Operator CREA_RESU

1 Goal

To create or enrich a structure of data result starting from fields with the nodes or by elements. Possible assignment of the fields for various sequence numbers.

The user must make sure of the coherence of the various fields used to build or enrich the structure of data result .

The assignment via one cham_no of function product by CREA_CHAMP [U4.72.04] is carried out by evaluating each function using the parameter representing the time provided under the keywords LIST_INST or INST.

The concept produced by this operator is, for the moment, of type evol_elas, evol_noli, evol_ther, mult_elas, fourier_elas, fourier_ther, evol_varc, evol_char, mode_meca, dyna_trans or dyna_harmo.

Moreover, three particular features are accessible in this operator:

- the creation of a concept of the type evol_char by assignment of field or an analytical formula;
- the projection of a thermal transient 1D on an axisymmetric grid 3D.
- The creation of transitory evolutions of loadings second member of the type evol_char or dyna_trans by product of matrices assembled by fields kinematics, by conversion and combination of transitory results or by assemblies of space-time loads of standard plane wave or nodal forces.

2 Syntax

```
resu [result]= CREA_RESU  (

          ◊  reuse = resu,

          ♦  OPERATION =  / 'AFFE',
                          / 'ECLA_PG', # not to use directly.
                          / 'PERM_CHAM',
                          / 'PROL_RTZ',
                          / 'PREP_VRC1',
                          / 'PREP_VRC2',
                          / 'ADZE',
```

 # Construction of Dsultat by successive assignments or evaluations
# of cham_no: (OPERATION: 'AFFE')

- the creation of a concept `result` simulating the reorganization of the fuel assemblies;

```
          ♦  TYPE_RESU                =  'MULT_ELAS',
          ♦  NOM_CHAM                 =  nomcham,     [K16]
          ♦  AFFE = _F (
                 ♦  CHAM_GD           =  chno,        [cham_no]
                 ◊  NOM_CAS           =  nomc,        [KN]
                 ◊  MODEL             =  Mo,          [model]
                 ◊  CHAM_MATER        =  chmat,       [cham_mater]
                 ◊  CARA_ELEM         =  carac,       [cara_elem]
                 ◊  LOAD           =    tank         /[char_meca]
                                                     /[char_cine_meca]
                    ),

          ♦  TYPE_RESU                =  / 'EVOL_ELAS',
          ♦  NOM_CHAM                 =  nomcham,     [K16]
          ◊  EXCIT  =_F (
                    ♦  LOAD           =     tank,           [char_meca]
                    ◊  FONC_MULT      =     fonc,           [function]
                    ◊  TYPE_CHARGE    =  / typc             [l_Kn]
                                         / 'FIXES'
                           ),
                 ♦  AFFE = _F (
                 ♦  CHAM_GD           =  chno,        [cham_no]
                 ◊  MODEL             =  Mo,          [model]
                 ◊  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,       [DEFECT]
                 ◊  CRITERION         =  / 'RELATIVE',[DEFECT]
                                         / 'ABSOLUTE',
                    ),
```

```
                ♦  TYPE_RESU                =  / 'EVOL_NOLI',
                ♦  NOM_CHAM                 =  nomcham,     [K16]
                ◊  BEHAVIOR              =_F (to see the document [U4.51.11]),
                ◊  EXCIT  =_F (
                       ♦  LOAD              =  tank,              [char_meca]
                       ◊  TYPE_CHARGE       =  / 'FIXE_CSTE'
                                               / 'FIXE_PILO'
                                               / 'FIXE_PILO'
                                               / 'SUIV'
                                               / 'DIDI'
                       ♦  /  FONC_MULT      =  fonc,              [function]
                          /  ♦  DEPL        =  depl,              [function]
                             ♦  QUICKLY        =  quickly,
   [function]
                             ♦  ACCE        =  acce,              [function]
                          ◊  MULT_APPUI     =  / 'YES',
                                               / 'NOT'           [DEFECT]
                          ◊  DIRECTION      =  (d1, d2, d3),      [l_R]
                          ◊  GROUP_NO  =  lgrno,             [l_gr_noeud]
                          ),
                       ♦  AFFE = _F (
                       ♦  CHAM_GD          =  chno,        [cham_no]
                       ◊  MODEL            =  Mo,          [model]
                       ◊  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,       [DEFECT]
                       ◊  CRITERION        =  / 'RELATIVE',[DEFECT]
                                              / 'ABSOLUTE',
                       ),

                ♦  TYPE_RESU                =  'FOURIER_ELAS',
                ♦  NOM_CHAM                 =  nomcham,     [K16]
                ♦  AFFE = _F (
                       ♦  CHAM_GD          =  chno,        [cham_no]
                       ◊  MODEL            =  Mo,          [model]
                       ◊  CHAM_MATER       =  chmat,       [cham_mater]
                       ◊  CARA_ELEM        =  carac,       [cara_elem]
                       ◊  NUME_MODE        =  num,         [I]
                       ◊  TYPE_MODE        =  / 'SYME',    [DEFECT]
                                              / 'ANTI',
                                              / 'ALL',
                       ◊  LOAD          =    tank         /[char_meca]
                                             /[char_cine_meca]
                       ),
                ♦  TYPE_RESU                =  'FOURIER_THER',
                ♦  NOM_CHAM                 =  nomcham,     [K16]
                ♦  AFFE = _F (
                       ♦  CHAM_GD          =  chno,        [cham_no]
                       ◊  MODEL            =  Mo,          [model]
                       ◊  CHAM_MATER       =  chmat,       [cham_mater]
                       ◊  CARA_ELEM        =  carac,       [cara_elem]
                       ◊  NUME_MODE        =  num,         [I]
                       ◊  TYPE_MODE        =  / 'SYME',    [DEFECT]
                                              / 'ANTI',
```

```
                                    / 'ALL',
            ),

♦  TYPE_RESU                =  'EVOL_THER',
♦  NOM_CHAM                 =  nomcham,     [K16]
 ◊ EXCIT  =_F (
          ♦  LOAD            =  tank,              [char_ther]
          ◊  FONC_MULT       =  fonc,              [function]
             ),
          ♦  AFFE = _F (
          ♦  CHAM_GD         =  chno,        [cham_no]
          ◊  MODEL           =  Mo,          [model]
          ◊  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,       [DEFECT]
          ◊  CRITERION       =  / 'RELATIVE',[DEFECT]
                                / 'ABSOLUTE',
             ),

♦  TYPE_RESU                =  'EVOL_VARC',
♦  NOM_CHAM                 =  nomcham,     [K16]
♦  AFFE = _F (
          ♦  CHAM_GD         =  chno,        [cham_no]
          ◊  MODEL           =  Mo,          [model]
          ◊  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,       [DEFECT]
          ◊  CRITERION       =  / 'RELATIVE',[DEFECT]
                                / 'ABSOLUTE',
             ),

♦  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]
          ◊  MODEL           =  Mo,          [model]
          ◊  CHAM_MATER      =  chmat,       [cham_mater]
          ◊  CARA_ELEM       =  carac,       [cara_elem]
          ◊  FREQ            =  freq,        [l_R8]
          ♦  NUME_MODE       =  numo,        [I]
          ◊  AXIS               =  axis,        [K16]

             ),

♦  TYPE_RESU                =  'DYNA_TRANS',
♦  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]
          ◊  MODEL          =  Mo,           [model]
          ◊  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,        [DEFECT]
          ◊  CRITERION      =  / 'RELATIVE', [DEFECT]
                               / 'ABSOLUTE',
          ),

♦  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]
          ◊  MODEL          =  Mo,           [model]
          ◊  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,        [DEFECT]
          ◊  CRITERION      =  / 'RELATIVE', [DEFECT]
                               / 'ABSOLUTE',
          ),
```

/ # Construction of a concept of the type EVOL_CHAR by assignment or evaluation
# of one cham_no

```
♦  TYPE_RESU                =  'EVOL_CHAR',
♦  NOM_CHAM                 =  nomcham,     [K16]
♦  AFFE = _F (
          ♦  CHAM_GD        =  chno,         [cham_no]
          ◊  MODEL          =  Mo,           [model]
          ◊  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,        [DEFECT]
          ◊  CRITERION      =  / 'RELATIVE', [DEFECT]
                               / 'ABSOLUTE',
          ),
```

/ # Construction of a result on a grid burst for visualization or
 # postprocessing. This keyword should not be called directly.

```
♦  TYPE_RESU                =  …
♦  ECLA_PG= _F   (          to see [U4.44.14]          ),
```

/ **# Construction of a result dedicated to the fuel assemblies**
 # (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,     [DEFECT]
                ◊  CRITERION                     =  / 'ABSOLUTE',
                                                 / 'RELATIVE',
                ♦  MAILLAGE_INIT              =  ma_1,         [grid]
                ♦  RESU_FINAL                 =  resu,         [evol_noli]
                ♦  MAILLAGE_FINAL             =  mo_2,         [grid]
                ♦  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,     [DEFECT]
                                ),
```

/ **# Projection of a transient 1D on an axisymmetric grid**
 # (OPERATION = 'PROL_RTZ')

```
                ♦  TYPE_RESU                  =  'EVOL_THER'
                ♦  PROL_RTZ=_F (
                   ♦  MAILLAGE_FINAL          =  ma_3D,        [grid]
                   ♦  TABLE                   =  post_1D,      [table]
                   ◊  /  INST                 =  inst,         [l_R]
                      /  LIST_INST            =  linst,        [LSTIR8]
                   ◊  PRECISION               =  / prec,
                                                 / 1.0E-6,     [DEFECT]
                   ◊  CRITERION                  =  / 'ABSOLUTE',
                                                 / 'RELATIVE', [DEFECT]
                   ◊  PROL_DROITE             =  / 'EXCLUDED', [DEFECT]
                                                 / 'LINEAR',
                                                 / 'CONSTANT',
                   ◊  PROL_GAUCHE             =  / 'EXCLUDED', [DEFECT]
                                                 / 'LINEAR',
                                                 / 'CONSTANT',

                   ♦  REFERENCE MARK                    =  'CYLINDRICAL',
                   ♦  ORIGIN                 =  (ori1, ori2, ori3), [l_R]
                   ♦  AXE_Z                     =  (axe1, axe2, axe3), [l_R]
                                ),
```

/ **# Construction of a result of the type EVOL_THER to calculate**
 # temperature in the layers of the hulls of the multi-layer type to leave
# of a field of functions of time and space (thickness)
 # (OPERATION: 'PREP_VRC1')

```
                ♦  TYPE_RESU    =  'EVOL_THER'
                ♦  PREP_VRC1 = _F  (
                   ♦  CHAM_GD              =  chno,          [cham_no]
                   ♦  MODEL                =  Mo,            [model]
                   ♦  CARA_ELEM            =  carac,         [cara_elem]
                   ♦  INST                 =  inst,          [l_R8]
                                        ),
```

```
/ # Construction of a result of the type EVOL_THER to calculate
  # temperature in the layers of the hulls multi-layer from one
  # evol_ther "hull" container TEMP_MIL/TEMP_INF/TEMP_SUP
  # (OPERATION: 'PREP_VRC2')
                    ♦  TYPE_RESU     =  'EVOL_THER'
                    ♦  PREP_VRC2 = _F  (
                       ♦  EVOL_THER          =  evol,          [evol_ther]
                       ♦  MODEL              =  Mo,            [model]
                       ♦  CARA_ELEM          =  carac,         [cara_elem]

              #  Possible selection of a subset of elements to be treated:
                    ◊  /  ALL                =     'YES',      [DEFECT]
                       /  GROUP_MA           =     lgma,       [l_gr_maille]

                                          ),
/   # Creation by assembly of structures of data result evol_ther:
    # (OPERATION: 'ADZE')

                    ♦  TYPE_RESU              =  'EVOL_THER'
                    ♦  ADZE =       _F  (
                       ♦  RESULT           =  evol,          [evol_ther]
                       ◊  TRANSLATION      =  /  tr,         [R]
                                              /  ~~~         [DEFECT]
                                          ),
      )
/ # Construction of a result of the type EVOL_CHAR or DYNA_TRANS from
  # products of assembled matrices and fields kinematics
  # (OPERATION: 'KUCV')

                 ♦  TYPE_RESU                 =  / 'EVOL_CHAR',
                                                 / 'DYNA_TRANS',
                 ♦  KUCV =_F  (
                    ♦  RESU_INIT              =  resuI,      / [dyna_trans]
                                                             / [evol_char]
                                                             / [evol_noli]
                    ♦  MATR_AMOR              =  matr_has,   [matr_asse_depl_r]
                    ◊  MATR_RIGI              =  matr_k,     [matr_asse_depl_r]
                    ♦  /  INST                =  inst,       [l_R]
                       /  LIST_INST           =  linst,      [Listr8]
                    ◊  CRITERION                 =  / 'ABSOLUTE',
                                                 / 'RELATIVE',[DEFECT]
                    ◊  PRECISION              =  / prec,
                                                 / 1.0E-6,   [DEFECT]
                              ),

/ # Construction of a result of the type DYNA_TRANS by assemblies of loads
  # (OPERATION: 'CONV_CHAR')

                 ♦  TYPE_RESU                 =  / 'DYNA_TRANS',
                 ♦  CONV_TANK =_F(
                    ♦  LOAD                   =  Ltank,      [l_char_meca]
                    ♦  MATR_RIGI              =  matr_k,     [matr_asse_depl_r]
                    ♦  /  INST                =  inst,       [l_R]
                       /  LIST_INST           =  linst,      [listr8]
                    ◊  CRITERION                 =  / 'ABSOLUTE',
                                                 / 'RELATIVE',[DEFECT]
                    ◊  PRECISION              =  / prec,
                                                 / 1.0E-6,   [DEFECT]
                              ),
```

/ **# Construction of a result of the type EVOL_CHAR or DYNA_TRANS**
 # by conversion and combination of transitory results
 # (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             =  digital,        [nume_ddl]
                   ♦  /  INST                 =  inst,        [l_R]
                      /  LIST_INST            =  linst,       [Listr8]
                   ◊  CRITERION                  =  / 'ABSOLUTE',
                                                 / 'RELATIVE',[DEFECT]
                   ◊  DDL_EXCLUS              =  / 'DX',
                                                 / 'DY',
                                                 / 'DZ',
                                                 / 'DRX',
                                                 / 'DRY MARTINI',
                                                 / 'DRZ',
                   ◊  PRECISION               =  / prec,
                                                 / 1.0E-6,    [DEFECT]
                   ◊  COEFF                   =  / coeff ,
                                                 / 1.0,       [DEFECT]
                   /  ♦  GROUP_NO_INTERF  =  gni             [grno]
                      ♦  VITE_ONDE     =     vite_onde       [R]
                      ♦  DIRECTION     =     to dir             [R]
                      ♦  COOR_REFE     =     crefe           [R]
                                          )
                                ),
```

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

3 Operands

3.1 Operand OPERATION

♦ OPERATION = defines the type of operation to carry out with this operator:

'AFFE' : creation of a structure of data result starting from fields. C' is to the user to make sure of the coherence of the fields provided to create the structure of data and to check that they are based on the same model.
'ECLA_PG' : creation of a structure of data on a grid burst for visualization,
'PERM_CHAM' : reorganization of the fuel assemblies,
'PROL_RTZ' : prolongation of a field $1D$ on an axisymmetric structure,
'PREP_VRC1' : calculation of the temperature in the layers of a hull on the basis of a temperature $TEMP = f(EPAIS, INST)$,
'PREP_VRC2' : calculation of the temperature in the layers of a hull on the basis of a temperature calculated by aster with a model of hulls (TEMP_MIL/TEMP_INF/TEMP_SUP),
'ADZE' : creation of a structure of data result starting from several structures of end to end put data result.
'KUCV' : creation transitory evolutions of loadings second member of the type evol_char or dyna_trans by product of matrices assembled by fields kinematics speeds and/or displacements.
'CONV_RESU' : creation transitory evolutions of loadings second member of the type evol_char or dyna_trans by conversion and combination of transitory results.
'CONV_CHAR' : creation transitory evolutions of loadings second member of the type dyna_trans by assemblies of space-time loads of standard plane wave or nodal forces.

The structure of data result perhaps réentrante for the operations AFFE and ECLA_PG. It is it always for PERM_CHAM
Pour OPERATION = 'AFFE', Lbe existing fields can be replaced according to the value of the variable of access INST by using the values indicated behind the keywords PRECISION and CRITERION. When there is replacement of an existing field, the code transmits a message of alarm, if not the fields are stored at the end of the structure of data.

3.2 Operand TYPE_RESU

♦ TYPE_RESU
Type of the structure of data result created.
In the case of a result of the type EVOL_VARC and of an evaluation of a field of functions (time and Espace), one vérifiE coherence enters the nature of the field of functions and the name of the field given under NOM_CHAM . If for example, the field of functions is of the type NOEU_NEUT_F the name of the field must be NEUT .

3.3 Operand NOM_CHAM

♦ NOM_CHAM
Reference symbol of the field to be affected. This name must be coherent with the structure of data modified or created. It can take for example the value
'DEPL','VARI_ELGA','TEMP','FLUX_ELNO','IRRA', etc.
In the case of a result of the type EVOL_VARC and of an evaluation of a field of functions (time and space) *Code_Aster* vérifie coherence enters the nature of the field of functions and the name of the field given under NOM_CHAM . If for example, the field of functions is of the type NOEU_NEUT_F the name of the field must be NEUT .

In the case of a result of type 'EVOL_CHAR', the fields which one can to create are:

NEAR	Fields of pressure (N/m^2) , component $PRES$
FVOL_3D	Voluminal fields of forces (N/m^3) , components FX , FY , FZ
FVOL_2D	Voluminal fields of forces (N/m^3) , components FX , FY
FSUR_3D	Surface fields of forces (N/m^2) , components FX , FY , FZ
FSUR_2D	Surface fields of forces (N/m^2) , components FX , FY
VITE_VENT	Fields with the nodes of speed of the wind (m/s) , components DX , DY , DZ
T_EXT	Map of outside temperature, component $TEMP$
COEF_H	Map of coefficient of exchange, component H
FLUN	Flux plot normal, component $FLUN$
FORC_NODA	Fields with the nodes, componentbe FX , FY , FZ

3.4 Operand BEHAVIOR

The syntax of this keyword common to several orders is described in the document [U4.51.11]. This keyword must be indicated in the case of non-linear mechanics because it is used in recovery as calculation in STAT_NON_LINE and DYNA_NON_LINE to check the compatibility of the behaviors (many internal variables in particular).

At the time of the operation AFFE in the case of non-linear mechanics,

- without reuse , one news structure of data result will be created with the well informed behavior. SI BEHAVIOR is not renseigné, it behavior taken by default is rubber band (COMPORTEMENT=' ELAS') in small deformations (RELATION=' PETIT').
- One uses reuse to add or replace fields in LE result. If BEHAVIOR is informed, one replaces his map of behavior for every moment; if not, one keep the behavior for the existing moment and one recovers that preceding moment for the non-existent moment (elastic in small deformations without preceding moment).

At the end of the L' operation AFFE, one checks, pour each moment, coherence between its map of behavior and the number of the internal variables. In spite of the inconsistencies, authorized cases allow a resumption of this field (VARI_ELGA) in a non-linear calculation, and calculation should occur well (autocorrection in the operator).

3.5 Operands EXCIT

So that a result resulting from the order CREATED_RESU that is to say exploitable by other orders, it is necessary to build and inform the structure of data by specifying the associated loads. The keyword factor EXCIT is used for TYPE_RESU : EVOL_ELAS, EVOL_NOLI and EVOL_THER. One will refer to the respective documents U4.51.01, U4.51.03 and U4.54.01.

3.6 Keyword CHAM_GD

3.6.1 Operand CHAM_GD

♦ CHAM_GD = chno

chno is:

1) that is to say one CHAM_NO of function created by the order CREA_CHAMP [U4.72.04] and in this case one evaluates for each node the function and each moment defined behind LIST_INST or INST one is created CHAM_NO realities,
2) that is to say one cham_no or is one CHAM_ELEM realities created by the order CREA_CHAMP (word ofAFFE or EXTR) and this field is duplicated as many times as the list of moments defined behind LIST_INST or INST requires.

3.6.2 Operands MODEL, CHAM_MATER, CARA_ELEM, LOAD

These operands optional are used to allow the filling of the structures of data result. This filling is essential if the order CREA_RESU is called by MACRO_ELAS_MULT to then use the orders of postprocessing which will search this information in the structure of data.

◊ MODEL = Mo,

Name of the model whose elements are the object of calculation.

◊ CHAM_MATER = chmat,

Name of the material field.

◊ CARA_ELEM = carac,

Name of the characteristics of the structural elements (beam, hull, discrete,…) if they are used in the model. When OPERATION takes the value PREP_VRC1 or PREP_VRC2, the components there are recovered THICK and COQU_NCOU.

◊ LOAD = tank,

Name of a concept of the type char_meca product by AFFE_CHAR_MECA or by AFFE_CHAR_MECA_F [U4.44.01] starting from the model Mo. One can also give the name of a “kinematic load” (standard char_cine_meca) result of the operators AFFE_CHAR_CINE or AFFE_CHAR_CINE_F [U4.44.03].

3.6.3 Operands LIST_INST / LIST_FREQ/ NUME_INIT / NUME_FIN

♦ LIST_INST = litps

List of realities produced by DEFI_LIST_REEL [U4.34.01].

♦ LIST_FREQ = lifreq

List of realities produced by DEFI_LIST_REEL [U4.34.01].

◊ NUME_INIT = nuini

◊ NUME_FIN = nufin

The moments of calculation are those defined in the concept litps taken between nuini and it nufin number of moment. In the absence of keyword NUME_FIN, it is the size of the list of realities which is taken into account.

3.6.4 Operands INST

♦ INST = linst

List of realities: list of the moments for which cham_no of function will be evaluated, or it cham_no realities will be affected.

Note:

The sequence number created in the concept result is recovered starting from the value of the variable of access INST when it is present, that is to say affected with the maximum value immediately above.

3.6.5 Operands FREQ

In the case MODE_MECA/MODE_MECA_C :

◊ FREQ = freq

Value of the frequency.

This operand is optional, that allows, in the case of a réentrant concept, to be able to declare another field for the same number of mode (NUME_MODE) without having to provide the frequency.

It should be noted that if the user declares a field (for example EFGE_ELNO) for one NUME_MODE for which another field already exists with an associated frequency (for example DEPL) and that it informs the operand FREQ with a different value, the concept will not be able by being treated by COMB_SISM_MODAL.

In the other cases:

♦ FREQ = lfreq

List of realities: list of the frequencies for which cham_no of function will be evaluated, or it cham_no realities will be affected.

Note:

The sequence number created in the concept result is recovered starting from the value of the variable of access FREQ when it is present, that is to say affected with the maximum value immediately above.

3.6.6 Operands PRECISION / CRITERION

These operands make it possible to refine the access by real variables of access of time or the frequency.

l PRECISION = / prec [R]
/ 0.0 or 1.0D-3 or 1.0D-6 [DEFECT]

This keyword makes it possible to indicate that one searches all the fields whose moment (respectively the frequency) is in the interval "inst ± prec"(confer CRITERION).

If OPERATION = 'AFFE', the value by default prec is fixed at 0.0 to avoid crushing a field to which the value of the moment is close to that one treats. the provided moment is not used to recover a field in the structure of data, it is an attribute which it should be associated with the field that one stores. In general, the fields which one stores correspond all to moments
different.
In the case very rare Où the user would wish to crush one of the fields containedS in the structure of data, it will have to use the keyword PRECISION. A message of alarm then indicates the name of the fields concerned with their moments of storage, and the precision provided by the user:

l CRITERION = / 'RELATIVE' [DEFECT]
/ 'ABSOLUTE'

'RELATIVE' : the interval of research is: [inst (1 - prec), inst (1 + prec)]
'ABSOLUTE' : the interval of research is: [inst - prec, inst + prec].

3.6.7 Operands NUME_MODE / TYPE_MODE

In the case MODE_MECA/MODE_MECA_C :

♦ NUME_MODE = num

Entirety indicating the number of the mode in the case TYPE_RESU=' MODE_MECA'.

In the case FOURIER_ELAS :

◊ NUME_MODE = num

Entirety indicating the number of the harmonic of Fourier of the field stored in a concept of the type fourier_elas.

◊ TYPE_MODE = / 'SYME'
/ 'ANTI'
/ 'ALL'

The type of the mode of stored Fourier defines.

'SYME': symmetrical harmonic
'ANTI': antisymmetric harmonic
'ALL': symmetrical and antisymmetric harmonic

3.6.8 Operand AXIS

Disponible Dyears the case MODE_MECA/MODE_MECA_C only :

◊ AXIS = / 'X'
/ 'Y'
/ 'Z'

Allows to define a direction for a given sequence number so that the concept at exit can be provided to the operand MODE_CORR of COMB_SISM_MODAL [U4.84.01].

3.6.9 Operand NOM_CAS

♦ NOM_CAS = nomc

Character string defining the variable of access of the field stored in a concept of the type mult_elas.

3.6.10 Operands MATR_RIGI/MATR_MASS

If TYPE_RESU=' MODE_MECA', 'DYNA_HARMO' or 'DYNA_TRANS' :

◊ MATR_RIGI = matr_k
Matrix of rigidity corresponding to the stored fields.

◊ MATR_MASS = matr_m
Matrix of mass corresponding to the stored fields.

4 Operands associated with the fields at the points with integration

4.1 Keyword ECLA_PG

It is strongly disadvised using the order directly CREA_RESU, the macro order should be used, MACR_ECLA_PG (See [U4.44.14]).

5 Operands associated with the fuel assemblies

5.1 Operands RESU_INIT

♦ RESU_INIT = rinit

Name of SD evol_noli containing the fields to be transferred on the new grid.

5.2 Operands INST_INIT / PRECISION/CRITERE

♦ INST_INIT = iinit

Moment characterizing in the SD evol_noli indicated under RESU_INIT, fields to be transferred on the other grid. By default, the filed last moment is selected

♦ PRECISION = prec

Precision used to search the moment specified by INST_INIT in the structure of data evol_noli associated with RESU_INIT.

♦ CRITERION = / 'RELATIVE' [DEFECT]
/ 'ABSOLUTE'

Criterion used to search the moment specified by INST_INIT in the structure of data evol_noli associated with RESU_INIT.

5.3 Operands MAILLAGE_INIT

♦ MAILLAGE_INIT = maillagei

Name of the grid on which was defined SD evol_noli indicated under RESU_INIT.

5.4 Operands RESU_FINAL

♦ RESU_FINAL = resu

Name of the structure of data evol_noli defined on the new grid on which the fields will be transferred. The structure of data resu (it must exist will have been created for example by the order STAT_NON_LINE) and must contain one sequence number.

5.5 Operands MAILLAGE_FINAL

♦ MAILLAGE_FINAL = mailfin

Name of the structure of data grid created on the new grid on which the fields will be transferred.

5.6 Keyword PERM_CHAM

5.6.1 Operands GROUP_MA_FINAL

♦ GROUP_MA_FINAL = gma_2

Name of the group of meshs of MAILLAGE_FINAL, place where the fields are transferred in RESU_FINAL.

5.6.2 Operands GROUP_MA_INIT

♦ GROUP_MA_INIT = gma_1

Name of the grid on which the structure of data was defined evol_noli indicated under RESU_INIT.

5.6.3 Operand TRAN

♦ TRAN = (tx, ty, tz)

Vector translation allowing to obtain geometrically GROUP_MA_FINAL from GROUP_MA_INIT. It is necessary to provide 3 values exactly.

5.6.4 Operand PRECISION

◊ PRECISION = prec

Absolute precision making it possible to check the good adequacy between the initial meshs and the final meshs, by default the value is fixed at 10^{-3}.

6 Operands associated with projection on an axisymmetric grid

6.1 Keyword PROL_RTZ

Construction of a thermal transient on an axisymmetric grid (3D) starting from the data of a thermal transient calculated on a grid 1D. The transient 1D is given in the form of a structure of data TABLE exit of the order POST_RELEVE_T having the following parameters:

- the definition of the moments ('INST'),
- coordinates of the nodes of the grid 1D ('COOR_X')
- the value of the temperatures to the nodes ('TEMP').

The coordinates of the table must necessarily have for origin the node of coordinate 0.
The values of the temperatures can possibly be prolonged regularly or interpolated linearly according to the coordinate 'COOR_X'.

6.1.1 Operands MAILLAGE_FINAL

♦ MAILLAGE_FINAL = mailfin

Name of the grid on which one carries out projection, the operator checks that the grid is three-dimensional .

6.1.2 Operands TABLE

♦ TABLE = table

Name of a structure of data TABLE exit of the order POST_RELEVE_T containing the thermal transient 1D. The parameters of this table are obligatorily : 'INST' , 'COOR_X' and 'TEMP'.

6.1.3 Operands INST / LIST_INST / PRECISION / CRITERION

◊ INST = litps

List of actual values.

◊ LIST_INST = litps

List of realities produced by DEFI_LIST_REEL [U4.34.01].

◊ PRECISION = / prec [R]
/ 1-0D-6 [DEFECT]

Precision used to search the moment specified in TABLE post_1D.

◊ CRITERION =/ 'RELATIVE',
/ 'ABSOLUTE,'

Criterion used to search the moment specified in TABLE post_1D.

6.1.4 Operands PROL_DROITE and PROL_GAUCHE

The projection of the transient is carried out according to the coordinate COOR_X regarded as the coordinate R in the cylindrical reference mark of the grid 3D. One can define using these two operands the way of prolonging the field beyond the terminals defined by the beach of variation of the parameter 'COOR_X' in the table.

◊ PROL_DROITE and PROL_GAUCHE =

Define the type of prolongation on the right (on the left) of the field of definition of the variable:

- 'CONSTANT' for a prolongation with the last (or first) value of the function,
- 'LINEAR' for a prolongation along the first definite segment (PROL_GAUCHE) or of the last definite segment (PROL_DROITE),
- 'EXCLUDED' if the extrapolation of the values apart from the field of definition of the parameter is prohibited (in this case if a calculation requires a value of the function out of field of definition, the code will stop in fatal error).

6.1.5 Operand REPERE/ORIGINE/AXE_Z

♦ REFERENCE MARK = 'CYLINDRICAL'

The reference mark of work to project the transient is supposed to be cylindrical, the transient 1D being regarded as the radial variation of the field of temperature. The two operands following make it possible to carry out a change of reference mark.

♦ ORIGIN = (ori1, ori2, ori3)

Corresponds to the position of the origin of the grid 1D compared to the origin of the grid 3D.

♦ AXE_Z = (axe1, axe2, axe3)

Definition of the axis of the cylindrical reference mark.

7 Operands associated with the preparation with the variables with order

7.1 Keywords PREP_VRC1 and PREP_VRC2

the thermal evolution which one can associate with the material field by AFFE_MATERIAU/AFFE_VARC must be ready to be used by the finite elements of the mechanical model. A problem arises for the elements of type hull or pipe which use a temperature varying in the thickness on the various layers. For these elements, it is necessary to prepare the calculation of the temperature on the layers upstream of the order AFFE_MATERIAU. For that, the user must use the order CREA_RESU with one of the operations PREP_VRC1 or PREP_VRC2 ("Preparation of the Variables of Order"):

- OPERATION = 'PREP_VRC1' : calculation of the temperature in the layers of a hull on the basis of a temperature TEMP= F (THICK, INST)

- OPERATION = 'PREP_VRC2' : calculation of the temperature in the layers of a hull on the basis of a temperature calculated by aster with a model of hulls (TEMP_MIL/TEMP_INF/TEMP_SUP).

7.1.1 Operand CHAM_GD

♦ CHAM_GD = chgd

chgd is a map of functions of time and thickness.

7.1.2 Operand EVOL_THER

♦ EVOL_THER = evol

evo is a structure of data EVOL_THER of standard "hull", i.e. containing the components TEMP_MIL/TEMP_INF/TEMP_SUP.

7.2 Operands ALL/GROUP_MA

Only in the case OPERATION = 'PREP_VRC2'

◊ / ALL = 'YES', [DEFECT]

This keyword makes it possible to carry out the operation on all the meshs of the grid.

/ GROUP_MA = lgma,

This keyword makes it possible to carry out the operation on a list of groups of meshs of the grid.

8 Operands associated with the assembly with structure of data of type result

8.1 Keyword ADZE

Allows to assemble several structures of data evol_ther by putting them end to end by relocating the value of the parameter time.

8.1.1 Operand RESULT

♦ RESULT = resu

resu is a structure of data evol_ther.

All the fields present in the structure of data are treated, that concerns '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'.

8.1.2 Operand TRANSLATION

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

tr is the actual value which will be added to the value of the attribute INST for each field of the structure of data resu before insertion in the structure of data result.

9 Operands associated with constitution of temporal evolutions second member

9.1 Keyword KUCV

Construction of a result of the type evol_char or dyna_tranS second member, to affect then like charges in DYNA_VIBRA or DYNA_NON_LINE, starting from products of assembled matrices of damping and of rigidity by fields kinematics the speeds and displacements extracted from an already calculated evolution. That makes it possible to proceed to the products $KU+CV$ which one usually reaches by successive calls to PROD_MATR_CHAM in a temporal loop.

9.1.1 Operand RESU_INIT

♦ RESU_INIT = resui

Name of result of calculated evolution of type dyna_tranS, evol_char or evol_noli with partyR from which one extracts fields kinematics speeds and displacements used in the products by assembled matrices.

9.1.2 Operands MATR_AMOR / MATR_RIGI

♦ MATR_ AMOR = matr_ has
◊ MATR_RIGI = matr_k,

NameS of assembled matrices of damping and of rigidity (optional) that one uses in the products $KU+CV$ constituting the produced evolution.

9.2 Keyword CONV_RESU

Construction of a result of the type evol_char or dyna_tranS (second member), to affect then like charges in DYNA_VIBRA or DYNA_NON_LINE, to leave of an already calculated evolution. That allows to if required change the type of result as well as the produced field. Thus, one can according to the type of entry required in the dynamic operators to have to change an evolution of the type evol_char and of field 'FORC_NODA' in an evolution of the type 'dyna_trans' and of field 'DEPL' or conversely. Or of to change an evolution of the type dyna_trans and of field 'FORC_NODA' in another evolution of type dyna_trans but of field 'DEPL'.

9.2.1 Operand RESU_INIT

♦ RESU_INIT = resui

Name of result of calculated evolution of type evol_char, dyna_tranS or evol_noli that one converts then into another result of type evol_char or dyna_tranS by possibly changing the field name.

9.2.2 Operand NOM_CHAM_INIT

◊ NOM_CHAM_INIT =/'DEPL',
/ 'ACCE',
/ 'FORC_NODA'
/ 'REAC_NODA',

Name of field of result DE L"evolution of departure calculated of type evol_char or dyna_tranS that one converts then in the field of the result of the produced evolution: that is to say inevitably 'FORC_NODA'for type evol_char or 'DEPL' for dyna_tranS.

9.2.3 Operands NUME_DDL / MATR_RIGI

♦ / MATR_RIGI = matr_K
/ NUME_DDL = digital

Entries from which one can obtain the classification of reference or conversion of the produced evolution . The data of the operand MATR_RIGI is advised if an evolution of the type is produced dyna_trans .

9.2.4 Operand COEFF

◊ COEFF = / coeff [R]
/ 1.0 [DEFECT]

Real coefficient of combination of produced evolution.

9.2.5 Operand DDL_EXCLUS

◊ DDL_EXCLUS = nom_cmp [TXM]

Name of component to be excluded at once from the initial result which one converts. Applies in general to a field 'FORC_NODA'. All the field is taken if the keyword is not indicated.

9.2.6 Operand GROUP_NO_INTERF

♦ GROUP_NO_INTERF = gni [grno]

With this keyword, one defines the group of nodes for which one wishes to generate a space-time seismic field.

9.2.7 Operand VITE_ONDE

♦ VITE_ONDE = vite_onde, [R]

The propagation velocity of the wave.

9.2.8 Operand DIRECTION

♦ DIRECTION = to dir, [R]

Direction of propagation the wave (directional vector: 3 values).

9.2.9 Operand COOR_REFE

♦ COOR_REFE = crefe [R]

This keyword makes it possible to inform the coordinates (3D) of a point of reference for the calculation of dephasing (time of arrival of the wave).

9.3 Keyword CONV_TANK

Construction of a result of the type dyna_tranS second member, to affect then like charges in DYNA_VIBRA or DYNA_NON_LINE, to leave assembly for a list of moments of a list of loads defined by AFFE_CHAR_MECA_F varying at the same time in space and time like plane waves or tablecloths of forces.

9.3.1 Operand TANK

♦ LOAD = L tank,

Data of a continuation of NomS of conceptS of type char_meca defined by AFFE_CHAR_MECA_F and correspondent with loads varying at the same time in space and time like plane waves or forces of the type FORCE_NODALE or FORCE_ARETE constituted by the application of tablecloths. In the taken action pursuant, one can mix loads of the type different, that is to say plane waves, maybe of

the tablecloths of forces. The coefficient to be applied to the evolution produced via COEF_MULT in a load EXCIT_RESU will have to be worth 1.0 in all the cases.

9.3.2 Operand MATR_RIGI

♦ MATR_ RIGI = matr_ K

Entry of a matrix of rigidity being used as object of reference and from which one can obtain L E field of materials like L classification of reference of the produced evolution has of type dyna_trans .

9.4 Operands INST / LIST_INST / PRECISION / CRITERION

◊ INST = litps

List of actual values moments of calculation of the produced evolutions.

◊ LIST_INST = litps

List of realities produced by DEFI_LIST_REEL [U4.34.01] moments of calculation of the produced evolutions.

◊ PRECISION = / prec [R]
/ 1.0D-6 [DEFECT]

Precision used to search the moment specified in the result of origin .

◊ CRITERION = / 'RELATIVE',
/ 'ABSOLUTE,'

Criterion used to search the moment specified in the result of origin .

10 Example of use

Construction of a thermal transient starting from a function:

One defined below the principal orders used to build a concept result of type evol_ther.

Definition of a list of moments.

```
lr8 =  DEFI_LIST_REEL  (  BEGINNING = 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),)
                       )
```

Definition of a function of the parameter '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 = 'LINEAR',
                          PROL_GAUCHE = 'LINEAR',
                       )
```

Construction of a field to the nodes of function, one affects the same function fct1 with the whole of the nodes of the grid.

```
CH = CREA_CHAMP  (    TYPE_CHAM=' NOEU_TEMP_F', OPERATION= 'AFFE',
                      MAILLAGE=ma,
                      AFFE=_F (TOUT=' OUI', NOM_CMP=' TEMP',
                      VALE_F=fonction1,),
                  )
```

…

Creation of the concept result TEMPLE, built starting from the field with the nodes of function CH. One limits to the sequence number 20 correspondent to value 0.1. The structure of data will comprise 20 sequence numbers from 1 to 20.

```
TEMPLE = CREA_RESU  (  OPERATION = 'AFFE',
                       TYPE_RESU = 'EVOL_THER',  NOM_CHAM = 'TEMP',
                       AFFE   = (_F (CHAM_NO   = CH,
                                     LIST_INST = lr8,
                                     NUME_FIN = 20,  ),
                                 )
                     )
…
END ()
```