Opérateur CREA_RESU

1 But

Créer ou enrichir une structure de données resultat à partir de champs aux nœuds ou par éléments. Affectation possible des champs pour différents numéros d'ordre.

L'utilisateur doit s'assurer de la cohérence des différents champs utilisés pour construire ou enrichir la structure de données resultat .

L'affectation par l'intermédiaire d'un cham_no de fonction produit par CREA_CHAMP [U4.72.04] s'effectue en évaluant chaque fonction à l'aide du paramètre représentant le temps fourni sous les mots clés LIST_INST ou INST.

Le concept produit par cet opérateur est, pour le moment, de type evol_elas, evol_noli, evol_ther, mult_elas, fourier_elas, fourier_ther, evol_varc, evol_char, mode_meca, dyna_trans ou dyna_harmo.

De plus, trois fonctionnalités particulières sont accessibles dans cet opérateur :

- la création d'un concept de type evol_char par affectation de champ ou une formule analytique ;
- la projection d'un transitoire thermique 1D sur un maillage axisymétrique 3D.
- La création d'évolutions transitoires de chargements second membre de type evol_char ou dyna_trans par produit de matrices assemblées par des champs cinématiques, par conversion et combinaison de résultats transitoires ou par assemblages de charges spatio-temporelles de type onde plane ou forces nodales.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

2 Syntaxe

```
resu [resultat]  = CREA_RESU  (
          ◊  reuse = resu,
          ♦   OPERATION    =  / 'AFFE',
                              / 'PERM_CHAM',
                              / 'PROL_RTZ',
                              / 'PREP_VARC',
                              / 'ASSE',
```

 # Construction d'un résultat par affectations ou évaluations successives
 # de cham_no : (OPERATION : 'AFFE')

- la création d'un concept `resultat` simulant la réorganisation des assemblages combustibles ;

```
          ♦  TYPE_RESU                =  'MULT_ELAS' ,
          ♦  NOM_CHAM                 =  nomcham,     [K16]
          ♦  AFFE = _F (
                    ♦  CHAM_GD        =  chno,        [cham_no]
                    ◊  NOM_CAS        =  nomc,        [Kn]
                    ◊  MODELE         =  mo,          [modele]
                    ◊  CHAM_MATER     =  chmat,       [cham_mater]
                    ◊  CARA_ELEM      =  carac,       [cara_elem]
                    ◊  CHARGE          =  char        /[char_meca]
                                                      /[char_cine_meca]
                       ),

          ♦  TYPE_RESU                =  / 'EVOL_ELAS',
          ♦  NOM_CHAM                 =  nomcham,     [K16]
          ◊  EXCIT  =_F(
                       ♦  CHARGE      =     char,           [char_meca]
                       ◊  FONC_MULT   =     fonc,           [fonction]
                       ◊  TYPE_CHARGE =  / typc             [l_Kn]
                                         / 'FIXE'
                            ),
                    ♦  AFFE = _F (
                    ♦  CHAM_GD        =  chno,        [cham_no]
                    ◊  MODELE         =  mo,          [modele]
                    ◊  CHAM_MATER     =  chmat,       [cham_mater]
                    ◊  CARA_ELEM      =  carac,       [cara_elem]
                    ♦  /   INST       =  linst,       [l_R8]
                       /   LIST_INST  =  litps,       [listr8]
                    ◊  NUME_INIT      =  numi,        [I]
                    ◊  NUME_FIN       =  numf,        [I]
                    ◊  PRECISION      =  /prec,       [R]
                                         / 0.0,       [DEFAUT]
                    ◊  CRITERE        =  / 'RELATIF', [DEFAUT]
                                         / 'ABSOLU',
                       ),
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
♦  TYPE_RESU                =  /'EVOL_NOLI',
♦  NOM_CHAM                 =  nomcham,              [K16]
◊  COMPORTEMENT             =   _F(voir le document [U4.51.11]),
◊  VERI_VARI                =  /'OUI'                [DEFAUT]
                               /'NON'
◊  EXCIT  =_F(
          ♦  CHARGE         =  char,                 [char_meca]
          ◊  TYPE_CHARGE    =  / 'FIXE_CSTE'
                               / 'FIXE_PILO'
                               / 'FIXE_PILO'
                               / 'SUIV'
                               / 'DIDI'
          ♦  /  FONC_MULT   =  fonc,                 [fonction]
             /  ♦  DEPL     =  depl,                 [fonction]
                ♦  VITE     =  vite,                 [fonction]
                ♦  ACCE     =  acce,                 [fonction]
             ◊  MULT_APPUI  =  / 'OUI',
                               / 'NON'                [DEFAUT]
             ◊  DIRECTION   =  (d1, d2, d3),         [l_R]
             ◊  GROUP_NO  =  lgrno,              [l_gr_noeud]
             ),
          ♦  AFFE = _F (
          ♦  CHAM_GD         =  chno,         [cham_no]
          ◊  MODELE          =  mo,           [modele]
          ◊  CHAM_MATER      =  chmat,        [cham_mater]
          ◊  CARA_ELEM       =  carac,        [cara_elem]
          ♦  /  INST         =  linst,        [l_R8]
             /  LIST_INST    =  litps,        [listr8]
          ◊  NUME_INIT       =  numi,         [I]
          ◊  NUME_FIN        =  numf,         [I]
          ◊  PRECISION       =  /prec,        [R]
                                / 0.0,        [DEFAUT]
          ◊  CRITERE         =  / 'RELATIF',  [DEFAUT]
                                / 'ABSOLU',
             ),

♦  TYPE_RESU                =  'FOURIER_ELAS',
♦  NOM_CHAM                 =  nomcham,      [K16]
♦  AFFE = _F (
          ♦  CHAM_GD         =  chno,         [cham_no]
          ◊  MODELE          =  mo,           [modele]
          ◊  CHAM_MATER      =  chmat,        [cham_mater]
          ◊  CARA_ELEM       =  carac,        [cara_elem]
          ◊  NUME_MODE       =  num,          [I]
          ◊  TYPE_MODE       =  / 'SYME',     [DEFAUT]
                                / 'ANTI',
                                / 'TOUS',
          ◊  CHARGE           =  char          /[char_meca]
                                               /[char_cine_meca]
             ),
♦  TYPE_RESU                =  'FOURIER_THER',
♦  NOM_CHAM                 =  nomcham,      [K16]
♦  AFFE = _F (
          ♦  CHAM_GD         =  chno,         [cham_no]
          ◊  MODELE          =  mo,           [modele]
          ◊  CHAM_MATER      =  chmat,        [cham_mater]
          ◊  CARA_ELEM       =  carac,        [cara_elem]
          ◊  NUME_MODE       =  num,          [I]
          ◊  TYPE_MODE       =  / 'SYME',     [DEFAUT]
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
                                    / 'ANTI',
                                    / 'TOUS',
                ),

♦  TYPE_RESU                   =   'EVOL_THER',
♦  NOM_CHAM                    =   nomcham,      [K16]
 ◊ EXCIT  =_F(
          ♦  CHARGE            =   char,               [char_ther]
          ◊  FONC_MULT         =   fonc,               [fonction]
             ),
          ♦  AFFE = _F (
          ♦  CHAM_GD           =   chno,         [cham_no]
          ◊  MODELE            =   mo,           [modele]
          ◊  CHAM_MATER        =   chmat,        [cham_mater]
          ◊  CARA_ELEM         =   carac,        [cara_elem]
          ♦  /   INST          =   linst,        [l_R8]
             /   LIST_INST     =   litps,        [listr8]
          ◊  NUME_INIT         =   numi,         [I]
          ◊  NUME_FIN          =   numf,         [I]
          ◊  PRECISION         =   / prec,       [R]
                                   / 0.0,        [DEFAUT]
          ◊  CRITERE           =   / 'RELATIF',  [DEFAUT]
                                   / 'ABSOLU',
             ),

♦  TYPE_RESU                   =   'EVOL_VARC',
♦  NOM_CHAM                    =   nomcham,      [K16]
♦  AFFE = _F (
          ♦  CHAM_GD           =   chno,         [cham_no]
          ◊  MODELE            =   mo,           [modele]
          ◊  CHAM_MATER        =   chmat,        [cham_mater]
          ◊  CARA_ELEM         =   carac,        [cara_elem]
          ♦  /   INST          =   linst,        [l_R8]
             /   LIST_INST     =   litps,        [listr8]
          ◊  NUME_INIT         =   numi,         [I]
          ◊  NUME_FIN          =   numf,         [I]
          ◊  PRECISION         =   / prec,       [R]
                                   / 0.0,        [DEFAUT]
          ◊  CRITERE           =  / 'RELATIF',   [DEFAUT]
                                   / 'ABSOLU',
             ),

♦  TYPE_RESU                   =   'MODE_MECA',
♦  NOM_CHAM                    =   nomcham,      [K16]
◊  MATR_RIGI                   =   matr_k,       [matr_asse_depl_r]
◊  MATR_MASS                   =   matr_m,       [matr_asse_depl_r]
♦  AFFE = _F (
          ♦  CHAM_GD           =   chno,         [cham_no]
          ◊  MODELE            =   mo,           [modele]
          ◊  CHAM_MATER        =   chmat,        [cham_mater]
          ◊  CARA_ELEM         =   carac,        [cara_elem]
          ◊  FREQ              =   freq,         [l_R8]
          ♦  NUME_MODE         =   numo,         [I]
          ◊  AXE               =   axe,          [K16]

             ),

♦  TYPE_RESU                   =   'DYNA_TRANS',
♦  NOM_CHAM                    =   nomcham,      [K16]
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
◊  MATR_RIGI                =  matr_k,       [matr_asse_depl_r]
◊  MATR_MASS                =  matr_m,       [matr_asse_depl_r]
♦  AFFE = _F (
          ♦  CHAM_GD        =  chno,         [cham_no]
          ◊  MODELE         =  mo,           [modele]
          ◊  CHAM_MATER     =  chmat,        [cham_mater]
          ◊  CARA_ELEM      =  carac,        [cara_elem]
          ♦  /  INST        =  linst,        [l_R8]
             /  LIST_INST   =  litps,        [listr8]
             /  NUME_ORDRE  =  nuor,         [I]
          ◊  PRECISION      =  /prec,        [R]
                               / 0.0,        [DEFAUT]
          ◊  CRITERE        =  / 'RELATIF',  [DEFAUT]
                               / 'ABSOLU',
          ),

♦  TYPE_RESU                =  'DYNA_HARMO',
♦  NOM_CHAM                 =  nomcham,     [K16]
◊  MATR_RIGI                =  matr_k,       [matr_asse_depl_r]
◊  MATR_MASS                =  matr_m,       [matr_asse_depl_r]
♦  AFFE = _F (
          ♦  CHAM_GD        =  chno,         [cham_no]
          ◊  MODELE         =  mo,           [modele]
          ◊  CHAM_MATER     =  chmat,        [cham_mater]
          ◊  CARA_ELEM      =  carac,        [cara_elem]
          ♦  /  FREQ        =  lfreq,        [l_R8]
             /  LIST_FREQ   =  lifreq,       [listr8]
             /  NUME_ORDRE  =  nuor,         [I]
          ◊  PRECISION      =  /prec,        [R]
                               / 0.0,        [DEFAUT]
          ◊  CRITERE        =  / 'RELATIF',  [DEFAUT]
                               / 'ABSOLU',
          ),
```

/ # Construction d'un concept de type EVOL_CHAR par affectation ou évaluation
 # d'un cham_no

```
♦  TYPE_RESU                =  'EVOL_CHAR',
♦  NOM_CHAM                 =  nomcham,     [K16]
♦  AFFE = _F (
          ♦  CHAM_GD        =  chno,         [cham_no]
          ◊  MODELE         =  mo,           [modele]
          ◊  CHAM_MATER     =  chmat,        [cham_mater]
          ♦  /  ♦  INST     =  linst,        [l_R8]
             /  ♦  LIST_INST =  litps,       [listr8]
                ◊  NUME_INIT =  numi,        [I]
                ◊  NUME_FIN  =  numf,        [I]
          ◊  PRECISION      =  / prec,       [R]
                               / 0.0,        [DEFAUT]
          ◊  CRITERE        =  / 'RELATIF',  [DEFAUT]
                               / 'ABSOLU',
          ),
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

/ **# Construction d'un résultat dédié aux assemblages combustibles**
 # (OPERATION : 'PERM_CHAM')

```
                ♦  TYPE_RESU                =  'EVOL_NOLI',

                ♦  NOM_CHAM                 =  nomcham,     [K16]
                ♦  RESU_INIT                =  resu_2,      [evol_noli]
                ◊  INST_INIT                =  tf,          [R]
                ◊  PRECISION                =  / prec,
                                               / 1.0E-6,    [DEFAUT]
                ◊  CRITERE                  =  / 'ABSOLU',
                                               / 'RELATIF',
                ♦  MAILLAGE_INIT            =  ma_1,        [maillage]
                ♦  RESU_FINAL               =  resu,        [evol_noli]
                ♦  MAILLAGE_FINAL           =  mo_2,        [maillage]
                ♦  PERM_CHAM =_F(
                   ♦  GROUP_MA_FINAL        = gma_2,        [gr_ma]
                   ♦  GROUP_MA_INIT         = gma_1,        [gr_ma]
                   ♦   TRAN                 =  tx,ty,tz),   [l_R]
                   ◊  PRECISION             =  / prec ,
                                               / 1.0E-3,    [DEFAUT]
                                ),
```

/ # Projection d'un transitoire 1D sur un maillage axisymétrique
 # (OPERATION = 'PROL_RTZ')

```
                ♦  TYPE_RESU                =  'EVOL_THER'
                ♦  PROL_RTZ=_F (
                   ♦  MAILLAGE_FINAL        =  ma_3D,       [maillage]
                   ♦  TABLE                 =  post_1D,     [table]
                   ◊  /  INST               =  inst,        [l_R]
                      /  LIST_INST          =  linst,       [listR8]
                   ◊  PRECISION             =  / prec,
                                               / 1.0E-6,    [DEFAUT]
                   ◊  CRITERE               =  / 'ABSOLU',
                                               / 'RELATIF', [DEFAUT]
                   ◊  PROL_DROITE           =  / 'EXCLU',   [DEFAUT]
                                               / 'LINEAIRE',
                                               / 'CONSTANT',
                   ◊  PROL_GAUCHE           =  / 'EXCLU',   [DEFAUT]
                                               / 'LINEAIRE',
                                               / 'CONSTANT',
                   ♦  REPERE                =  'CYLINDRIQUE',
                   ♦  ORIGINE               =  (ori1,ori2,ori3),   [l_R]
                   ♦  AXE_Z                 =  (axe1,axe2,axe3),   [l_R]
                                ),
```

/ # Construction d'un résultat de type EVOL_THER pour calculer la température
 # dans les couches des coques de type multicouche à partir soit :
 # cham_gd de fonctions du temps et de l'espace
 # evol_ther "coque" contenant TEMP_MIL/TEMP_INF/TEMP_SUP

```
        ♦  TYPE_RESU = 'EVOL_THER'
        ♦  PREP_VARC = _F   (
            ♦ /  CHAM_GD                =  chno,
[cham_no]
              EVOL_THER           =  evol,
[evol_ther]
           ♦  MODELE                    =  mo,                    [modele]
           ♦  CARA_ELEM                 =  carac,
[cara_elem]
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
                #  Si CHAM_GD
               ♦  /  INST              =  inst,                          [l_R8]
                  /  LIST_INST         =  linst,                      [listr8]
                #  Si EVOL_THER
               ◊  /  INST              =  inst,                          [l_R8]
                  /  LIST_INST         =  linst,                      [LISTR8]
                  ◊   CRITERE           =  / 'ABSOLU',
                                           / 'RELATIF',                 [DEFAUT]
                  ◊  PRECISION         =  / PREC,
                                           / 1.0E-6,                    [DEFAUT]
               ◊  /  TOUT              =  'OUI',                       [DEFAUT]
                  /  GROUP_MA          =  lgma ,
   [l_gr_maille]
         ),
/  # Création par assemblage de structures de données résultat evol_ther :
   # (OPERATION : 'ASSE')

                    ♦  TYPE_RESU              =  'EVOL_THER'
                    ♦  ASSE =        _F  (
                       ♦  RESULTAT            =  evol,           [evol_ther]
                       ◊  TRANSLATION         =  /  tr,          [R]
                                                 /  ~~~          [DEFAUT]
                                          ),
      )
/ # Construction d'un résultat de type EVOL_CHAR ou DYNA_TRANS à partir de
  # produits de matrices assemblées et champs cinématiques
  # (OPERATION : 'KUCV' )

                  ♦  TYPE_RESU                =  / 'EVOL_CHAR',
                                                 / 'DYNA_TRANS',
                  ♦  KUCV =_F  (
                     ♦  RESU_INIT             =  resui,       / [dyna_trans]
                                                              / [evol_char]
                                                              / [evol_noli]
                     ♦  MATR_AMOR             =  matr_a,      [matr_asse_depl_r]
                     ◊  MATR_RIGI             =  matr_k,      [matr_asse_depl_r]
                     ♦  /  INST               =  inst,        [l_R]
                        /  LIST_INST          =  linst,       [listr8]
                     ◊  CRITERE               =  / 'ABSOLU',
                                                 / 'RELATIF', [DEFAUT]
                     ◊  PRECISION             =  / prec ,
                                                 / 1.0E-6,    [DEFAUT]
                              ),

/ # Construction d'un résultat de type DYNA_TRANS par assemblages de charges
  # (OPERATION : 'CONV_CHAR' )

                  ♦  TYPE_RESU                =  / 'DYNA_TRANS',
                  ♦  CONV_CHAR =_F(
                     ♦  CHARGE                =  lchar,       [l_char_meca]
                     ♦  MATR_RIGI             =  matr_k,      [matr_asse_depl_r]
                     ♦  /  INST               =  inst,        [l_R]
                        /  LIST_INST          =  linst,       [listr8]
                     ◊  CRITERE               =  / 'ABSOLU',
                                                 / 'RELATIF', [DEFAUT]
                     ◊  PRECISION             =  / prec ,
                                                 / 1.0E-6,    [DEFAUT]
                                 ),
/ # Construction d'un résultat de type EVOL_CHAR ou DYNA_TRANS
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

 # par conversion et combinaison de résultats transitoires
 # (OPERATION : 'CONV_RESU')

```
♦  TYPE_RESU                  =  / 'EVOL_CHAR',
                                 / 'DYNA_TRANS',
♦  CONV_RESU =_F(
   ♦  RESU_INIT               =  resui,        / [dyna_trans]
                                                / [evol_char]
                                                / [evol_noli]
   ◊  NOM_CHAM_INIT=          =  / 'DEPL',
                              =  / 'ACCE',
                              =  / 'FORC_NODA',
                              =  / 'REAC_NODA',
   ♦  /  MATR_RIGI            =  matr_k,       [matr_asse_depl_r]
      /  NUME_DDL             =  nume,         [nume_ddl]
   ♦  /  INST                 =  inst,         [l_R]
      /  LIST_INST            =  linst,        [listr8]
   ◊  CRITERE                 =  / 'ABSOLU',
                                 / 'RELATIF',  [DEFAUT]
   ◊  DDL_EXCLUS              =  / 'DX',
                                 / 'DY',
                                 / 'DZ',
                                 / 'DRX',
                                 / 'DRY',
                                 / 'DRZ',
   ◊  PRECISION               =  / prec ,
                                 / 1.0E-6,     [DEFAUT]
   ◊  COEF                    =  / coef ,
                                 / 1.0,        [DEFAUT]
   /  ♦  GROUP_NO_INTERF  =  gni               [grno]
      ♦  VITE_ONDE     =     vite_onde        [R]
      ♦  DIRECTION     =     dir              [R]
      ♦  COOR_REFE     =     crefe            [R]
   /  ♦  FONC_DX          =  fx               [fonction]
      ◊  FONC_DY          =  fy               [fonction]
      ◊  FONC_DZ          =  fz               [fonction]
      ◊  FONC_DRX         =  frx              [fonction]
      ◊  FONC_DRY         =  fry              [fonction]
      ◊  FONC_DRZ         =  frz              [fonction]
                          )
                  ),
```

Si TYPE_RESU : 'MULT_ELAS' alors resu de type mult_elas
Si TYPE_RESU : 'FOURIER_ELAS' alors resu de type fourier_elas
Si TYPE_RESU : 'FOURIER_THER' alors resu de type fourier_ther
Si TYPE_RESU : 'EVOL_THER' alors resu de type evol_ther
Si TYPE_RESU : 'EVOL_VARC' alors resu de type evol_varc
Si TYPE_RESU : 'EVOL_ELAS' alors resu de type evol_elas
Si TYPE_RESU : 'EVOL_NOLI' alors resu de type evol_noli
Si TYPE_RESU : 'EVOL_CHAR' alors resu de type evol_char
Si TYPE_RESU : 'MODE_MECA' alors resu de type mode_meca
Si TYPE_RESU : 'DYNA_TRANS' alors resu de type dyna_trans
Si TYPE_RESU : 'DYNA_HARMO' alors resu de type dyna_harmo

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

3 Opérandes

3.1 Opérande OPERATION

♦ OPERATION = définit le type d'opération à effectuer avec cet opérateur :

' AFFE ' : création d'une structure de données résultat à partir de champs.
C'est à l'utilisateur de s'assurer de la cohérence des champs fournis pour créer la structure de données et de vérifier qu'ils s'appuient sur le même modèle.

'PERM_CHAM' : réorganisation des assemblages combustibles,

'PROL_RTZ' : prolongement d'un champ $1D$ sur une structure axisymétrique,

' PREP_VARC ' : calcul de la température dans les couches d'une coque soit en partant :

- d'une carte de fonctions créée à partir d'une température donnée par une NAPPE ou une FORMULE du type :

$$temp = f(inst, epais) \quad ; \quad temp = f(inst, excent) \quad ; \quad temp = f(inst, x, y, z)$$

où :

- $inst$ est le temps,
- $epais$ l'épaisseur de la coque dans $[-h/2, h/2]$,
- $excent$ est la position dans la couche en tenant compte de l'excentrement de la coque dans $[excent - h/2, excent + h/2]$.
- x, y, z coordonnées du centre de gravité de la couche.

- d'une température calculée par aster avec un modèle de coques (TEMP_MIL / TEMP_INF / TEMP_SUP),

' ASSE ' : création d'une structure de données résultat à partir de plusieurs structures de données résultat mises bout à bout.

' KUCV ' : création d'évolutions transitoires de chargements second membre de type evol_char ou dyna_trans par produit de matrices assemblées par des champs cinématiques de vitesses et/ou déplacements.

' CONV_RESU ' : création d'évolutions transitoires de chargements second membre de type evol_char ou dyna_trans par conversion et combinaison de résultats transitoires.

' CONV_CHAR ' : création d'évolutions transitoires de chargements second membre de type dyna_trans par assemblages de charges spatio-temporelles de type onde plane ou forces nodales.

La structure de données résultat peut-être réentrante pour le opération AFFE. Elle l'est toujours pour PERM_CHAM
Pour OPERATION = 'AFFE', les champs existants peuvent être remplacés suivant la valeur de la variable d'accès INST en utilisant les valeurs renseignées derrière les mots clés PRECISION et CRITERE. Quand il y a remplacement d'un champ existant, le code émet un message d'alarme, sinon les champs sont stockés à la fin de la structure de données.

3.2 Opérande TYPE_RESU

♦ TYPE_RESU
Type de la structure de données résultat créée.
Dans le cas d'un résultat de type EVOL_VARC et d'une évaluation d'un champ de fonctions (temps et espace), on vérifie la cohérence entre la nature du champ de fonctions et le nom du champ donné sous NOM_CHAM . Si par exemple, le champ de fonctions est du type NOEU_NEUT_F le nom du champ doit être NEUT .

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

3.3 Opérande NOM_CHAM

♦ NOM_CHAM
Nom symbolique du champ à affecter. Ce nom doit être cohérent avec la structure de données modifiée ou créée. Il peut prendre par exemple la valeur 'DEPL', 'VARI_ELGA', 'TEMP', 'FLUX_ELNO', 'IRRA', etc .
Dans le cas d'un résultat de type EVOL_VARC et d'une évaluation d'un champ de fonctions (temps et espace) *Code_Aster* vérifie la cohérence entre la nature du champ de fonctions et le nom du champ donné sous NOM_CHAM . Si par exemple, le champ de fonctions est du type NOEU_NEUT_F le nom du champ doit être NEUT .

Dans le cas d'un résultat de type 'EVOL_CHAR', les champs que l'on peut créer sont :

PRES	Champs de pression (N/m^2) , composante $PRES$
FVOL_3D	Champs de forces volumiques (N/m^3) , composantes FX , FY , FZ
FVOL_2D	Champs de forces volumiques (N/m^3) , composantes FX , FY
FSUR_3D	Champs de forces surfaciques (N/m^2) , composantes FX , FY , FZ
FSUR_2D	Champs de forces surfaciques (N/m^2) , composantes FX , FY
VITE_VENT	Champs aux nœuds de vitesse du vent (m/s) , composantes DX , DY , DZ
T_EXT	Carte de température extérieure, composante $TEMP$
COEF_H	Carte de coefficient d'échange, composante H
FLUN	Carte de flux normaux, composante $FLUN$
FORC_NODA	Champs aux nœuds, composantes FX , FY , FZ

3.4 Opérande COMPORTEMENT

La syntaxe de ce mot-clé commun à plusieurs commandes est décrite dans le document [U4.51.11]. Ce mot-clef doit être renseigné dans le cas de la mécanique non-linéaire car il sert en reprise de calcul dans STAT_NON_LINE et DYNA_NON_LINE pour vérifier la compatibilité des comportements (nombre de variables internes en particulier).

Lors de l'opération AFFE dans le cas de la mécanique non-linéaire,
- sans reuse , une nouvelle structure de données résultat sera créée avec le comportement renseigné. Si COMPORTEMENT n'est pas renseigné, le comportement pris par défaut est élastique (COMPORTEMENT='ELAS') en petites déformations (RELATION='PETIT').
- On utilise reuse pour ajouter ou remplacer des champs dans le résultat. Si COMPORTEMENT est renseigné, on remplace sa carte de comportement pour tous les instants ; sinon, on garde le comportement pour l'instant existant et on récupère celui du précédent instant pour l'instant inexistant (élastique en petites déformations sans précédent instant).

En fin de l'opération AFFE, on vérifie, pour chaque instant, la cohérence entre sa carte de comportement et le nombre des variables internes (voir mot-clef VERI_VARI). Malgré les incohérences, des cas autorisés permet une reprise de ce champ (VARI_ELGA) dans un calcul non-linéaire, et le calcul devrait bien se passer (correction automatique dans l'opérateur).

3.5 Opérande VERI_VARI

◊ VERI_VARI = /'OUI' [DEFAUT]
/'NON'

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Ce mot-clef sert à vérifier la cohérence entre la taille des champs de type VARI_ELGA et la définition du comportement non-linéaire.
S'il y a des différences, des alarmes sont émises.

3.6 Opérandes EXCIT

Pour qu'un résultat issu de la commande CREA_RESU soit exploitable par d'autres commandes, il est nécessaire de construire et de renseigner la structure de données en précisant les charges associées. Le mot clé facteur EXCIT est utilisé pour les TYPE_RESU : EVOL_ELAS, EVOL_NOLI et EVOL_THER. On se reportera aux documents respectifs U4.51.01, U4.51.03 et U4.54.01.

3.7 Mot clé CHAM_GD

3.7.1 Opérande CHAM_GD

♦ CHAM_GD = chno
chno est :
1) soit un CHAM_NO de fonction créé par la commande CREA_CHAMP [U4.72.04] et dans ce cas on évalue pour chaque nœud la fonction et pour chaque instant défini derrière LIST_INST ou INST on crée un CHAM_NO de réels,
2) soit un cham_no ou soit un CHAM_ELEM de réels créé par la commande CREA_CHAMP (mot d'AFFE ou EXTR) et ce champ est dupliqué autant de fois que la liste d'instants définie derrière LIST_INST ou INST le nécessite.

3.7.2 Opérandes MODELE, CHAM_MATER, CARA_ELEM, CHARGE

Ces opérandes facultatifs sont utilisés pour permettre le remplissage des structures de données résultat. Ce remplissage est indispensable dans le cas où la commande CREA_RESU est appelée par MACRO_ELAS_MULT pour utiliser ensuite les commandes de post-traitement qui vont rechercher cette information dans la structure de données.

◊ MODELE = mo,
Nom du modèle dont les éléments font l'objet du calcul.

◊ CHAM_MATER = chmat,
Nom du champ de matériau.

◊ CARA_ELEM = carac,
Nom des caractéristiques des éléments structuraux (poutre, coque, discret, …) s'ils sont utilisés dans le modèle. Lorsque OPERATION prend la valeur PREP_VARC, on y récupère les composantes EPAIS et COQU_NCOU.

◊ CHARGE = char,
Nom d'un concept de type char_meca produit par AFFE_CHAR_MECA ou par AFFE_CHAR_MECA_F [U4.44.01] à partir du modèle mo. On peut également donner le nom d'une "charge cinématique" (type char_cine_meca) résultat des opérateurs AFFE_CHAR_CINE ou AFFE_CHAR_CINE_F [U4.44.03].

3.7.3 Opérandes LIST_INST / LIST_FREQ / NUME_INIT / NUME_FIN

♦ LIST_INST = litps
Liste de réels produite par DEFI_LIST_REEL [U4.34.01].

♦ LIST_FREQ = lifreq
Liste de réels produite par DEFI_LIST_REEL [U4.34.01].

◊ NUME_INIT = nuini

◊ NUME_FIN = nufin
Les instants de calcul sont ceux définis dans le concept litps pris entre le nuini et le nufin numéro d'instant. En l'absence du mot clé NUME_FIN, c'est la taille de la liste de réels qui est prise en compte.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

3.7.4 Opérandes INST

♦ INST = linst
Liste de réels : liste des instants pour lesquels le cham_no de fonction sera évalué, ou bien le cham_no de réels sera affecté.

Remarque :
Le numéro d'ordre créé dans le concept resultat est soit récupéré à partir de la valeur de la variable d'accès INST lorsque elle est présente, soit affecté à la valeur maximum immédiatement supérieure.

3.7.5 Opérandes FREQ

Dans le cas MODE_MECA/MODE_MECA_C :

◊ FREQ = freq
Valeur de la fréquence.

Cette opérande est facultative, cela permet, dans le cas d'un concept réentrant, de pouvoir déclarer un autre champ pour un même numéro de mode (NUME_MODE) sans avoir à fournir la fréquence.

A noter que si l'utilisateur déclare un champ (par exemple EFGE_ELNO) pour un NUME_MODE pour lequel un autre champ existe déjà avec une fréquence associée (par exemple DEPL) et qu'il renseigne l'opérande FREQ avec une valeur différente, le concept ne pourra par être traité par COMB_SISM_MODAL.

Dans les autres cas :
♦ FREQ = lfreq
Liste de réels : liste des fréquences pour lesquelles le cham_no de fonction sera évalué, ou bien le cham_no de réels sera affecté.

Remarque :
Le numéro d'ordre créé dans le concept resultat est soit récupéré à partir de la valeur de la variable d'accès FREQ lorsque elle est présente, soit affecté à la valeur maximum immédiatement supérieure.

3.7.6 Opérandes PRECISION / CRITERE

Ces opérandes permettent d'affiner l'accès par variables d'accès réelles du temps ou de la fréquence.

```
l PRECISION =  / prec                        [R]
               / 0.0 ou 1.0D-3 ou 1.0D-6     [DEFAUT]
```

Ce mot clé permet d'indiquer que l'on recherche tous les champs dont l'instant (respectivement la fréquence) se trouve dans l'intervalle "inst ± prec" (confer CRITERE).

Dans le cas où OPERATION = 'AFFE', la valeur par défaut prec est fixée à 0.0 pour éviter d'écraser un champ dont la valeur de l'instant est proche de celui que l'on traite. l'instant fourni ne sert pas à récupérer un champ dans la structure de données, c'est un attribut qu'il faut associer au champ que l'on stocke. En général, les champs que l'on stocke correspondent tous à des instants différents.
Dans le cas très rare où l'utilisateur souhaiterait écraser l'un des champs contenus dans la structure de données, il devra utiliser le mot clé PRECISION. Un message d'alarme indique alors le nom des champs concernés avec leurs instants de stockage, et la précision fournie par l'utilisateur:

```
l CRITERE  =  / 'RELATIF'          [DEFAUT]
              / 'ABSOLU'
```

'RELATIF' : l'intervalle de recherche est : [inst (1 - prec), inst (1 + prec)]
'ABSOLU' : l'intervalle de recherche est : [inst - prec, inst + prec].

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

3.7.7 Opérandes NUME_MODE / TYPE_MODE

Dans le cas MODE_MECA/MODE_MECA_C :

♦ NUME_MODE = num

Entier désignant le numéro du mode dans le cas TYPE_RESU='MODE_MECA'.

Dans le cas FOURIER_ELAS :

◊ NUME_MODE = num

Entier désignant le numéro de l'harmonique de Fourier du champ stocké dans un concept de type fourier_elas.

◊ TYPE_MODE = / 'SYME'
/ 'ANTI'
/ 'TOUS'

Définit le type du mode de Fourier stocké.

'SYME' : harmonique symétrique
'ANTI' : harmonique antisymétrique
'TOUS' : harmonique symétrique et antisymétrique

3.7.8 Opérande AXE

Disponible dans le cas MODE_MECA/MODE_MECA_C uniquement :

◊ AXE = / 'X'
/ 'Y'
/ 'Z'

Permet de définir une direction pour une numéro d'ordre donné afin que le concept en sortie puisse être fourni à l'opérande MODE_CORR de COMB_SISM_MODAL [U4.84.01].

3.7.9 Opérande NOM_CAS

♦ NOM_CAS = nomc

Chaîne de caractères définissant la variable d'accès du champ stocké dans un concept de type mult_elas.

3.7.10 Opérandes MATR_RIGI/MATR_MASS

Dans le cas où TYPE_RESU='MODE_MECA','DYNA_HARMO' ou 'DYNA_TRANS' :

◊ MATR_RIGI = matr_k
Matrice de rigidité correspondant aux champs stockés.

◊ MATR_MASS = matr_m
Matrice de masse correspondant aux champs stockés.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

4 Opérandes associés aux assemblages combustibles

4.1 Opérandes RESU_INIT

♦ RESU_INIT = rinit
Nom de la SD evol_noli contenant les champs à transférer sur le nouveau maillage.

4.2 Opérandes INST_INIT / PRECISION/CRITERE

♦ INST_INIT = iinit
Instant caractérisant dans la SD evol_noli indiquée sous RESU_INIT, les champs à transférer sur l'autre maillage. Par défaut, le dernier instant archivé est sélectionné

♦ PRECISION = prec
Précision utilisée pour rechercher l'instant spécifié par INST_INIT dans la structure de données evol_noli associée à RESU_INIT.

♦ CRITERE = / 'RELATIF' [DEFAUT]
/ 'ABSOLU'

Critère utilisé pour rechercher l'instant spécifié par INST_INIT dans la structure de données evol_noli associée à RESU_INIT.

4.3 Opérandes MAILLAGE_INIT

♦ MAILLAGE_INIT = maillagei
Nom du maillage sur lequel a été définie la SD evol_noli indiquée sous RESU_INIT.

4.4 Opérandes RESU_FINAL

♦ RESU_FINAL = resu
Nom de la structure de données evol_noli définie sur le nouveau maillage sur lequel seront transférés les champs. La structure de données resu doit exister (elle aura été créée par exemple par la commande STAT_NON_LINE) et ne doit contenir qu'un seul numéro d'ordre.

4.5 Opérandes MAILLAGE_FINAL

♦ MAILLAGE_FINAL = mailfin
Nom de la structure de données maillage créée sur le nouveau maillage sur lequel seront transférés les champs.

4.6 Mot clé PERM_CHAM

4.6.1 Opérandes GROUP_MA_FINAL

♦ GROUP_MA_FINAL = gma_2
Nom du groupe de mailles du MAILLAGE_FINAL, lieu où les champs sont transférés dans RESU_FINAL.

4.6.2 Opérandes GROUP_MA_INIT

♦ GROUP_MA_INIT = gma_1
Nom du maillage sur lequel a été définie la structure de données evol_noli indiquée sous RESU_INIT.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

4.6.3 Opérande TRAN

♦ TRAN = (tx,ty,tz)
Vecteur translation permettant d'obtenir géométriquement GROUP_MA_FINAL à partir de GROUP_MA_INIT. Il est nécessaire de fournir exactement 3 valeurs.

4.6.4 Opérande PRECISION

◊ PRECISION = prec
Précision absolue permettant de vérifier la bonne adéquation entre les mailles initiales et les mailles finales, par défaut la valeur est fixée à 10^{-3} .

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

5 Opérandes associés à la projection sur un maillage axisymétrique

5.1 Mot clé PROL_RTZ

Construction d'un transitoire thermique sur un maillage axisymétrique (3D) à partir de la donnée d'un transitoire thermique calculé sur un maillage 1D. Le transitoire 1D est donné sous la forme d'une structure de données TABLE issue de la commande POST_RELEVE_T possédant les paramètres suivants :

- la définition des instants ('INST'),
- les coordonnées des nœuds du maillage 1D ('COOR_X')
- la valeur des températures aux nœuds ('TEMP').

Les coordonnées de la table doivent nécessairement avoir pour origine le nœud de coordonnée 0.
Les valeurs des températures peuvent éventuellement être prolongées de façon constante ou bien interpolées linéairement en fonction de la coordonnée 'COOR_X'.

5.1.1 Opérandes MAILLAGE_FINAL

♦ MAILLAGE_FINAL = mailfin
Nom du maillage sur lequel on effectue la projection, l'opérateur vérifie que le maillage est tridimensionnel .

5.1.2 Opérandes TABLE

♦ TABLE = table
Nom d'une structure de données TABLE issue de la commande POST_RELEVE_T contenant le transitoire thermique 1D. Les paramètres de cette table sont obligatoirement : 'INST' , 'COOR_X' et 'TEMP'.

5.1.3 Opérandes INST / LIST_INST / PRECISION / CRITERE

◊ INST = litps
Liste de valeurs réelles.

◊ LIST_INST = litps
Liste de réels produite par DEFI_LIST_REEL [U4.34.01].

◊ PRECISION = / prec [R]
/ 1-0D-6 [DEFAUT]
Précision utilisée pour rechercher l'instant spécifié dans la TABLE post_1D.

◊ CRITERE = / 'RELATIF',
/ 'ABSOLU,'
Critère utilisé pour rechercher l'instant spécifié dans la TABLE post_1D.

5.1.4 Opérandes PROL_DROITE et PROL_GAUCHE

La projection du transitoire est effectuée selon la coordonnée COOR_X considérée comme la coordonnée r dans le repère cylindrique du maillage 3D. On peut définir à l'aide de ces deux opérandes la façon de prolonger le champ au-delà des bornes définies par la plage de variation du paramètre 'COOR_X' dans la table.

◊ PROL_DROITE et PROL_GAUCHE =
Définissent le type de prolongement à droite (à gauche) du domaine de définition de la variable :

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- 'CONSTANT' pour un prolongement avec la dernière (ou première) valeur de la fonction,
- 'LINEAIRE' pour un prolongement le long du premier segment défini (PROL_GAUCHE) ou du dernier segment défini (PROL_DROITE),
- 'EXCLU' si l'extrapolation des valeurs en dehors du domaine de définition du paramètre est interdite (dans ce cas si un calcul demande une valeur de la fonction hors du domaine de définition, le code s'arrêtera en erreur fatale).

5.1.5 Opérande REPERE/ORIGINE/AXE_Z

♦ REPERE = 'CYLINDRIQUE'
Le repère de travail pour projeter le transitoire est supposé cylindrique, le transitoire 1D étant considéré comme la variation radiale du champ de température. Les deux opérandes suivants permettent d'effectuer un changement de repère.

♦ ORIGINE = (ori1,ori2,ori3)
Correspond à la position de l'origine du maillage 1D par rapport à l'origine du maillage 3D.

♦ AXE_Z = (axe1,axe2,axe3)
Définition de l'axe du repère cylindrique.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

6 Opérandes associés à la préparation des variables de commande

6.1 Mot clé PREP_VARC

L'évolution thermique que l'on peut associer au champ de matériau par AFFE_MATERIAU/AFFE_VARC doit être prête à être utilisée par les éléments finis du modèle mécanique. Un problème se pose pour les éléments de type coque qui utilisent une température variant dans l'épaisseur sur les différentes couches. Pour ces éléments, il est nécessaire de préparer le calcul de la température sur les couches en amont de la commande AFFE_MATERIAU. Pour cela, l'utilisateur doit utiliser la commande CREA_RESU avec l'opération PREP_VARC (PREParation des VaRiables de Commande).

6.1.1 Opérande CHAM_GD

♦ CHAM_GD = chgd
chgd est une carte de fonctions créée à partir d'une température donnée par une NAPPE ou une FORMULE du type :

$$temp = f(inst, epais) \ ; temp = f(inst, zexcen) \ ; temp = f(inst, x, y, z)$$

où :
- $inst$ est le temps,
- $epais$ l'épaisseur de la coque dans $[-h/2, h/2]$,
- $excent$ est la position dans la couche en tenant compte de l'excentrement de la coque, dans $[excent - h/2, excent + h/2]$.
- x, y, z coordonnées, dans le repère du maillage, du centre de gravité de la couche.

6.1.2 Opérande EVOL_THER

♦ EVOL_THER = evol
evo est une structure de données EVOL_THER de type « coque », c'est à dire contenant les composantes TEMP_MIL/TEMP_INF/TEMP_SUP.

6.2 Opérandes TOUT / GROUP_MA

Uniquement dans le cas EVOL_THER .

◊ / TOUT = 'OUI' , [DEFAUT]
Ce mot clé permet d'effectuer l'opération sur toutes les mailles du maillage.
/ GROUP_MA = lgma,
Ce mot clé permet d'effectuer l'opération sur une liste de groupes de mailles du maillage.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

7 Opérandes associés à l'assemblage de structure de données de type résultat

7.1 Mot clé ASSE

Permet d'assembler plusieurs structures de données evol_ther en les mettant bout à bout en translatant la valeur du paramètre temps.

7.1.1 Opérande RESULTAT

♦ RESULTAT = resu
resu est une structure de données evol_ther.

Tous les champs présents dans la structure de données sont traités, cela concerne
'TEMP','FLUX_ELGA','FLUX_ELNO','FLUX_NOEU','META_ELNO','META_NOEU',
'DURT_ELNO','DURT_NOEU','HYDR_ELNO','HYDR_NOEU','DETE_ELNO',
'DETE_NOEU','SOUR_ELGA','COMPORTHER','ERTH_ELEM','ERTH_ELNO',
'ERTH_NOEU'.

7.1.2 Opérande TRANSLATION

◊ TRANSLATION = / tr, [R]
/ 0. [DEFAUT]

tr est la valeur réelle qui sera ajoutée à la valeur de l'attribut INST pour chaque champ de la structure de données resu avant insertion dans la structure de données résultat.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

8 Opérandes associés à la constitution d'évolutions temporelles second membre

8.1 Mot clé KUCV

Construction d'un résultat de type evol_char ou dyna_trans second membre, à affecter ensuite comme charge dans DYNA_VIBRA ou DYNA_NON_LINE, à partir de produits de matrices assemblées d'amortissement et de rigidité par des champs cinématiques de vitesses et déplacements extraits d'une évolution déjà calculée. Cela permet de procéder aux produits $KU+CV$ auxquels on accède habituellement par des appels successifs à PROD_MATR_CHAM dans une boucle temporelle.

8.1.1 Opérande RESU_INIT

♦ RESU_INIT = resui
Nom du résultat d'évolution calculée de type dyna_trans, evol_char ou evol_noli à partir de laquelle on extrait des champs cinématiques de vitesses et déplacements utilisés dans les produits par des matrices assemblées.

8.1.2 Opérandes MATR_AMOR / MATR_RIGI

♦ MATR_ AMOR = matr_ a
◊ MATR_RIGI = matr_k,
Noms des matrices assemblées d'amortissement et de rigidité (facultative) que l'on utilise dans les produits $KU+CV$ constituant l'évolution produite.

8.2 Mot clé CONV_RESU

Construction d'un résultat de type evol_char ou dyna_trans (second membre), à affecter ensuite comme charge dans DYNA_VIBRA ou DYNA_NON_LINE, à partir d'une évolution déjà calculée. Cela permet de changer éventuellement le type de résultat ainsi que le champ produit. Ainsi, on peut selon le type d'entrée à fournir dans les opérateurs dynamiques avoir à changer une évolution de type evol_char et de champ 'FORC_NODA' en une évolution de type 'dyna_trans' et de champ 'DEPL' ou inversement. Ou bien de changer une évolution de type dyna_trans et de champ 'FORC_NODA' en une autre évolution de type dyna_trans mais de champ 'DEPL'.

8.2.1 Opérande RESU_INIT

♦ RESU_INIT = resui
Nom du résultat d'évolution calculée de type evol_char, dyna_trans ou evol_noli qu'on convertit ensuite en un autre résultat de type evol_char ou dyna_trans en changeant éventuellement le nom de champ.

8.2.2 Opérande NOM_CHAM_INIT

◊ NOM_CHAM_INIT = / 'DEPL',
/ 'ACCE',
/ 'FORC_NODA'
/ 'REAC_NODA',
Nom du champ du résultat de l''évolution de départ calculée de type evol_char ou dyna_trans qu'on convertit ensuite dans le champ du résultat de l'évolution produite :soit forcément 'FORC_NODA' pour le type evol_char ou 'DEPL' pour le dyna_trans.

8.2.3 Opérandes NUME_DDL / MATR_RIGI

♦ / MATR_RIGI = matr_k

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

/ NUME_DDL = nume

Entrées à partir desquelles on peut obtenir la numérotation de référence ou conversion de l'évolution produite . La donnée de l'opérande MATR_RIGI est conseillée si on produit une évolution de type dyna_trans .

8.2.4 Opérande COEF

◊ COEF = / coef [R]
/ 1.0 [DEFAUT]

Coefficient réel de combinaison de l' évolution produite.

8.2.5 Opérande DDL_EXCLUS

◊ DDL_EXCLUS = nom_cmp [TXM]

Nom de composante à exclure sur le champ du résultat initial qu'on convertit. S'applique en général sur un champ 'FORC_NODA'. On prend tout le champ si le mot clé n'est pas renseigné.

8.2.6 Opérande GROUP_NO_INTERF

♦ GROUP_NO_INTERF = gni [grno]

Avec ce mot-clé, on définit le groupe de nœuds pour lesquels on souhaite générer un champ sismique spatio-temporel par déphasage dû à un passage d'onde .

8.2.7 Opérande VITE_ONDE

♦ VITE_ONDE = vite_onde, [R]

La vitesse de propagation de l'onde.

8.2.8 Opérande DIRECTION

♦ DIRECTION = dir, [R]

La direction de propagation de l'onde (vecteur directionnel : 3 valeurs).

8.2.9 Opérande COOR_REFE

♦ COOR_REFE = crefe [R]

Ce mot-clé permet de renseigner les coordonnées (3D) d'un point de référence pour le calcul du déphasage (temps d'arrivée de l'onde).

8.2.10 Opérandes FONC_DX / FONC_DY / FONC_DZ / FONC_DRX / FONC_DRY / FONC_DRZ

♦ FONC_DX = fx [fonction]
◊ FONC_DY = fy [fonction]
◊ FONC_DZ = fz [fonction]
◊ FONC_DRX = frx [fonction]
◊ FONC_DRY = fry [fonction]
◊ FONC_DRZ = fry [fonction]

Ces mots-clés permettent de définir pour chacune des composantes 'DX', 'DY', 'DZ', 'DRX', 'DRY', 'DRZ', et seulement si nécessaire, des fonctions ou nappes ou formules dépendant de certaines de ces composantes, ainsi qu'éventuellement du paramètre 'INST', pour des champs de l'évolution entrée par RESU_INIT. Cela permet notamment de tourner une évolution ou bien d'obtenir une évolution différentielle par rapport à un mouvement ponctuel.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

8.3 Mot clé CONV_CHAR

Construction d'un résultat de type dyna_trans second membre, à affecter ensuite comme charge dans DYNA_VIBRA ou DYNA_NON_LINE, à partir de l'assemblage pour une liste d'instants d'une liste de charges définies par AFFE_CHAR_MECA_F variant à la fois dans l'espace et dans le temps comme des ondes planes ou des nappes de forces.

8.3.1 Opérande CHAR

♦ CHARGE = l char,

Donnée d'une suite de noms de concepts de type char_meca définies par AFFE_CHAR_MECA_F et correspondant à des charges variant à la fois dans l'espace et dans le temps comme des ondes planes ou des forces de type FORCE_NODALE ou FORCE_ARETE constituées de l'application de nappes. Dans la suite donnée, on peut mélanger des charges de type différent, soit des ondes planes, soit des nappes de forces. Le coefficient à appliquer à l'évolution produite via COEF_MULT dans une charge EXCIT_RESU devra valoir 1.0 dans tous les cas.

8.3.2 Opérande MATR_RIGI

♦ MATR_ RIGI = matr_ k

Entrée d'une matrice de rigidité servant d'objet de référence et à partir de la quelle on peut obtenir l e champ de matériaux ainsi que l a numérotation de référence de l'évolution produite de type dyna_trans .

8.4 Opérandes INST / LIST_INST / PRECISION / CRITERE

◊ INST = litps
Liste de valeurs réelles d'instants de calcul des évolutions produites.

◊ LIST_INST = litps
Liste de réels produite par DEFI_LIST_REEL [U4.34.01] d'instants de calcul des évolutions produites.

◊ PRECISION = / prec [R]
/ 1.0D-6 [DEFAUT]
Précision utilisée pour rechercher l'instant spécifié dans le résultat d'origine .

◊ CRITERE = / 'RELATIF',
/ 'ABSOLU,'
Critère utilisé pour rechercher l'instant spécifié dans le résultat d'origine .

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

9 Exemple d'utilisation

Construction d'un transitoire thermique à partir d'une fonction :
On a défini ci-dessous les principales commandes utilisées pour construire un concept `resultat` de type `evol_ther`.

Définition d'une liste d'instants.

```
lr8 =  DEFI_LIST_REEL  (  DEBUT = 0.E0,
                          INTERVALLE=(_F(JUSQU_A=5.e-3,NOMBRE=10 ),
                                      _F(JUSQU_A=5.e-2,NOMBRE= 9  ),
                                      _F(JUSQU_A=4.e-0,NOMBRE=79 ),
                                      _F(JUSQU_A=6.e-0,NOMBRE=20 ),)
                       )
```

Définition d'une fonction du paramètre `'INST'`.

```
fct1 = DEFI_FONCTION   (  NOM_PARA = 'INST'
                          VALE= (   0.0,      20.0,
                                    0.5,      25.0,
                                    2.0,      54.0,
                                    10.0,     134.0,)
                          PROL_DROIT  = 'LINEAIRE',
                          PROL_GAUCHE = 'LINEAIRE',
                       )
```

Construction d'un champ au nœuds de fonction, on affecte la même fonction `fct1` à l'ensemble des nœuds du maillage.

```
ch = CREA_CHAMP  (   TYPE_CHAM='NOEU_TEMP_F', OPERATION= 'AFFE',
                     MAILLAGE=ma ,
                     AFFE=_F(TOUT='OUI', NOM_CMP='TEMP', VALE_F=fct1,),
                  )
...
```

Création du concept résultat `TEMPE`, construit à partir du champ aux nœuds de fonction `ch`. On se limite au numéro d'ordre 20 correspondant à la valeur 0.1. La structure de données comportera 20 numéros d'ordre de 1 à 20.

```
TEMPE = CREA_RESU   (  OPERATION = 'AFFE',
                       TYPE_RESU = 'EVOL_THER',  NOM_CHAM = 'TEMP',
                       AFFE   = ( _F( CHAM_NO   = ch ,
                                      LIST_INST = lr8,
                                      NUME_FIN  = 20 ,  ),
                                )
                    )
...
FIN()
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)