
Procedure TEST_RESU

1 Goal

To compare a value extracted from a structure of data with a value of reference provided by the user.

This order makes it possible to test a digital value: entirety, reality or complex extracted from a concept already calculated. Today, one can test a component of one `cham_no` or of one `cham_elem`, a component of a field extracts from one `result`, a parameter of one `result`, a 'total' value extracted from a field or contents from an unspecified object from a concept user.

The procedure writes a conventional message then:

- "OK"(if it is good),
- "NOOK"(if not).

follow-up of the found value, the value of reference and percentage of error.

Systematically, a value of nonregression is checked, and when that is possible, an analytical value of reference, coming from an external source or another calculation with Code_Aster.

Orders `TEST_FONCTION` [U4.92.02] and `TEST_TABLE` [U4.92.03] allow to test the values extracted from the functions and the tables.

2 Syntax

```
TEST_RESU (
  ◆ / CHAM_NO= (_F ( ◆ CHAM_GD = chno, [cham_no]
                    / TYPE_TEST = / 'SOMM_ABS',
                              / 'SOMM',
                              / 'MAX',
                              / 'MIN',
                    ◇ NOM_CMP = ncmp, [K8]
                    / ◆ GROUP_NO = grno, [group_no]
                    ◆ NOM_CMP = nomcmp, [K8]
                    ◇ LEGEND = legend, [K16]
                    # See definition of the value of reference
                    ),)
  / CHAM_ELEM= (_F ( ◆ CHAM_GD = chel, [cham_elem]
                    / TYPE_TEST = / 'SOMM_ABS',
                              / 'SOMM',
                              / 'MAX',
                              / 'MIN',
                    ◇ NOM_CMP = ncmp, [K8]
                    / ◆ GROUP_MA = gma1, [group_ma]
                    ◆ / NOT = nupoint, [I]
                    / GROUP_NO = grno, [group_no]
                    ◇ SOUS_POINT = nuspl,
                    ◆ NOM_CMP = nomcmp, [K8]
                    ◇ LEGEND = legend, [K16]
                    # See definition of the value of reference
                    ),),
  / CARTE= (_F ( ◆ CHAM_GD = chel, [cham_elem]
                 ◆ NOM_CMP = nomcmp, [K8]
                 ◆ GROUP_MA = gma1, [group_ma]
                 ◇ LEGEND = legend, [K16]
                 # See definition of the value of reference
                 ),),
  / MAILLAGE= (_F ( ◆ GRID = my, [grid]
                   ◆ CARA =
                     / 'NB_MAILLE'
                     / 'NB_NOEUD'
                     / 'NB_GROUP_MA'
                     / 'NB_NB_GROUP_NO'
                     / 'EXI_GROUP_MA'
                     ◆ NOM_GROUP_MA = gma, [group_ma]
                     / 'EXI_GROUP_NO'
                     ◆ NOM_GROUP_NO = gno, [group_no]
                   # See definition of the value of reference
                   ),),
  / RESU = (_F ( ◆ RESULT = LMBO,
                 [resultat_sdaster]
```

```

♦ / NUME_ORDRE = nuor, [I]
  / NUME_MODE = numo, [I]
  / INST = inst, [R]
  / FREQ = freq, [R]
  / NOEUD_CMP = (node, cmp), [1_Kn]
  / NOM_CAS = nocas, [KN]
  / ANGLE =  $\alpha$ , [R]

♦ / PARA = para, [K16]
  / NOM_CHAM = nosymb, [K16]
    ◊/ NOM_CMP = ncmp, [K8]
      / NOM_VARI = nvari, [K16]

  / TYPE_TEST = / 'SOMM_ABS',
                / 'SOMM',
                / 'MAX',
                / 'MIN',

  / ♦ GROUP_NO = grno, [group_no]
    / ♦ GROUP_MA= gma1, [group_ma]
      ♦ / NOT = nupoint, [I]
        / GROUP_NO = grno, [group_no]
          ◊ SOUS_POINT = nusp,
            ♦ NOM_CMP = nomcmp, [K8]

# See definition of the value of reference
),),
  / GENE = ( _F ( ♦ RESU_GENE = LMBO ,
[VECT_ASSE_GENE] NUME_CMP_GENE = ncmp, [I]
~ RESU_GENE = LMBO ,
[MODE_GENE] ♦ / PARA = para, [K16]
  / NOM_CHAM = nosymb, [K16]
    NUME_CMP_GENE = ncmp, [I]

♦ / NUME_ORDRE = nuor, [I]
  / NUME_MODE = numo, [I]
  / FREQ = freq, [R]

RESU_GENE = LMBO , [HARM
_GENE]
NOM_CHAM = nosymb, [K16]
NUME_CMP_GENE = ncmp, [I]

♦ / NUME_ORDRE = nuor, [I]
  / FREQ = freq, [R]

RESU_GENE = LMBO ,
[TRAN_GENE]
NOM_CHAM = nosymb, [K16]
NUME_CMP_GENE = ncmp, [I]

♦ / NUME_ORDRE = nuor, [I]
  / INST = inst, [R]

# See definition of the value of reference
),),

```

```
 / OBJECT = ( _F (
                ◆ NAME = nomobj , [K24]
                # See definition of the value of reference
                ), ),

 / TEST_NAN = / 'NOT' , [DEFECT]
              / 'YES' ,

 )

# Definition of the value of reference

◆ / VALE_CALC = valley , [R]
  / VALE_CALC_C = valley , [C]
  / VALE_CALC_I = valley , [I]
  / VALE_CALC_K = valley , [K*]
# if VALE_CALC = 0.
◇ ORDRE_GRANDEUR = ordgrd , [R]

◇ LEGEND = legend , [K16]

◇ VALE_ABS = / 'NOT' , [DEFECT]
             / 'YES' ,

◇ / | TOLE_MACHINE = / 1.0D-6 , [DEFECT]
   |                / prec , [R]
   | CRITERION = / 'RELATIVE' ,
[DEFECT]           / 'ABSOLUTE' ,
# only for RESU or GENE
 / | TOLE_MACHINE = / (prec1, prec2), [l_R]
   | CRITERION = / (crit1, crit2), [l_Kn]

# except in TEST_FICHER
◇ REFERENCE = / 'ANALYTICAL',
              / 'SOURCE_EXTERNE',
              / 'AUTRE_ASTER',

If REFERENCE is indicated
◆ / VALE_REFE = valley , [R]
  / VALE_REFE_C = valley , [C]
  / VALE_REFE_I = valley , [I]
  / VALE_REFE_K = valley , [K*]
◇ PRECISION = / 1.0D-3 , [DEFECT]
              / prec , [R]
```

Notice 1

The definition of the value of reference is common to the orders TEST_RESU, TEST_TABLE, TEST_FONCTION and TEST_FICHER. All the keywords are not available in all the orders, to see the comments.

The whole type (_I) do not exist in TEST_FONCTION and TEST_FICHER. The type complexes (_C) do not exist in TEST_FICHER and for OBJECT. The type chains (_K) exist only in TEST_FICHER and TEST_TABLE.

VALE_ABS do not exist for MAP and TEST_FICHER.

Notice 2

Keywords CHAM_NO, CHAM_ELEM and MAP allow to test them cham_no, them cham_elem and them map. The keyword RESU is reserved for concepts of the type result. The keyword GRID allows to test some characteristics (whole) of a grid.

3 General information

This order makes it possible to test a scalar digital value recovered in a concept of the type `cham_no`, `cham_elem`, `map` or `result`, compared to a value of nonregression and, when it is possible, compared to an analytical value of reference, coming from an external source or another calculation with Code_Aster.

Three types of digital values can be tested:

- a component of a field (`cham_no`, `cham_elem`, `map` or field which is part of one `result`),
- a parameter contained in a concept of `result`,
- a global value of a field [§4.4].

To test a component of field, a field [§ should be chosen4.1] then to choose a component [§4.3].

To test a parameter, it is necessary to choose a sequence number [§4.1.4] and to choose the name of the parameter.

The expected digital value (real, complex or whole) is provided in accordance with [§4.7].

Notice concerning the tests of nonregression

One systematically makes a test of nonregression compared to a computed value previously. The tolerance associated with this test (`TOLE_MACHINE`) must be very weak and should not be higher than the value by default. In particular, the value should be the same one (on at least 8 decimals) on all the platforms. Moreover, this computed value should change only when algorithm is modified, corrected. Any other variation owes alerted the developer on the reliability of the programming.

Remarks concerning the tests in the structures of “generalized” data:

One can to test the generalized components (displacements, speeds or accelerations of a transient in modal space). It is advisable nevertheless to be circumspect with this kind of test. Indeed the value of a generalized component depends entirely on the standard of the mode. However that - C_i is given in an arbitrary way. Thus without preliminary standardisation of the standards, the value of a generalized size is arbitrary. Lastly, there is no possibility in Code_Aster of fixing the direction of a mode. For a multiple mode, that wants to say that, even once the normalized modes, a generalized size can take an unspecified value. In the case of a simple mode, it can be directed in a direction or the opposite direction. One then obtains a generalized value or his opposite.

4 Operands

4.1 Selection of a field

In order to test a field which can be an isolated field (`cham_no` or `cham_elem` or `map`), or a field extracted from a result, or will use the keyword factors: `CHAM_NO`, `CHAM_ELEM`, `MAP` or `RESU`.

4.1.1 “Isolated” fields

- ◆ `CHAM_GD = field`

Name of `cham_no`, `cham_elem` or `map` which one wants to extract a value.

4.1.2 Operand **RESULT**

- ◆ `RESULT = LMBO`

Name of the concept `result` treaty.

4.1.3 Operand NOM_CHAM

- ◆ / NOM_CHAM = nosymb
Reference symbol of the field to be selected.

4.1.4 Selection of a sequence number

- ◆ / NUME_ORDRE = nuor,
Sequence number of the field (or the parameter) required.

/ NUME_MODE = numo,
/ INST = inst,
/ FREQ = freq,
/ NOEUD_CMP = (node, cmp),
/ NOM_CAS = nocas,
/ ANGLE = α ,

These keywords make it possible to identify a sequence number in one `result` [U4.71.00].

They are called “variables of access”.

They all are not valid for all the types of `result`.

When the access is not made an actual value (ANGLE, FREQ, INST) the value given should not be ambiguous (cf [§4.7]).

4.2 Selection of a parameter in a result

To select a parameter in a result, it is necessary to specify the desired sequence number [§4.1.4] and to give the name of the parameter.

- ◆ PARA = para
Name of the sought parameter. This name is attached to the type of the concept `result` treaty.

4.3 Selection of a component of a field

The access to a size is done for one `cham_no` by:

- the name of the node which carries this size.

The access to a size is done for one `cham_elem` by:

- the name of the mesh which supports the element,
- something which specifies:
 - that is to say the name of a node of this mesh for `cham_elem` “with the nodes” (ELNO).
 - that is to say the number of the point of GAUSS for `cham_elem` “at the points of GAUSS” (ELGA).

The access to a size is done for one `map` by:

- the name of the mesh,

- ◆ GROUP_MA = gma1

Allows to indicate the mesh for which one wants to test it `cham_elem` or `map`.
LE group should contain only one nets.

- ◆ GROUP_NO = grno

Name of group containing it node which one wants to check a component.
LE groups must be reduced with only one node .

- / NOT = nupoint

Entirety `nupoint` specify the number of the point of GAUSS which one wants to test the value (case of `cham_elem` "at the points of GAUSS").

```
◇ SOUS_POINT = nusp
```

Entirety `nusp` specify the number of the under-point of which one wishes to obtain the value (case of `cham_elem` under-points, used by the elements of structure: beam, pipes, hulls).

In the case of the plates and of the multi-layer hulls, the number of the under-point corresponds to the level in the whole of the layers. Each layer is described by a lower, average and higher skin. By convention, for N layers, this number varies between 1 and $3N$ where the first point is at the level of the lower skin of the first layer and it $3N$ ème not on the level of the higher skin of the last layer (cf [R3.07.03] and [R3.07.04] for the numeration of the layers).

In the case of the multifibre beams, this entirety is the number of the fibre whose classification is described in documentation [U4.26.01] and [R3.08.08].

In the case of the pipes, it is necessary to refer to the description made in the document [R3.08.06].

```
/ NOM_CMP = ncmp
```

Name of the component which one wants to test [U2.01.04].

```
/ NOM_VARI = nvari
```

For the fields of the internal variables (`VARI_*`), one can give the name of the internal variable (see [U4.51.11] for the rules of naming of the internal variables) instead of the name of the component (`V1,...`).

4.4 To test a field "overall"

Once a field selected [§4.1], one can test a quantity calculated overall on all the field. For that, `NOM_CMP` does not have to be well informed to allow to take into account all the components of the field.

```
/ TYPE_TEST = 'SOMM_ABS'
```

The sum of the absolute values of the components of the field.

```
/ TYPE_TEST = 'SOMM'
```

The sum of the values of the components of the field.

```
/ TYPE_TEST = 'MAX'
```

The maximum of the values of the components of the field.

```
/ TYPE_TEST = 'MIN'
```

Minimum of the values of the components of the field.

4.5 To test the contents of an object JEVEUX

This functionality is reserved for the developers of the Code. To use it, it is necessary to know the names of the objects `JEVEUX` composing the concepts of the user. It is intended to check to it not regression of the structures of data other than them `RESULT`, `FIELDS`, `TABLE` and `FUNCTION`.

4.5.1 Operand NAME

```
NAME = nomobj
```

Name of the object `jeveux` that one wants to test.

Note:

The objective of this kind of test (NAME + VALE_CALC/_I) is to test a whole vector overall. The "amount" which is tested is unfortunately a bad "check sum" of the object: a permutation within the vector does not change this sum. A more careful test consists in printing the object in a file (IMPR_CO) then to test the contents of this file with a truth "check sum" (TEST_FICHER).

4.5.2 Operands CRITERION and PRECISION

See below [§4.7].

4.6 Keyword factor GRID.

This keyword makes it possible to validate the orders of the code which produce (or modify) grids. It makes it possible to test some characteristics (whole) grids: many meshes, of nodes, groups of meshes, groups of nodes, many meshes in a group of meshes and many nodes in a group of nodes.

4.6.1 Operand GRID

GRID = netted

Name DU grid that one wants to test.

4.6.2 Operand CARA

Allows to choose the characteristic of the grid to be tested: 'NB_MAILLE', 'NB_NOEUD', 'NB_GROUP_MA', 'NB_GROUP_NO', 'EXI_GROUP_MA' or 'EXI_GROUP_NO'.

4.6.3 Operands NOM_GROUP_MA and NOM_GROUP_NO

These operands are to be used for CARA=' EXI_GROUP_MA' (resp. CARA=' EXI_GROUP_NO').

It make it possible to indicate the group which one wants to test.

In this case, the value tested (VALE_CALC_I or VALE_REFE_I) must contain the number of meshes (or nodes) of the group.

4.7 Definition of the value of nonregression and reference

One systematically makes a test of nonregression compared to a value previously calculated, with a very weak tolerance: keywords VALE_CALC and TOLE_MACHINE.

As often as possible, one adds a test compared to a value of reference compared to an analytical solution, a value obtained of an external source or another modeling: keywords REFERENCE, VALE_REFE, PRECISION.

♦ / VALE_CALC = valley

Actual value of nonregression. It is the computed value by Code_Aster.

When this value is worthless (lower than 1.e-16 in absolute value), the keyword should be informed ORDRE_GRANDEUR = ordgrd. It has direction only in absolute.

It is checked whereas: $|val| \leq tole * ordgrd$

If ORDRE_GRANDEUR is not provided whereas VALE_CALC is null, the test of nonregression is ignored (appears with SKIP in the file result). Attention in this case, a test with VALE_REFE is obligatory.

/ VALE_CALC_C = valley

Value complexes of nonregression.

/ VALE_CALC_I = valley

Whole value of nonregression.

/ VALE_CALC_K = valley

Character string of nonregression. Only by TEST_FICHER and TEST_TABLE.

◇ TOLE_MACHINE = sheet

Precision requested (by default 1.D-6) to accept the computed value compared to the value of nonregression (VALE_CALC). See ORDRE_Grandeur above.

◇ VALE_ABS

= 'NOT' the value of reference and the computed value by Aster are compared such as they are.

= 'YES' the value of reference and the computed value by Aster are compared in absolute values.

◇ CRITERION =

Type of test to be carried out. Applies to the test of nonregression and the test compared to a value of reference if necessary.

If v is the extracted value, the test will carry for:

- 'RELATIVE' on: $|val - v| \leq prec \cdot |val|$
- 'ABSOLUTE' on: $|val - v| \leq prec$

◇ REFERENCE =

/ 'ANALYTICAL' : the provided value of reference is "analytical"

/ 'SOURCE_EXTERNE' : the provided value of reference comes from a program other than Code_Aster, of a bibliographical reference, a measurement, etc.

/ 'AUTRE_ASTER' : the provided value of reference is that obtained by another calculation with Code_Aster (another order, another modeling, option of calculation,...)

It is the presence of the keyword REFERENCE who indicates that one has an external reference and conditions the presence of the keywords VALE_REFE [_I/_C] and PRECISION.

◆ VALE_REFE, VALE_REFE_C, VALE_REFE_I, VALE_REFE_K

Similar to the keywords VALE_CALC above. They define the value obtained by the external source.

◇ PRECISION =

Precision requested (by default 1.D-3) to accept the computed value compared to the value of reference (VALE_REFE).

Note:

When the definition of the sequence number of one RESULT is done by a real variable of access (FREQ, INST, ANGLE), it is not necessary that there is ambiguity on this sequence number. For that the user defines a small interval around the value requested thanks to the keywords CRITERION and TOLE_MACHINE.

In this case ("real" access) keywords CRITERION and TOLE_MACHINE will thus wait 2 values each one: (crit1, crit2) and (prec1, prec2).

crit1 and prec1 relate to the value of nonregression.

crit2 and prec2 allow to choose the interval of research of the sequence number.

The values by default of crit1 and prec1 are 'RELATIVE' and 1.D-6.

The values by default of crit2 and prec2 are 'RELATIVE' and 1.D-3.

One cannot define explicitly *crit2* and *prec2* without defining *crit1* and *prec1*.

Note:

Tests of not-regression (keywords *VALE_CALC**) are systematically ignored in the tests of validation but they must nevertheless be indicated (with an unspecified value) because required by the catalogue of the order. LES tests of validation are identified like such by the presence of *testlist validation* in their file *.export*.
I L has there however an exception for the tests on the character strings (*VALE_CALC_K*) who are not ignored because there are not a tolerance/precision on these tests.

4.8 Addition of one specificity to the value tested

◇ LEGEND =

Character string of to more the 16 characters describing the test carried out.
The user thus has the possibility of commenting on his test.

4.9 Keyword TEST_NAN

```
| TEST_NAN = / 'NOT' , [DEFECT]  
/ 'YES' ,
```

This keyword is used to validate the operation of *NaN* (Not-a-Number) of Code_Aster. This keyword is to be used only in optics to carry out tests. Its use must cause a fatal error in *FPE* (Floating Not Exception) in mode *debug* in TEST_RESU.

5 Examples

```
TEST_RESU ( CHAM_NO = _F ( CHAM_GD = F,  
GROUP_NO = 'GN2',  
NOM_CMP = 'DX',  
REFERENCE = 'ANALYTICAL',  
VALE_CALC = 1.1999996845E-5,  
VALE_REFE = 1.2E-5,  
PRECISION = 1D-4 ,))
```

```
TEST_RESU ( MAP = _F ( CHAM_GD = CART1,  
GROUP_MA = 'GM3',  
NOM_CMP = 'X4',  
REFERENCE = 'AUTRE_ASTER',  
VALE_CALC_I = 3,  
VALE_REFE_I = 3,))
```

```
TEST_RESU ( CHAM_ELEM= (_F ( CHAM_GD = SIGGA,  
GROUP_MA = ' G M3',  
NOT = 3,  
NOM_CMP = 'SIXX',  
VALE_CALC = 3.4E6, ),  
  
_F ( CHAM_GD = SIGNO,  
GROUP_MA = 'GM5',  
GROUP_NO = 'GNR1',  
NOM_CMP = 'SIXX',  
VALE_CALC = 3.4E6 , ,))
```

```
TEST_RESU ( RESU = (_F ( RESULT = evolth,  
NOM_CHAM = 'FLUX_ELGA',  
TYPE_TEST = 'MAX',  
REFERENCE = 'ANALYTICAL',  
VALE_CALC = 154.35000201404,  
VALE_REFE = 154.35),))
```

```
TEST_RESU ( OBJECT = _F ( NAME = 'CH1 .CHME.LIGREL.LIEL',  
VALE_CALC_I = 1278484,))
```

```
TEST_RESU ( GRID = (  
_F ( MAILLAGE= my, CARA= 'NB_MAILLE'  
VALE_CALC_I = 1000, ),  
_F ( MAILLAGE= my, CARA= 'NB_GROUP_NO'  
VALE_CALC_I = 12, ),  
_F ( MAILLAGE= my, CARA= 'EXI_GROUP_MA'  
NOM_GROUP_MA=' GAUCHE',  
VALE_CALC_I = 25, ),  
))
```