To introduce new features with `CALC_EUROPLEXUS`

Summary:

This document describes the stages to be followed to add new features to the macro-order `CALC_EUROPLEXUS`. It can be a question:

- elements,
- laws of behavior,
- loadings.

Contents

1 Introduction 3
2 Structure 4
2.1 Stages 4
2.1.1 Reading of the data and translation 4
2.1.2 Recovery and working of the results 4
2.2 Catalogue translation 4
3 Addition of a new law of behavior 5
3.1 Catalogue of order 5
3.2 Dictionary of the behaviors 5
3.2.1 Stages 5
3.2.2 Details on the internal transformations of variables. 5
3.2.2.1 Passage EPX 5thrs Code_Aster 5
3.2.2.2 Notwise Code_Aster 5thrs EPX 6
3.3 Dictionary of the laws or materials 6
3.3.1 Stages 6
3.4 Synthesis 7
4 Addition of a new type of element 8
4.1 Dictionary of modelings 8
4.2 Addition again CARA_ELEM 9
4.2.1 Dictionary of the characteristic elementary 9
4.2.2 Stages 9
4.3 Passage de forced with effort 10
5 Addition of a new loading 11
5.1 Addition of a connection 11
5.1.1 Dictionary of the connections 11
5.1.2 Stages 11
5.2 Addition of a loading 12
5.2.1 Dictionary loadings 12
5.2.2 Stages 12
5.3 Treatment of the multiple choices 12

1 Introduction

The macro-order CALC_EUROPLEXUS is to launch a calculation with software EUROPLEXUS (EPX in the continuation of the document) starting from a setting in classical data of *Code_Aster*.

Among the features of *Code_Aster*, little is available in CALC_EUROPLEXUS. That is explained on the one hand by the fact why there must be a very close correspondence between the functionality in question in *Code_Aster* and equivalent functionality in EPX. In addition this functionality must be integrated into the macro-order in order to make a correct translation enters *Code_Aster* and EPX.

This document describes in way the most complete possible procedure to follow to add a desired functionality. In a first part, one describes in a brief way the structure of the macro-order while pointing on essential information for the developer. That will be followed by three other parts describing:

- the addition of a law of behavior;
- the addition of a kind of elements (and a keyword ofAFFE_CARA_ELEM);
- the addition of a loading.

In each one of these cases, one leaves the postulate which the developer knows in EPX the functionality equivalent to that it wishes to add in *Code_Aster*.

2 Structure

2.1 Stages

The macro-order `CALC_EUROPLEXUS` breaks up into three principal stages:
1) reading of the data input and translation;
2) launching of calculation EPX ;
3) recovery and working of the results.

At the time of the addition of a functionality the two stages concerned are 1) and 3).

2.1.1 Reading of the data and translation

This stage corresponds exactly to the use of `CALC_EUROPLEXUS` with `LANCEMENT=' NON'`. It produces the command file EPX and grid MED (possibly comprising an initial state).

Dedicated modules

The body the macro one (`calc_europlexus_ops.py`) draft successively the various entries (model, characteristics elementary, behavior and materials, loadings,…). In most case, the effective treatments are made in dedicated modules python. These modules are in the repertoire `Calc_epx/`.

`calc_epx_geom.py`	Treatment of the model
`calc_epx_mate.py`	Treatment of materials and behaviors
`calc_epx_cara.py`	Treatment of the elementary characteristics
`calc_epx_char.py`	Treatment of the loadings
`calc_epx_poutre.py`	Special treatment of orientations of the beams and discrete

Writing of the command file

The command file EPX is a textual file functioning by keyword of four characters. It has a rather regular structure (although many exceptions exist). The module `calc_epx_struc.py` was created to standardize to the maximum the setting in data EPX by seeking the smallest possible regular units to go up until the directives (principal keywords). Various objects were created to facilitate the development of macro and to produce a readable and structured command file.

2.1.2 Recovery and working of the results

The results resulting from EPX are contained in a file MED. Reading of this file and working of the results in an object `evol_noli` is made by one second macro-order: `LIRE_EUROPLEXUS` called by `CALC_EUROPLEXUS`. It is specified however that this macro-order can be called in a way independent of `CALC_EUROPLEXUS`.

2.2 Catalogue translation

The catalogue of translation (`calc_epx_cata.py`) is the central module of `CALC_EUROPLEXUS` and `LIRE_EUROPLEXUS` from the point of view of a developer wishing to add a functionality. It is in this file that all the features authorized like their translation EPX are detailed (as far as possible).

The addition of any functionality thus starts with the edition of this file.

3 Addition of a new law of behavior

One lists the various actions to be operated on the code to add a new law of behavior. Certain actions are detailed in dedicated paragraphs when that is necessary.

3.1 Catalogue of order

To add the name of the new behavior in the file `c_comportement.capy` in the part `COMMAND == ‘CALC_EUROPLEXUS’`.

3.2 Dictionary of the behaviors

In the catalogue of translation (`calc_epx_cata.py`), to duplicate a key couple/value of the dictionary `cata_compor`.
For example for the behavior `VMIS_ISOT_TRAC` this couple is the following:

```
‘VMIS_ISOT_TRAC’: {
        ‘LAW’    : [‘ELAS’, ‘TRACTION’],
        ‘NEED’: [‘O’, ‘O’,],
        ‘REPEAT’: [‘, ‘],
        ‘NOM_EPX’: ‘VMIS ISOT’,
        ‘NOM_EPX_CH_MED’: ‘ISOT’,
        ‘NB_VAR_ASTER’: 2,
        ‘NB_VAR_EPX’   : 5,
        ‘TRANSFORMER’      : True,
            },
```

3.2.1 Stages

1) To replace the existing key by the new behavior.
2) To inform the keywords materials necessary to the operation of the law in the key `‘LAW’` .
3) It is enough to recopy the list `mc_mater` module python of the behavior (`./bibpyt/Comportement/` *`mon_comportement`* `.py`).
4) To say for each one of them in `‘NEED’` if their presence is obligatory or optional (`‘O’` / `‘F’`) and in `‘REPEAT’` if they can be repeated or not (`‘there’, ‘`). In the majority of the cases the presence of these keywords in material is obligatory and they cannot be repeated (installation for specificities of `GLRC_DAMAGE`).
5) In `‘NOM_EPX’` to give the name of material corresponding in EPX (keyword of the directive `MATT`). In certain cases, the chain of four characters making it possible to know material EPX associated with a field given in file MED is not that formed by the first four characters of `‘NOM_EPX’`. In this case it is necessary to add the key `‘NOM_EPX_CH_MED’` with the good value.
6) To indicate respectively in `‘NB_VAR_ASTER’` and `‘NB_VAR_EPX’` the number of internal variables of the law in *Code_Aster* and EPX (field `ECRO` in EPX).
7) To indicate in the keyword `‘TRANSFORMER’` if a routine of transformation of the internal variables of EPX towards *Code_Aster* is necessary and thus to develop (`True/False`).
8) If need be, to develop the transformation of the internal variables of *Code_Aster* towards EPX to send a field of internal variables (not no ones) in initial state.

3.2.2 Details on the internal transformations of variables.

3.2.2.1 Passage EPX 5thrs *Code_Aster*

If `‘TRANSFORMER’` is `False` and that there are more components side *Code_Aster* that side EPX, the reading of the internal variables will fail. It is necessary has minima to activate `‘TRANSFORMER’` and to program the addition amongst missing components in their affecting value 0.

If `‘TRANSFORMER’` is `False` and that there are at least as many components side EPX side *Code_Aster*, The component i EPX is copied from the component i of *Code_Aster* if $i \leq NbVarAster$. The other components are forgotten.

If 'TRANSFORMER' is True , the transformation should then be programmed. For that it is necessary to add a new routine (while duplicating already presents) in the module /bibpyt/Calc_epx/trans_var_int.py. The name of this routine must be tr_e2a_ *mon_comportement*. This routine must then to be connected with the rest of the code in the method transfo_var_int class LireEPX () of lire_europlexus_ops.py.

3.2.2.2 Notwise *Code_Aster* 5thrs EPX

When the keyword VARI_INT of ETAT_INIT is put at 'YES', then a field of internal variables is added at the initial state sent to EPX. At the time of the addition of a new law of behavior, if nothing is done, the initial field of internal variables will be null on the meshs on which the new behavior is affected.
So that the values of the internal variables are transferred it is necessary to develop the transformation of *Code_Aster* towards EPX of the law. For that it is necessary:

- To create a new routine of transformation on the model of tr_a2e_vmis_john_cook in the module ./bibpyt/Calc_epx/trans_var_int.py. It will be named tr_e2a_ *mon_comportement*.
- To add the call to the new routine in var_int_a2e same module.

3.3 Dictionary of the laws or materials

In the catalogue of translation (calc_epx_cata.py), to duplicate a key couple/value of the dictionary cata_lois . The keys of this dictionary are formed by the name of the behavior and the keyword material (of DEFI_MATERIAU) that one wishes to define separated by one '/' .

Here the first three couples of this dictionary:

```
'ELAS/ELAS': {
        'PARA'     : ['E', 'NAKED', 'RHO', 'AMOR_ALPHA', 'AMOR_BETA'],
        'PARA_EPX': ['YOUNG', 'NAKED', 'RO', 'KRAY', 'MRAY'],
        'NEED'   : ['O', 'O', 'O', 'F', 'F'],
        'STANDARD'     : ['reality', 'reality', 'reality', 'reality', 'reality'],
                  },
'VMIS_ISOT_TRAC/ELAS': {
        'PARA'     : ['E', 'NAKED', 'RHO',],
        'PARA_EPX': ['YOUNG', 'NAKED', 'RO',],
        'NEED'   : ['O', 'O', 'O',],
        'STANDARD'     : ['reality', 'reality', 'reality',],
                  },
'VMIS_ISOT_TRAC/TRACTION': {
        'PARA'     : ['SIGM',],
        'PARA_EPX': [['ELAS', 'EXAM NERVES'],],
        'NEED'   : ['O',],
        'STANDARD'     : ['fonc',],
                  },
```

3.3.1 Stages

1) To replace the existing key by the key to be added.
2) In 'PARA', to enter names of the parameters of material *Code_Aster*.
3) In 'PARA_EPX', to enter names of the parameters corresponding EPX. In certain cases, there is not perfect correspondence between the parameters and a special treatment must be carried out. One must then make so that the list 'PARA_EPX' with the index in question another list contains in order to be able to detect that a special treatment is to be made. This treatment is to program in a routine to add in the module calc_epx_mate.py and to connect in the routine get_para_all same module after the line:
```
        standard yew (para_epx) is list:
```
4) To say for each one of them in 'NEED' if their presence is obligatory or optional ('O'/'F').
5) To give the type of each one of them in 'STANDARD' , reality or function ('real'/'fonc').

3.4 Synthesis

In the case of a perfectly identical law of behavior enters *Code_Aster* and EPX, the addition are summarized with modifications of catalogues/dictionaries.

For the other cases, targeted specific treatments are to be programmed. One counts some for the moment two. The transformation of the internal components of variables and correspondence of the parameters of the laws. For these two treatments, it is enough to duplicate existing it and to adapt it to the case in question.

4 Addition of a new type of element

Contrary to *Code_Aster*, a modeling or geometry EPX (directing GEOM) is associated with a mesh single support. It is important to have that with the spirit before undertaking the addition of a new element.

4.1 Dictionary of modelings

The first stage consists in adding to the dictionary cata_modelisa catalogue of translation a key of the name of modeling *Code_Aster*. Here several keys of this dictionary:

```
'Q4GG': {
        'MODE_EPX': {
            'TRIA3': ['T3GS'],
            'QUAD4': ['Q4GS'],
                    },
        'ETAT_INIT': True,
        'RESU_ELEM': True,
        'CONT_ASTER': ['NXX', 'NYY', 'NXY', 'MXX', 'MYY', 'MXY', 'QX', 'QY'],
        'MC_CARA': 'HULL',
        'MODI_REPERE': 'HULL',
              },
'POU_D_E': {
        'MODE_EPX': {
            'SEG2': ['POUT']
                    },
        'ETAT_INIT': False,
        'RESU_ELEM': False,
              },
'DIS_TR': {
        'MODE_EPX': {
            'POI1': ['APPU', 'PMAT']
                    },
        'ETAT_INIT': False,
        'RESU_ELEM': False,
              },
```

1) In the dictionary 'MODE_EPX', to add a key of the name of the mesh support of modeling EPX to be added.
2) The value associated with this key must be a list, in most case length 1, containing the names of corresponding modelings EPX. If the list has several elements that means that there exist several possible choices. In the existing cases for the moment one slices between multiple modelings starting from contained information in the structure of data cara_elem . Generally, in these typical cases, should be added specific treatments. This point is detailed in the part 4.2.2 with item 7 .
3) To put ETAT_INIT with True if an initial stress field on this element can be sent to EPX. **That requires developments in EPX.** If not False.
4) To put RESU_ELEM with True if the stress fields (and internal variables) resulting from EPX can be recovered and formatted to be intégrés with a result of *Code_Aster*. If not False.
5) If 'RESU_ELEM' is True , to indicate in the list 'CONT_ASTER' components of constraints *Code_Aster* of this element. Attention they must be given in the order of the corresponding components in EPX. It is specified that it is necessary obligatorily that the constraints in *Code_Aster* and EPX correspond perfectly except for the order and of a forced transformation/effort.
6) If 'RESU_ELEM' and that it is necessary to operate a transformation forced/effort on the stress field for this element, to indicate in 'MC_CARA' the keyword of AFFE_CARA_ELEM to which this element refers.
7) If it is necessary to apply to the constraints a change of reference mark, it should be indicated in 'MODI_REPERE'. In these rare cases, developments will be to make to treat this need.

4.2 Addition again `CARA_ELEM`

The element to be added can need contained information in keyword for `AFFE_CARA_ELEM`. If this keyword is not programmed yet in `CALC_EUROPLEXUS` (see dictionary `cata_cara_elem` catalogue of translation), it is necessary to add it.

4.2.1 Dictionary of the characteristic elementary

For that it is necessary to duplicate a key couple/value of the dictionary `cata_cara_elem` and to replace the key by the keyword in question. Here the first values of this dictionary are given:

```
'HULL': [
        {
            'SELECT': {'THICK': Nun,
                      },
            'TITLE': 'HULLS',
            'DIRECTING': 'COMPLEMENT',
            'MOT_CLE_ASTER': 'THICK',
            'MOT_CLE_EPX': 'THICK',
            'VERIF': {
                      'VECTOR': Nun,  # taken into account elsewhere
                      'ANGL_REP': Nun,  # taken into account elsewhere
                      'COQUE_NCOU': [1],
                      'A_CIS': Nun,            # No taking into account by Exp
                      'COEF_RIGI_DRZ': Nun,    # No taking into account by Exp
                      'MODI_METRIQUE': ['NOT',],
                      },
        }
    ],
    'BEAM': [
        {
            'SELECT': {'SECTION': 'RIGHT-ANGLED',
                      },
            'TITLE': 'ELEMENTS BEAMS',
            'DIRECTING': 'COMPLEMENT',
            'MOT_CLE_EPX': 'GEOP RECT',
            'CARA_ASTER': ['','', '', 'HY', 'HZ'],
            'CARA_EPX': ['VX', 'VY', 'VZ', 'AY', 'AZ'],
            'VERIF': {
                      'VARI_SECT': ['CONSTANT',],
                      },
        },

        {
            'SELECT': {'SECTION': 'CIRCLE',
                      },
            'TITLE': 'ELEMENTS BEAMS',
            'DIRECTING': 'COMPLEMENT',
            'MOT_CLE_EPX': 'GEOP CIRC',
            'CARA_ASTER': ['','', '', 'R'],
            'CARA_EPX': ['VX', 'VY', 'VZ', 'DEXT'],
            'COEF_MULT': [1. , 1. , 1. , 2.],
            'VERIF': {
                      'VARI_SECT': ['CONSTANT',],
                      },
        }
```

The values of the keys are lists of dictionaries. That seems useless above in the examples but that makes it possible to treat the complex case of the key `'DISCRETE'`. One gives then the significances of the various keys allowed in these dictionaries. That is Dû with the difficulty in establishing simple correspondences between information of *Code_Aster* (not organized same manner according to the keywords of `AFFE_CARA_ELEM`) and those of EPX (seldom uniform).

4.2.2 Stages

1) To inform in 'SELECT', the keywords making it possible to make the choice enters the various cases suggested. In the case of the beams, that makes it possible to distinguish the beams rectangularS steel ringsS. In the case of hulls, itthere has that a possibility, however it is necessary when to even inform a keyword so that functions. If the value of the key does not have

importance, "Nun should be given". The only presence of the keyword makes it possible to make the choice.

2) To give a title, which will be in comment in the command file EPX for this elementary characteristic.
3) To indicate in which directives EPX must be integrated the relative information into this keyword. In most case it is the directive `COMP` or `COMPLEMENT` but information on the discrete elements must be in the directive `MATT`.
4) To indicate in `'MOT_CLE_EPX'`, corresponding keyword EPX.
5) If values of only one keyword *Code_Aster* are to be recovered, it is then necessary to inform `'MOT_CLE_ASTER'` with the keyword *Code_Aster* in question. If the value of only one characteristic of the keyword `'CARA'` is to be recovered and that EPX does not use another keyword only the value of `'MOT_CLE_EPX'` to declare this value, it is also necessary to inform `'MOT_CLE_ASTER'` with the name of the characteristic.
6) If not, to inform in `'CARA_ASTER'` the list of the names of the characteristics *Code_Aster* keyword `CARA` to recover.
7) If the fact of assigning these elementary characteristics to a group of meshs given makes it possible to take a decision among several possible modelings EPX, to indicate in `'MODE_EPX'` modeling in question.
8) Generally, when several values are to be recovered, EPX associates a keyword with each one of them. It is then necessary to indicate in `'CARA_EPX'` names of these keyword (characteristics). In the ideal case, the characteristic of index i in the list must correspond to the value of index i in the list of the recovered values. It can happen that certain characteristics for which need EPX has oneselfinT not transmitted by *Code_Aster via* this way. They should however be indicated nevertheless, one will see in the following points how to make the good connections.
9) So characteristics of EPX are provided by this way and only `'CARA_ASTER'` is present, the lists `CARA_EPX` and `CARA_ASTER` must be of the same length all the same. Thus if the characteristic of index i in the list `CARA_EPX` does not have correspondence in the keyword `CARA`, it is necessary to put `''` in index i of `CARA_ASTER`.
10) So characteristics of EPX are provided by this way and only `'MOT_CLE_ASTER'` is present, it is necessary to inform the keyword '`IS_VALE_ASTER`'whose value will be of a the same list length than the list `CARA_EPX`, comprising `True` in index i if the value associated with of the same characteristic index with the list `CARA_EPX` present in the list of value is recovered and `False` if not.

Note:

> *It is essential that the value of index i list of the recovered values corresponds to of the same characteristic index in the list `CARA_EPX` to which one would have withdrawn from the not provided characteristics.*

11) To connect the characteristics provided by another way, the dictionary should be enriched `dic_gr_cara_supp`. These data currently come from the treatment of the model and the keyword `ORIENTATION` of `AFFE_CARA_ELEM`.
12) Lastly, if need be, to add the key `'VERIF'`, while following the existing examples, to make the checking that certain keyword *Code_Aster* take one of the expected values well.

4.3 Passage de forced with effort

The stress fields in EPX are always expressed in constraints and never in efforts even for the elements of structure. It is sometimes necessary to operate a transformation of the constraints into efforts at the time of the addition of a new element. It was specified in 4.1 with item 6, to add the key `'MC_CARA'` with information on the element. If the transformation is not programmed yet for this keyword of `AFFE_CARA_ELEM` , it is necessary to add this additional case in the method `prep_cont_2_eff` class `Lire_EPX ()` module `lire_europlexus_ops.py` .

5 Addition of a new loading

Loadings within the meaning of *Code_Aster* are divided into two distinct directives in EPX: LINK and TANK. LE catalogue translation follows this rule by separating the loadings and the connections in two different dictionaries: cata_liais and cata_char.

5.1 Addition of a connection

5.1.1 Dictionary of the connections

To add a new connection it is necessary to duplicate a key couple/value of the dictionary cata_liais.

```
cata_liais = {
    ‘DDL_IMPO’: {
        ‘MOT_CLE_EPX’: [‘BLOQ’, ‘DEPL’],
        # the choice between BLOQ and DEPL is made into hard in calc_epx_char
        # it is made according to the presence of a function or not
        # if function => DEPL, if not BLOQ
        ‘ASTER’     : [‘DX’, ‘DY’, ‘DZ’, ‘DRX’, ‘DRY MARTINI’, ‘DRZ’],
        ‘EPX’       : [‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’],
        ‘ENTITY’    : [‘GROUP_MA’, ‘GROUP_NO’],
        ‘NB_CLE_MAX’: {‘BLOQ’: 6,
                        ‘DEPL’: 1,
                      },
        ‘VALE_IMPO’: {‘BLOQ’: 0. ,
                       ‘DEPL’: Nun,
                      },
        ‘FONC_MULT’: {‘BLOQ’: False,
                       ‘DEPL’: True,
                      },
                  },
    ‘RELA_CINE_BP’: {
        ‘MOT_CLE_EPX’: [‘LCAB’],
        ‘ASTER’: [‘CABLE_BP’],
        ‘EPX’: Nun,
        ‘MOT_CLE_VERIF’: [‘SIGM_BPEL’, ‘RELA_CINE’, ‘TYPE_EPX’],
        ‘VALE_VERIF’: [‘NOT’, ‘YES’, [‘ADHE’, ‘GLIS’, ‘FROT’]],
        ‘FONC_MULT’: False,
    },
    ‘LIAISON_MAIL’: {
        ‘MOT_CLE_EPX’: [‘RELA’],
        ‘FONC_MULT’: False,
    },
}
```

5.1.2 Stages

1) To replace the name of the key by the name of the keyword of AFFE_CHAR_MECA to add.
2) To indicate in ‘MOT_CLE_EPX’ the name of the corresponding connections in EPX. The value must be a list, generally made up of only one element. Several elements present in the list indicate that the keyword can correspond to several connections in EPX. The procedure to follow in this case is described in 5.3 .
When ‘MOT_CLE_EPX’ takes value RELA, as for LIAISON_MAIL, CALC_EUROPLEXUS will translate the list of relations contained in a table attached to the concept char_meca. One does not follow then the other points (except the point 8). A connection will be on the other hand necessary in AFFE_CHAR_MECA so that the table containing the relations is created (see the routine bibfor/load/calirc. F90).
3) To indicate in ‘ASTER’ the list of the keywordsS to translate.
4) To indicate in ‘EPX’ the list of the translations of these keywords. To indicate Nun if syntax EPX of the connection does not make it possible to return within this framework (attention it is not possible that if the list ‘ASTER’ only one element has).
5) To indicate in ‘ENTITY’ keywords (among GROUP_MA and GROUP_NO) being able to describe the unit to which the connection applies. Not to indicate this keyword if syntaxes *Code_Aster* and/or EPX do not return within this framework.

6) If the list `'ASTER'` is higher than 1, to indicate in `'NB_CLE_MAX'` the number of keywordsS of this list being able to be present in the same occurrence of the connection. If this number is 1, it is not necessary to indicate this key, it is the value by default.
7) If values taken by the keywords of the list `'ASTER'` are imposed, to indicate this value in `'VALE_IMPO'`.
8) To indicate in `'FONC_MULT'` if the connection must be accompanied by a multiplying function (`True/False`).
9) To indicate in `'MOT_CLE_VERIF'` the keywords which one must only check the value, if there are some.
10) To indicate in `'VALE_VERIF'` with the same index the values which the keywords must take.
11) Each keyword present in the occurrence of the connection *Code_Aster* must be in lists `'ASTER'` , `'ENTITY'` or `'MOT_CLE_VERIF'`. That prevents the user from using keywords not taken into account without he being informed by it.
12) If need be, to indicate in `'COEF_MULT'` the coefficient by which must be multiplied the values resulting from *Code_Aster* during the translation in EPX.

5.2 Addition of a loading

5.2.1 Dictionary loadings

To add a new loading, it is necessary to duplicate a key couple/value of the dictionary `cata_char`. Here a key of this dictionary is given:

```
cata_charge = {
    'FORCE_COQUE': {
        'TYPE_CHAR'  : 'FACTO',
        'MOT_CLE_EPX': ['CLOSE COQU'],
        'ASTER'      : ['NEAR'],
        'EPX'        : Nun,
        'COEF_MULT'  : -1,
        'ENTITY'     : ['GROUP_MA'],
                  },
              }
```

5.2.2 Stages

The stages are exactly the same ones as for a connection with two exceptions near:

1) To indicate in `'TYPE_CHAR'`, type EPX of the loading (`FACTO` or `CONST`). In the case `FACTO`, the loading must be accompanied by a multiplying function, but not in the case `CONST`.
2) Not to give the keyword `'FONC_MULT'` since it is implied by the type of loading.

5.3 Treatment of the multiple choices

It can happen that a loading or a connection of *Code_Aster* can correspond to several loadings or connections in EPX.

First of all it is necessary to identify what will determine which possibility is retained. The analysis of this criterion and the choice which results from this must be programmed into hard in the module `calc_epx_char.py` in the routine `export_charge` after the list of code:

```
yew len (mot_cle_epx) > 1:
```

The dictionary must then be adapted to these multiple possibilities with following logic. Each key whose value is not identical for each case must be a dictionary whose keys are the various names of the loadings or possible connections and whose values are the values desired for the key in question. The connection `DDL_IMPO` in is an illustration.