

Structure de données sd_eigensolver

Résumé :

Ce document décrit la structure de données `EIGENSOLVER`. Celle-ci explicite le problème modal traité (GEP et QEP) ainsi que les paramètres attenants au solveur modal choisi.

Table des Matières

1 Généralités.....	3
2 Arborescence.....	3
3 Contenu des objets de base.....	3
3.1 Vecteur ESVK.....	3
3.2 Vecteur ESVI.....	4
3.3 Vecteur ESVR.....	4

1 Généralités

Cet objet de type `EIGENOLVER` a pour fonction de stocker et de véhiculer entre les différentes routines du code (pour l'instant qu'en interne de la commande `CALC_MODES`¹), les informations décrivant un problème modal au sens *Code_Aster* (GEP ou QEP) ainsi que les paramètres attenants au solveur modal choisi (`SORENSEN`, `TRI_DIAG`, `JACOBI` ou `QZ`). Cet objet est créé sur la base volatile.

Il ne doit être créé et rempli que *via* `VPINIS`. La cohérence de ses paramètres et celles de certains objets sous-jacents (par ex. les matrices définissant le problème) sont contrôlées *via* `VPVERS`. Cette routine fait aussi intervenir, optionnellement, la définition au sens Python de cet objet (`sd_eigensolver.py`) *via* un appel à la routine `cheksd`. Dans `CALC_MODES` cette option est toujours activée.

D'autres routines dédiées (type « méthode » au sens C++) permettent en outre de manipuler cet objet :

- Le relire entièrement et mettre à jour certaines variables auxiliaires : `VPLECS`.
- Lire un coefficient donné de cet objet : `VPLECI`.
- Ecrire un coefficient donné de cet objet : `VPECRI`.

Normalement, tous les champs de cet objet, une fois qu'ils ont été remplis et vérifiés, ne doit plus être modifiés. Les seules exceptions concernent, pour l'instant, dans `CALC_MODES`:

- `ESVK(2)` et `ESVK(3)` : afin d'intervertir les matrices de travail si `OPTION='PLUS_Grande'`,
- `ESVI(1)` et `ESVI(2)` : recalcul/correction du nombre de modes et de la taille de l'espace de projection.

Lors d'une modification de cette structure de données il faut donc veiller à :

- *mettre en cohérence, si nécessaire, les sources mentionnées ci-dessus,*
- *mettre à jour, si nécessaire, le catalogue `sd_eigensolver.py`,*
- *mettre à jour les documentations (cette doc. D et si nécessaire les doc. U concernant le calcul modal),*
- *enrichir ou modifier, si nécessaire, quelques cas-tests.*

2 Arborescence

```
SOLVEUR (K19) ::=record
  ♦ '.ESVK' : OJB S V K24 long=20 (initialisé à ' ')
  ♦ '.ESVR' : OJB S V R long=15 (initialisé à r8vide())
  ♦ '.ESVI' : OJB S V I long=15 (initialisé à isnnem())
```

3 Contenu des objets de base

3.1 Vecteur `ESVK`

`ESVK` :

Paramètres généraux décrivant le problème modal

`ESVK(1)` : type de résultat ('DYNAMIQUE', 'MODE_FLAMB' ou 'GENERAL').

`ESVK(2)` : nom de la matrice fournie *via* le mot-clé `MATR_RIGI` si `DYNAMIQUE/MODE_FLAMB`, `MATR_A` si `GENERAL`.

`ESVK(3)` : nom de la matrice fournie *via* le mot-clé `MATR_MASS` si `DYNAMIQUE`, `MATR_RIGI_GEOM` si `MODE_FLAMB`, `MATR_B` si `GENERAL`.

¹ Routine `op0045.F90`.

ESVK(4) : nom de la matrice fournie *via* le mot-clé MATR_AMOR si DYNAMIQUE, MATR_B si GENERAL.

ESVK(5) : type de calcul. Les valeurs possibles sont :
'PLUS_PETITE', 'CENTRE', 'BANDE' ou 'TOUT' (si MODE_FLAMB ou GENERAL),
Idem plus 'PLUS_GRADE' (si DYNAMIQUE).

ESVK(6) : nom du solveur modal ('SORENSEN', 'TRI_DIAG', 'JACOBI' ou 'QZ').

ESVK(7) : valeur de l'option de pré-capture des modes rigides ('SANS' ou 'MODE_RIGIDE').

ESVK(8) : comportement à adopter en cas de bande de calcul vide, option 'STOP_BANDE_VIDE' ('OUI' ou 'NON').

ESVK(9) : nom de la table générée par INFO_MODE (option TABLE_FREQ/TABLE_CHAR_CRIT).

ESVK(10) : comportement à adopter en cas d'erreur, option 'STOP_ERREUR' ('OUI' ou 'NON').

ESVK(11) : activation du test de Sturm, option 'STURM' ('OUI' ou 'NON').

ESVK(12) à ESVK(15) : inutilisés.

Paramètres spécifiques aux solveurs modaux (cf. ESVK(6))

ESVK(16) : type d'approche en QEP ('R', 'I' ou 'C').
si ESVK(6)='QZ'
ESVK(17) : type de méthode QZ ('QZ_SIMPLE', 'QZ_EQUI' ou 'QZ_QR').
Sinon :
ESVK(17) : inutilisé.
ESVK(18) à ESVK(20) : inutilisés.

3.2 Vecteur ESVI

ESVI :

Paramètres généraux décrivant le problème modal

ESVI(1) : nombre de modes à calculer.

ESVI(2) : valeur du paramètre DIM_SOUS_ESPACE.

ESVI(3) : valeur du paramètre COEF_DIM_ESPACE.

ESVI(4) : valeur du paramètre NMAX_ITER_SHIFT.

ESVI(5) : nombre de fréquences ou charges critiques saisies par l'utilisateur.

ESVI(6) à ESVI(10) : inutilisés.

Paramètres spécifiques aux solveurs modaux (cf. ESVK(6))

si ESVK(6)='TRI_DIAG'
ESVI(11) : valeur de NMAX_ITER_ORTHO .
ESVI(12) : valeur de NMAX_ITER_QR .
ESVI(13) à ESVI(15) : inutilisés.

si ESVK(6)='JACOBI'
ESVI(11) : valeur de NMAX_ITER_BATHE .
ESVI(12) : valeur de NMAX_ITER_JACOBI .
ESVI(13) à ESVI(15) : inutilisés.

si ESVK(6)='SORENSEN'
ESVI(11) : valeur de NMAX_ITER_SORENSEN .
ESVI(12) à ESVI(15) : inutilisés.

si ESVK(6)='QZ'
ESVI(11) à ESVI(15) : inutilisés.

3.3 Vecteur ESVR

ESVR :

Paramètres généraux décrivant le problème modal

ESVR(1) : première borne de recherche saisie par l'utilisateur (mot-clé `FREQ` ou `CHAR_CRIT`).
ESVR(2) : deuxième borne de recherche saisie par l'utilisateur (mot-clé `FREQ` ou `CHAR_CRIT`).
ESVR(3) : valeur du paramètre `CALC_FREQ/PREC_SHIFT`.
ESVR(4) : valeur définissant le « zéro modal » : $(2.\pi.\text{SEUIL_FREQ})^{**2}$ si `DYNAMIQUE`,
`SEUIL_CHAR_CRIT` sinon.
ESVR(5) : valeur du paramètre `VERI_MODE/PREC_SHIFT` .
ESVR(6) : valeur du paramètre `VERI_MODE/SEUIL` .
ESVR(7) à ESVR(10) : inutilisés.

Paramètres spécifiques aux solveurs modaux (cf. `ESVK(6)`)

si `ESVK(6)='TRI_DIAG'`
 ESVR(11) : valeur de `PREC_ORTHO` .
 ESVR(12) : valeur de `PREC_LANCZOS` .
 ESVR(13) à ESVR(15) : inutilisés.
si `ESVK(6)='JACOBI'`
 ESVR(11) : valeur de `PREC_BATHE` .
 ESVR(12) : valeur de `PREC_JACOBI` .
 ESVR(13) à ESVR(15) : inutilisés.
si `ESVK(6)='SORENSEN'`
 ESVR(11) : valeur de `PREC_SOREN` .
 ESVR(12) : valeur de `PARA_ORTHO_SOREN` .
 ESVR(13) à ESVR(15) : inutilisés.
si `ESVK(6)='QZ'`
 ESVR(11) à ESVR(15) : inutilisés.