

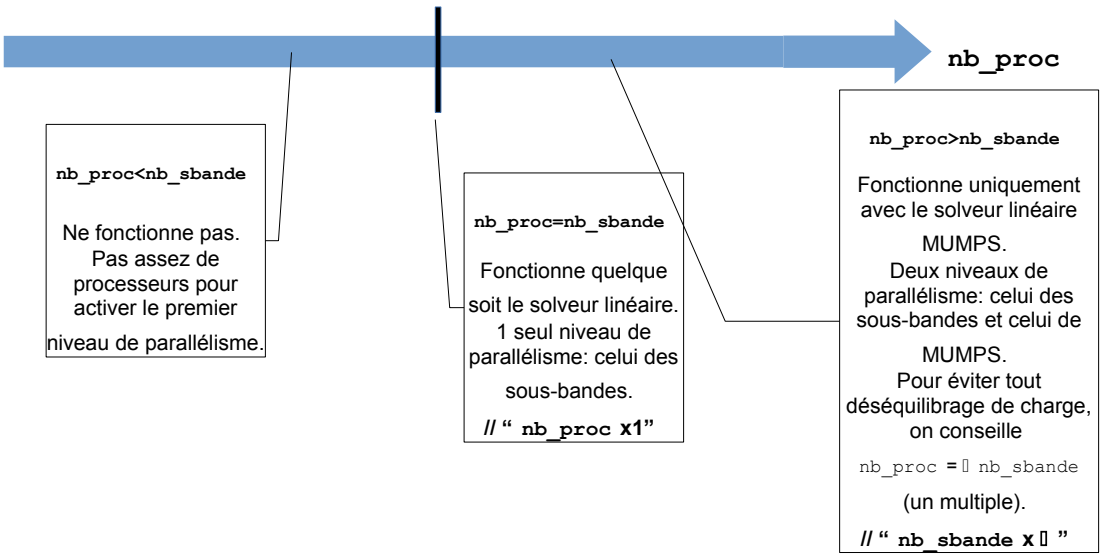


Figure 4-3a. INFO_MODE seul ou en pré-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes.
Périmètre d'utilisation avec NIVEAU_PARALLELISME='COMPLET'.

Bien sûr, s*i on ne souhaite pas utiliser le solveur linéaire MUMPS*, mais plutôt un des deux autres solveurs directs séquentiels[56] (SOLVEUR=_F(METHODE='MULT_FRONT' ou 'LDLT')), on peut quand même profiter du premier niveau de parallélisme en paramétrant un nombre de processeurs rigoureusement identique au nombre de sous-bandes: nb_proc=nb_sbande.

L'exemple de la figure 4.4 illustre ce cas de figure sur sur nb_proc=10 processeurs avec un découpage en nb_sbande=10 sous-bandes : parallélisme mono-niveau dit « 10x1 ».

Si nb_proc<nb_sbande, le code s'arrête en erreur fatale car il n'y a pas assez de données à paralléliser. Il faut alors ajuster nb_proc ou nb_sbande.
Si nb_proc>nb_sbande, le code s'arrête en erreur fatale pour signifier le caractère sous-optimale du calcul: on allait utiliser, dans le second niveau, plusieurs processeurs pour effectuer une factorisation matricielle séquentielle. Il faut alors ajuster nb_proc ou nb_sbande, soit paramétrer MUMPS (soit changer NIVEAU_PARALLELISME).

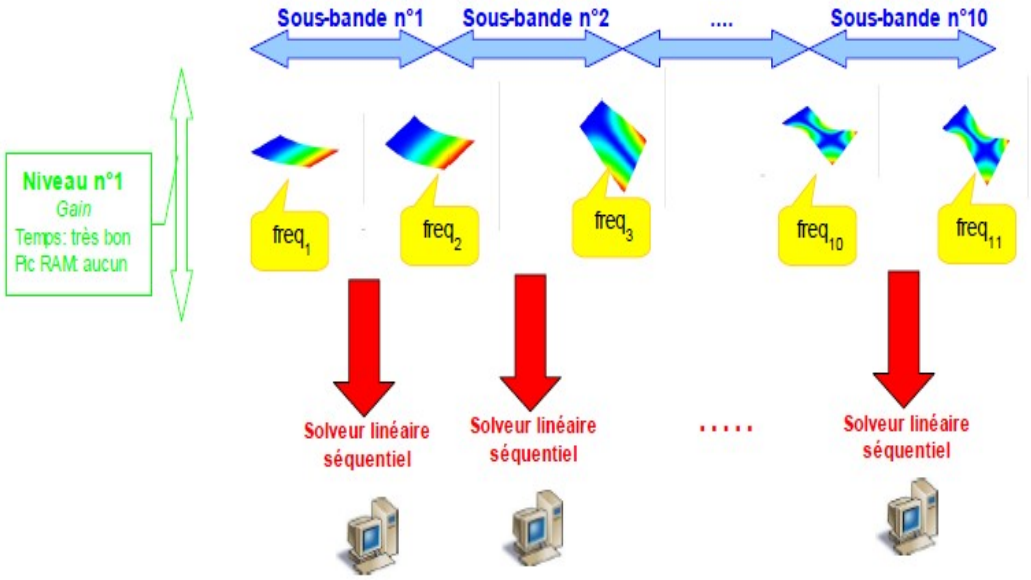


Figure 4-4. Un seul niveau de parallélisme.
INFO_MODE seul ou en pré-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes.
Exemple de distribution du test de Sturm sur nb_proc=10 processeurs avec un découpage en nb_sbande=10 sous-bandes (parallélisme « 10x1 »).
On utilise ici un solveur linéaire séquentiel[57](METHODE='MULT_FRONT' ou 'LDLT') et la paramétrage du parallélisme par défaut ('COMPLET').

Remarque:

- *Ce cas de figure fonctionne aussi de la même manière avec le solveur MUMPS. Sur 10 processeurs, seul le premier niveau étant activé, il fonctionne alors en séquentiel pour chacune des sous-bandes.*

- **En changeant le paramétrage du parallélisme ('PARTIEL')**
  Pour activer cette option, il faut impérativement paramétrer le solveur MUMPS[58]. Le nombre de processeurs peut alors prendre toutes les valeurs possibles par rapport au nombre de sous-

56 Séquentiel en MPI. Car MULT_FRONT bénéficie potentiellement d'un parallélisme openMP.
57 Séquentiel pour MPI. MULT_FRONT peut être partiellement parallélisée en OpenMP. Pour l'instant, ces deux types de parallélisme ne sont pas combinables cf. [U2.08.06].
58 Sinon on s'arrête en erreur fatale.

bandes, car celles-ci sont traitées en séquentiel. Seul le second niveau est parallélisé (cf. figure 4.2/4.3b).

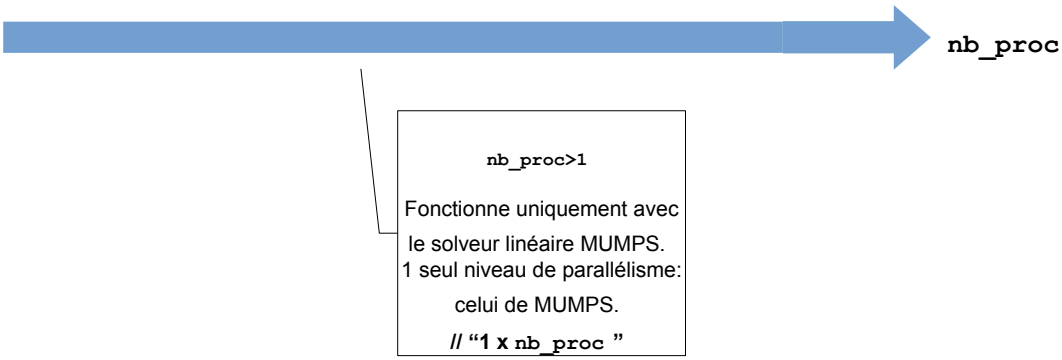


Figure 4-3b. INFO_MODE seul ou en pré-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes .
Périmètre d'utilisation avec NIVEAU_PARALLELISME='PARTIEL'.

Remarque:

- Lorsque le nombre de fréquences ou de modes de flambement est égal à 2, nb_sbande=1. On retombe alors dans le cas de figure qui suit (type INFO_MODE de post-traitement dans CALC_MODES avec option 'BANDE' découpée en sous-bandes).

4.3.2 INFO_MODE de post-traitement dans CALC_MODES avec option 'BANDE' découpée en sous-bandes

Dans les cas précédents , on gère les incompatibilités éventuelles entre le nombre de sous-bandes et de processeurs ainsi que le paramétrage (cf. figures 4.3a/b). Mais une fois que le calcul modal est accepté, puis effectué, l' INFO_MODE de post-vérification ne peut plus remettre en cause la conformité du paramétrage par rapport au nombre de bandes à tester. Donc toutes les combinaisons en terme de paramétrage et de nombre de processeurs doivent être acceptées. Sinon, la situation peut devenir ingérable pour l'utilisateur !

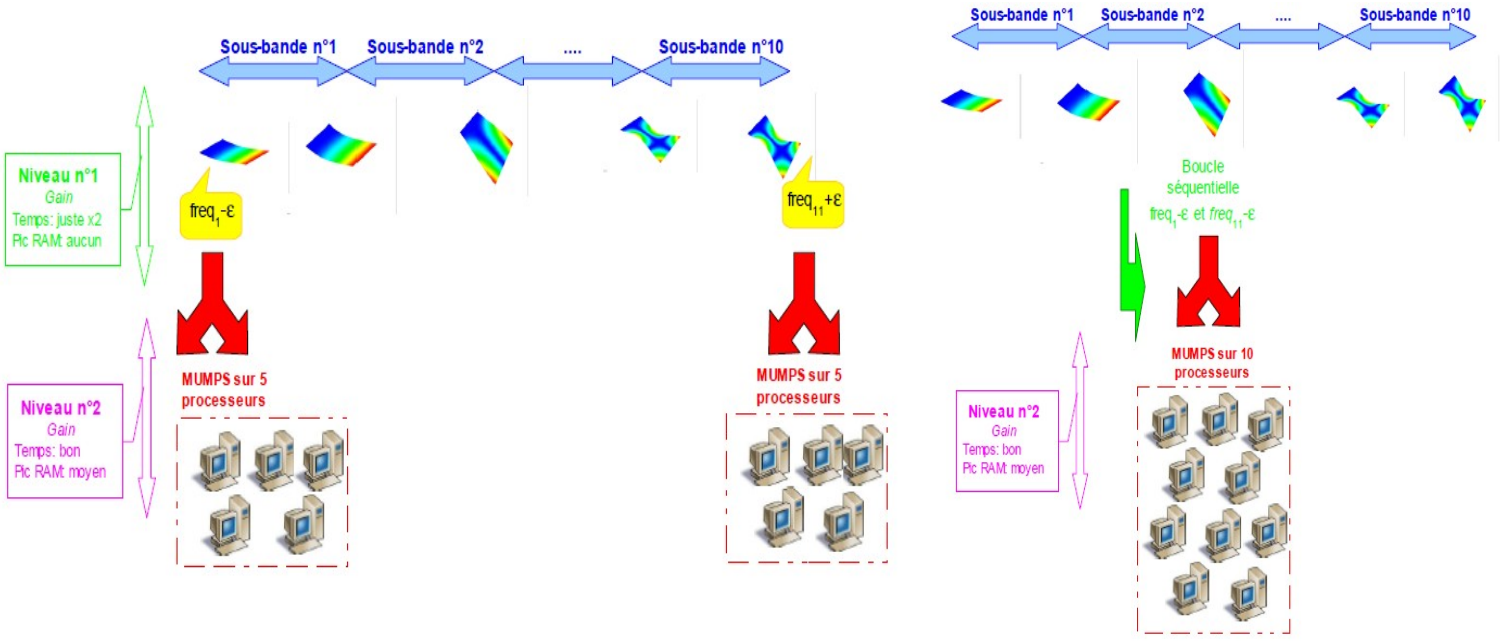


Figure 4-5a/b. Deux ou un seul niveau de parallélisme.

INFO_MODE en post-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes.
Exemple de distribution du test de Sturm sur nb_proc=10 processeurs avec un découpage initial en nb_sbande=10 sous-bandes (parallélisme « 2x5 » ou « 1x10 »).
On utilise ici le solveur linéaire MUMPS et les deux valeurs de GESTION_PARALLELISME='COMPLET' (par défaut) /'PARTIEL'.

En fait, dans ce cas de figure, on ne doit en tester qu'une seule sous-bande: **elle est constituée des bornes extrêmes de l'ensemble des modes calculés** moins un petit décalage (pour que la matrice dynamique reste inversible). Ces modes calculés concernent l'ensemble des sous-bandes si ' GLOBAL ' (valeur par défaut) ou simplement la sous-bande courante avec ' LOCAL '.
Il doit faire au mieux avec les capacités de calcul dont il dispose.

Pour optimiser leurs coûts et faciliter leur utilisation, ces INFO_MODEs de post-vérification bénéficient donc de traitements particuliers (cf. figures 4.5/4.6):

- Comme ils ne concernent qu'une seule bande de fréquences, on organise le premier niveau de parallélisme en deux paquets: un pour chaque fréquence. **On adopte alors un schéma parallèle basé sur les fréquences plutôt qu'un schéma basé sur les sous-bandes.**

- Si le second niveau ne peut pas s'activer car le solveur linéaire retenu est nativement séquentiel, on n'arrête pas le calcul pour autant. Il est sous-optimal mais cette étape étant peu coûteuse[59] par rapport au processus complet de CALC_MODES avec option 'BANDE' découpée en sous-bandes, **on tolère ici ce type de fonctionnement.**

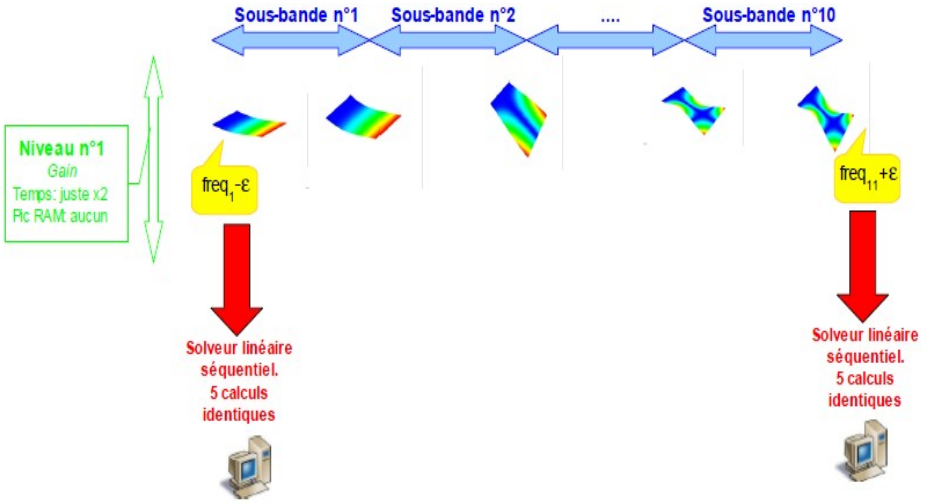


Figure 4-6. Un niveau de parallélisme.
INFO_MODE en post-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes.
Exemple de distribution du test de Sturm sur nb_proc=10 processeurs avec un découpage initial en nb_sbande=10 sous-bandes (parallélisme « 2x1 »).
On utilise ici un solveur linéaire séquentiel [60] (METHODE='MULT_FRONT' ou 'LDLT') et les deux valeurs de GESTION_PARALLELISME='COMPLET' [61] .

59 Ne pas accorder cette exception rendrait caduque l'utilisation de CALC_MODES avec option 'BANDE' découpée en sous-bandes parallélisées avec un autre solveur linéaire que MUMPS (sauf à débrancher le test de Sturm final, ce qui n'est pas conseillé).
60 Séquentiel pour MPI. MULT_FRONT peut être partiellement parallélisée en OpenMP. Pour l'instant, ces deux types de parallélisme ne sont pas combinables cf. [U2.08.06].
61 La valeur 'PARTIEL' est ici prohibée. L'opérateur CALC_MODES avec option 'BANDE' découpée en sous-bandes, arrête le calcul dès sa phase de lancement.

Ces tests de Sturm ne subissent donc pas les restriction de périmètre décrites dans les figures 4.3a/b. En fait, comme il n'y a que deux fréquences à distribuer et que l'on dispose d'au moins deux processeurs, il n'y a aucune restriction fonctionnelle. A part celles imposées par l'étape d'initialisation de CALC_MODES avec option 'BANDE' découpée en sous-bandes.
Cela permet donc d'avoir une validation des schémas parallèles plus exhaustive car elle concerne tous les solveurs linéaires. On peut aussi fonctionner sans MUMPS, le cas échéant, tout en procurant des gains appréciables: uniquement les gains en temps du premier niveau (x2).

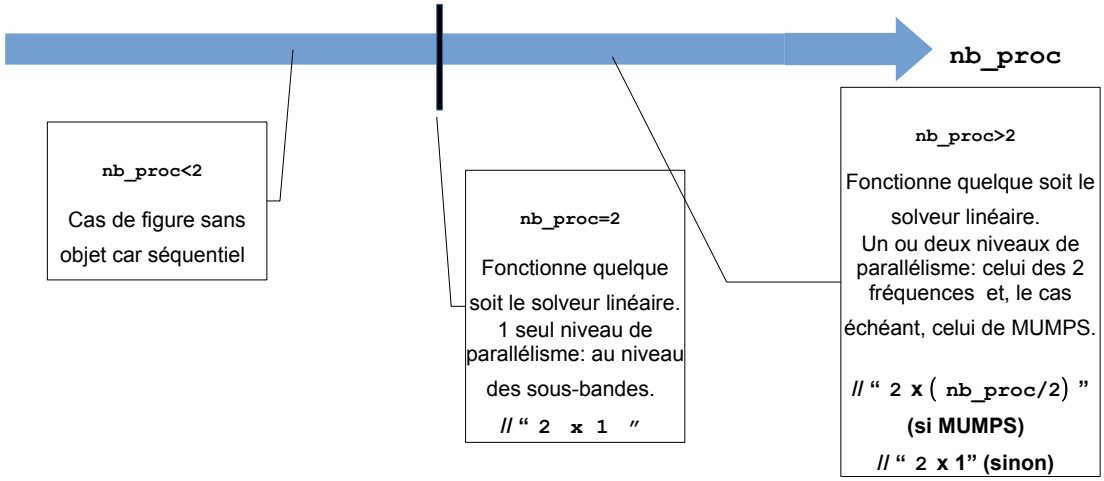


Figure 4-7a. INFO_MODE de post-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes.
Périmètre d'utilisation avec NIVEAU_PARALLELISME='COMPLET'.

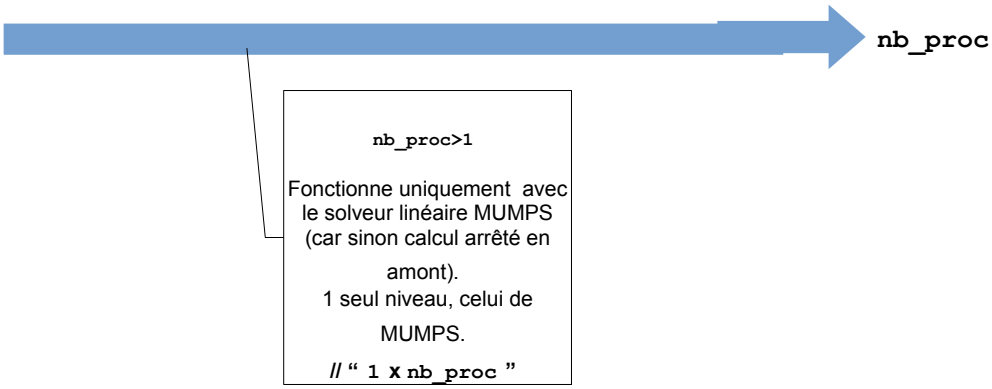


Figure 4-7b. INFO_MODE de post-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes.
Périmètre d'utilisation avec NIVEAU_PARALLELISME='PARTIEL'.

Remarque:

- Dans le cas STURM='GLOBAL' on utilise directement l'opérateur INFO_MODE depuis le python de la macro-commande. Dans l'autre configuration, STURM='LOCAL', on reste dans le F77 de l'opérateur dédié à CALC_MODES avec OPTION parmi ['BANDE','CENTRE','PLUS_*','TOUT']. Mais dans les deux cas, on exploite bien sûr les

mêmes schémas parallèles via les mêmes sources. Les comportements et les résultats sont identiques.

4.3.3 Autres tests de Sturm

Il reste des tests de Sturm ne bénéficiant, éventuellement, que d'un seul niveau de parallélisme (celui de MUMPS). Il est activé par défaut dès qu'on utilise plusieurs processeurs et qu'on a choisit MUMPS. Il s'agit des tests de Sturm de :

- CALC_MODES avec OPTION='BANDE' (sans relecture d'un INFO_MODE préalable),
- CALC_MODES avec OPTION parmi ['BANDE','CENTRE','PLUS_*','TOUT'] + STURM='OUI',
- CALC_MODES avec OPTION parmi ['SEPARE','AJUSTE'].

Remarque :

- Le parallélisme de ces tests de Sturm se comporte comme si le paramètrage NIVEAU_PARALLELISME='PARTIEL' était activé. A la différence que l'on permet, pour l'instant le gaspillage de ressources (comme ce sont des opérateurs à réserver aux petits problèmes). On peut par exemple utiliser un CALC_MODES paramétré avec MULT_FRONT dans un run parallèle MPI. Cette souplesse permet de tester différentes fonctionnalités dans un même fichier de commande.

4.4 Déséquilibrages de charge

4.4.1 Généralités

D'un point de vue fonctionnel et performance, on prend en compte certains déséquilibrages de charge. Ainsi, l'utilisation du parallélisme reste très simple et cela ne grève pas trop ses performances.

Avant de détailler tous les cas de figure, on peut en résumer les causes :

- **Distribution des sous-bandes ou des fréquences sur les processeurs**. Déséquilibrage fonctionnel qui peut se produire dans tous les cas de figure (INFO_MODE seul, en pré ou post-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes). Il est pris en charge automatiquement et peu préjudiciable pour les performances.
- Distribution des fréquences par sous-bande. Déséquilibrage d'ordre plutôt informatique (pour INFO_MODE seul) et fonctionnel (en pré-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes). Il n'est pas trop préjudiciable pour les performances des études et il peut s'améliorer (dans le premier cas de figure) facilement.
- Déséquilibrage des tâches numérique au niveau du solveur linéaire parallèle (ici MUMPS).

On ne traite directement que les éventuels déséquilibrages du premier niveau de parallélisme (par sous-bandes ou par fréquences).
Ceux du second niveau sont directement gérés par MUMPS et non paramétrables. Même si le paramétrage numérique[62] du solveur linéaire permet parfois d'influencer ces déséquilibrages, on ne peut pas les manipuler directement. Il faut donc faire confiance au produit. Toutefois, en pratique, on constate rarement de très mauvais déséquilibrages à ce niveau.

4.4.2 INFO_MODE seul ou en pré-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes

Une cause évidente de déséquilibrage concerne la distribution « sous-bandes/processeurs ». Lorsque le nombre de processeurs de la classe batch n'est pas un multiple de nb_sbande

$$\text{nb\_proc} = \alpha \times \text{nb\_sbande} + \beta \quad (\beta < \text{nb\_sbande})$$ (cf. figure 4.8),

62 Renumeroteur, parallélisme centralisé ou distribué...

on distribue le reliquat de processeurs en priorisant les premières sous-bandes. Chacun des β premiers processeurs gère une sous-bande en plus ($\alpha+1$). Les $nb\_sbande-\beta$ suivant gèrent, chacun, α sous-bandes.

Afin de limiter les coûts de communication MPI, on réserve de manière contigüe chaque paquet de α (ou $\alpha+1$ en cas de déséquilibrage) processeurs traitant une sous-bande. La figure 4.8 illustre ce type de déséquilibrage avec un calcul de 10 sous-bandes distribuées (nb_sbande=10) sur 41 processeurs (nb_proc=41): **parallélisme dit « 1x5 + 9x4 ».**

Remarques:

- Comme ici le processeur maître travaille plus que les autres (cf. point suivant), cette redistribution essaie de compenser un peu ce déséquilibrage.
- Toutefois, concernantCALC_MODES avec option 'BANDE' découpée en sous-bandes, le déséquilibrage est principalement dû au calculs modaux proprement dit (même si ceux-ci traitent des sous-bandes homogènes en nombre de modes). Dans ce cas de figure, nos stratégies correctives autour du simple test de Sturm ne compensent que partiellement ce déséquilibrage. Pour bien faire, il faudrait un solveur modal parallèle avec équilibrage dynamique de la charge (inséré entre le premier et le deuxième niveau de parallélisme, cf. stratégie ERAM de C.Calvin).

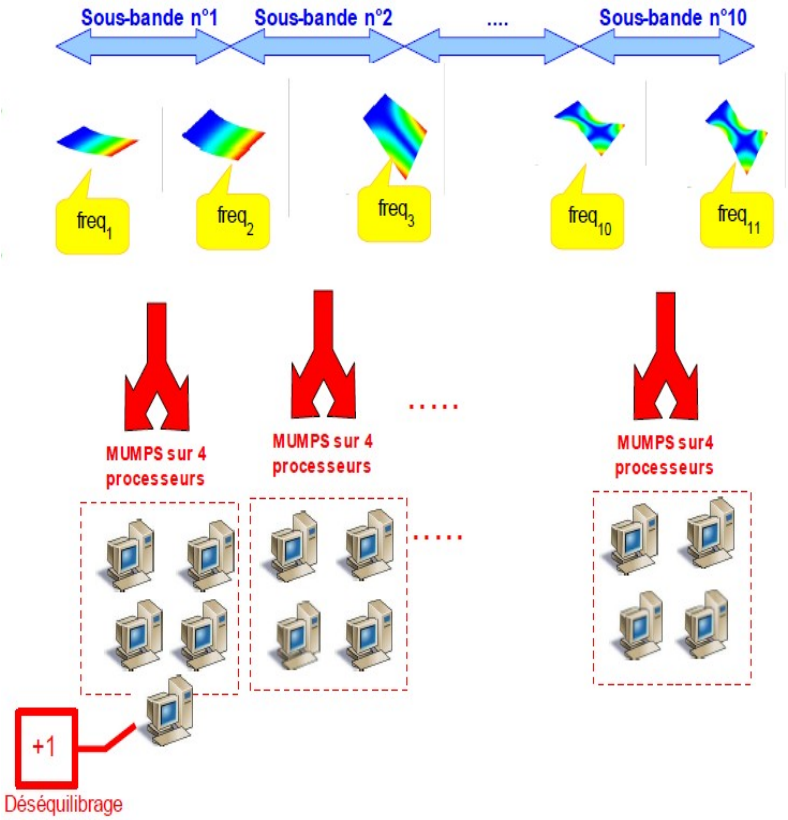


Figure 4-8. Déséquilibrage de charge « sous-bandes/processeurs ».
INFO_MODE seul ou en pré-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes..
Exemple de distribution du test de Sturm sur nb_proc=41 processeurs avec un découpage en nb_sbande=10 sous-bandes (parallélisme dit « 1x5 + 9x4 »).

En fait, pour ne pas ré-écrire la programmation du test de Sturm et conserver les mêmes source en parallèle et en séquentiel, **on a gardé une organisation déséquilibrée des calculs du premier niveau**. La première sous-bande traite les deux premières fréquences ($freq_1$ et $freq_2$) tandis que les autres ne traitent qu'une seule fréquence ($freq_j$). La figure 4.9 illustre ce déséquilibrage.

D'où un déséquilibrage théorique de l'ordre de 2, égal au déséquilibrage «fréquences/sous-bandes». En pratique, on constate que ce déséquilibrage est moins sévère et conduit souvent à une efficacité parallèle proche de 0.6 (au lieu de 0.5).

En pratique, celle-ci n'est pas véritablement préjudiciable car les gains globaux procurés sont déjà potentiellement importants. De plus, souvent une des fréquence s'avère suffisamment proche d'une mode propre pour que l'on active l'algorithme de décalage (cf. algorithme 5). A chaque étape de cet algorithme on ré-effectue un calcul de position modal (on passe de σ_j à $\tilde{\sigma}_j$). Donc le processeur concerné va prendre du retard et, pour peu que cela ne soit pas une des deux premières bornes, ce retard va devenir plus important que le surcoût du processeur maître.

D'autre part, concernant un calcul INFO_MODE parallèle préalable à un calcul CALC_MODES avec option 'BANDE' découpée en sous-bandes parallélisées, il peut être plus confortable pour l'utilisateur de paramétrer ces deux calculs avec le même nombre de processeurs. Et non pas, l'un sur nb_proc et l'autre sur nb_proc-1 (=nb_sbande).

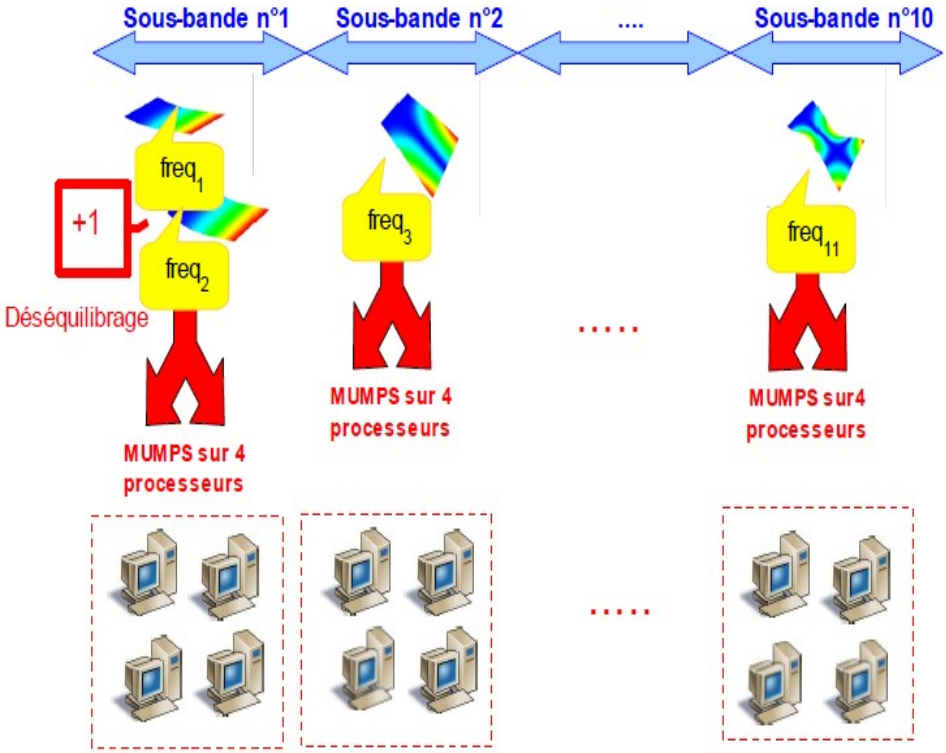


Figure 4-9. Déséquilibrage de charge « fréquences/sous-bandes ».
INFO_MODE seul ou en pré-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes.

4.4.3 INFO_MODE en post-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes

Dés que le nombre de processeurs choisi est impair, on a fonctionnellement un déséquilibrage « fréquences/processeurs ». Il est géré de la même manière que dans le cas précédent:

- la première fréquence se verra dédier $\frac{nb\_proc+1}{2}$ processeurs,
- alors que la seconde n'en aura que $\frac{nb\_proc-1}{2}$.

Comme le nombre de processeurs est souvent grand par rapport à 2, l'efficacité parallèle est peu impactée et elle reste proche de 1.

Ce schéma parallèle, déjà orienté « fréquence », n'est bien sûr pas concerné par le déséquilibrage «fréquences/sous-bandes».

Remarque:

- Par contre, comme le parallélisme du deuxième niveau est plus sollicité dans ce cas de figure, l'éventuel déséquilibrage de MUMPS est ici potentiellement plus prégnant.
  Toutefois, le coût calcul de cette étape étant négligeable par rapport à celui du CALC_MODES avec option 'BANDE' découpée en sous-bandes, l'impact global de ces déséquilibrages est faible.

4.4.4 Autres tests de Sturm

Le déséquilibrage est limité à celui de second niveau de parallélisme dans **MUMPS** déjà évoqué.

4.5 Gains procurés par le parallélisme

4.5.1 INFO_MODE seul ou en pré-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes

En terme d'efficacité parallèle en temps, le premier niveau de parallélisme est proche de 0.6 sur une large plage de processeurs. On peut ainsi distribuer le calcul de plusieurs dizaines de sous-bandes sur autant de paquets de processeurs avec une bonne efficacité.

Tandis que celle du second niveau est de l'ordre de 0.4, à condition toutefois que le problème traité soit assez volumineux (>0.5M de degrés de liberté). Surtout, cette efficacité s'étiole à partir de 4, 8 ou 16 processeurs dévolus au traitement d'une sous-bande. Tout dépend du cas de figure, du paramétrage numérique de MUMPS (renumeroteur, pré/post-traitements...) et des caractéristiques de la machine (architecture mémoire, réseaux...).
En bref, le premier niveau de parallélisme est un plus efficace en terme de consommations en temps et surtout plus stable que le second. Par contre, le premier niveau ne procure pas de gain en pic mémoire RAM, alors que le second permet de gagner sur cet élément souvent dimensionnant du calcul.

Donc, en activant seulement le premier niveau de parallélisme, un calcul découpé en nb_sbande et parallélisé sur nb_proc=nb_sbande peut gagner en temps au moins un facteur 0.6*nb_sbande (sans surcoût/gain mémoire et sans perte de précision).
Si on distribue le calcul sur plus de processeurs (nb_proc>nb_sbande) en utilisant le solveur MUMPS (en ajoute ainsi le 2nd niveau), les gains en temps vont être améliorés d'un facteur proche de 2 dès qu'on rajoute 5*nb_bande processeurs supplémentaires.
Et ce, avec des gains en mémoire pouvant aller jusqu'à un facteur 2[63].

Ainsi, le calcul sur nb_sbande=10 illustré à la figure 4.1, si il coûte $1000s$ en séquentiel, il n'en consomme plus que $1000/(0.6\times 10)=166s$ sur nb_proc=10.
Et lorsqu'on rajoute 30 nouveaux processeurs (nb_proc=40) pour activer aussi le second niveau [64] , le temps total est proche de $166/(0.4\times 4)=104s$.
D'où une accélération globale en temps, sur 40 processeurs, de $1000/104=9.6$. Soit presque un gain d'un ordre de grandeur sur 40 processeurs !

Niveau de parallélisme	Niveau 1 (sous-bande)	Niveau 2 (MUMPS)
Gain en pic mémoire RAM	Aucun	Moyen
Gain en temps de calcul	Très bon	Bon

63 Lors des tests de Sturm, les consommations mémoire de MUMPS sont particulièrement optimisées (on ne stocke pas la matrice factorisée). Les gains que peut lui procurer le parallélisme sont donc, en proportion, moins importants.

64 On passe d'un parallélisme « 4x1 » à un parallélisme « 4x4 ».

Efficacité parallèle en temps	De l'ordre de 0.6	De l'ordre de 0.4
Périmètre/stabilité de l'efficacité parallèle en temps	Sur une large plage de processeurs (par ex. nb_sbande=2 à 50) Efficacité stable quelque soit le problème considéré et la plate-forme.	Efficacité qui diminue à partir de 4, 8 ou 16 processeurs. Efficacité variable suivant le problème considéré, le paramétrage de MUMPS et la plate-forme.
Conditions de mise en oeuvre	Avec NIVEAU_PARALLELISME='COMPLET' : dès que nb_proc >1 (quelque soit le solveur linéaire choisit). Avec NIVEAU_PARALLELISME='PARTIEL' : pas mis en oeuvre, sans objet.	Avec NIVEAU_PARALLELISME='COMPLET' : dès que nb_proc > nb_sbande et METHODE='MUMPS'. Avec NIVEAU_PARALLELISME='PARTIEL' : dès que nb_proc > 1 et METHODE='MUMPS'.
Périmètre/fonctionnalités du test de Sturm	Pas d'impact	Uniquement avec MUMPS.
Robustesse	Pas d'impact	Pas d'impact
Comportement numérique	Pas d'impact	Parfois, peut générer des messages d'ALARME ou d'ERREUR différents suivant le nombre de processeurs impliqués dans ce second niveau. Mais le problème soulevé sera, dans tous les cas, identique.

Tableau 4-1. Synoptique du parallélisme INFO_MODE seul ou pré-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes

4.5.2 INFO_MODE en post-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes

Ici l'efficacité parallèle en temps du premier niveau est excellente (proche de 1). Mais elle n'est active que sur 2 paquets de processeurs ! Au delà elle est complétée par l**e second niveau MUMPS dont l'efficacité en temps est proche de 0.4 .**
Donc, en activant seulement le premier niveau de parallélisme, le test de Sturm de post-traitement parallélisé sur nb_proc=2 peut gagner en temps un facteur 2 (sans surcoût/gain mémoire et sans perte de précision).
Si on distribue ce calcul sur plus de processeurs (nb_proc>2) en utilisant le solveur MUMPS (en ajoute ainsi le 2nd niveau), les gains en temps vont être améliorés d'un facteur proche de 2 dès qu'on rajoute 10 processeurs supplémentaires.
Et ce, avec des gains en mémoire pouvant aller jusqu'à un facteur 2.

Ainsi, le post-traitement illustré à la figure 4.5a, si il coûte 100s en séquentiel, il n'en consomme plus que $100/(2\times 0.4\times 5)=25s$ sur nb_proc=10. Soit un gain global en temps d'un facteur 4.

Niveau de parallélisme	Niveau 1 (fréquence)	Niveau 2 (MUMPS)
Gain en pic mémoire RAM	Aucun	Moyen
Gain en temps de calcul	Excellent	Bon
Efficacité parallèle en temps	De l'ordre de 1	De l'ordre de 0.4
Périmètre/stabilité de l'efficacité parallèle en temps	Bornée à 2	Efficacité qui diminue à partir de 4, 8 ou 16 processeurs. Efficacité variable suivant le

		problème considéré, le paramétrage de MUMPS et la plate-forme.
Conditions de mise en oeuvre	Avec NIVEAU_PARALLELISME='COMPLET' : dès que nb_proc >1 (quelque soit le solveur linéaire choisit). Avec NIVEAU_PARALLELISME='PARTIEL': pas mis en oeuvre, sans objet.	Avec NIVEAU_PARALLELISME='COMPLET' : dès que nb_proc > 2 et METHODE='MUMPS'. Avec NIVEAU_PARALLELISME='PARTIEL' : dès que nb_proc > 1 et METHODE='MUMPS'.
Périmètre/fonctionnalités du test de Sturm	Pas d'impact	Pas d'impact
Robustesse	Pas d'impact	Pas d'impact
Comportement numérique	Pas d'impact	Parfois, peut générer des messages d'ALARME ou d'ERREUR différents suivant le nombre de processeurs impliqués dans ce second niveau. Mais le problème soulevé sera, dans tous les cas, identique.

Tableau 4-2. Synoptique du parallélisme INFO_MODE en post-traitement de CALC_MODES avec option 'BANDE' découpée en sous-bandes

4.5.3 Autres tests de Sturm

L'efficacité est limitée à celle de MUMPS (cf. les deux paragraphes précédent).

4.6 Garde-fous logiciels et conseils méthodologiques

L'introduction du parallélisme MPI dans un code riche, générique et à « développement soutenu[65] » tel que *Code_Aster* pouvait poser certains problèmes: failles d'AQ, restriction de périmètre, problème d'ergonomie, et surtout, les risques de résultats faux !

Pour circonscrire ces problèmes, un certain nombre de garde-fous logiciels et de conseils méthodologiques ont donc été mis en place depuis une dizaine d'année:

- **Se limiter à des opérations MPI simples**. Elles sont informatiquement contingentées dans des routines « chapeaux » dédiées afin de ne pas être saupoudrées dans le code.

- En particulier, ne faire que des **communications collectives simples** (moins de risque de 'deadlocks'). Laisser les implémentations subtiles type « asynchronisation » ou « bufférisation » à la librairie MPI elle-même et limiter, autant que faire se peut, les communications point-à-point.

- **Communiquer le plus tôt possible et des quantités simples**: vecteurs d'entiers, de réels ou de complexes.

- **Typer les structures de données potentiellement impactées par le parallélisme** (MPI_(IN)COMPLET). Elles peuvent être complètes ou incomplètes. On les complète si un traitement numérique particulier l'exigent: produit matrice-vecteur, archivage...

65 Une cinquantaine de contributeurs chaque année avec quasiment une release hebdomadaire.

- **Limiter le parallélisme aux tâches les plus coûteuses, les plus transverses et les plus modulaires**: principalement construction des matrices/vecteurs et résolution de systèmes linéaires.

- Tous **les opérateurs du code lisent leurs données d'entrées et écrivent leurs données de sorties** (*via* les bases globale et volatile) **de manière centralisée et en séquentiel**. En fin d'opérateur, on complète donc, si nécessaire, les structures de données distribuées afin de pouvoir passer sans encombre le relais à l'opérateur suivant.

L'introduction d'un parallélisme multi-niveaux dans les calculs modaux a un peu complexifié la situation... mais en procurant la contre-partie notable de gros bénéfices en performances. **La principale nouveauté concerne la gestion simultanée de plusieurs sous-communicateurs MPI du communicateur courant**[66].
Pour continuer à assurer la maintenabilité, la développabilité et l'AQ du code, la liste des garde-fous logiciels et des conseils méthodologiques s'est enrichie:

- Pour ne pas passer le sous-communicateur concernant « une tâche/groupe de processeurs » donnés, en paramètre d'entrée des routines, on stocke celui ci dans un objet dédié de la base globale. **En début de chaque routine manipulant du MPI**, on interroge cet objet **pour connaître le sous-communicateur courant**[67]. On ne manipule d'ailleurs jamais directement cet objet. On passe toujours par une « méthode » d'accès[68].

- On ne traite que des **sous-communicateurs emboîtés** afin de n'oublier aucun processeur (équivalent du deadlock pour les sous-communicateurs).

- On ne **change que sous-communicateur que dans des zones circonscrites** du code; Et pas dans des routines de bas niveau (type calcul élémentaire ou résolution de système), plutôt des routines chef d'orchestre. On réinitialise, dès que possible, le communicateur courant au classique `MPI_COMM_WORLD`.

- Avant d'**utiliser une librairie externe parallèle en mode reprise** (par ex. le solve de MUMPS), on vérifie que le communicateur qui lui est associé est bien le même que le communicateur courant.

- **Détruire un sous-communicateur** seulement après s'être assuré que les occurrences de produits externes auxquelles il était associé (si c'est le cas), ont bien été détruites. Sinon la destruction de cette occurrence risque de poser problème: fuite mémoire, comportement erratique, planton...

- **A chaque changement de sous-communicateur,** prévoir une étape transitoire de barrière sur le `MPI_COMM_WORLD`. Cela permet de s'assurer que tous les processeurs actifs sont présents à cette étape du calcul.

- Lors des communications de gros volumes de données (type groupe de vecteurs propres), on peut jouer sur les sous-communicateurs afin d'éviter d'écrouler les réseaux ! On peut, par exemple, **communiquer en cascade entre différents niveaux de sous-communicateurs** emboîtés. Au delà des aspects robustesse, c'est souvent bénéfique en terme de performance.

Remarque :

66 Le fameux `MPI_COMM_WORLD`.
67 *Via* la fonction `COMCOU.f`.
68: *Via* la routine `MPIEXE.F`.

- Ces gains sont effectifs dès à présent dans INFO_MODE et CALC_MODES avec option 'BANDE' découpée en sous-bandes. En réutilisant les mêmes mécanismes[69], ils peuvent probablement se transposer à d'autres fonctionnalités du même type[70] dans le code.

69 Les fonctionnalités afférentes, principalement gestion des sous-communicateurs et interrogations du sous-communicateur courant, ont été encapsulées dans des fonctions dédiées. Elles sont appelables par tous les opérateurs du code.

70 Schémas de calculs dissociables en plusieurs niveaux avec, si possible, un niveau supérieur de type 'embarrassingly parallel'.

5 Récapitulatif du paramétrage

Rappelons les usages de la procédure de dénombrement en fonction des opérateurs de *Code_Aster*:

Sur l'axe réel (méthode de Sturm standard) : GEPs standards (matrices réelles symétriques)

- INFO_MODE,
- CALC_MODES avec OPTION='BANDE',
- CALC_MODES avec OPTION parmi ['BANDE','CENTRE','PLUS_*','TOUT'] + STURM='OUI',
- CALC_MODES avec OPTION parmi ['SEPARE','AJUSTE'],
- CALC_MODES avec option 'BANDE' découpée en sous-bandes.

Dans le plan complexe (méthode APM) : GEPs atypiques (matrices complexes ou non symétriques) et QEPs.

- INFO_MODE.

Méthode	*Mot-clé*	*Valeur par défaut*	*Références*
Sturm Standard	SEUIL_FREQ	0.01	§3.2
	PREC_SHIFT	0.05	§3.2
	NMAX_ITER_SHIFT	3	§3.2
	NIVEAU_PARALLELISME	'COMPLET'	§4
	SOLVEUR	'MULT_FRONT'	§4
APM	NBPOINT_CONTOUR	40	§3.3
	NMAX_ITER_CONTOUR	3	§3.3

Tableau 5.1. Récapitulatif du paramétrage des méthodes de dénombrement .

6 Bibliographie

6.1 Livres/articles/proceedings/thèses

[BP99] O.Bertrand (encadré par B.Philippe-INRIA/IRISA). *Procédure de dénombrement de valeurs propres*. Thèse de l'Université de Rennes 1 (**1999**).
[GLL09][S.Graillat, P.Langlois & N.Louvet. *Algorithms for accurate, validated and fast polynomial evaluation* . Japan J. Indust. Appl. Math., vol.26, pp191-214 (**2009**).
[GM99] B.Gleyse & M.Moflith. *Exact computation of the number of zeros of a real polynomial in the open disk by a determinant representation* . Computers and Mathematics with Applications, vol. 38, pp257-263 (**1999**).
[GM11] S.Graillat & V.Ménissier-Morain. *Compensated Horner scheme in complex floating point arithmetic*. A paraître dans Information and Computation (2011).
[KP11] E.Kamgnia & B.Philippe. *Couting eigenvalues in domains of the complex field.* Rapport INRIA n°7770 (**2011**).
[KP12] E.Kamgnia & B.Philippe. *Calcul de déterminants pour grandes matrices.* Communication à la journée "Valeurs propres" de l'INRIA Rennes (**2012**).
[Hau80] Y.Haugazeau. *Application du théorème de Sylvester à la localisation des valeurs propres.* RAIRO Analyse Numérique, vol.14, 1, pp25-41 (**1980**).
[HSZ08] M.Hochenstenbach, D.Singer & P.Zachlin. *Eigenvalue inclusion regions from inverses of shifted matrices* . Linear Algebra and its Applications, vol.429, pp2481-2496, (**2008**).
[JKL01] H.Jung, D.Kim & I.Lee. *Technique of checking missed eigenvalues for eigenproblem with damping matrix* . Int. J. Numer. Meth. Engng., vol.50, pp55-66 (**2001**).
[JHKL03] J.Jo, H.Jung, M.Ko & I.Lee. *Eigenvalue-counting methods for non-proportionally damped systems* . Int. J. of Solids and Structures, vol.40, pp6457-6472 (**2003**).
[JJKL08] H.Jung, J.Jo, B.Kim & I.Lee. *Improvement of the eigenvalue-couting method based of the argument principle* . Journal of Engineering Mechanics, vol.134, pp907-912 (**2008**).
[LL08] P.Langlois & N.Louvet. *Compensated Horner algorithm in K times the working precision* . Conference RNC-8 (2008).
[NP12] L.B.Nguenang & B.Philippe. *La méthode EIGENCNT et sa version parallèle* . Communication à la journée "Valeurs propres" de l'INRIA Rennes (**2012**).
[Var04] R.S.Varga. *Gershgorin and his circles* . Springer Series in Computational Mathematics, vol.36, Springer-Verlag (**2004**).
[RH97] S.Rombouts & K.Heyde. *An accurate and efficient algorithm for the computation of the characteristic polynomial of a general square matrix* . Journal of the Computational Physics, vol. 140, pp453-458 (**1997**).
[ROO08] S.M.Rump, T.Ogita & S.Oishi. *Accurate floating-point summation part I & II* . SIAM, J.Sci.Comput., vol.31, pp189-224/pp1269-1302 (**2008**).

6.2 Documents internes EDF

[Boi08] O.Boiteau. *Extension des solveurs QZ et Sorensen aux problèmes modaux non symétriques de Code_Aster* . CR-I23/2008/044 (**2008**).
[Boi10a/b/c/d] O.Boiteau. *Documents Code_Aster Utilisateur et de Référence sur le calcul modal* . Doc. U4.52.02, R5.01.01/02 (**2010**).
[Boi11] O.Boiteau. *Procédure de dénombrement de valeurs propres: étude bibliographique, maquettage et chantier logiciel dans Code_Aster*. Note HI-23-2011-02589 (**2011**).

6.3 Ressources Internet

[GMP] *GMP: GNU Multiple Precision arithmetic library* . http://gmplib.org .
[MPFR] G.Hanrot and al. *The MPFR library* . http://www.mpfr.org
[QD] Y.Hida and al. *Double-double and quad-double QD package* . http://crd.lbl.gov/~dhbailey/mpdist .
[XD] W.Demmel and al. XBLAS: A reference implantation for the extended and mixed precision BLAS. http://crd.lbl.gov/~xiaoye.

7 Description des versions du document

Version Aster	Auteur(s) Organisme(s)	Description des modifications
V11.2.13	O.BOITEAU EDF-R&D/SINETICS	Prise en compte EL16710.
V11.3.10/17	O.BOITEAU EDF-R&D/SINETICS	Prise en compte EL17078/20604.

8 Ann exe n°1: Démonstration de la propriété 2

Pour imposer des conditions aux limites linéaires, on utilise une technique de double dualisation[R3.03.01] qui conduit au système généralisé transformé

$$(\tilde{\mathbf{A}}-\sigma\tilde{\mathbf{B}})\tilde{\mathbf{u}}=\left(\begin{pmatrix}\mathbf{A} & \mathbf{C}^{\mathrm{T}} & \mathbf{C}^{\mathrm{T}}\\ \mathbf{C} & -\mathbf{Id} & \mathbf{Id}\\ \mathbf{C} & \mathbf{Id} & -\mathbf{Id}\end{pmatrix}-\sigma\begin{pmatrix}\mathbf{B} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{pmatrix}\right)\begin{pmatrix}\mathbf{u}\\ \mathbf{v}\\ \mathbf{w}\end{pmatrix}=\mathbf{0}\qquad(\tilde{S})$$

On notera par la suite $\mathbf{v_i^j}$ le vecteur colonne nul de taille p $(i=1..\mathrm{p}\,;\,j=2,3)$ sauf à l'indice i, pour lequel il vaut 1, et $\mathbf{0_n}$, le vecteur colonne nul de taille n. Maintenant considérons les trois familles de vecteurs suivantes:

- $n-p$ vecteurs $\mathbf{V_i^1}=\begin{pmatrix}\mathbf{u_i^1}\\ \mathbf{v_i^1}\\ \mathbf{v_i^1}\end{pmatrix}$ qui sont les vecteurs propres du système $(\tilde{S})$. Clairement, d'après la proposition 2 de [R5.01.01], ils sont $\tilde{\mathbf{A}}$ et $\tilde{\mathbf{B}}$-orthogonaux (d'après les propriétés de $\mathbf{A}$ et $\mathbf{B}$) et on note leurs valeurs propres λ_i
- p vecteurs indépendants $\mathbf{V_i^2}=\begin{pmatrix}\mathbf{0_n}\\ \mathbf{v_i^2}\\ \mathbf{v_i^2}\end{pmatrix}$,
- p vecteurs indépendants $\mathbf{V_i^3}=\begin{pmatrix}\mathbf{0_n}\\ \mathbf{v_i^3}\\ -\mathbf{v_i^3}\end{pmatrix}$.

Ces $n+p$ vecteurs forment une base B de l'espace $E=\left\{\tilde{\mathbf{u}}\begin{pmatrix}\mathbf{u}\\ \mathbf{v}\\ \mathbf{w}\end{pmatrix}\ /\ \mathbf{C}\,\mathbf{u}=0\right\}$ des solutions admissibles du problème dualisé (en tant que famille libre d'un espace de dimension finie).
Considérons, pour un nombre réel σ donné, la forme quadratique associée à la matrice $(\tilde{\mathbf{A}}-\sigma\tilde{\mathbf{B}})$

$$\Phi_\sigma(\tilde{\mathbf{u}})=\langle(\tilde{\mathbf{A}}-\sigma\tilde{\mathbf{B}})\tilde{\mathbf{u}},\tilde{\mathbf{u}}\rangle,\quad \tilde{\mathbf{u}}\in E$$

En décomposant sur la base B engendrée par les vecteurs précédents

$$\tilde{\mathbf{u}}=\sum_{i=1,n-p}a_i^1\mathbf{V_i^1}+\sum_{i=1,n-p}a_i^2\mathbf{V_i^2}+\sum_{i=1,n-p}a_i^3\mathbf{V_i^3}$$

on obtient

$$\Phi_\sigma(\tilde{\mathbf{u}})=\sum_{i=1,n-p}(a_i^1)^2(\lambda_i-\sigma)(\mathbf{u_i^{1T}}\mathbf{B}\,\mathbf{u_i^1})+\sum_{i=1,p}(-4)(a_i^3)^2$$

(on a utilisé en particulier les propriétés d'orthogonalité des familles de vecteurs $\mathbf{V}_i^j$ et la relation $(\mathbf{u}_i^1,\mathbf{C}^{\mathrm{T}}\mathbf{v}_i^2)=(\mathbf{C}\,\mathbf{u}_i^1,\mathbf{v}_i^2)=0$). D'où, en notant L_i la forme linéaire qui associe à $\tilde{\mathbf{u}}$ sa $i^{\text{ème}}$ coordonnée dans la base B :

$$\Phi_\sigma(\tilde{\mathbf{u}})=\sum_{i=1,n-p}L_i^2(\tilde{\mathbf{u}})(\lambda_i-\sigma)(\mathbf{u_i^{1T}}\mathbf{B}\,\mathbf{u_i^1})+\sum_{i=1,p}0\,L_{n-p+i}^2(\tilde{\mathbf{u}})+\sum_{i=1,p}(-4)\,L_{n+i}^2(\tilde{\mathbf{u}})$$

Clairement les $(L_i)_{i=1,n+p}$ sont linéairement indépendantes d'où une lecture immédiate des termes de la signature suivant le signe des facteurs. D'ailleurs, on remarque que cette décomposition comporte obligatoirement p zéros et p signes moins. Les relations de la propriété se déduisent alors tout naturellement, en utilisant le principe d'inertie de Sylvester (qui nous assure que cette décomposition est

invariante), les relations de $\tilde{\mathbf{A}}$ et $\tilde{\mathbf{B}}$ - orthogonalité de la propriété 2, les propriétés du tableau 3.3-1 et en remarquant que:

$$\tilde{\mathbf{u}}_i^{\mathrm{T}} \tilde{\mathbf{B}} \tilde{\mathbf{u}}_j = \mathbf{u}_i^{\mathrm{T}} \mathbf{B} \mathbf{u}_i = \frac{\tilde{\mathbf{u}}_i^{T} \tilde{\mathbf{A}} \tilde{\mathbf{u}}_j}{\lambda_j} = \frac{\mathbf{u}_i^{T} \mathbf{A} \mathbf{u}_j}{\lambda_j} \text{ pour } \lambda_j \neq 0$$

La restriction $\sigma \geq 0$ du cas de figure (2) provient du fait que si $\mathbf{B}$ est définie positive alors le spectre du problème (S) est positif et donc le shift avec une valeur strictement négative n'a aucun intérêt (on se retrouve dans le cadre d'application du corollaire 1).