Note of use of connections 1D-3D

Summary:

This note aims to guide the user in the choice of a connection Beam – 3D in dynamics in Code_Aster.

Methodologies of connection suggested are of two type:

- Not-intrusive surface connection called rocks 1D-3D (option `3D_POU` of `LIAISON_ELEM`)
- Intrusive voluminal connection within the framework Harlequin (option `3D_POU_ARLEQUIN` of `LIAISON_ELEM`)

Contents

1 Introduction

For the calculation of localised effects in space and time, a model beam makes it possible to gain in computing times in absence as of nonlinear effects, and a mixed model beam-3D makes it possible to take into account the nonlinear effects limited in space.

2 Rock not-intrusive 1D-3D

A rocker of a model beam to a mixed model beam-3D makes it possible to gain considerably in computing times for a precision equivalent to a model whole 3D [bib2].

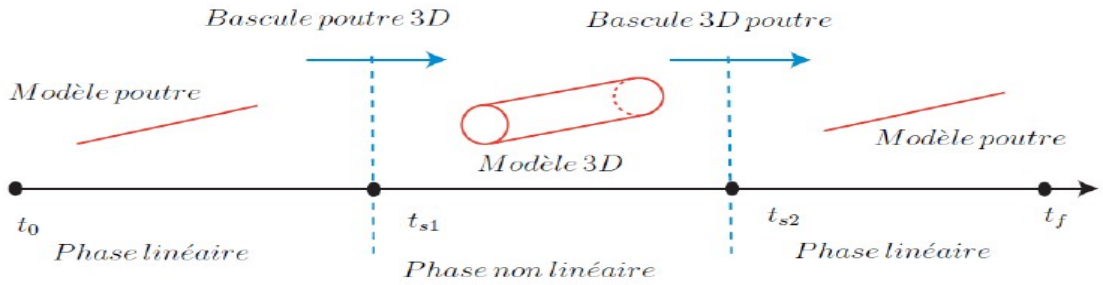


Image 1.1-a weight functions within the framework Harlequin

Code_Aster allows to initialize a dynamic calculation with fields of displacements, speeds and accelerations nonworthless (ETAT_INIT in the operators of dynamic calculation). These fields of initialization can be calculated beforehand, then transformed into a field in *Code_Aster*.

2.1 Theoretical principle of the rocker

The rocker consists in passing from a model of Beam ($\mathbf{U_p}$ is the solution beam system $M_p\ddot{U}_p+C_p\dot{U}_p+K_pU_p=f_p$) with a model 3D (solution $\mathbf{U_{3D}}$ to initialize) via a relevant initialization of the 3D solution. The basic principle is the following:

1. starting from the solution beam $\mathbf{U_p}$, to create a 3D solution $\mathbf{PU_p}$ for the rigid assumption of section

2. to write the fields displacement 3D as being the sum of the extruded field of Beam type $\mathbf{Pu_p}$ and of a correct field in noted displacement $\mathbf{U_{3Dc}}$, fascinating of account deformation of the section **:**

$$U_{3D}=PU_p+U_{3Dc}$$

3. The dynamics of the model 3D will be written then as follows:

$$M_{3D}\left(P\ddot{U}_p+\ddot{U_{3Dc}}\right)+C_{3D}\left(P\dot{U}_p+\dot{U_{3Dc}}\right)+K_{3D}\left(PU_p+U_{3Dc}\right)=f_{3D}$$

4. to correct displacements, **in statics**, under the effect of a vector F forces$_{3Dc}$ calculated as follows:

$$f_{3Dc}=K_{3D}U_{3Dc}=f_{3Dc}\left(t=t_b\right)-M_{3D}P\ddot{U}_p-C_{3D}P\dot{U}_p-K_{3D}PU_p$$

The static correction is carried out on 3 pas de successive times: the moment of rocker T_B like that of before ($T_{b-1} = T_B - \Delta T$) and that according to ($T_{b+1} = T_B + \Delta T$) where ΔT is the step of computing time. One from of deduced a correction of speed which is written as follows:

$$\dot{U}_{3D}=\frac{\left[PU_p+U_{3Dc}\right]_{(t_{b+1})}-\left[PU_p+U_{3Dc}\right]_{(t_{b-1})}}{2\Delta t}$$ (Centered Finished Differences)

Accelerations, as for them, are is calculated in the same way according to a diagram of Centered Finished the Differences type,

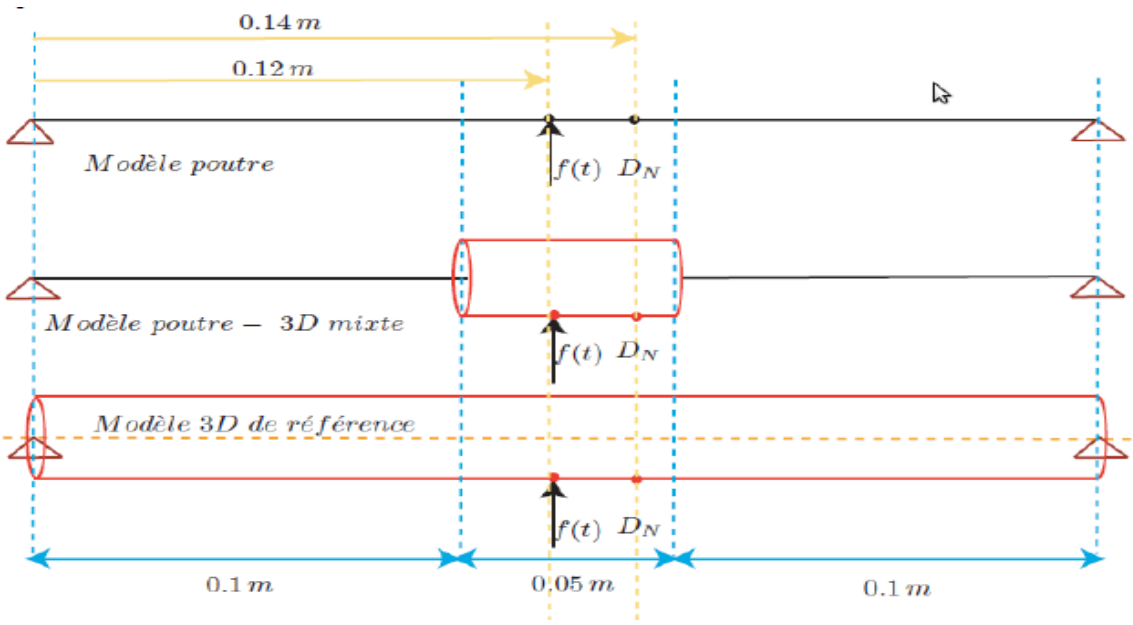
$$\ddot{U}_{3D}=\frac{\left[PU_p+U_{3Dc}\right]_{(t_{b-1})}+\left[PU_p+U_{3Dc}\right]_{(t_{b+1})}-2\left[PU_p+U_{3Dc}\right]_{(t_b)}}{\Delta t^2}$$

that is to say given in an automatic way by *Code_Aster* in the case of the use of an implicit diagram of integration in time.

The correction comprises a correction of the arrow as well as a correction to take into account the deformation of the section.

2.2 Example of implementation

Methodology is clarified below for a problem of bi--supported beam subjected to a sinusoidal dynamic excitation lasting 3s.



Simulation starts by a model of beam. Dynamic calculation on the mono-model of beam takes place of the initial moment t0 until the moment of rocker to the model 3D (here, t0=0s, tb=2.4s).

```
Ti=0;
Tf=3.0;
Tb=2.4;

dt=0.0015;

Tb_1=Tb-dt;
Tb_2=Tb+dt;
```

The following operations will be used for the rocker.

```
# Creation of the empty grid Python
mail_py = MAIL_PY ()
```

```
# Conversion of the grid Aster into grid Python
mail_py.FromAster ('E-mail')

# Coordinates of the nodes
COORD_3D = mail_py.cn

iPouAllNode = mail_py.gno.get ('AllNode')

iPouRP1 = mail_py.gno.get ('RP1')
iPouRP2 = mail_py.gno.get ('RP2')

C3DZ = zeros ((len (COORD_3D), 1))
for I in arranges (len (C3DZ)):
   C3DZ [I] = COORD_3D [I] [2]

C1DZ1 = []
min=1112
while (min! =1111):
   I = 0
   S = int ((ABS (C3DZ [I]) +ajust) *Pow) /float (Pow)
   for I in arranges (len (C3DZ)):
      yew (int ((C3DZ [I] +ajust) *Pow) /float (Pow))==min:
         C3DZ [I] = 1111
      elif ((int ((C3DZ [I] +ajust) *Pow) /float (Pow)) < S):
         S = int ((C3DZ [I] +ajust) *Pow) /float (Pow)
   yew (min! =s):
      min=s
      C1DZ1.append (min)

C1D = zeros ((len (C1DZ1) - 1.3))
for I in arranges (len (C1D)):
   C1D [I] [0] = 0
   C1D [I] [1] = 0
   C1D [I] [2] = C1DZ1 [I]

# Function to seek the section which corresponds has a value of Z
def OrdreSect (Z):
   jk=0
   while (C1D [jk] [2]! =z):
      jk=jk+1
    return jk
```

At the moment of rocker Tb, one recovers displacements Up, Vp speeds and Ap accelerations of neutral fibre of the beam by the operator CREA_CHAMP, operation 'EXTR'. Then, components (DX, DY, DZ, DRX, DRY MARTINI, DRZ) are extracted in tables Python.

```
Time = NP.array ([Tb, Tb_1, Tb_2])
nbpas = temps.shape [0]

U0= []
U1= []
U2= []

for I in arranges (nbpas):

   not = time [I]
```

```
DEP=CRÉA_CHAMP (OPERATION=' EXTR',
                TYPE_CHAM=' NOEU_DEPL_R',
                RESULTAT=DLT,
                NOM_CHAM=' DEPL',
                INST=pas,);

Upx = DEP.EXTR_COMP ('DX', ['AllNode']) .valeurs;
Upy = DEP.EXTR_COMP ('DY', ['AllNode']) .valeurs;
Upz = DEP.EXTR_COMP ('DZ', ['AllNode']) .valeurs;
Uprx = DEP.EXTR_COMP ('DRX', ['AllNode']) .valeurs;
Upry = DEP.EXTR_COMP ('DRY MARTINI', ['AllNode']) .valeurs;
Uprz = DEP.EXTR_COMP ('DRZ', ['AllNode']) .valeurs;

ACC=CRÉA_CHAMP (OPERATION=' EXTR',
                TYPE_CHAM=' NOEU_DEPL_R',
                RESULTAT=DLT,
                NOM_CHAM=' ACCE',
                INST=pas,);

ddotUpx = ACC.EXTR_COMP ('DX', ['AllNode']) .valeurs;
ddotUpy = ACC.EXTR_COMP ('DY', ['AllNode']) .valeurs;
ddotUpz = ACC.EXTR_COMP ('DZ', ['AllNode']) .valeurs;
ddotUprx = ACC.EXTR_COMP ('DRX', ['AllNode']) .valeurs;
ddotUpry = ACC.EXTR_COMP ('DRY MARTINI', ['AllNode']) .valeurs;
ddotUprz = ACC.EXTR_COMP ('DRZ', ['AllNode']) .valeurs;
```

By an assumption of type rigid body, one calculates tables Python containing displacements PUp, PVp speeds and accelerations PAP of the mixed model by extrusion of the fields beam in the thickness.

Then, tables PUp Python and ddotPUp are calculated and contain the fields of displacement and acceleration of the nodes of the mixed model according to topology (1D or 3D).

```
 dictLNo = []
dictPUpx = []
dictPUpy = []
dictPUpz = []
dictPUprx = []
dictPUpry = []
dictPUprz = []
dictAccx = []
dictAccy = []
dictAccz = []
dictAccrx = []
dictAccry = []
dictAccrz = []
for I in arranges (len (PUp)):
   j=i+1
   yew len (PUp [I]) ==3:
      dictLNo.append ('N%i'%j);
      dictPUpx.append (PUp [I] [0]);
      dictPUpy.append (PUp [I] [1]);
      dictPUpz.append (PUp [I] [2]);
      dictAccx.append (ddotPUp [I] [0]);
      dictAccy.append (ddotPUp [I] [1]);
      dictAccz.append (ddotPUp [I] [2]);
   else:
      dictLNo.append ('N%i'%j);
      dictPUpx.append (PUp [I] [0]);
      dictPUpy.append (PUp [I] [1]);
```

```
        dictPUpz.append (PUp [I] [2]);
        dictPUprx.append (PUp [I] [3]);
        dictPUpry.append (PUp [I] [4]);
        dictPUprz.append (PUp [I] [5]);
        dictAccx.append (ddotPUp [I] [0]);
        dictAccy.append (ddotPUp [I] [1]);
        dictAccz.append (ddotPUp [I] [2]);
        dictAccrx.append (ddotPUp [I] [3]);
        dictAccry.append (ddotPUp [I] [4]);
        dictAccrz.append (ddotPUp [I] [5]);
```

These tables Python are transformed thereafter into lists then in fields of Code_Aster.

```
TD=CRÉA_TABLE (LISTE= (_F (LISTE_K=dictLNo, PARA=' NOEUD'),
                       _F (LISTE_R=dictPUpx, PARA=' DX'),
                       _F (LISTE_R=dictPUpy, PARA=' DY'),
                       _F (LISTE_R=dictPUpz, PARA=' DZ'),
                       _F (LISTE_R=dictPUprx, PARA=' DRX'),
                       _F (LISTE_R=dictPUpry, PARA=' DRY'),
                       _F (LISTE_R=dictPUprz, PARA=' DRZ'),))

TA=CRÉA_TABLE (LISTE= (_F (LISTE_K=dictLNo, PARA=' NOEUD'),
                       _F (LISTE_R=dictAccx, PARA=' DX'),
                       _F (LISTE_R=dictAccy, PARA=' DY'),
                       _F (LISTE_R=dictAccz, PARA=' DZ'),
                       _F (LISTE_R=dictAccrx, PARA=' DRX'),
                       _F (LISTE_R=dictAccry, PARA=' DRY'),
                       _F (LISTE_R=dictAccrz, PARA=' DRZ'),))

DeplPUp=CRÉA_CHAMP (TYPE_CHAM=' NOEU_DEPL_R',
                    OPERATION=' EXTR',
                    MAILLAGE=Mail,
                    TABLE=TD,);

AccIni=CRÉA_CHAMP (TYPE_CHAM=' NOEU_DEPL_R',
                   OPERATION=' EXTR',
                   MAILLAGE=Mail,
                   TABLE=TA,);
```

The products matrix-fields **K3D*PUp** and **M3D*PAp** are carried out by PROD_MATR_CHAM.

```
FORCE1=PROD_MATR_CHAM (MATR_ASSE=RIGIDITE,
                       CHAM_NO=DeplPUp,
                       TITRE=' PROD_MATR_CHAM1',);

TBLf1=POST_RELEVE_T (ACTION=_F (OPERATION=' EXTRACTION',
                     INTITULE=' f1',
                     CHAM_GD=FORCE1,
                     GROUP_NO=' AllNode',
                        NOM_CMP= ('DX', 'DY', 'DZ', 'DRX', 'DRY MARTINI',
'DRZ',),),);

VARf1=TBLf1.EXTR_TABLE ()

FORCE2=PROD_MATR_CHAM (MATR_ASSE=MASSE,
                       CHAM_NO=AccIni,
                       TITRE=' PROD_MATR_CHAM2',);

TBLf2=POST_RELEVE_T (ACTION=_F (OPERATION=' EXTRACTION',
```

```
                        INTITULE=' f2',
                        CHAM_GD=FORCE2,
                        GROUP_NO=' AllNode',
                            NOM_CMP= ('DX', 'DY', 'DZ', 'DRX', 'DRY MARTINI',
'DRZ',),),);

   VARf2=TBLf2.EXTR_TABLE ()
```

According to the forces in presence in with the dealt problem, the loading **f3D (Tb) - K3D*PUp-M3D*PAp** is calculated in the form of a dictionary by nodes then stored as follows:

```
CHARGE3D=AFFE_CHAR_MECA (MODELE=MODELE,
                         FORCE_NODALE= (dictCharge),);
```

Then, a field of correction of the deformation of the U3Dc section is calculated by MECA_STATIQUE like the answer, at the moment of rocker, model 3D of reference to the loading **f3D (Tb) - K3D*PUp-M3D*PAp**.

```
 Mstat=MECA_STATIQUE (MODELE=MODELE,
                      CHAM_MATER=CHMAT,
                      CARA_ELEM=Carel3D,
                      EXCIT= (_F (CHARGE=CondLim,),
                              _F (CHARGE=CHARGE3D,),),);
```

The answer which results from it is raised then stored in a table:

```
tableUc=POST_RELEVE_T (ACTION=_F (OPERATION=' EXTRACTION',
                       INTITULE=' U3Dc',
                       RESULTAT=Mstat,
                       NOM_CHAM=' DEPL',
                       PRECISION=1e-15,
                       GROUP_NO=' AllNode',
                           NOM_CMP= ('DX', 'DY', 'DZ', 'DRX', 'DRY MARTINI',
'DRZ',),),);

TC=tableUc.EXTR_TABLE ()
```

Lastly, the field of displacement of the patch 3D of the mixed model at the moment **Tb** will be the sum of the two fields **U3Dc** and **PUp** .

```
# dictionary of displacements at the time of the rocker
dictDeplx = []
dictDeply = []
dictDeplz = []
dictDeplrx = []
dictDeplry = []
dictDeplrz = []
for I in arranges (len (U0)):
   j=i+1
   yew len (U0 [I]) ==3:
      dictDeplx.append (U0 [I] [0]);
      dictDeply.append (U0 [I] [1]);
      dictDeplz.append (U0 [I] [2]);
   else:
      dictDeplx.append (U0 [I] [0]);
      dictDeply.append (U0 [I] [1]);
      dictDeplz.append (U0 [I] [2]);
      dictDeplrx.append (U0 [I] [3]);
      dictDeplry.append (U0 [I] [4]);
```

```
      dictDeplrz.append (U0 [I] [5]);

Dini=CRÉA_TABLE (LISTE= (_F (LISTE_K=dictLNo, PARA=' NOEUD'),
                        _F (LISTE_R=dictDeplx, PARA=' DX'),
                        _F (LISTE_R=dictDeply, PARA=' DY'),
                        _F (LISTE_R=dictDeplz, PARA=' DZ'),
                        _F (LISTE_R=dictDeplrx, PARA=' DRX'),
                        _F (LISTE_R=dictDeplry, PARA=' DRY'),
                        _F (LISTE_R=dictDeplrz, PARA=' DRZ'),))

Dep0=CRÉA_CHAMP (TYPE_CHAM=' NOEU_DEPL_R',
                 OPERATION=' EXTR',
                 MAILLAGE=Mail,
                 TABLE=Dini,);
```

The fields of speed and acceleration are also calculated according to a diagram of Centered Differences, by using the fields Beam and corrected displacements, speeds and accelerations at the moment of rocker **Tb** and at the moments **Tb – ΔT** and **Tb + ΔT**.

```
# Calculation calibrated airspeeds
dictVitex = []
dictVitey = []
dictVitez = []
dictViterx = []
dictVitery = []
dictViterz = []
for I in arranges (len (U1)):
    j=i+1
    yew len (U1 [I]) ==3:
       dictVitex.append ((U2 [I] [0] - U1 [I] [0])/(2*dt));
       dictVitey.append ((U2 [I] [1] - U1 [I] [1])/(2*dt));
       dictVitez.append ((U2 [I] [2] - U1 [I] [2])/(2*dt));
    else:
       dictVitex.append ((U2 [I] [0] - U1 [I] [0])/(2*dt));
       dictVitey.append ((U2 [I] [1] - U1 [I] [1])/(2*dt));
       dictVitez.append ((U2 [I] [2] - U1 [I] [2])/(2*dt));
       dictViterx.append ((U2 [I] [3] - U1 [I] [3])/(2*dt));
       dictVitery.append ((U2 [I] [4] - U1 [I] [4])/(2*dt));
       dictViterz.append ((U2 [I] [5] - U1 [I] [5])/(2*dt));

Vini=CRÉA_TABLE (LISTE= (_F (LISTE_K=dictLNo, PARA=' NOEUD'),
                        _F (LISTE_R=dictVitex, PARA=' DX'),
                        _F (LISTE_R=dictVitey, PARA=' DY'),
                        _F (LISTE_R=dictVitez, PARA=' DZ'),
                        _F (LISTE_R=dictViterx, PARA=' DRX'),
                        _F (LISTE_R=dictVitery, PARA=' DRY'),
                        _F (LISTE_R=dictViterz, PARA=' DRZ'),))

Vit0=CRÉA_CHAMP (TYPE_CHAM=' NOEU_DEPL_R',
                 OPERATION=' EXTR',
                 MAILLAGE=Mail,
                 TABLE=Vini,);

# Calculation of corrected accelerations
dictAccex = []
dictAccey = []
dictAccez = []
dictAccerx = []
dictAccery = []
```

```
dictAccerz = []
for I in arranges (len (U1)):
    j=i+1
    yew len (U1 [I]) ==3:
       dictAccex.append ((U2 [I] [0] - 2*U0 [I] [0] +U1 [I] [0])/(dt ** 2));
       dictAccey.append ((U2 [I] [1] - 2*U0 [I] [1] +U1 [I] [1])/(dt ** 2));
       dictAccez.append ((U2 [I] [2] - 2*U0 [I] [2] +U1 [I] [2])/(dt ** 2));
    else:
       dictAccex.append ((U2 [I] [0] - 2*U0 [I] [0] +U1 [I] [0])/(dt ** 2));
       dictAccey.append ((U2 [I] [1] - 2*U0 [I] [1] +U1 [I] [1])/(dt ** 2));
       dictAccez.append ((U2 [I] [2] - 2*U0 [I] [2] +U1 [I] [2])/(dt ** 2));
         dictAccerx.append ((U2 [I] [3] - 2*U0 [I] [3] +U1 [I] [3])/(dt **
2));
         dictAccery.append ((U2 [I] [4] - 2*U0 [I] [4] +U1 [I] [4])/(dt **
2));
         dictAccerz.append ((U2 [I] [5] - 2*U0 [I] [5] +U1 [I] [5])/(dt **
2));

Aini=CRÉA_TABLE (LISTE= (_F (LISTE_K=dictLNo, PARA=' NOEUD'),
                         _F (LISTE_R=dictAccex, PARA=' DX'),
                         _F (LISTE_R=dictAccey, PARA=' DY'),
                         _F (LISTE_R=dictAccez, PARA=' DZ'),
                         _F (LISTE_R=dictAccerx, PARA=' DRX'),
                         _F (LISTE_R=dictAccery, PARA=' DRY'),
                         _F (LISTE_R=dictAccerz, PARA=' DRZ'),))

Acc0=CRÉA_CHAMP (TYPE_CHAM=' NOEU_DEPL_R',
                 OPERATION=' EXTR',
                 MAILLAGE=Mail,
                 TABLE=Aini);
```

The fields of initialization thus built are used to initialize dynamic calculation (ETAT_INIT in the operators of dynamic calculation).

```
Bascule=DYNA_LINE_TRAN (MODELE=MODELE,
                        CHAM_MATER=CHMAT,
                        MATR_MASS=MASSE,
                        MATR_RIGI=RIGIDITE,
                        SCHEMA_TEMPS=_F (...),
                        ETAT_INIT=_F (DEPL=Dep0,
                                      VITE=Vit0,
                                      ACCE=Acc0,),
                        EXCIT= (_F (VECT_ASSE=F_Xass,
                                    FONC_MULT=FSIN),),
                        INCREMENT=_F (LIST_INST=LISTE,),);
```

The nonintrusive procedure of rocker of a model of beam to a model 3D in transitory dynamics, developed and validated in Code_Aster (CAS-test SDNV139), makes it possible to gain in computing times because the total model of standard mean structure can replace the local model 3D in the absence of nonlinearities located in time. In addition, the use of preexistent connection 3D-POU in Code_Aster makes it possible to reduce the size of the model for cases of nonlinearities located in space.

3 Connection 3D-Beam within the framework Harlequin

The framework Harlequin (option 3D_POU_ARLEQUIN keyword LIAISON_ELEM of the operator AFFE_CHAR_MECA) is a flexible method of connection of models 1D and 3D with covering [bib2]. It thus makes it possible also to gain considerably in computing times for a precision equivalent to a model whole 3D.

3.1 Principal ingredients Harlequin

The method Harlequin rests on the following principles:

- The connection of under fields via a weak formulation: the introduction of the multipliers of Lagrange into the zone of joining guarantees the coupling of the models, the continuity of the kinematics quantities, as well as the control of the variations of the constraints and the deformations between the coupled zones;

- Distribution of energy between fields and models: with an aim of not twice counting the energy of the total system in the zone of covering, the virtual work associated with the two models is distributed between the under-fields coupled through the zone of joining by the means of weight functions $(\gamma, 1-\gamma)$ who form a partition of the unit (the sum of the two functions is equal to 1) on the whole of the field of study.

Weighting makes it possible to avoid taking into account energy several times in the same zone and authorizes thus a certain freedom with the user for the choice of the prevalent model. Indeed, it makes it possible to put the weight on the model which one wishes to make express.

The matrices of coupling obtained are transformed into relations kinematics. This connection thus results in linear relations connecting displacements of the whole of nodes 3D (3 degrees of freedom per node) dependent with the whole of the nodes of beam (6 degrees of freedom per node).

3.2 Example of implementation

It is pointed out that the connection 3D-Beam Harlequin, as developed today in Code_Aster, must meet a very important requirement. Indeed, Lbe grids 1D and 3D must be "hierarchically compatible", with the direction where all the elements 3D are included in the cylindrical space of the element 1D in opposite (not of elements 3D to horse between two elements beam).

The grid associated with mixed model 1D-3D is read by LIRE_MAILLAGE :

```
MAIL=LIRE_MAILLAGE (UNITE=20,);
```

The reading of the grids 1D and 3D can also be done separately. Then, the two grids are assembled via the operator ASSE_MAILLAGE.

Then, the elements of structure and voluminal are assigned to the model:

```
MODELE=AFFE_MODELE (MAILLAGE=MAIL,
                           AFFE= (_F (GROUP_MA= ('Ref3D'),
                                      PHENOMENE=' MECANIQUE',
                                      MODELISATION=' 3D',),
                                  _F (GROUP_MA= ('Beam'),
                                      PHENOMENE=' MECANIQUE',
                                      MODELISATION=' POU_D_T',),),);
```

The geometrical characteristics of the elements of beam are also affected:

```
CAREL3D=AFFE_CARA_ELEM (MODELE=MODELE,
                                POUTRE=_F (GROUP_MA= ('Beam'),
                                           SECTION=' CERCLE',
                                           CARA=' R',
                                           VALE=0.005,),);
```

The management of weightings in the zones free and joining Harlequin is left with the care of the user. As it is about a coupling of the L2 type, it is enough to impose the weighting coefficients in the

definition of the parameters materials (density RHO and Young modulus E) relative to the various zones of covering.

```
# Material used in the zones out of covering
      MATL=DEFI_MATERIAU (ELAS=_F (E=2.E11*1.0,
                                   NU=0.3,
                                   RHO=7800.0*1.0,),);

      # Material used in the zone of joining included in covering
      MATC=DEFI_MATERIAU (ELAS=_F (E=2.E11*0.5,
                                   NU=0.3,
                                   RHO=7800.0*0.5,),);

      # Material used in the free zone 3D included in covering
      MATR3D=DEFI_MATERIAU (ELAS=_F (E=2.E11*0.99,
                                     NU=0.3,
                                     RHO=7800.0*0.99,),);

      # Material used in the free zone 1D included in covering
      MATR1D=DEFI_MATERIAU (ELAS=_F (E=2.E11*0.01,
                                     NU=0.3,
                                     RHO=7800.0*0.01,),);
```

Apart from the zone of covering, the materials are given by the user as he hears it (material MATL for GROUP_MA=' Poutre'). For the zone of covering, the partition of the unit is ensured by assigning to the groups of predefined meshs already balanced materials. In this case, modelings are balanced (weight = 50%) in the same way in the zone of joining (material MATC for GROUP_MA= ('1DCol', '3DCol')). In the free zone (with the free direction of the conditions of connection Harlequin), we chose to put the weight at 99% on the voluminal model (material MATR3D for GROUP_MA=' 3DLib') and to 1% on the model Beam (material MATR1D for GROUP_MA=' 1DLib').

```
CHMAT=AFFE_MATERIAU (MAILLAGE=MAIL,
                AFFE= (_F (GROUP_MA=' Poutre',
                           MATER=MATL,),
                       _F (GROUP_MA=' 1DLib',
                           MATER=MATR1D,),
                       _F (GROUP_MA=' 1DCol',
                           MATER=MATC,),
                       _F (GROUP_MA=' 3DLib',
                           MATER=MATR3D,),
                       _F (GROUP_MA=' 3DCol',
                           MATER=MATC,),
                        ),);
```

The condition of connection Harlequin is then specified under the option 3D_POU_ARLEQUIN keyword LIAISON_ELEM of the operator AFFE_CHAR_MECA. For each occurrence of the connection Harlequin, the user must define the following operands :

- The group of meshs 3D of the zone of joining (keyword GROUP_MA_1)
- The group of meshs 1D of the zone of joining (keyword GROUP_MA_2)
- The concept CHAM_MATER defining materials (being used to ensure the partition of the unit)
- The concept CARA_ELEM defining the geometrical characteristics being used with calculation of the matrices as coupling

```
ARLE=AFFE_CHAR_MECA (MODELE=MODELE,
                     LIAISON_ELEM= (_F (OPTION='3D_POU_ARLEQUIN',
                                        GROUP_MA_1= ' 3DCol',
                                        GROUP_MA_2= ' 1DCol',
                                        CHAM_MATER=CHMAT,
```

```
                                    CARA_ELEM=CAREL3D,),
                                ),);
```

In the sight of a modal or dynamic calculation on modal basis, the matrices of mass and stiffness can be assembled with, inter alia, the load resulting from the coupling within the framework Harlequin.

```
ASSEMBLY (MODELE=MODELE,
              CHAM_MATER=CHMAT,
              CARA_ELEM=CAREL3D,
              CHARGE= (CondLim,ARLE),
              NUME_DDL=CO ('NUMEDDL'),
              MATR_ASSE= (_F (MATRICE=CO ('RIGIDITY'),
                              OPTION=' RIGI_MECA',),
                          _F (MATRICE=CO ('MASS'),
                              OPTION=' MASS_MECA',),
                          ),);
```

An example of calculation in transitory linear dynamics in presence on the basis of mixed model 1D-3D connected within the framework Harlequin is given below (cf CAS-tests SSLV156 and SSLV160):

```
RefM3D=DYNA_LINE_TRAN (MODELE=MODELE,
                       CHAM_MATER=CHMAT,
                       CARA_ELEM=CAREL3D,
                       MATR_MASS=MASSE,
                       MATR_RIGI=RIGIDITE,
                       SCHEMA_TEMPS=_F (SCHEMA=' NEWMARK',
                                        GAMMA=0.5+alpha,
                                        BETA= (1+alpha) ** 2/4,),
                       EXCIT= (_F (CHARGE=CondLim,),
                               _F (CHARGE=ARLE,),
                               _F (CHARGE=Charge,
                                   FONC_MULT=FSIN),),
                       INCREMENT=_F (LIST_INST=LISTE3D,),);
```

4 Bibliography

[1] M.Tannous, "Development and evaluation of coupled approaches of digital modeling 1D and 3D of the contact rotor-stator", Thesis of the Central School Nantes.

[2] A. Ghanem, "Contribution to the advanced modeling of the revolving machines in transitory dynamics within the framework Harlequin", Thesis of INSA de Lyon.