Structures de données sd_carte , sd_cham_no , sd_cham_elem et sd_resuelem

Résumé :

Table des Matières

1 Généralités

Les structures de données représentant les champs sont :

- sd_cham_no : champ sur les nœuds d'un maillage
- sd_carte : champ sur les mailles d'un maillage
- sd_cham_elem : champ sur les éléments d'un ligrel
- sd_resuelem : champ de matrices (ou vecteurs) élémentaires sur les éléments d'un ligrel

Nous appelons "grandeur" un "vecteur" de composantes (CMP) du champ.
Par exemple, pour un champ de déplacement : ('DX','DY','DZ').
Un champ discrétisé est un ensemble de grandeurs localisées sur des nœuds, des points de Gauss ou des mailles.
Toutes les grandeurs d'un champ n'ont pas forcément les mêmes composantes : par exemple, sur certaines parties du maillage, les nœuds peuvent avoir 6 CMPS de déplacement (éléments de poutre) alors que sur d'autres parties, les nœuds n'ont que 3 CMPS (éléments volumiques).

Les composantes d'une grandeur sont un sous-ensemble des CMPS déclarées dans le catalogue des grandeurs [D4.04.01]. Pour décrire une grandeur, outre ses valeurs numériques, il faut savoir de quelles CMPS il s'agit ; pour cela, on utilise la notion de "descripteur_grandeur" qui décrit la présence (ou non) de l'ensemble des CMPS du catalogue. Cette notion est décrite ci-dessous.

- Les sd_carte sont des champs discrétisés sur les mailles d'un maillage (ou les mailles tardives d'un ligrel). Il existe 1 grandeur par maille,
- les sd_cham_no sont des champs discrétisés sur les nœuds d'un maillage (ou les nœuds tardifs d'un ligrel). Il existe 1 grandeur par nœud,
- les sd_cham_elem sont des champs discrétisés sur les éléments d'un ligrel. Il peut exister plusieurs grandeurs par élément (par exemple une grandeur par point de Gauss ou par nœud). Les points de dicrétisation (nœuds ou point de Gauss) peuvent avoir des sous-points ; si c'est le cas, tous les points ont le même nombre de sous-points,
- les sd_resuelem sont des champs discrétisés sur les éléments d'un ligrel. Les grandeurs associées à de tels champs sont les grandeurs dites "élémentaires" : matrices élémentaires ou vecteurs élémentaires. L'ensemble des valeurs d'un resuelem peut être volumineux, c'est pourquoi l'objet contenant ces valeurs (.RESL) a une structure de collection dispersée.

Remarque importante :

Les structures de données décrites ici ne sont pas d'une utilisation facile. Ce sont des SD normalement utilisées dans des opérations de bas niveaux : calculs élémentaires, assemblages, résolutions... Lorsqu'on veut lire ou écrire dans de telles SD, il est souvent préférable de les transformer préalablement en SD plus commodes à utiliser (champs "simples"). Les routines de transformation ad hoc : CNOCNS, CNSCNO, CELCES, CARCES... sont décrites dans [D4.06.06].

2 Arborescences

```
sd_carte  (K19)  ::=record
  ♦ '.NOMA'      :   OJB   S   E  K8
  ◊ '.NOLI'      :   OJB   S   V  K24
  ♦ '.DESC'      :   OJB   S   V  I
  ◊ '.LIMA'      :   OJB   XC  V  I
  ♦ '.VALE'      :   OJB   S   V  R/C/K8/K24/...

sd_cham_no  (K19)  ::=record
  ♦ '.DESC'      :   OJB   S   V  I
  ♦ '.REFE'      :   OJB   S   V  K24
  ♦ '.VALE'      :   OJB   S   V  R/C/K8

sd_cham_elem  (K19)  ::=record
  ♦ '.CELK'      :   OJB   S   V  K24
  ♦ '.CELD'      :   OJB   S   V  I
  ♦ '.CELV'      :   OJB   S   V  R/C/K8/...

sd_resuelem  (K19)  ::=record
  ♦ '.NOLI'      :   OJB   S   V  K24
  ♦ '.DESC'      :   OJB   S   V  I
  ♦ '.RESL'      :   OJB   XD  V  R/C
  ◊ '.RSVI'      :   OJB   XC  V  I
```

3 Contenu des objets JEVEUX

3.1 DESCRIPTEUR_GRANDEUR

C'est un vecteur d'entiers. Il décrit les CMPS présents effectivement dans une grandeur.

Toutes les CMPS possibles d'une grandeur sont décrites dans le catalogue des GRANDEURS. Elles y sont ordonnées. Pour décrire les CMPS effectivement présentes dans une grandeur on décide de garder un vecteur de booléens qui répond à la question suivante : la ième CMP (dans l'ordre du catalogue des grandeurs) est-elle présente dans la grandeur que l'on veut décrire ? Pour économiser de la place mémoire (et disque), on décide de "coder" ce vecteur de booléens sur un vecteur d'entiers : sur chaque entier (appelé entier_codé), on code 30 booléens.

Exemple :

Si la grandeur 'DEPL_R' était décrite dans le catalogue par :

DX	DY	DZ	DRX	DRY	DRZ	LAGR

Sur un élément de type poutre le descripteur_grandeur vaut 126. En effet :

	DX	DY	DZ	DRX	DRY	DRZ	LAGR
	1	1	1	1	1	1	0
126 =	2^1	$+ 2^2$	$+ 2^3$	$+ 2^4$	$+ 2^5$	$+ 2^6$	

Sur un élément de type volumique le descripteur_grandeur vaut 14. En effet:

	DX	DY	DZ	DRX	DRY	DRZ	LAGR
	1	1	1	0	0	0	0
14 =	2^1	+ 2^2	+ 2^3				

Sur un nœud supplémentaire crée pour l'introduction de condition cinématique par dualisation, le descripteur_grandeur vaut 128. En effet :

	DX	DY	DZ	DERX	DRY	DRZ	LAGR
	0	0	0	0	0	0	1
128 =							2^7

Un descripteur_grandeur est un vecteur d'entier_codés : V de dimension n_ec où n_ec est le nombre d'entier_codés nécessaires à la description de la grandeur décrite dans le catalogue.

n_ec	nombre de CMPS dans le catalogue
1	1 à 30
2	31 à 60
3	...

Le i^{ème} entier_codé renseigne sur la présence (ou non) des CMPS numérotés de 30*(i-1)+1 → 30*i.

4 SD carte

4.1 Généralités

Une carte est un champ discrétisé par maille. Chaque maille peut être "affectée" d'une grandeur (au plus). Les cartes sont en général des SD crées à partir des données de l'utilisateur. Sa structure est faite pour stocker (avec le moins de volume possible) les informations concernant l'affectation des grandeurs sur des "morceaux" du maillage.

Remarque :

La structure choisie est économique en espace mais elle ne répond pas rapidement à la question : quelle grandeur est affectée sur la maille M1 ? Pour répondre à cette question, il faut "étendre" la carte (cela créer des objets temporaires plus volumineux) ; c'est l'objet de la routine ETENCA appelée par CALCUL. Une carte est donc une liste ordonnée de couples (grandeur, zone_affectée). L'ordre des couples est important car il sert à prendre en compte le principe de surcharge des affectations : la dernière affectation prime sur les précédentes.

Une zone_affectée peut être :

- l'ensemble des mailles du maillage (TOUT: 'OUI'),
- l'ensemble des mailles tardives d'un ligrel,
- un GROUP_MA du maillage,
- une liste de mailles du maillage,
- une liste de mailles tardives d'un ligrel.

4.2 Objet .NOMA

Nom du maillage associé à la carte.

4.3 Objet .DESC

```
'.DESC' S  V  I  DIM = 3 + (2+n_ec)*n_gd_max
```

Le champ ' DOCU ' de l'objet . DESC contient : ' CART '

DESC(1)	gd (numéro de la grandeur associée à la carte)
DESC(2)	n_gd_max (majorant du nombre de zone_affectée)
DESC(3)	n_gd_edit (nombre réel de zone_affectée) n_gd_edit peut être = 0
DESC(3+1)	code_1er_zone ("code" de la première zone_affectée)
DESC(3+2)	numéro de la 1ère zone_affectée
.....	
DESC(3+2*n_gd_max-1)	code_der_ent (code de la dernière zone_affectée)
DESC(3+2*n_gd_max)	numéro de la dernière zone_affectée

Attention :

il faut n_gd_max=1 pour une carte constante sur toutes les mailles tardives (par exemple CHTIME dans me2mme.f)

Le "code" d'une zone_affectée peut valoir :

1	l'ensemble des mailles du maillage (TOUT: 'OUI '),
-1	l'ensemble des mailles tardives d'u n ligrel ,
2	un GROUP_MA du maillage ,
3	une liste de mailles du maillage ,
-3	une liste de mailles tardives d'un ligrel .

Si code=1 (ou -1):
le numéro de la zone_affectée correspondante ne sert à rien.

Si code=2:
le numéro de la zone_affectée correspondante est le numéro du group_ma
dans la collection mailla.GROUPEMA

Si code=3 (ou -3):
le numéro de la zone_affectée correspondante est le numéro de l'objet de la collection .LIMA
qui contient les numéros des mailles composant la zone_affectée.

Vient ensuite dans l'objet .DESC une suite de descripteur_grandeur décrivant les différentes grandeurs affectées.

Soit n_ec le nombre d'entier_codé nécessaires à décrire les CMPS de la grandeur gd :

DESC(3+2*n_gd_max+1)	début du premier descripteur_grandeur
....	....
DESC(3+2*n_gd_max +(n_gd_max-1)*n_ec +1)	début du dernier descripteur_grandeur

Remarque :

Pour un champ constant (1 seule grandeur affectée à toutes les mailles du maillage). On a alors :
DESC(2) = 1
DESC(3) = 1
DESC(4) = 1
DESC(5) = peu importe
DESC(6) = début du descripteur_grandeur de la zone_affectée (TOUT : 'OUI'
Dans ce cas .LIMA et .NOLI ne sont pas alloués (économie de place).

4.4 Objet .NOLI

Cet objet n'est présent que si la carte concerne des mailles tardives.

C'est un vecteur de K24 de dimension nb_gd_max. En face de izone on trouve, si cette zone_affectée est une liste de mailles tardives, le nom du ligrel ou sont définies ces mailles.

```
izone ---> nom_ligrel
```

4.5 Objet .LIMA

C'est une famille contiguë numérotée de vecteurs d'entiers.

```
LIMA(izone) : V(I)
```

V contient les numéros des mailles constituant la zone_affectée.

Les numéros de mailles de la liste sont des numéros relatifs au ligrel référencé dans .NOLI(izone).

si un numéro de maille est > 0 , c'est une maille du maillage associé à la carte.
si un numéro de maille est < 0 , c'est une maille du supplémentaire du ligrel.

4.6 Objet .VALE

C'est un vecteur de scalaires dimensionné à nb_gd_max * nb_cmp_max, si nb_cmp_max est le nombre de CMPS dans la catalogue pour la grandeur associée à la carte.

La grandeur associée à la zone_affectée izone commence dans .VALE à l'indice :

```
izone -->  .VALE( (izone-1)*nb_cmp_max + 1 )
```

Attention :

Seules les CMPS affectées sont stockées (consécutivement et dans l'ordre du catalogue) dans l'objet .VALE Par exemple, pour une carte de DEPL_R, si la 1ère zone est affectée par : (DX=2. et DZ=4.)
VALE(1) = 2.
VALE(2) = 4.

5 SD cham_no

5.1 Objet .DESC

Le champ 'DOCU' de l'objet .DESC contient : 'CHNO'

DESC(1)	gd (grandeur associée au cham_no)
DESC(2)	num
DESC(3), ..., DESC(3 + n_ec - 1)	descripteur_grandeur de la grandeur dans le cas où num est < 0

Si num est négatif num = "-" nb_cmp

Si num est < 0, sa valeur absolue est le nombre de CMP de la grandeur pour TOUS les nœuds du maillage (par ex. le champ de géométrie). Dans ce cas le champ ne concerne que les nœuds du maillage (pas de nœuds tardifs) et on suppose que tous les nœuds ont la même représentation de la grandeur.

Le descripteur_grandeur est alors stocké de DESC(3) à DESC (3 + n_ec - 1). Si num est positif, il existe alors une structure de type prof_chno référencée dans l'objet .REFE.

5.2 Objet .REFE

REFE(1)	nom du MAILLAGE.
REFE(2)	nom d'un prof_chno [D4.06.07] (si DESC(2)>0) La SD prof_chno décrit les CMPS portées par les nœuds du cham_no. Elle sert à pointer dans l'objet .VALE qui contient les valeurs.
REFE(3)	inutilisé
REFE(4)	inutilisé

5.3 Objet .VALE

Cet objet contient les "valeurs" du champ aux nœuds sur les nœuds du maillage ou sur les nœuds tardifs des ligrel utilisés dans le prof_chno.

6 Dans le cas général

La description de l'objet .VALE dans le cas où le cham_no n'est pas à "représentation constante" est faite dans [D4.06.07 §3].

7 Dans le cas où le cham_no est à "représentation constante"

Soit :
nb_no : le nombre de nœuds du maillage.
ncmp : le nombre de CMPS portées par tous les nœuds du maillage.

```
LONG(.VALE) = nb_no * ncmp
```

VALE(1)	valeur de la 1ère CMP portée par le 1er nœud
VALE(2)	valeur de la 2ème CMP portée par le 1er nœud
...	...
VALE(ncmp)	valeur de la dernière CMP portée par le 1er nœud
VALE(ncmp+1)	valeur de la 1ère CMP portée par le 2ème nœud
...	...

L'ordre des CMPS est celui du catalogue des grandeurs (objet '&CATA.GD.NOMGD' [D4.04.01]).

8 SD champ_elem

8.1 Cas des cham_elem possédant des sous-points (éléments de structure)

Le nombre de points de discrétisation (nœuds, points de Gauss, ...) d'un cham_elem sur une maille est déterminé a priori par le nombre de points défini dans le catalogue du type_elem associé à la maille. Pour les éléments de type "structure", on veut pouvoir stocker plus de grandeurs que de points définis dans le catalogue.

Lors d'un calcul non-linéaire sur une coque (par exemple), l'intégration choisie pour le comportement non-linéaire nécessite de stocker l'état de contraintes en plusieurs points dans l'épaisseur : il faut discrétiser l'épaisseur de la coque. Pour cela, on dira que chaque point de Gauss positionné sur la surface de l'élément (leur nombre est fixé dans le catalogue du type_elem), est composé de n sous-points représentant la discrétisation de la normale à l'élément en ce point.

De la même façon, un élément non-linéaire de tuyau, pourra discrétiser sa section (anneau circulaire) en la découpant en secteurs et en couches.

Pour un élément donné, tous les points de discrétisation ont obligatoirement le même nombre de sous-points.

Attention :

Avant de créer un cham_elem avec sous-points, il faut dire pour tous les éléments du ligrel le nombre de sous-points voulu. Pour cela, on utilise un cham_elem_s de la grandeur DCEL_I (argument DCELZ de la routine alchml.f). Lorsqu'on appelle la routine de calculs élémentaires (calcul.f), le passage de cet argument est sous-terrain : le cham_elem_s doit avoir le même nom que le cham_elem (OUT) qu'il sert à dimensionner.

8.2 Cas des cham_elem ne possédant pas de sous-points

Informatiquement, tous les cham_elem ont des sous-points. Un cham_elem qui n'a pas besoin de cette notion est en fait un cham_elem pour lequel chaque point de discrétisation n'a qu'un seul sous-point ; on confond alors le point et son unique sous-point.

8.3 Cas des cham_elem de la grandeur VARI_R

La grandeur VARI_R est la grandeur "réservée" qui sert à représenter une grandeur dont le nombre de composantes (CMP) est indéterminée au niveau des catalogues de type_elem.

On se sert par exemple de cette grandeur pour représenter les variables internes des lois de comportement, car chaque loi peut avoir un nombre différent de telles variables.

Dans le catalogue des grandeurs, cette grandeur n'a qu'une seule CMP : VARI.

Au moment de la création d'un cham_elem_VARI_R, on doit dire pour chacun des éléments, combien de composantes aura la grandeur VARI_R. Ces composantes s'appelleront alors : 'V1', 'V2', ..., 'Vn'. Pour cela, on utilise le même mécanisme que pour déclarer le nombre des sous-points (voir ci-dessus).

8.4 Objet .CELK

CELK(1)	nom du ligrel associé au cham_elem.
CELK(2)	nom de l'option de calcul associée au cham_elem.
CELK(3)	/ 'ELNO' : CHAM_ELEM aux nœuds / 'ELGA' : CHAM_ELEM aux points de Gauss / 'ELEM' : CHAM_ELEM constant par élément
CELK(4)	nume_couche : numéro de la couche (cadré à gauche) pour un CHAM_ELEM calculé sur une couche d'élément de coque.
CELK(5)	nive_couche : position dans la couche pour un CHAM_ELEM calculé sur une couche d'élément de coque : /'INF' / 'MOY' / 'SUP'
CELK(6)	Nom du paramètre de l'option associée au cham_elem
CELK(7)	/'MPI_COMPLET' /'MPI_INCOMPLET'

CELK(7) :

- 'MPI_INCOMPLET' : le calcul a été distribué (MPI) et l'objet .CELV n'est pas complet sur tous les processeurs. Chaque processeur n'a calculé qu'un sous-ensemble des éléments.
- 'MPI_COMPLET' : sinon. C'est à dire que le calcul n'a pas été distribué, ou bien, le cham_elem (MPI_INCOMPLET à sa création) a été "complété" en appelant l'utilitaire sdmpic.f

8.5 Objet .CELD

.CELD : vecteur d'entiers. Le champ 'DOCU' de l'objet .CELD contient : 'CHML'

Cet objet est le descripteur de l'objet contenant les valeurs du cham_elem (.CELV).

CELD(1)	gd	grandeur associée au cham_elem .
CELD(2)	nb_gr	nombre de grel du ligrel associé .
CELD(3)	mxsp	maximum du nombre de sous-points pour les éléments du ligrel
CELD(4)	mxcmp	maximum du nombre de CMP (grandeur VARI_R) pour les éléments du ligrel. 0 si grandeur différente de VARI_R
CELD(4+1)	debu_grel_1	adresse (-1) dans .CELD du début des informations concernant le 1er GREL
...		
CELD(4+nb_gr)	debu_grel_n	adresse (-1) dans .CELD du début des informations concernant le dernier GREL

puis on stocke bout à bout la description du champ pour chaque GREL du ligrel

CELD(debu_grel +1)	nel	nombre d'élément du GREL
CELD(debu_grel +2)	modelo	mode_local associé au champ local (ou 0 si champ inexistant sur le GREL)
CELD(debu_grel +3)	lgcata	longueur du champ local au vu du catalogue. c'est à dire sans tenir compte des sous-points et des composantes multiples de VARI_R. (= 0 si modelo = 0)
CELD(debu_grel +4)	lggrel	longueur total du segment contenant toutes les valeurs du champ sur le GREL
puis:do iel = 1, nel		(si modelo > 0)
CELD(debu_grel+4+4*(iel-1)+1)	nbsp	nombre de sous_points pour l'élément iel
CELD(debu_grel+4+4*(iel-1)+2)	ncdyn	nombre de CMP (VARI_R) pour l'élément iel
CELD(debu_grel+4+4*(iel-1)+3)	lgchel	nombre de valeurs du champ local pour l'élément iel. lgchel= lgcata * nbsp * ncdyn
CELD(debu_grel+4+4*(iel-1)+4)	adiel	adresse dans l'objet .CELV de la 1ere valeur de l'élément iel

8.6 Objet .CELV

C'est un vecteur contenant bout à bout les valeurs des champs locaux des différents éléments.

La description du segment concernant un élément est donnée par le mode_local défini pour le type_elem. Cette description est éventuellement complétée par la donnée du nombre de sous-points et du nombre de CMPS (VARI_R).

Pour un champ de grandeur (différente de VARI_R) n'ayant pas de sous-points, tous les éléments d'un même grel ayant le même type_elem, leurs champs locaux ont tous la même longueur et la même organisation.

On se déplace dans l'objet .CELV grâce à l'objet .CELD.

On peut décrire l'organisation de l'objet .CELV par ces définitions :

CELV(ligrel)	suite de CELV(GREL) mis bout à bout
CELV(GREL)	suite de CELV(élément) mis bout à bout
CELV(élément)	suite de CELV(point) mis bout à bout
CELV(point)	suite de CELV(sous-point) mis bout à bout
CELV(sous-point)	suite de CMP(scalaire) mises bout à bout

8.7 Quelques "formules" fréquemment utilisées dans la programmation

8.7.1 LIGREL

numéro de la grandeur associée au CHAM_ELEM :

```
NUMGD=ZI(JCELD-1+1)
```

nombre de GREL du LIGREL associé au CHAM_ELEM :

```
NGREL=ZI(JCELD-1+2)
```

nombre maxi. de sous-points des éléments d'un CHAM_ELEM: (peut-être = 0)

```
MXSP=ZI(JCELD-1+3)
```

nombre maxi. de CMPS (VARI_R) des éléments d'un CHAM_ELEM:

```
(/=0 <=> VARI_R)
MXCDY=ZI(JCELD-1+4)
```

8.7.2 GREL : IGR

nombre d'éléments d'un GREL (IGR):

```
NEL=ZI(JCELD-1+ZI(JCELD-1+4+IGR) +1)
```

mode_local d'un GREL (IGR):

```
IMOLO=ZI(JCELD-1+ZI(JCELD-1+4+IGR) +2)
```

longueur cumulée des éléments d'un GREL (IGR):

```
LGGREL=ZI(JCELD-1+ZI(JCELD-1+4+IGR) +4)
```

adresse (dans .CELV) du début du GREL IGR :

```
DEBUGR=ZI(JCELD-1+ZI(JCELD-1+4+IGR)+8)
puis : ZR(JCELV -1 +DEBUGR) = ...
```

longueur (CATALOGUE) d'un élément d'un GREL (IGR):

```
LGCATA=ZI(JCELD-1+ZI(JCELD-1+4+IGR) +3)
```

8.7.3 Élément IEL du GREL IGR

adresse (dans .CELV) du début de élément IEL du GREL IGR :

```
ADIEL=ZI(JCELD-1+ZI(JCELD-1+4+IGR) +4 +4*(IEL-1)+4)
puis : ZR(JCELV -1 +ADIEL) = ...
```

longueur de élément IEL du GREL IGR :

```
LGIEL=ZI(JCELD-1+ZI(JCELD-1+4+IGR) +4 +4*(IEL-1)+3)
```

nombre de sous-points de élément IEL du GREL IGR :

```
NBSPT=ZI(JCELD-1+ZI(JCELD-1+4+IGR) +4 +4*(IEL-1)+1)
il n'y a pas de sous-points : NBSPT=0 (ou 1 ; remarque de JD)
```

nombre de CMPS (VARI_R) de élément IEL du GREL IGR :

```
NCDYN=ZI(JCELD-1+ZI(JCELD-1+4+IGR) +4 +4*(IEL-1)+2)
0 -> la grandeur n'est pas VARI_R
```

9 SD resuelem

9.1 Objet .NOLI

NOLI(1)	nom du ligrel associé au resuelem .
NOLI(2)	nom de l'option associée au resuelem .
NOLI(3)	/'MPI_COMPLET' /'MPI_INCOMPLET'
NOLI(4)	/' ' /'VOISIN_VF'

NOLI(3) :

- ' MPI_INCOMPLET' : le calcul a été distribué (MPI) et l'objet . RESL n'est pas complet sur tous les processeurs. Chaque processeur n'a calculé qu'un sous-ensemble des éléments.
- ' MPI_COMPLET ' : sinon. C'est à dire que le calcul n'a pas été distribué, ou bien, le resuelem (MPI_INCOMPLET à sa création) a été "complété" en appelant l'utilitaire sdmpic.f

NOLI(4) : cette valeur n'a de sens que pour les resuelem "matrice"

- ' ' : les matrices élémentaires sont "ordinaires". Pour un élément, la matrice élémentaire est un "carré" (ou un demi-carré si la matrice est symétrique) concernant les ddls de l'élément
- ' VOISIN_VF ' : les matrices élémentaires sont de type " VOISIN_VF " (plus grandes). Dans ce cas, on stocke bout-à-bout les matrices concernant le couplage des ddls de l'élément avec ceux de ces voisins. Voir les détails dans l'objet . RSVI .

9.2 Objet .DESC

Le champ 'DOCU' de l'objet .DESC contient : 'RESL'

DESC(1)	gd (grandeur associée au resuelem)
DESC(2)	nb_gr (nombre de GREL de .NOLI(1))
DESC(2+1)	mode_1er_gr (mode_local des champs locaux du premier GREL)
...	
DESC(2+nb_gr)	mode_der_gr (mode_local du dernier GREL)

9.3 Objet .RESL

C'est une collection dispersée de vecteurs de R (ou C) . L'accès à cette collection se fait par le numéro de GREL : .RESL(IGREL) → V

Si ncmpel est le nombre de scalaires représentant le champ local pour un élément du GREL ,

V(1,...,ncmpel)	valeurs du champ sur le 1er élément du GREL
V(ncmpel+1,...,2*ncmpel)	valeurs du champ sur le 2ème élément du GREL
...	...

Attention :

Pour les resuelem " matrice " de type " VOISIN_VF ", le nombre de valeurs par élément n'est pas ncmpel . Pour se déplacer dans . RESL , il faut utiliser l'objet . RSVI . Voir ci-dessous.
A cause de la "distribution" des éléments (AFFE_MODELE / PARTITION / PARALLELISME = 'GROUP_ELEM ') les objets de la collection . RESL peuvent exister au sens de JEEXIN sans que l'on puisse faire un JEVEUO(...,'L',...) dessus car ils n'ont pas d'image disque. Pour tester si ces objets existent vraiment, il faut utiliser la routine JAEXIN .

9.4 Objet .RSVI

Cet objet n'existe que lorsque le resuelem est de type " matrice " et qu'il est de type " VOISIN_VF ". C'est une collection contigüe de vecteurs de I . L'accès à cette collection se fait par le numéro de GREL : .RSVI(IGREL) → V

Si nel est le nombre d'éléments du GREL , V est de longueur nel+1

V(1)	(=1) : indice dans . RESL du début de la matrice élémentaire de iel=1
V(2)	indice dans . RESL du début de la matrice élémentaire de iel=2
V(3)	indice dans . RESL du début de la matrice élémentaire de iel=3
...	...
V(nel+1)	(longueur cumulée des matrices élémentaires du grel) +1

Soit :

iel1, iel2, ..., ielp les p voisins de iel0
nddl1, nddl2, ..., nddlp le nombre de ddls des p voisins de iel0
nddl0 : nombre de ddls de iel0

Alors :

la matrice élémentaire MEL de iel0 est de dimension nddl0*(nddl0+nddl1+...+nddlp)
MEL(1) -> couplage (iel0,1) X (iel0,1)
MEL(2) -> couplage (iel0,1) X (iel0,2)
...
MEL(nddl0+1) -> couplage (iel0,2) X (iel0,1)
...
MEL(nddl0*nddl0) -> couplage (iel0,nddl0) X (iel0,nddl0)

MEL(nddl0*nddl0+1) -> couplage (iel0,1) X (iel1,1)
...
MEL(nddl0*(nddl0+nddl1)+1) -> couplage (iel0,1) X (iel2,1)
...

10 Exemples

10.1 SD carte

```
 CARTE = CREA_CHAMP (TYPE_CHAMP : 'CART_META_R', OPERATION : 'AFFE',
                 MAILLAGE : MAILLA
                 AFFE : ( TOUT    : 'OUI'
                          NOM_CMP : ( 'ZF' 'ZP' 'ZB' 'ZM' 'P' )
                          VALE    : ( 0.0   0.0  0.0  0.0  0.0) )
                 AFFE : ( GROUP_MA    : GM2
                          NOM_CMP : ( 'ZF' 'ZP' )
                          VALE    : ( 0.2  0.3  ) )
                 AFFE : ( MAILLE    : T2
                          NOM_CMP : ( 'ZP' 'ZM' 'P' )
                          VALE    : ( 0.4  0.5  0.6 ) )
                ) ;


 IMPRESSION SEGMENT DE VALEURS >CARTE              .DESC        <
      1 -           64            3            3            3            1
      6 -            3            2            3            3          254
     11 -          254          254
--------------------------------------------------------------------------------
 IMPRESSION DE LA COLLECTION : CARTE              .LIMA
 IMPRESSION OBJET DE COLLECTION CONTIGUE>CARTE              .LIMA<  OC :      1
      1 -            1            3
 IMPRESSION OBJET DE COLLECTION CONTIGUE>CARTE              .LIMA<  OC :      2
      1 -            2
 IMPRESSION OBJET DE COLLECTION CONTIGUE>CARTE              .LIMA<  OC :      3
      1 -            4            5
--------------------------------------------------------------------------------
 IMPRESSION SEGMENT DE VALEURS >CARTE              .NOLI        <
      1 - >                        <>                        <
      3 - >                        <
--------------------------------------------------------------------------------
 IMPRESSION SEGMENT DE VALEURS >CARTE              .NOMA        <
      1 - >MAILLA  <
--------------------------------------------------------------------------------
 IMPRESSION SEGMENT DE VALEURS >CARTE              .VALE        <
      1 -  2.00000E-01  3.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
      6 -  0.00000E+00  0.00000E+00  2.00000E-01  4.00000E-01  0.00000E+00
     11 -  5.00000E-01  0.00000E+00  0.00000E+00  6.00000E-01  0.00000E+00
     16 -  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
     21 -  0.00000E+00
```

Remarque :

Le contenu des objets imprimés ci-dessus peut surprendre : il ne correspond pas à ce qui est dit plus haut. En effet cette carte a été "terminée" par un appel à la routine `tecart.f`.
Cette action facultative a pour but de permettre une surcharge "fine" des valeurs affectées dans la commande `CREA_CHAMP`.

10.2 SD cham_no

```
cham_no = CREA_CHAMP ( MAILLAGE : mailla, TYPE_CHAMP : 'NOEU_DEPL_R',
          OPERATION : 'AFFE',
          AFFE:(GROUP_NO:gn1
```

```
            nom_cmp: 'DX' VALE_R: 1.0)
          AFFE:(NOEUD:(n2,n7)
            NOM_CMP: ('DX','DZ') vale_r: (2. ,4.) )
            );


 IMPRESSION SEGMENT DE VALEURS >cham_no            .DESC        <
      1 -           32            6
--------------------------------------------------------------------------------
 IMPRESSION SEGMENT DE VALEURS >cham_no            .REFE        <
      1 - >MAILLA                  <>cham_no                 <
--------------------------------------------------------------------------------
 IMPRESSION SEGMENT DE VALEURS >cham_no            .VALE        <
      1 -  2.00000E+00  4.00000E+00  1.00000E+00  1.00000E+00  2.00000E+00
      6 -  4.00000E+00
```

10.3 SD cham_elem

```
FLUXN=CALC_CHAM_ELEM(    MODELE=MOTH,  TEMP=T2,
            CHAM_MATER=CHMAT,   OPTION='FLUX_ELNO')


IMPRESSION SEGMENT DE VALEURS >FLUXN              .CELD        <
 >>>>>
      1 -           47            2            1            0            6
      6 -           18            2         6520            8           16
     11 -            1            0            8            1            1
     16 -            0            8            9            3         6857
     21 -            6           18            1            0            6
     26 -           17            1            0            6           23
     31 -            1            0            6           29

 IMPRESSION SEGMENT DE VALEURS >FLUXN              .CELV        <
 >>>>>
      1 - -8.78595D-12 -4.27645D-12 -8.78595D-12 -4.08919D-12  6.96696D-12
      6 - -4.07954D-12  6.96696D-12 -4.77838D-12  4.96957D-12 -4.15161D-12
     11 -  4.96957D-12 -4.26679D-12 -1.33159D-12 -4.54543D-12 -1.33159D-12
     16 - -3.57760D-12  7.27596D-12 -8.41283D-12  7.27596D-12 -8.41283D-12
     21 -  7.27596D-12 -8.41283D-12  0.00000D+00 -8.86757D-12  0.00000D+00
     26 - -8.86757D-12  0.00000D+00 -8.86757D-12  0.00000D+00 -8.86757D-12
     31 -  0.00000D+00 -8.86757D-12  0.00000D+00 -8.86757D-12

 IMPRESSION SEGMENT DE VALEURS >FLUXN              .CELK        <
 >>>>>
      1 - >MOTH    .MODELE         <>FLUX_ELNO          <
      3 - >ELNO                    <>                        <
      5 - >                        <>PFLUX_R
```

10.4 SD resuelem

```
CHTH= AFFE_CHAR_THER(MODELE:MODEL  TEMP_IMPO:(NOEUD:N8 TEMP:3.4)
                     SOURCE:(TOUT:'OUI'  SOUR: 7.)        );
VECTEL=CALC_VECT_ELEM( CHARGE:CHTH OPTION:'CHAR_THER');
```

Le `resuelem` est extrait du `VECT_ELEM VECTEL : 'VECTEL .VE001'`

```
 IMPRESSION SEGMENT DE VALEURS >VECTEL  .VE001     .DESC        <
      1 -          105            3         5781         5648            0
--------------------------------------------------------------------------------
 IMPRESSION SEGMENT DE VALEURS >VECTEL  .VE001     .NOLI        <
```

```
      1 - >MODEL    .MODELE         <>CHAR_THER_SOUR_R        <
--------------------------------------------------------------------------------
 IMPRESSION DE LA COLLECTION : VECTEL  .VE001     .RESL
 IMPRESSION OBJET DE COLLECTION >VECTEL  .VE001     .RESL<  OC :       1
      1 -  3.50000E+00  3.50000E+00  3.50000E+00  4.66667E+00  4.66667E+00
      6 -  4.66667E+00
 IMPRESSION OBJET DE COLLECTION >VECTEL  .VE001     .RESL<  OC :       2
      1 -  4.08333E+00  4.66667E+00  4.66667E+00  4.08333E+00
```