Operator DEFI_BASE_REDUCED

The goal of the operator is to build the reduced base starting from a non-linear calculation (thermal or mechanical) or of a parametric linear calculation.
There exist two methods:

- For the non-linear problems, the operator rests on a sd result of the type evol_ther or evol_noli and a singular decomposition with the values (SVD) on the transient or a strategy of the incremental type POD realizes;
- For the parametric linear problems, one uses an algorithm of the glouton type. In this case O N defines the linear system to solve.

Two types of bases can be produced:

- bases known as “primal”: they are pressed on the fields of displacements for the mechanics and on the fields of temperatures for thermics;
- bases known as “dual”: they are pressed on the stress fields for mechanics and on the fields of flow for thermics.

The operator produces a concept of the type mode_empi .

Contents

1 Syntax

```
base = DEFI_BASE_REDUCED (

             ◊  BASE      =   base ,                          [base_empi]

 # Standard of operation

             ◊  OPERATION =   |' POD ' ,                       [ DEFECT ]
                              |' POD_INCR',
                              |' GLOUTON',

# If OPERATION=' POD_INCR'
             ◊  SHEET     =   /tole_incr,                     [R]
                              /1.0E-10,                       [DEFECT]

# If OPERATION=' POD_INCR' or 'POD'

# options for the results produced by STAT_NON_LINE

             ♦  RESULT =   resu,                          [evol_noli]

             ♦  NOM_CHAM  =   |'DEPL'
                              |' SIEF_NOEU'

# options for the results produced by THER_NON_LINE

             ♦  RESULT =   resu,                          [evol_ther]

             ♦  NOM_CHAM  =   |'TEMP'
                              |' FLUX_NOEU'

             ◊  TYPE_BASE =   |' 3D',                          [DEFECT]
                              |' LINEIQUE',

# options for TYPE_BASE = 'LINEAR'

             ♦  AXIS          =   |' OX',
                              |' OY',
                              |' OZ'

             ♦  SECTION   =   l_grno,                          [l_gr_noeud]

# options of selection amongst modes

             ◊  TOLE_SVD  =   /tole,                           [R]
                              /1.0E-6,                        [DEFECT]
             ◊  NB_MODE   =   nbmode,                          [I]
```

```
      # If OPERATION='GLOUTON'

                  ♦  NB_VARI_COEF    =   nbvaricoef,          [ I ]
                  ◊   TYPE_VARI_COEF  =   |' DIRECT ',         [DEFECT]

                  ◊   VARI_PARA          =   _F (
                     ♦   NOM_PARA     =   will nompara,              [ para_fonc ]
                     ♦   VALE_PARA       =   will valepara,             [ R ]
                     ♦    VALE_INIT      =    valeinit                  [ R ]
                                         ),

                  ◊   MATR_ASSE          =   _F (
                     ♦   MATRIX       =   matrix ,             / [ matr_ adze
      _DEPL_R]
                                                              /[matr_adze_DEPL_C]
                      ◊   COEF_R          =   coefr ,                 [ R ]
                      ◊   COEF_C          =    coefc ,                [ C ]
                     ◊    FONC_R          =   foncr ,                   [fonc  tion
/formule]
                      ◊   FONC_C          =    foncc ,                 [fonc tion
      /formule]
                                         ),

                  ◊   VECTOR          =   vector          [cham_no]
                  ◊   COEF_R             =   coefr ,                 [ R ]
                  ◊   COEF_C             =   coefc ,                 [ C ]
                 ◊   SOLVEUR          =   _F (see the document [U4.50.01]),

      # For all the operations
                 ◊   TITLE     =   title,                    [l_Kn]

                 ◊   INFORMATION      =   /1,                       [DEFECT]
                                      /2,
      )
```

2 Operands

The operator leaves a structure of data of type mode_empi (modes empirical).

2.1 Operand BASE

◊ BASE = base, [base_empi]

It is possible to enrich a base by empirical modes by the following operations. In this case, it here is provided.

2.2 Operand OPERATION

◊ OPERATION = |' POD ', [DEFECT]
|' POD_INCR',
|' GLOUTON',

This keyword makes it possible to choose the manner of calculating the empirical modes:

- By a POD (OPERATION=' POD'): one carries out a SVD on the matrix of the snapshots containing the results on a transient. This operation is exact (within the meaning of the extraction of the empirical modes) but perhaps potentially expensive (in CPU and memory);
- By an incremental POD (OPERATION=' POD_INCR'): one builds the empirical base in an incremental way (see [R5.01.50]). This method is less precise than the classical POD but much less expensive. Moreover it makes it possible to enrich an empirical base existing with new results;
- By a method glouton (OPERATION='GLOUTON', to see [R5.01.50]) on parametric problems.

2.3 Operands for the strategies POD

2.3.1 Operand RESULT

♦ RESULT = resu,

Name of the structure of data result to analyze to generate the reduced base. Two types of possible resultS : evol_noli or evol_ther.

Limitations of use:

- Function only in 3D (thermal and mechanical);
- Cannot use limiting conditions of Dirichlet by dualisation (AFFE_CHAR_MECA or AFFE_CHAR_THER). It is necessary to use limiting conditions of Dirichlet per elimination (AFFE_CHAR_CINE).

2.3.2 Operand NOM_CHAM

♦ NOM_CHAM =|'DEPL'
|' SIEF_NOEU'
|'TEMP'
|' FLUX_NOEU'

One specifies the type of basic field reduced:

- 'DEPL' or 'SIEF_NOEU' if the structure of data is of type evol_noli ;
- 'TEMP' or 'FLUX_NOEU' if the structure of data is of type evol_ther.

2.3.3 Operand TYPE_BASE

◊ TYPE_BASE =| '3D',
| 'LINEAR',

One specifies the basic type reduced.

The linear case is specific to the digital simulation of welding. This case requires to specify the axis of welding and information concerning the section of the welded zone, perpendicular to the axis of welding.

2.3.4 OperandS AXIS and SECTION

OperandS usedS in the linear case (specific to the digital simulation of welding).

♦ AXIS =|' OX',
|' OY',
|' OZ',

Operand allowing to specify the axis of welding.

♦ SECTION = l_grno,

Operand allowing to specify the group of nodes contained in the first section of the grid of the welded zone.

Note:
These operands make it possible to define a local classification used for the calculation of the reduced modes.

2.3.5 OperandS NB_MODE / TOLE_SVD

◊ TOLE_SVD = sheet

Désigne the tolerance retained for the SVD. The value by default is of 1.0E-6.

The selected modes are those whose singular value is higher than sheet X sing_maxi where sing_maxi is the maximum singular value given by the SVD.

Notice : the incremental POD (OPERATION=' POD_INCR') fact also a SVD but not on the matrix of the stereotypes. This parameter is thus also useful in this case.

◊ NB_MODE = nbmode

NRshade of modes retained for the construction of the reduced base.

The user must inform only one of the two operands to fix the number of modes retained for the construction of BESA reduced.

2.3.6 Operand SHEET

◊ SHEET = /tole_incr, [R]
/1.0E-10, [DEFECT]

This parameter makes it possible to regulate the precision of the incremental POD.

2.4 Operands for strategies GLOUTON

Method OPERATION=' GLOUTON' allows to build an empirical base on a parameterized problem. The basic principle is to build the linear system which solves the problem concerned and to give the list of the parameters which vary. The empirical base is then built by an algorithm of the type Glouton (greedy algorithm) to which it is necessary to provide the variation of the parameters.
The linear system will be written as follows:

$$\sum_{i=1}^{n_m} \alpha_i(\gamma_{j=1,n_p}) \boldsymbol{M}_i = \beta \boldsymbol{V} \quad (1)$$

It contains to the maximum $n_m = 8$ assembled matrices $\boldsymbol{M}_i$ (real or complex). These matrices can be built in a classical way in code_aster (CALC_MATR_ELEM and ASSE_MATRICE for example). In front of each matrix, there is a coefficient $\alpha_{i=1,n_m}$, which is either constant (reality or complex), or a function or a formula (real or complex) which depends to the maximum of $\gamma_{j=1,n_p=5}$ parameters.
Lastly, the second member $\boldsymbol{V}$ is a simple nodal field with a constant coefficient β reality or imaginary.
To build the empirical base, it is necessary to traverse the esapce of the parameters what the user defined in the §2.4.2.

2.4.1 Operands MATR_ASSE and VECTOR

These operands will build the linear system to solve.

```
◊   MATR_ASSE          =  _F (
   ♦   MATRIX       =   matrix ,            / [ matr_ adze _DEPL_R]
                                           /[matr_adze_DEPL_C]
    ◊   COEF_R          =   coefr ,              [ R ]
    ◊   COEF_C          =    coefc ,             [ C ]
   ◊    FONC_R          =   foncr ,               [fonc tion /formule]
    ◊   FONC_C          =    foncc ,              [fonc tion /formule]
                        ),
```

O N gives the list of the assembled matrices $\boldsymbol{M}_i$ by the keyword factor MATR_ASSE . The name of the matrix (coming for example from the order ASSE_MATRICE) is given in MATRIX .
One chooses then the coefficient in front of each matrix. This coefficient is constant (reality by COEF_R or complex by COEF_C), that is to say a function (real by FONC_R or complex by FONC_C). In the case of a function, it must depend on the parameters whose list (variation) is given by the keyword factor VARI_PARA (see § 2.4.2).

```
◊   VECTOR          =   vector          [cham_no]
◊    COEF_R             =   coefr ,              [ R ]
◊    COEF_C             =   coefc ,              [ C ]
```

C be keywords define the second member $\boldsymbol{V}$ by the keyword VECTOR and the coefficient β reality (COEF_R) or complex (COEF_C). The second member is inevitably constant.

2.4.2 Operands VARI_PARA , NB_VARI_COEF and TYPE_VARI_COEF

These operands define parametric space traversed to build the empirical base.

```
♦   NB_VARI_COEF    =   nbvaricoef,         [ I ]
 ◊   TYPE_VARI_COEF  =   |' DIRECT ',        [DEFECT]
```

NB_VARI_COEF give the number of parameters when parametric space is traversed. The keyword TYPE_VARI_COEF allows to say that one will give *explicitly* the list of the values of the parameters (only mode available for the moment).

```
 ◊  VARI_PARA        =   _F (
    ♦   NOM_PARA     =   will nompara,            [ para_fonc ]
    ♦   VALE_PARA       =   will valepara,           [ R ]
    ♦    VALE_INIT      =    valeinit             [ R ]
                        ),
```

L E keyword factor VARI_PARA give the list of the parameters $\gamma_{j=1,n_p=5}$ and their values. For each occurrence, one gives the name of the parameter in NOM_PARA . This parameter is inevitably used in the coefficients $\alpha_{i=1,n_m}$ in front of the matrices. For each parameter, one gives the list of his values by VALE_PARA . The length of the vector VALE_PARA is inevitably equal to NB_VARI_COEF . Also should be given an initial value VALE_INIT (which initializes the algorithm glouton).

2.4.3 Operand SOLVEUR

This keyword gives the parameters of the solvor used (see [U4.50.01]). Indeed the algorithm glouton will solve a large number of times the definite linear system.

3 Examples of use

Example of use of the incremental mode by enrichment (OPERATION=' POD_INCR'):

```
mat1  = AFFE_MATERIAU (AFFE=_F (TOUT=' YES', MATER=acier1))
resu1 = STAT_NON_LINE (CHAM_MATER = mat1,…)
mode1 = DEFI_BASE_REDUITE (RESULTAT=resu1,
                           OPERATION=' POD',
                           NOM_CHAM=' DEPL')
mat2  = AFFE_MATERIAU (AFFE=_F (TOUT=' YES', MATER=acier2))
resu2 = STAT_NON_LINE (CHAM_MATER = mat2,…)
mode1 = DEFI_BASE_REDUITE (reuse=mode1,
                           OPERATION=' POD_INCR',
                           RESULTAT=resu2,
                           NOM_CHAM=' DEPL')
```

The empirical base mode1 was built on two sets of parameters materials.

Note:

- In the example above, the first DEFI_BASE_REDUITE could have been used in incremental mode (OPERATION=' POD_INCR');
- With a very weak tolerance (keyword SHEET), the incremental mode tends towards a classical SVD.