Operator **DEFI_BASE_REDUCTED**

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, one defines the linear system to solve.

The operator also allows to truncate a base (primal or dual) on a reduced field (for example produced by the order **DEFI_DOMAINE_REDUIT**).

Two types of bases can be produced:

- bases known as “primal”: they are pressed on the fields of displacements for 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`.

**Warning:** The translation process used on this website is a "Machine Translation". It may be imprecise and inaccurate in whole or in part and is provided as a convenience.

Copyright 2020 EDF R&D - Licensed under the terms of the GNU FDL (http://www.gnu.org/copyleft/fdl.html)
Contents

1 Syntax .................................................................................................................................................. 3

2 Operands............................................................................................................................................. 5

2.1 Operand BASE .............................................................................................................................. 5

2.2 Operand OPERATION ....................................................................................................................... 5

2.3 Operands for the strategies POD .................................................................................................... 5

2.3.1 Operand RESULT .......................................................................................................................... 5

2.3.2 Operand MODEL ........................................................................................................................ 5

2.3.3 Operand TABL_COOR_REDUIT ................................................................................................. 6

2.3.4 Operand NOM_CHAM .................................................................................................................. 6

2.3.5 Operand TYPE_BASE ................................................................................................................ 6

2.3.6 Operands AXIS and SECTION .................................................................................................. 6

2.3.7 Operands NB_MODE / TOLE_SVD ............................................................................................. 7

2.3.8 Operand SHEET ........................................................................................................................ 7

2.4 Operands for strategies GLOUTON .............................................................................................. 7

2.4.1 Operands MATR_ASSE and VECT_ASSE .................................................................................. 7

2.4.2 Operands VARI_PARA , NB_VARI_COEF and TYPE_VARI_COEF ............................................ 8

2.4.3 Operand SOLVEUR .................................................................................................................... 9

2.4.4 Operand ORTHO_BASE ............................................................................................................. 9

2.4.5 Operand TYPE_BASE ................................................................................................................ 9

2.5 Other operands.................................................................................................................................. 9

3 Examples of use ................................................................................................................................... 10
1 Syntax

```plaintext
base = DEFI_BASE_REDUCE (  
  ◊ BASE = base, [base_empi]  
  
  # Standard of operation  
  ◊ OPERATION = ['POD ', [DEFECT]  
    | 'POD_INCR',  
    | 'GLOUTON',  
    | 'TRONCATURE',  
  
  # If OPERATION=' POD_INCR'  
  ◊ SHEET = /tole_incr, [R]  
    /1.0E-10, [DEFECT]  
  ◊ TABL_COOR_REDUIT = tabl_coor, [table]  
  
  # If OPERATION=' POD_INCR' or 'POD'  
  ◊ MODEL = model, [model]  
  
  # 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',  
    | 'LINEIQUE', [DEFECT]  
  
  # 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]  
```

Warning: The translation process used on this website is a "Machine Translation". It may be imprecise and inaccurate in whole or in part and is provided as a convenience.
# If OPERATION='GLOUTON'

◊ ORTHO_BASE = /'NOT', [DEFECT]
◊ TYPE_BASE = /'STANDARD', [DEFECT]
◊ TYPE_VARI_COEF = |'DIRECT', [DEFECT]
◊ NB_VARI_COEF = nbvaricof, [I]
◊ 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]  
   ◇ COEF_R = coefr, [R]  
   ◇ COEF_C = coeffc, [C]  
   ◇ FONC_R = foncr, [fonction/formule]  
   ◇ FONC_C = foncc, [fonction/formule]  
   ),
◊ VECT_ASSE = _F (  
   ◇ VECTOR = vector, [cham_no_aster]  
   ◇ COEF_R = coefr, [R]  
   ◇ COEF_C = coeffc, [C]  
   ◇ FONC_R = foncr, [fonction/formule]  
   ◇ FONC_C = foncc, [fonction/formule]  
   ),
◊ SOLVEUR = _F (see the document [U4.50.01]),

# If OPERATION='TRUNCATION'

◊ MODELE_REDUIT = model, [model]

# 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,

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',

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;

It is possible also of to truncate a base (primal or dual) on a reduced field (for example produced by the order DEFI_DOMAINE_REDUIT) with OPERATION='TRONCATURE'.

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 MODEL

◊ MODEL = model,

When the result given by the keyword RESULT does not contain the relative information with the model (this perhaps case when one handles it before with the orders CREA_RESU or LIRE_RESU), this keyword makes it possible to define explicitly the model.
2.3.3 Operand TABL_COOR_REDUIT

◊ TABL_COOR_REDUIT = tabl_coor ,

Lorsqu' one calculates an empirical base by the method of the incremental POD to enrich an already existing base, it is necessary to have the reduced coordinates of preceding calculation. These coordinates are stored in a structure of data table of name ‘COOR_REDUIT’ who is attached to the empirical base. One can recover it via the operator RECU_TABLE. For example:

coorredp=RECU_TABLE (CO=base, NOM_TABLE=' COOR_REDUIT')

But if you recover the empirical base previously calculated by an operator like LIRE_RESU (in particular with format MED), this table is not available. The operator TABL_COOR_REDUIT thus allows to give it to DEFI_BASE_REDUITE.

It is thus necessary to have saved this table upstream at the same time as the empirical base (by one IMPR_TABLE), then to recover it (by one LIRE_TABLE) to give it to DEFI_BASE_REDUITE. A typical succession of orders is thus the following one:

1/ Creation of a first reduced base
   • non-linear calculation
   • creation of the base with DEFI_BASE_REDUITE
   • recovery of the table of the coordinates reduced with RECU_TABLE
   • safeguard of the empirical base by IMPR_RESU
   • safeguard of the table of the coordinates reduced by IMPR_TABLE

2/ Enrichment bases to reduce
   • non-linear calculation
   • reading of the preceding empirical base by LIRE_RESU
   • reading of the table of the reduced coordinates preceding by LIRE_TABLE
   • enrichment of the base reduced by DEFI_BASE_REDUITE by giving the preceding base (keyword BASE) and the table of the reduced coordinates preceding (keyword TABL_COOR_REDUIT)

2.3.4 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.

2.3.5 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.

2.3.6 OperandS AXIS and SECTION

Warning: The translation process used on this website is a "Machine Translation". It may be imprecise and inaccurate in whole or in part and is provided as a convenience.

Copyright 2020 EDF R&D - Licensed under the terms of the GNU FDL (http://www.gnu.org/copyleft/fdl.html)
OperandS usedS in the linear case (specific to the digital simulation of welding).

♦ \texttt{AXIS} = |' OX',
   | ' OY',
   | ' OZ',

Operand allowing to specify the axis of welding.

♦ \texttt{SECTION} = l_grno,

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

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

\section*{2.3.7 OperandS \texttt{NB\_MODE} / \texttt{TOLE\_SVD}}

♦ \texttt{TOLE\_SVD} = sheet

Designates the tolerance retained for the SVD. The value by default is of \texttt{1.0E-6}.

The selected modes are those whose singular value is higher than \texttt{sheet} \times \texttt{sing\_maxi} where \texttt{sing\_maxi} is the maximum singular value given by the SVD.

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

♦ \texttt{NB\_MODE} = nbmode

NR shade 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.

\section*{2.3.8 Operand \texttt{SHEET}}

♦ \texttt{SHEET} = /tole\_incr, [R]
   /1.0E-10, [DEFECT]

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

\section*{2.4 Operands for strategies GLOUTON}

Method \texttt{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:

\begin{equation}
\sum_{i=1}^{n_m} \alpha_{i} \, Y_{j=1,n_j} \, M_{i} = \sum_{i=1}^{n_i} \beta_{i} \, Y_{j=1,n_j} \, V_{i}
\end{equation}

It contains \( n_m \) assembled matrices \( M_{i} \) (real or complex). These matrices can be built in a classical way in \texttt{code\_aster} (\texttt{CALC\_MATR\_ELEM} and \texttt{ASSE\_MATRICE} by example). In front of each matrix, there is a coefficient \( \alpha_{i=1,n_m} \), which is either constant (real or complex), or a function or a formula (real or complex) which depends to the maximum of \( Y_{j=1,n_j} \) parameters.
Lastly, the second member $V$ is also a linear combination of $n_v$ vectors (real or complex) with a coefficient $\beta_{1,1,n}$, which is either constant (reality or complex), or a function or a formula (real or complex) which depends to the maximum of $\gamma_{j=1,n}$ parameters. To build the empirical base, it is necessary to traverse space parameters what the user defined in the §2.4.2.

### 2.4.1 Operands MATR_ASSE and VECT_ASSE

These operands will build the linear system to solve.

- **MATR_ASSE**
  - $\text{MATRIX} = \_F ( \text{matrix}, /\text{matr_adze_DEPL_R} /\text{matr_adze_DEPL_C} )$
  - $\text{COEF_R} = \text{coefr}$
  - $\text{COEF_C} = \text{coefc}$
  - $\text{FONC_R} = \text{foncr}$
  - $\text{FONC_C} = \text{foncc}$

ON gives the list of the assembled matrices $M_j$ 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).

- **VECT_ASSE**
  - $\text{VECTOR} = \_F ( \text{vector}, /\text{cham_no_aster} )$
  - $\text{COEF_R} = \text{coefr}$
  - $\text{COEF_C} = \text{coefc}$
  - $\text{FONC_R} = \text{foncr}$
  - $\text{FONC_C} = \text{foncc}$

These keywords define the second member $V$ by the keyword VECT_ASSE. One chooses then the coefficient in front of each vector. 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).

### 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 ( \text{NOM_PARA} = will nompara, [para_fonc] \text{VALE_PARA} = will valepara, [R] \text{VALE_INIT} = valeinit [R] ),
LE keyword factor \textit{VARI\_PARA} give the list of the parameters $Y_{j=1,n}$ and their values. For each occurrence, one gives the name of the parameter in \textit{NOM\_PARA}. This parameter is inevitably used in the coefficients $Q_{i=1,n}$ in front of the matrices. For each parameter, one gives the list of his values by \textit{VALE\_PARA}. The length of the vector \textit{VALE\_PARA} is inevitably equal to \textit{NB\_VARI\_COEF}. Also should be given an initial value \textit{VALE\_INIT} (which initializes the algorithm glouton).

2.4.3 Operand \textbf{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.

2.4.4 Operand \textbf{ORTHO\_BASE}

\begin{verbatim}
◊ ORTHO\_BASE = '/NOT', [DEFECT]
   '/YES'
\end{verbatim}

This keyword allows to make a orthonormalisation of the base empirical when \textit{ORTHO\_BASE='OUI'}. This orthonormalisation is made with an algorithm of iterative the Graam-Schmidt type (IGS) according to the version of Kahan-Parlett.

2.4.5 Operand \textbf{TYPE\_BASE}

\begin{verbatim}
◊ TYPE\_BASE = '/STANDARD', [DEFECT]
   '/IFS\_STAB'
\end{verbatim}

When \textit{TYPE\_BASE='IFS\_STAB'}, one creates an empirical base which will be stable when it is used for models of interaction fluid-structure (model $\{u, p, \phi\}$ to see [R4.02.02]) in transitory dynamics. The strategy consists in building an empirical base by diagonalisation of three bases on each component (see [R5.01.50]). When this keyword is activated, a checking of the components present is made on the components of the result as starter.

2.5 Other operands

\begin{verbatim}
◊ MODELE\_REDUIT = model, [model]
\end{verbatim}

Method \textit{OPERATION='TRUNCATION'} allows to truncate an empirical base on a model more restricted than that which was used to build the base. This operation is essential when one shows very-reduction of model (keyword \textit{DOMAINE\_REDUIT} in \textit{STAT\_NON\_LINE} and \textit{THER\_NON\_LINE}).

The keyword \textit{BASE} as starter gives the empirical base to truncate. The keyword \textit{MODELE\_REDUIT} give the model on which the base will be truncated. It is indeed of the model and not the grid.
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,...)
model = 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,...)
model = DEFI_BASE_REDUITE (reuse=model,
                          OPERATION=' POD_INCR',
                          RESULTAT=resu2,
                          NOM_CHAM=' DEPL')

The empirical base model 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.