

Structures de données sd_num_ddl, sd_num_equa, sd_stockage et sd_prof_chno

Résumé :

Ce document décrit les structures de données NUME_DDL , NUME_EQUA , STOCKAGE et PROF_CHNO utilisées pour définir la numérotation des inconnues des systèmes linéaires et le stockage des matrices assemblées et des vecteurs solution et second membre.

Table des Matières

1 Généralités.....	3
2 Arborescences.....	4
3 L'objet NUME_DDL.....	5
4 L'objet NUME_EQUA.....	5
5 L'objet PROF_CHNO.....	5
5.1 Objet PRNO.....	6
5.2 Objet LILL.....	7
5.3 Objet NUEQ.....	7
5.4 Objet DEEQ.....	9
6 L'objet NUML_EQUA.....	10
7 L'objet STOCKAGE.....	11
7.1 L'objet STOC_LIGN_CIEL.....	11
7.2 L'objet STOC_MORSE.....	12
7.3 L'objet MULT_FRONT.....	13
8 Exemples.....	13

1 Généralités

Un NUME_DDL sert à définir la numérotation des inconnues (et des équations) d'un système linéaire. On rappelle que les inconnues d'un tel système sont des composantes portées par des nœuds du MAILLAGE (ou des nœuds tardifs de LIGREL).

Les matrices que l'on veut décrire sont carrées, creuses, symétriques ou non.

Quand une matrice est non-symétrique, sa structure creuse est symétrique.

La numérotation des inconnues d'un système est semblable à celle des composantes d'un CHAM_NO. D'ailleurs la SD PROF_CHNO décrite dans ce document est celle référencée par la SD CHAM_NO [D4.06.05 - Structure de Donnée champ].

La SD NUME_DDL contient également la description des tableaux de stockage des valeurs des matrices assemblées [D4.06.10 - Structure de Données *sd_matr_asse*]. On appelle STOCKAGE cette partie de la structure de données.

Les relations de dépendance entre ces objets peuvent se représenter par :

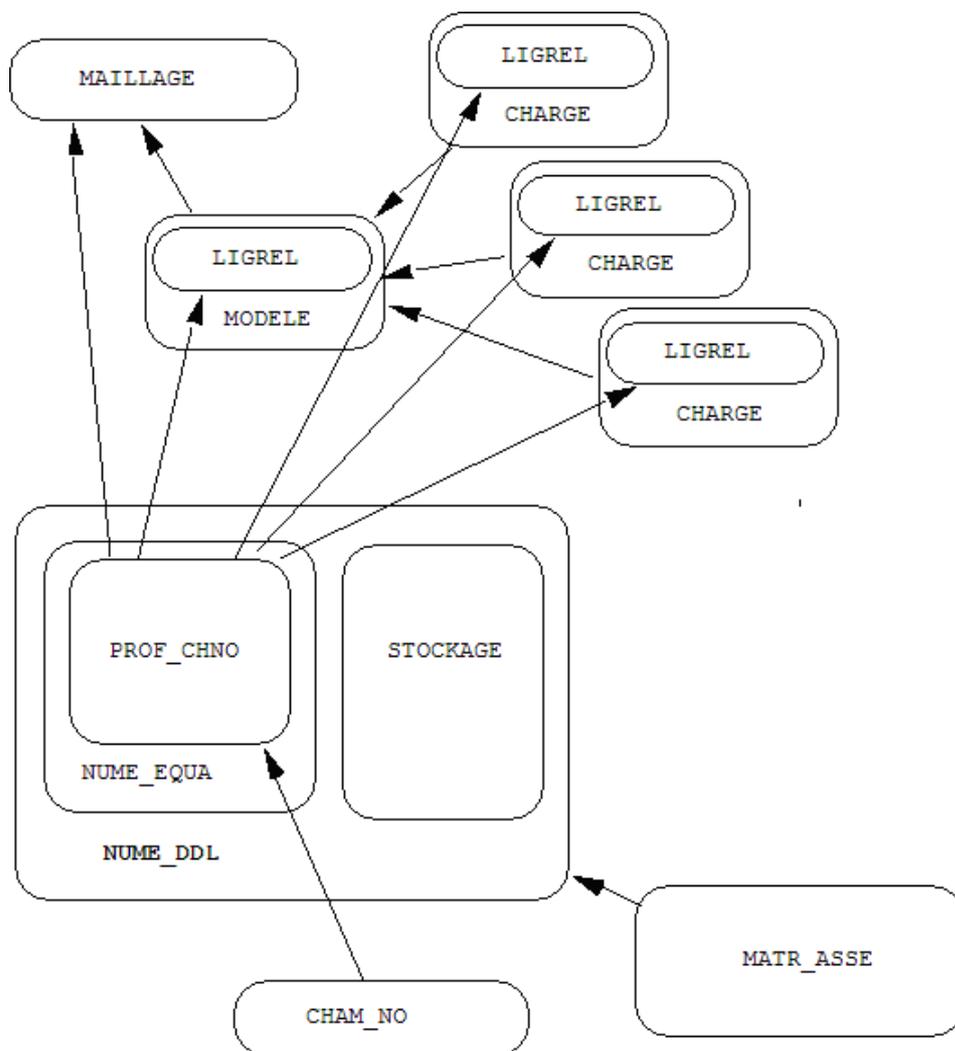


Figure 1-a: Liens des NUME_DDL, NUME_EQUA, STOCKAGE et PROF_CHNO avec les autres structures de données

2 Arborescences

Voici l'arborescence des différents objets présentés dans ce document.

```
NUME_DDL (K14) ::=record
  ◆ .NUME      NUME_EQUA
  ◆ .NUML      NUML_EQUA
  ◆ .NSLV      OJB S V K24 dim=1

NUME_EQUA (K19) ::=record
  ◆ .NEQU      OJB S V I dim=2
  ◆ .REFN      OJB S V K24 dim=4
  ◆ .DELG      OJB S V I

PROF_CHNO (K19) ::=record
  ◆ .PRNO      OJB XC V I NUM
  ◆ .LILI      OJB S N K24
  ◆ .NUEQ      OJB S V I
  ◆ .DEEQ      OJB S V I

NUML_EQUA (K19) ::=record
  ◆ .PRNO      OJB XC V I
  ◆ .NEQU      OJB S V I dim=2
  ◆ .DELG      OJB S V I
  ◆ .NUEQ      OJB S V I
  ◆ .NULG      OJB S V I
  ◆ .NUGL      OJB S V I

STOCKAGE (K14) ::=record
  ◆ .SMOS      STOC_MORSE # pour décrire la matrice creuse initiale
  % si factorisée par 'LDLT'
  .SLCS      STOC_LCIEL # pour décrire la matrice factorisée
  % si factorisée par 'MULT_FRONT'
  .MLTF      MULT_FRONT # pour décrire la matrice factorisée

STOC_LCIEL (K19) ::=record
  ◆ .SCBL      OJB S V I
  ◆ .SCDI      OJB S V I
  ◆ .SCDE      OJB S V I
  ◆ .SCHC      OJB S V I
  ◆ .SCIB      OJB S V I

STOC_MORSE (K19) ::=record
  ◆ .SMDI      OJB S V I
  ◆ .SMDE      OJB S V I
  ◆ .SMHC      OJB S V I

MULT_FRONT (K19) ::=record
  ◆ .ADNT      OJB S V I4
  ◆ .GLOB      OJB S V I4
  ◆ .LOCL      OJB S V I4
  ◆ .PNTI      OJB S V I4
```

3 L'objet `NUME_DDL`

L'objet `NUME_DDL` sert à définir la numérotation des inconnues (et des équations) d'un système linéaire.

```
NUME_DDL (K14) ::=record
  ♦ .NUME          NUME_EQUA
  ♦ .NUML          NUML_EQUA
  ♦ .NSLV          OJB S  V K24 dim=1
```

Il contient donc trois objets. Les premier fait référence à l'objet `NUME_EQUA` (voir §) :

```
NUME_DDL(1:14) // '.NUME'=NUME_EQUA
```

Le deuxième fait référence à l'objet `NUML_EQUA` (voir §) :

```
NUME_DDL(1:14) // '.NUML'=NUML_EQUA
```

Le dernier objet `.NSLV` stocke le nom de la `sd_solveur` qui sera utilisée par défaut.

```
NSLV(1) : nom de la sd_solveur qui sera utilisée par défaut.
```

4 L'objet `NUME_EQUA`

L'objet `NUME_EQUA` fait référence à la numérotation des équations. Il contient trois objets :

```
NUME_EQUA (K19) ::=record
  ♦ .NEQU          OJB S  V I      dim=2
  ♦ .REFN          OJB S  V K24    dim=4
  ♦ .DELG          OJB S  V I
```

L'objet `.REFN` stocke quelques informations :

- `REFN(1)` : nom du maillage sous-jacent au `NUME_DDL` ;
- `REFN(2)` : nom de la grandeur simple associée (`TEMP_R`, `DEPL_R`, `PRES_C`,...);
- `REFN(3)` : solveur linéaire par défaut (`MUMPS`, `MULT_FRONT`) ;
- `REFN(4)` : `i` utilisé.

L'objet `.NEQU` stocke les dimensions :

- `NEQU(1)` : nombre total d'équations (longueur du `.VALE` des champs nodaux) ;
- `NEQU(2)` : nombre total de degrés de liberté (longueur de `.NUEQ` dans le `PROF_CHNO`).

L'objet `.DELG` décrit (un peu) les équations du type « Lagrange » :

- `DELG(ieq)=-1` : si l'équation `ieq` correspond à la composante `LAGR` portée par le deuxième nœud d'une maille `SEG3` portant un élément de Lagrange (Lagrange 1) ;
- `DELG(ieq)=-2` : si l'équation `ieq` correspond à la composante `LAGR` portée par le troisième nœud d'une maille `SEG3` portant un élément de Lagrange (Lagrange 2) ;
- `DELG(ieq)=0` : l'équation n'est pas de type « Lagrange ».

5 L'objet `PROF_CHNO`

La structure de données `PROF_CHNO` définit le profil de numérotation, c'est-à-dire la description des inconnues. Dans `Code_Aster`, les inconnues sont nécessairement portées par des nœuds. Les nœuds d'un `PROF_CHNO` sont de deux types :

- Les nœuds « physiques » du maillage `ma` (concerné par le modèle) ;
- Les nœuds tardifs¹ d'un (ou plusieurs) `LIGREL` qui s'appuient sur le maillage `ma`.

1 Un nœud tardif dans `Code_Aster` correspond à un nœud ajouté par une condition limite dualisée (`DDL_IMPO`, `LIAISON_DDL`,...). Ce nœud a été ajouté lors de la création de la charge et stocké dans le *Manuel de développement*

Les objets de PROF_CHNO sont les suivants :

```
PROF_CHNO (K19) ::=record
  ◆ .PRNO      OJB XC V I NUM
  ◆ .LILI      OJB S  N K24
  ◆ .NUEQ      OJB S  V I
  ◆ .DEEQ      OJB S  V I
```

5.1 Objet PRNO

L'objet .PRNO est une collection contiguë numérotée de vecteur d'entiers.

```
◆ .PRNO      OJB XC V I NUM
```

Cette collection décrit quelles sont les composantes portées par les nœuds impliqués dans le PROF_CHNO en indiquant le nom du LIGREL de modèle et ceux des LIGREL à mailles et/ou à nœuds tardifs. S'il s'agit d'un LIGREL à mailles et à nœuds tardifs (DDL_IMPO, LIAISON_DDL...), cela permet d'identifier les nœuds tardifs impliqués dans le PROF_CHNO. Par contre, si il s'agit d'un LIGREL uniquement à mailles tardives (FORCE_NODALE...), il ne pointe vers aucun objet de collection du .PRNO.

Ce qui veut dire que l'objet PRNO(1) renseigne sur les degrés de liberté portés par les nœuds du maillage (sur lequel s'appuie nécessairement le modèle) sous-jacent au PROF_CHNO et que les autres objets de la collection .PRNO (quand ils existent) renseignent sur les degrés de liberté portés par les nœuds qui n'appartiennent pas au maillage (nœuds tardifs).

Les nœuds tardifs sont définis dans des LIGREL. A chaque LIGREL nomlig correspond un objet .PRNO(ili) d'indice ili. Le numéro du LIGREL est celui correspondant à nomlig dans le pointeur de nom .LILI.

Par exemple, la collection .PRNO contiendra :

- .PRNO(1) : nœuds du maillage *ma* ;
- .PRNO(2) : nœuds tardifs du LIGREL dont le nom est dans .LILI(2) ;
- .PRNO(3) : nœuds tardifs du LIGREL dont le nom est dans .LILI(3) ;
- ...

Si *nec* est le nombre d'entiers codés de la grandeur associée au PROF_CHNO :

- .PRNO(1) est de longueur $(nb_noeuds(ma)) * (2+nec)$;
- .PRNO(1ili) est de longueur $(nb_noeuds_tardifs(LIGREL(ili))) * (2+nec)$.

Chaque nœud est décrit par deux entiers et un vecteur d'entiers codés de longueur *nec* que l'on appelle le descripteur-grandeur [D4.06.05 - Structure de Donnée champ].

Prenons l'exemple des nœuds du maillage *ma* :

- *nb_cmp* est le nombre de composantes portées par le nœud *ino* du maillage ;
- *ieq* est l'adresse dans l'objet .NUEQ de la première composante portée par le nœud *ino*.

```
v = PRNO (1)
v (ino-1) * (2+nec)+1 = ieq
v (ino-1) * (2+nec)+2 = nb_cmp
                                <- .
v (ino-1) * (2+nec)+2+1      |
...                          |   Descripteur-grandeur du nœud ino
v (ino-1) * (2+nec)+2+nec   |
                                <-
```

Remarque :

Toutes les composantes portées par un même nœud sont consécutives dans .NUEQ. C'est pourquoi on ne stocke que l'adresse de la première. Les composantes sont ordonnées dans

LIGREL correspondant pour porter les Lagrange. Ils n'apparaissent pas dans la structure de donnée maillage et ils ont des numéros négatifs.

l'ordre du catalogue des grandeurs. Malheureusement, la grandeur associée au `.PRNO` n'est pas stockée dans le `PROF_CHNO`.

5.2 Objet LILI

L'objet `.LILI` est un pointeur de noms qui permet d'accéder aux éléments de la collection `PRNO`.

◆ `.LILI` OJB S N K24

Par convention, on stocke dans `LILI(1)` la valeur `&MAILLA`, ce qui veut dire que l'objet `PRNO(1)` renseigne sur les degrés de liberté portés par les nœuds du maillage sous-jacent au `PROF_CHNO`. Néanmoins on ne stocke pas le nom du maillage.

Les autres objets de la collection `.PRNO` (quand ils existent) renseignent sur les degrés de liberté portés par les nœuds qui n'appartiennent pas au maillage (nœuds tardifs). Ces nœuds tardifs sont définis dans des `LIGREL`. A chaque `LIGREL nomlig` correspond un objet `.PRNO(ili)`. Le numéro du `LIGREL` est celui correspondant à `nomlig` dans le pointeur de nom `.LILI`.

5.3 Objet NUEQ

L'objet `.NUEQ` est un vecteur d'entiers.

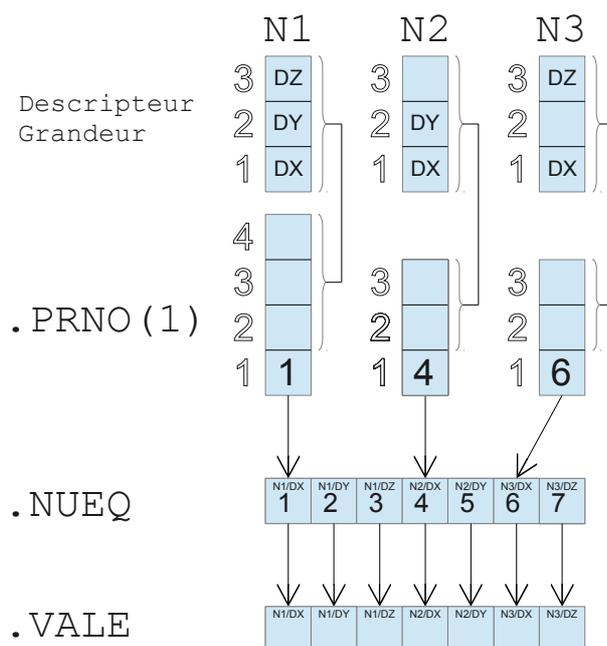
◆ `.NUEQ` OJB S V I

C'est un vecteur d'indirection entre l'objet `.PRNO` et l'objet `.VALE` des `CHAM_NO` qui référencent ce `PROF_CHNO`. Ce vecteur d'indirection permet de s'affranchir de la règle selon laquelle les composantes d'un nœud se suivent dans l'ordre du catalogue des grandeurs mais aussi que chaque composante correspond effectivement à une inconnue dans le `NUMD_DDL`.

Dans la pratique, ce vecteur est quasiment toujours l'identité ($NUEQ(I) = I$). Ce cas le plus général signifie deux choses :

1. Il y a autant d'inconnues dans que de composantes sur les nœuds correspondant au `LIGREL` du modèle et sur les nœuds tardifs des `LIGREL` tardifs des chargements dualisés ;
2. La numérotation est une tautologie : les inconnues sont dans l'ordre des composantes du catalogue sur chaque nœud.

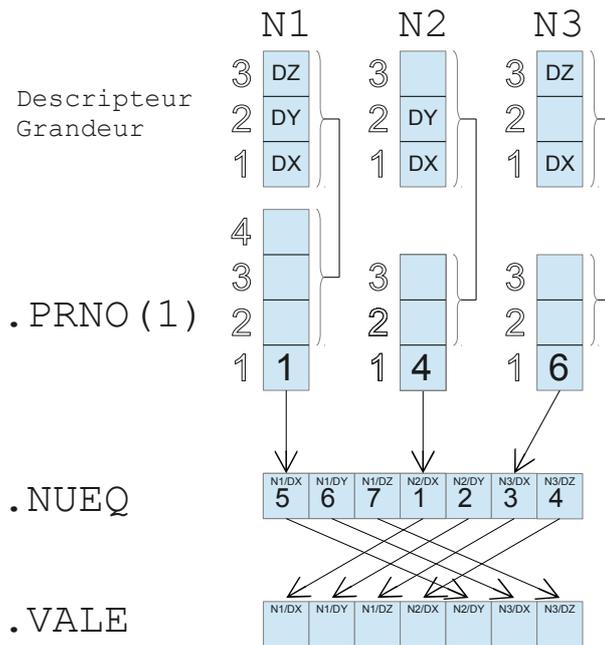
Sur l'exemple suivant, on a trois nœuds. Le premier nœud a pour composantes `DX`, `DY` et `DZ`, le deuxième nœud a pour composantes `DX` et `DY` et le troisième nœud a pour composantes `DX` et `DZ`. Dans ce cas, L'objet `.NUEQ` est tel que $NUEQ(I) = I$:



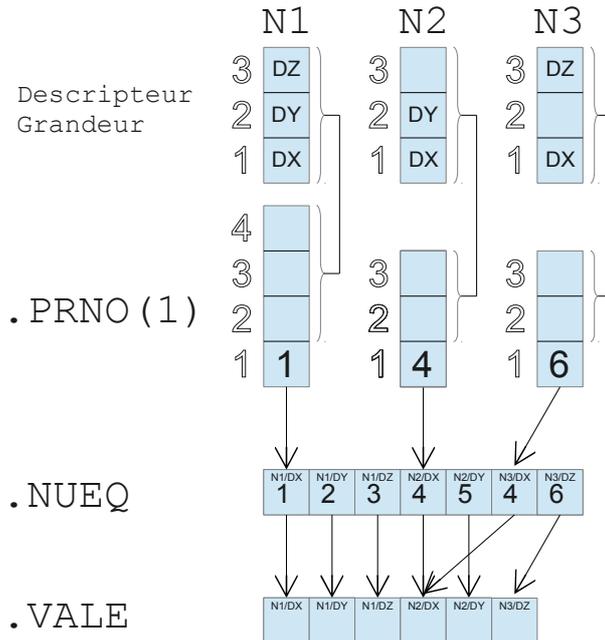
Néanmoins, ce vecteur n'est plus aussi trivial dans deux cas précis :

- Lorsque l'on crée des *macr_elem_stat* dans le but de séparer les degrés de liberté internes des degrés de liberté externes du macro-élément. On peut alors factoriser partiellement les matrices (la partie correspondant aux ddls internes) ;
- Lorsque l'on définit des relations d'identité entre certains degrés de liberté. On utilise ce cas dans certaines situations (XFEM et contact de type LAC).

Pour le premier cas, on a toujours *.NUEQ* et *.VALE* de même longueur. Par contre, l'ordre n'est pas forcément le même. Par exemple, on reprend le cas précédent en disant que le nœud 1 doit être en dernier (il correspond à un macro-élément qui doit se trouver en dernière position pour pouvoir bénéficier des factorisations partielles dans le solveur *LDLT*). Le schéma devient le suivant :



C'est toujours une bijection entre `.NUEQ` et `.VALE` mais ce n'est plus une tautologie ($NUEQ(I) \neq I$). Enfin le dernier cas est la prise en compte de relation d'identité entre deux degrés de liberté. Par exemple, on reprend le même cas, sauf que cette fois-ci, on écrit que le déplacement DX du deuxième nœud est égal au déplacement DX du troisième nœud.



Cette fois-ci, on n'a plus de bijection : il y a moins d'inconnues que de degrés de liberté.

Remarque :

Si le solveur est `MULT_FRONT`, on vérifie que le vecteur correspond à l'identité. Ce qui signifie qu'on ne peut pas utiliser ce solveur dans le cas des macro-éléments statiques ou lorsqu'il existe des relations d'identité entre degrés de liberté. Dans ces cas, il faut utiliser le solveur `MUMPS`.

5.4 Objet DEEQ

L'objet `.DEEQ` est un vecteur d'entiers de longueur $2 * neq$. Avec `neq` est le nombre d'équations du `PROF_CHNO`.

```
◆ .DEEQ      OJB S  V I DIM=2*neq
```

C'est un vecteur « inverse » de `.PRNO` qui décrit (partiellement) les équations.

Si `neq` est un numéro d'équation (i.e. adresse dans l'objet `.VALE`), alors :

```
V((neq-1)*2+1) : ino
```

```
V((neq-1)*2+2) : icmp
```

Avec la convention suivante :

- Si `ino > 0` et `icmp > 0` : `neq` est l'équation associée à la `icmp`^{ième} composante portée par le nœud `ino` du maillage.
- Si `ino > 0` et `icmp < 0` : `neq` est l'une des deux équations qui dualisent le blocage de la `icmp`^{ième} composante du nœud `ino` du maillage.
- Si `ino = 0` et `icmp = 0` : `neq` est une équation de dualisation d'une relation linéaire entre plusieurs composantes.

Remarque :

Dans le cas de la prise en compte des relations d'identité dans la numérotation, comme plusieurs nœuds/composantes correspondent à la même équation, par convention, on indique le premier couple nœud/composante.

6 L'objet `NUML_EQUA`

L'objet `NUML_EQUA` fait référence à la numérotation des équations si le solveur utilisé est `MUMPS` et que l'utilisateur a activé l'option `MATR_DISTRIBUE='OUI'`.

```
NUML_EQUA (K19) ::=record
  ◆ .PRNO      OJB XC V  I
  ◆ .NEQU      OJB S  V  I   dim=2
  ◆ .DELG      OJB S  V  I
  ◆ .NUEQ      OJB S  V  I
  ◆ .NULG      OJB S  V  I
  ◆ .NUGL      OJB S  V  I
```

- ◆ `.PRNO`
Cet objet est identique (dans son architecture) au `.PRNO` d'un `PROF_CHNO`. La différence réside dans le fait que la numérotation des degrés de liberté est locale pour le processeur considéré.
- ◆ `.NEQU`
 - `NEQU(1)` : nombre local d'équations (longueur du `.VALE` des champs nodaux) ;
 - `NEQU(2)` : nombre local de degrés de liberté (longueur de `.NUEQ` dans le `PROF_CHNO`).
- ◆ `.DELG dim=neq local`
Cet objet est similaire à l'objet `.DELG` d'un `NUME_EQUA`.
- ◆ `.NUEQ dim=neq local`
C'est un vecteur d'indirection entre l'objet `.PRNO` et l'objet `.VALE` des `CHAM_NO`. Voir `.NUEQ` d'un `NUME_EQUA`.
- ◆ `.NULG`
C'est un vecteur d'indirection qui permet de passer de la numérotation locale des degrés de liberté (du `.NUML`) à la numérotation globale (du `.NUME`)
- ◆ `.NUGL`
C'est un vecteur d'indirection qui permet de passer de la numérotation globale des degrés de liberté (du `.NUME`) à la numérotation locale (du `.NUML`)

7 L'objet STOCKAGE

Les matrices creuses de Code_Aster ont toutes une structure symétrique : si $a(i, j)$ existe, alors $a(j, i)$ existe également.

Pour une matrice symétrique creuse, on ne stocke que sa partie "supérieure" ($a(i, j)$ pour $j \geq i$). La structure de données `STOCKAGE` décrit comment sont stockés les termes non nuls de la partie supérieure de matrice.

Pour une matrice non symétrique, on stocke la partie supérieure et la partie inférieure. Comme ces 2 parties ont la même structure (à une transposition près), le stockage de la partie supérieure suffit.

Remarque :

Bien que théoriquement, les termes non nuls d'une matrice puissent être disposés de façon quelconque, Le remplissage des matrices de Code_Aster est tel que tous les DDLs portés par un nœud sont connectés à tous les DDLs portés par les autres nœuds jouxtant ce nœud via un élément fini. La matrice est donc formée de petits rectangles pleins correspondant à la connectivité des nœuds du modèle. En particulier, il existe des petits blocs sur toute la diagonale de la matrice.

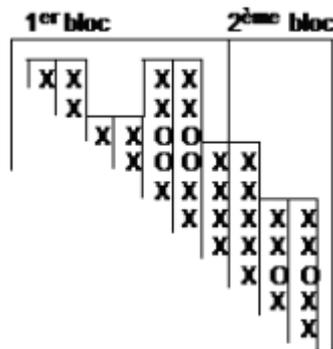
7.1 L'objet STOC_LIGN_CIEL

Pour le stockage en ligne de ciel, les objets sont les suivants :

```
STOC_LCIEL (K19) ::=record
    ◆ .SCBL      OJB  S  V  I
    ◆ .SCDI      OJB  S  V  I
    ◆ .SCDE      OJB  S  V  I
    ◆ .SCHC      OJB  S  V  I
    ◆ .SCIB      OJB  S  V  I
```

.SCDE	S V I dim=4	
	(1)	nombre d'équations <code>neq</code>
	(2)	taille des blocs de la matrice <code>t_bloc</code>
	(3)	nombre de blocs nécessaire au stockage des valeurs de la matrice <code>n_bloc</code>
	(4)	hauteur maximum des colonnes de la matrice
.SCHC	S V I dim= <code>neq</code>	
	(i)	hauteur de la $i^{\text{ème}}$ colonne
.SCDI	S V I dim= <code>neq</code>	
	(i)	adresse du terme diagonal de la $i^{\text{ème}}$ colonne dans son bloc
.SCBL	S V I dim= <code>n_bloc+1</code>	
	(1)	0
	(K+1)	numéro de la dernière colonne du bloc K
		remarque : une colonne ne peut appartenir qu'à un seul bloc
.SCIB	S V I dim= <code>neq</code>	
	(i)	numéro du bloc qui contient la $i^{\text{ème}}$ colonne de la matrice

Exemple :



SCDI (1) =	1	SCDI (6) =	17
SCDI (2) =	3	SCDI (7) =	22
SCDI (3) =	4	SCDI (8) =	6
SCDI (4) =	6	SCDI (9) =	11
SCDI (5) =	11	SCDI (10) =	17
SCHC (1) =	1	SCHC (6) =	6
SCHC (2) =	2	SCHC (7) =	5
SCHC (3) =	1	SCHC (8) =	6
SCHC (4) =	2	SCHC (9) =	5
SCHC (5) =	5	SCHC (10) =	6
SCBL (1) =	0	SCIB (1 à 7) =	1
SCBL (2) =	7	SCIB (7 à 10) =	2
SCBL (3) =	10		

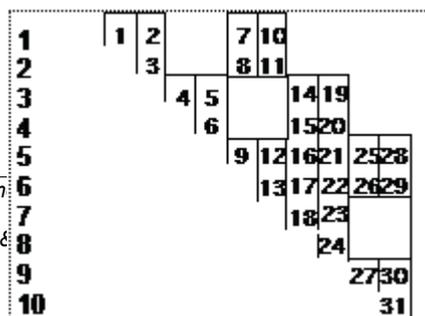
7.2 L'objet STOC_MORSE

Pour le stockage morse, les objets sont les suivants :

```
STOC_MORSE (K19) ::=record
    ♦ .SMDI      OJB  S  V  I
    ♦ .SMDE      OJB  S  V  I
    ♦ .SMHC      OJB  S  V  I
```

.SMDE	S V I dim=3
(1)	nombre d'équations <i>neq</i>
(2)	nombre de termes stockés dans la demi-matrice <i>n_termes</i>
(3)	nombre de bloc 1
.SMHC	S V I dim= <i>n_termes</i>
(i)	numéro de ligne du <i>i</i> ^{ème} terme stocké
.SMDI	S V I dim= <i>neq</i>
(i)	adresse du terme diagonal de la <i>i</i> ^{ème} colonne dans le bloc Il faut donc que tous les termes diagonaux soient stockés dans la matrice

Exemple :



```
SMDI (1) = 1          SMDI (6) = 13
SMDI (2) = 3          SMDI (7) = 18
SMDI (3) = 4          SMDI (8) = 24
SMDI (4) = 6          SMDI (9) = 27
SMDI (5) = 9          SMDI (10) = 31
```

```
SMHC (1) = 1
SMHC (2) = 1
SMHC (3) = 2
SMHC (4) = 3
SMHC (5) = 3
SMHC (6) = 4
SMHC (7) = 1
SMHC (8) = 2
SMHC (9) = 5
...
SMHC (28) = 5
SMHC (29) = 6
SMHC (30) = 9
SMHC (31) = 10
```

7.3 L'objet `MULT_FRONT`

Pour le stockage `MULT_FRONT.`, les objets sont les suivants :

```
MULT_FRONT (K19) ::=record
  ◆ .ADNT      OBJ  S  V  I4
  ◆ .GLOB      OBJ  S  V  I4
  ◆ .LOCL      OBJ  S  V  I4
  ◆ .PNTI      OBJ  S  V  I4
```

Soit `lgind`, la somme du nombre des voisins des super-nœuds.

- ◆ `.GLOB` OBJ S V I4 dim=`lgind`
Ce vecteur donne l'ensemble des voisins des super-nœuds
- ◆ `.LOCL` OBJ S V I4 dim=`lgind`
Ce vecteur établit pour les numéros de lignes des super-nœuds, la correspondance entre la numérotation locale du fils et la numérotation locale du père.
- ◆ `.ADNT` OBJ S V I4 dim=`matr_init`
La dimension `matr_init` est la taille de la matrice initiale (morse). C'est le vecteur des adresses des termes initiaux dans la matrice factorisée.
- ◆ `.PNTI` OBJ S V I4 dim= `19*neq+10`
Cumul dans un même vecteur d'une suite de tables de travail.

8 Exemples

Un exemple de `NUMD_DDL` associé aux trois solveurs linéaires ('LDLT', 'MULT_FRONT', 'GCPC') est donné dans le document [D4.06.10 - Structure de Données `sd_matr_asse`].