

Structure de Données sd_l_charge

Table des Matières

1 Généralités.....	3
2 Objet SD_L_CHARGES – Ancienne version	3
2.1 Arborescence.....	3
2.2 Contenu des objets.....	3
2.2.1 Objet .INFC.....	3
2.2.2 Objet .LCHA.....	5
2.2.3 Objet .FCHA.....	5
3 Objet SD_L_CHARGES – Nouvelle version	5
3.1 Principes	5
3.1.1 Un chargement, qu'est-ce que c'est ?.....	5
3.1.2 Fonction multiplicatrice.....	5
3.1.3 Définition du type de charge.....	5
3.1.4 La notion de genre de charge.....	6
3.1.5 La notion de mot-clef de charge.....	6
3.1.6 Multiplicité des genres/mots-clefs.....	6
3.1.7 Notion d'objet et de préfixe.....	7
3.2 Contenu de la SD et accès aux informations	7
3.2.1 Arborescence.....	7
3.2.2 Routines utilitaires	7
3.2.3 Vérifications de la liste des charges.....	8
3.3 Calcul des chargements	8
3.3.1 Principe général.....	8
3.3.2 Les routines de calcul des chargements.....	8
3.3.3 Les routines de pré-assemblage.....	8
3.4 Ajout d'un nouveau chargement.....	9

1 Généralités

Un objet de type `sd_l_charges` est créé et utilisé dans les commandes globales utilisant les charges. Il donne des informations sur les chargements utilisés dans la commande. Cette SD sert à la fois dans les opérateurs de calcul mais aussi dans le post-traitement. Elle est donc créée sur la base globale ou volatile suivant les cas :

- Pour les opérateurs de statique linéaire (`MECA_STATIQUE`) ou `NON_LINEAIRE` (`STAT_NON_LINE`), pour la thermique (`THER_LINE` , `THER_NON_LINE` et `THER_LINE_MO`) et pour la dynamique `non_LINEAIRE` (`DYNA_NON_LINE`), la SD est créée sur la base globale, elle est attachée à la SD résultat produit par ces opérateurs ;
- Pour les opérateurs de dynamique linéaire (`DYNA_VIBRA`), pour la mécanique de la rupture (`CALC_G`), la SD est créée sur la base volatile ;
En post-traitement, elle est soit créée localement (mot-clef `EXCIT`), soit récupérée dans les SD résultat la stockant. C'est le cas des opérateurs `CALC_CHAMP` et `POST_ELEM` .
On la crée également dans `LIRE_RESU` .
Il existe actuellement deux versions de cette SD :
- L'ancienne version est la plus répandue ;
- La nouvelle version, introduite dans l'opérateur `DYNA_VIBRA` a pour vocation à remplacer l'ancienne version à terme ;

2 Objet `SD_L_CHARGES` – Ancienne version

2.1 Arborescence

```
sd_l_charges (K19) ::=record
  (o)  '.INFC' : OJB  S  V  I
  (o)  '.LCHA' : OJB  S  V  K24
  (o)  '.FCHA' : OJB  S  V  K24
```

2.2 Contenu des objets

2.2.1 Objet `.INFC`

Soit `nchar` le nombre de charges utilisées dans la commande globale (nombre d'occurrences du mot-clef `EXCIT`)

- dimension = 4 x `nchar` + 3 en mécanique
- dimension = 2 x `nchar` + 1 en thermique

En mécanique

`.INFC(1)` = `nchar`

Les valeurs `INFC(2)` à `INFC(1+nchar)` sont réservées aux charges de type Dirichlet:

Pour $1 \leq \text{ichar} \leq \text{nchar}$

<code>INFC(1+ichar)</code>	= code
	= 0 si pas de charge de Dirichlet (dualisée) (ou si la charge ne contient que des relations éliminées)
	= 1 si la charge est issue de <code>AFFE_CHAR_MECA</code>
	= 2 si la charge est issue de <code>AFFE_CHAR_MECA_F</code> et si elle est indépendante du temps
	= 3 si la charge est issue de <code>AFFE_CHAR_MECA_F</code> et si elle est dépendante du temps
	= 4 force suiveuse
	= 5 force pilotée
	= -1 si la charge est issue de <code>AFFE_CHAR_CINE</code>
	= -2 si la charge est issue de <code>AFFE_CHAR_CINE_F</code> et si elle est indépendante du temps

= -3 si la charge est issue de AFFE_CHAR_CINE_F et si elle est dépendante du temps

-

Les valeurs infc (2+nchar) à infc (1+2*nchar) sont réservées aux charges mécaniques de type Neuman :

```
Pour 1 ≤ ichar ≤ nchar
  INFC (1+nchar+ichar)
    = 0 si pas de charge
    = 1 si la charge est issue de AFFE_CHAR_MECA
    = 2 si la charge est issue de AFFE_CHAR_MECA_F et si elle est indépendante du temps
    = 3 si la charge est issue de AFFE_CHAR_MECA_F et si elle est dépendante du temps

  .INFC (1+2*nchar+1) = inutilisé
  .INFC (1+2*nchar+2) = nombre de charges donnant des forces de Laplace.
```

Les valeurs infc (3+3*nchar+1) à infc (3+4*nchar) sont réservées aux charges différentielles:

```
Pour 1 ≤ ichar ≤ nchar
  INFC (3+3*nchar+ichar)
    = 1 si la charge est différentielle
    = 0 sinon
```

En thermique

```
.INFC (1) = nchar
```

Les valeurs INFC (2) à INFC (1+nchar) sont réservées aux charges de type Dirichlet :

```
Pour 1 ≤ ichar ≤ nchar

  INFC (1+ichar) = code
    = 0 si pas de charge de Dirichlet (dualisée)
      (ou si la charge ne contient que des relations éliminées)
    = 1 si la charge est issue de AFFE_CHAR_THER
    = 2 si la charge est issue de AFFE_CHAR_THER_F et si elle est indépendante du temps
    = 3 si la charge est issue de AFFE_CHAR_THER_F et si elle est dépendante du temps
    = 4 force suiveuse
    = -1 si la charge est issue de AFFE_CHAR_CINE
    = -2 si la charge est issue de AFFE_CHAR_CINE_F et si elle est indépendante du temps
    = -3 si la charge est issue de AFFE_CHAR_CINE_F et si elle est dépendante du temps
```

Les valeurs INFC (1+nchar+1) à INFC (1+2*nchar) sont réservées aux charges de type Neuman :

```
Pour 1 ≤ ichar ≤ nchar

  INFC (1+nchar+ichar)
    = 0 si pas de charge
    = 1 si la charge est issue de AFFE_CHAR_THER
```

- = 2 si la charge est issue de `AFFE_CHAR_THER_F` et si elle est indépendante du temps
- = 3 si la charge est issue de `AFFE_CHAR_THER_F` et si elle est dépendante du temps

2.2.2 Objet `.LCHA`

`.LCHA` : S V K24 dimension = nchar
`LCHA` contient le nom de toutes les charges impliquées dans la commande globale.

2.2.3 Objet `.FCHA`

`.FCHA` : S V K24 dimension = nchar
`FCHA` contient le nom de la fonction multiplicatrice appliquée à la charge.

3 Objet `SD_L_CHARGES` – Nouvelle version

3.1 Principes

3.1.1 Un chargement, qu'est-ce que c'est ?

Un chargement est en général défini en deux temps :

- Description du chargement dans les opérateurs `AFFE_CHAR_*` ;
- Application du chargement dans la commande (sous le mot-clef `EXCIT/CHARGE`)

Il existe néanmoins un deuxième moyen de définir un chargement :*

- Définition d'un `CHAM_NO ( VECT_ASSE )` ou d'un `VECT_ASSE_GENE` à l'aide des opérateurs de manipulations des vecteurs (`CALC_VECT_ELEM`, `ASSE_VECTEUR`, etc) ou suite à un calcul précédent ;
 - Application du chargement dans la commande (sous le mot-clef `EXCIT/VECT_ASSE`) ;
- Cette deuxième manière de faire est très utilisée en dynamique.

Dans le mot-clef `EXCIT`, on peut également définir :

- une fonction multiplicatrice du temps réelle ou complexe ;
- Un type d'application du chargement (mot clef `TYPE_CHARGE`) utilisé surtout dans les opérateurs non-linéaires ;

3.1.2 Fonction multiplicatrice

En interne des commandes, on utilise toujours une fonction multiplicatrice (`FONC_MULT` ou `FONC_MULT_C`). Si l'utilisateur n'a pas précisé, c'est une fonction constante unité définie en interne, ou aussi une fonction non-unité avec un coefficient donné par l'utilisateur par `COEF_MULT` ou `COEF_MULT_C`. Dans le cas complexe, il est possible de donner la phase (`PHAS_DEG`) et la puissance de la pulsation (`PUIS_PULS`) du chargement complexe, écrit sous forme d'exponentielle complexe.

3.1.3 Définition du type de charge

Quand l'utilisateur définit un chargement dans `AFFE_CHAR_*`, il utilise un mot-clef particulier. `AFFE_CHAR_*` procède alors de plusieurs manières différentes :

1. Création d'une `CARTE` contenant l'information nécessaire sur tout le modèle. Par exemple, si le chargement est une pression répartie, ça veut dire que la pression sera définie non-nulle uniquement sur les mailles affectées par le chargement. Sur le reste du `MODELE`, la pression sera nulle¹ ;
2. Création d'une `CARTE` sur une partie du modèle (`GROUP_MA` défini) ;
3. Création d'objets spécifiques qui ne sont pas des cartes.

Le cas 1 est le plus fréquent. Le cas 2 est beaucoup plus rare: il ne concerne en mécanique que les chargements de Dirichlet et `FORCE_NODALE`.

¹ Une pression nulle et non une pression non-définie. Ce qui implique en pratique qu'un élément fini doit être capable au moins de traiter le cas d'une charge nulle.

Le cas 3 concerne quelques chargements (en mécanique: EVOL_CHAR, FORCE_ELEC et AFFE_CHAR_CINE par exemple).

L'information sur le type de chargement est retrouvable par le nom de l'objet (CARTE ou autre objet) créée par AFFE_CHAR_MECA . Mais il y a quelques cas ambigus (information perdue). E n tout état de cause, le lieu d'application de la charge est perdu car, en général, on défini la CARTE sur tout le MODELE (cas 2).

3.1.4 La notion de genre de charge

Un **genre** de charge regroupe les chargements ayant les points communs suivants:

- Unité phénoménologique : Neumann/Dirichlet sur un PHENOMENE donné ;
- Unité de description : le chargement est décrit dans le même opérateur (AFFE_CHAR_* par exemple) et est construit de la même façon: même objet (carte) et un paramètre associé , un LIGREL , une option ;
- Unité de programmation : le calcul du chargement est fait dans une seule routine

Il y a (actuellement) quinze genres de charge :

- DIRI_DUAL : AFFE_CHAR_MECA avec Dirichlet dualisés ;
- DIRI_ELIM : AFFE_CHAR_CINE ;
- NEUM_MECA : Chargement de Neumann en mécanique ;
- PRE_SIGM : comme son nom l'indique ;
- VITE_FACE : comme son nom l'indique ;
- IMPE_FACE : comme son nom l'indique ;
- EVOL_CHAR : comme son nom l'indique ;
- SIGM_CABLE : comme son nom l'indique ;
- FORCE_ELEC : comme son nom l'indique ;
- INTE_ELEC : comme son nom l'indique ;
- ONDE_FLUI : comme son nom l'indique ;
- ONDE_PLANE : comme son nom l'indique ;
- VECT_ASSE_CHAR : VECT_ASSE défini par AFFE_CHAR_MECA ;
- VECT_ASSE : CHAMNO directement en entrée d'EXCIT ;
- VECT_ASSE_GENE : VECT_ASSE_GENE directement en entrée d'EXCIT ;

Un genre regroupe plusieurs **types** de chargement. Par exemple NEUM_MECA regroupe des chargements de type Neumann en mécanique, définis dans AFFE_CHAR_MECA* et avec des mots-clefs aussi divers que FORCE_NODALE ou FORCE_COQUE.

3.1.5 La notion de mot-clef de charge

Identifiable par le nom de la carte ou de l'objet (sauf les cas ambigus). Ne sert que dans quelques cas (impressions de débogage, repérer des chargements particuliers comme la pesanteur ou commande CALC_G).

3.1.6 Multiplicité des genres/mots-clefs

Dans un seul AFFE_CHAR_*, on peut définir plusieurs chargements très différents (plusieurs genre et mots-clefs). Or le nom de la charge (concept du AFFE_CHAR_* ou nom du VECT_ASSE/VECT_ASSE_GENE) sert d'itérateurs dans la sd_l_charges.

Il y a autant de charges que d'occurrences du mots-clef facteur EXCIT. Mais il y a plusieurs chargements par occurrence.

Pour identifier le genre, on utilise donc un entier codé et donc 30 genres de charges différents sont possibles. Par décodage de cet entier, on peut dire si ce genre ou pas est présent dans l'occurrence de la charge.

Pour identifier le mot-clef de charge, on utilise donc deux entiers codés et donc 60 mots-clefs de charges différents sont possibles.

3.1.7 Notion d'objet et de préfixe

Un chargement est défini par un ou plusieurs objets.

Le nom de l'objet est toujours préfixé de la même manière dans le cas d'AFFE_CHAR_*. Le **préfixe** PREFOB est construit sur la base suivante :

- PREFOB(1:8) : nom du concept issu d'AFFE_CHAR_* ;
- PREFOB(9:13) : chaîne identifiant le phénomène soit .CHAC , .CHME ou .CHTH (respectivement, acoustique: AFFE_CHAR_ACOU , mécanique: AFFE_CHAR_MECA , ou thermique: AFFE_CHAR_THER) ;

Avec ce préfixe :

- On identifie l'objet : une carte ou un autre objet . E n identifiant l'objet, on peut identifier le genre et, éventuellement le mot-clef ;
- Pour les chargements à LIGREL réduit, on peut construire le nom du LIGREL à partir du préfixe ;

3.2 Contenu de la SD et accès aux informations

3.2.1 Arborescence

```
sd_l_charges (K19) ::=record
  (o) '.NCHA' : OJB S V K8 LONUTI=nchar
  (o) '.CODC' : OJB S V I LONUTI=nchar
  (o) '.TYPC' : OJB S V K8 LONUTI=nchar
  (o) '.TYPA' : OJB S V K16 LONUTI=nchar
  (o) '.PREO' : OJB S V K24 LONUTI=nchar
  (o) '.NFON' : OJB S V K8 LONUTI=nchar
  (o) '.TFON' : OJB S V K16 LONUTI=nchar
  (o) '.VFON' : OJB S V R LONUTI=nchar
```

Tous ces objets sont indicé par le numéro de charge *ichar* , sachant que *nchar* correspond au nombre d'occurrences du mot-clef facteur *EXCIT* .

- '. NCHA ' contient le nom des charges (concept issu d' AFFE_CHAR_* ou VECT_ASSE) – Accès en lecture par *lislch.f* ;
- '. CODC ' contient le genre des charges (entier codé) – Accès en lecture par *lislco.f* ;
- '. TYPC ' contient le type de la charge (complexe, réel, fonction) : REEL , COMP , FONC_F0 (fonction ² quelconque) et FONC_FT (fonction du temps) – Accès en lecture par *lisltc.f* ;
- '. TYPA ' contient le type d'application de la charge (chargement fixe, piloté, suiveur, Dirichlet différentiel) : FIXE_CSTE , FIXE_PILO , SUIV et DIDI – Accès en lecture par *lislta.f* ;
- '. PREO ' contient le préfixe des objets de la charge – Accès en lecture par *lislle.f* ;
- '. NFON ' contient le nom de la fonction multiplicatrice – Accès en lecture par *lislnf.f* ;
- '. TFON ' contient le type de la fonction multiplicatrice : fonction ou constante réelle ou complexe (FONCT_REEL , FONCT_COMP , CONST_REEL et CONST_COMP) – Accès en lecture par *lisltf.f* ;
- '. VFON ' contient les paramètres de l'exponentielle complexe dans le cas d'une fonction multiplicatrice complexe – Accès en lecture par *lispcp.f* ;

Important : il est interdit d'accéder directement aux objets de la SD par leur nom, il faut utiliser les routines d'accès.

3.2.2 Routines utilitaires

On crée la SD (objets vides) dans la routine *lisrcs.f*.

On imprime son contenu (mode INFO=2) grâce à *lisimp.f*.

On lit le mot-clef *EXCIT* et on remplit la SD dans *lislec.f* .

3.2.3 Vérifications de la liste des charges

Quelques vérifications sont proposées en standard dans la SD. La routine faisant ces vérifications est *lischk.f*, elle est appelée systématiquement après la création et le remplissage de la SD. Ces vérifications sont :

- 2 Le type de la charge peut être une fonction définie par AFFE_CHAR_*_F Mais dans ce cas, ce ne peut être qu'une fonction à variable *réelle* .

- Cohérence des modèles : toutes les charges reposent sur le même modèle et sont cohérentes avec le modèle du calcul (cette limitation est provisoire, en attendant de réfléchir au cas des transitoires) – routine `liscom.f` ;
- Cohérence entre les chargements et le phénomène : toutes les charges reposent sur le même phénomène et sont cohérentes avec le phénomène de l'opérateur – routine `liscap.f` ;
- Cohérence entre les chargements et la commande : le genre de chargement est calculable sur la commande – routine `liscom.f` ;
- Interdiction des doublons : on interdit que le même chargement soit présent deux fois – routine `lisdbl.f` ;
- Vérifications du type charge. Pour l'instant, pas assez automatique, quelques vérifications diverses (pilotage, chargements suiveurs). À terme, ces incompatibilités seront probablement plus automatisées – routine `lisver.f` ;

3.3 Calcul des chargements

3.3.1 Principe général

Pour calculer effectivement un chargement (et donc créer le second membre pour l'intégrer dans une résolution), il y a deux cas :

1. Les chargements standards : ils se construisent à partir de l'assemblage dans un `CHAM_NO` de vecteurs élémentaires. Il y a donc une phase de calcul de ces `VECT_ELEM` (appel à `CALCUL`) et une phase d'assemblage ;
2. Les chargements non-standards : le `CHAM_NO` est recopié de l'objet déjà construit par ailleurs (`VECT_ASSE` par exemple) ou c'est un chargement particulier (contact par exemple) ;

3.3.2 Les routines de calcul des chargements

La routine la plus importante est `vechme.f`. Calcule le genre `NEUM_MECA`, `EVOL_CHAR` et d'autres choses en glutant parfois. Dans la nouvelle version, `vechme.f` est remplacé par `vechms.f`, et calcule uniquement le genre `NEUM_MECA`, il est séparé en deux morceaux :

- Préparation des champs d'entrée (standard avec une carte de paramètre) – Routine `vechmp.f` ;
- Calcul effectif des `VECT_ELEM` – Routine `vechmx.f` ;

Pour la sensibilité : on passe de `vechde.f` à `vechd2.f` (provisoire avant résorption sensibilité)

Pour les Dirichlet dualisés : on passe de `vedime.f` à `vedimd.f`

Pour `EVOL_CHAR`, on passe de `vechme.f` à `veevoc.f`. En pratique un `EVOL_CHAR` contient des chargements de type `NEUM_MECA`. Donc, pour le calcul `veevoc.f` va appeler `vechmp.f/vechmx.f` en construisant une `sd_l_charges` provisoire.

Le traitement des `VECT_ASSE/VECT_ASSE_GENE` est prévu dans `cnvesl.f`.

Le traitement de `VECT_ASSE` venant d'`AFFE_CHAR_MECA` (genre `VECT_ASSE_CHAR`) est prévu dans `veassc.f`.

3.3.3 Les routines de pré-assemblage

Comme tous les chargements sont construits à l'aide d'une fonction multiplicatrice, on fait l'assemblage en deux séquences :

- Pré-assemblage : les `VECT_ELEM` sont assemblés dans une liste de `CHAM_NO`, dans la routine `asvepr.f` qui construit une liste ;
- Combinaison linéaire des `CHAM_NO` : on combine les `CHAM_NO` pré-assemblés avec les fonctions multiplicatrices dans la routine `ascomb.f` ;

3.4 Ajout d'un nouveau chargement

Pour ajouter un nouveau chargement, il faut :

- Faire les impacts nécessaires dans `AFFE_CHAR_*` ;
- Identifier le genre de la charge par comparaison avec ce qui existe déjà ;

- Identifier les opérateurs permettant l'utilisation de cette charge ;
- Écrire les routines qui calculeront cette charge (§ 8 et éventuels calculs élémentaires) ;
- Impacter la routine principale `lisdef.f` (compléter les DATA) ;
- Ajouter des protections d'usage dans `lischk.f` (identifier les opérateurs permettant l'utilisation de cette charge par exemple, rendre incompatible avec le pilotage, etc...)