Note of use of the boundary conditions treated by elimination

Summary

Treatment of the boundary conditions of the Dirichlet type by elimination (AFFE_CHAR_CINE) do not offer the same general information as by dualisation (AFFE_CHAR_MECA for example).

This treatment is to be used when one searches to improve the execution times of a calculation or if one wishes to work with positive definite matrices.

Let us note that the boundary conditions available in AFFE_CHAR_* (* = MECA/THER/ACOU) cannot all be eliminated and treated by AFFE_CHAR_CINE.

In this document, it is shown how to use the “loads kinematics” in the command sets Aster.

There are 3 cases (simpler with most complicated):

- One uses a “total” ordering of calculation (THER_LINEAIRE, STAT_NON_LINE,…). In this case, the loads kinematics are used like the other loads.
- One wishes to do a calculation of clean modes. It is then necessary to add an argument in the order ASSE_MATRICE.
- One wishes to do a calculation “step by step” and to solve the linear systems with the orders TO FACTORIZE and TO SOLVE. In this case, the order should be used CALC_CHAR_CINE.

1 Principle of elimination

One seeks to solve in $\mathbb{R}^n$ the problem of minimization under constraint (Pb1) according to:

$$\min_{u\in U_G}\left(\frac{1}{2}u^T K u - u^T f\right) \quad \text{with} \quad U_G=\left\{u\in\mathbb{R}^n,\, u\big|_G=u_0\right\}$$

where

- $u_0\in\mathbb{R}^p$ is known ($1\le p\le n$)
- G is the subset of $N=\{1,\dots,n\}$, of cardinal p : $G=g_1\dots g_p$
- $u\big|_G$ is the projection of u on under space generated by $\{u_i\}_{i\in G}$
- where $(u_i)_j=\delta_{ij},\ \forall\, j\in N$
- K is a symmetrical matrix $n\times n$,
- $f\in\mathbb{R}^n$ is fixed.

The constraint $u\big|_G=u_0$ or not represent boundary conditions of the homogeneous Dirichlet type.

If one notes $L=C_N G$ the complementary one to G in N , one can, using u_i defined previously, to break up $\mathbb{R}^n$ all in all direct of V_G = vector space generated by $\{u_i\}_{i\in G}$ and of V_L = vector space generated by $\{u_i\}_{i\in L}$;

Consequently, we have $\mathbb{R}^n=V_G\oplus V_L$
and one notes $u=u_G\oplus u_K$ where $u_G=u\big|_G$ and $u_L=u\big|_L$
that is to say still in vectorial notation $u=\begin{Bmatrix}u_G\\u_L\end{Bmatrix}$

The problem (Pb1) can thus be written in the form of the problem (Pb2):

$$\begin{cases}\min\left(\frac{1}{2}u_G^T K_{GG} u_G+\frac{1}{2}u_L^T K_{LL} u_L+u_L^T K_{LG} u_G-u_L^T f_L-u_G^T f_G\right)\\ u_G\in V_G\\ u_L\in V_L\\ u_G=u_0\end{cases}$$

What amounts writing:

$$(\text{Pb1})\Leftrightarrow(\text{Pb2}) \qquad \begin{cases}\min\left(\frac{1}{2}u_L^T K_{LL} u_L+u_L^T K_{LG} u_0-u_L^T f_L\right)\\ u_L\in V_L\\ u=u_0\oplus u_L\end{cases}$$

One then eliminated u_G problem of minimization.

We now will search the matric problem associated with (Pb3).

One searches u_L minimizing

$$\frac{1}{2}u_L^T K_{LL} u_L + u_L^T K_{LG} u_0 - u_L^T f_L$$

what amounts solving the following matric problem:

$$K_{LL} u_L = F_L - K_{LG} u_0$$

One can thus write:

$$(\text{Pb1}) \Leftrightarrow (\text{Pb2}) \Leftrightarrow (\text{Pb3}) \Leftrightarrow \begin{bmatrix} K_{LL} & 0 \\ 0 & I_G \end{bmatrix} \begin{bmatrix} u_L \\ u_G \end{bmatrix} = \begin{bmatrix} f_L - K_{LG} u_0 \\ u_0 \end{bmatrix}$$, that is to say $K' \begin{bmatrix} u_L \\ u_G \end{bmatrix} = f'$

2 Treatment in *Aster*

2.1 The loads kinematics

A kinematic load (standard *Aster* : char_cine_* [* = meca/ther/acou]) makes it possible to characterize the unit G of the imposed ddl and them $(u_0)_i$ for $i \in G$ who are the values assigned to these ddl.

The definition of a kinematic load is done via the operator AFFE_CHAR_CINE for $(U_0)_I$ constant or functions of the geometry or time.

2.2 The vectors kinematics

The kinematic vector is a cham_no_* which represents the vector $\begin{bmatrix} 0 \\ u_0 \end{bmatrix}$.

With each kinematic load corresponds a kinematic vector.

This operation is carried out by the operator CALC_CHAR_CINE.

2.3 Calculation of K'

K' is directly calculated at the assembly time by the operator ASSE_MATRICE provided naturally that one provides in argument a list of loads kinematics.

The structure of data MATR_ASSE_* was modified in order to be able to store K' when that is necessary.

2.4 Calculation of f'

After L " operator TO FACTORIZE the concept of the matr_asse_* type produced, contains factorized of K' and the matrix K_{LG} unchanged.

The calculation of f' is carried out at the time of the resolution: it is necessary to provide the operator TO SOLVE in argument the kinematic vector corresponding to $\begin{bmatrix} 0 \\ u_0 \end{bmatrix}$ via the keyword CHAM_CINE.

This operator calculates then f' before solving $\text{fact}(K') \begin{bmatrix} u_L \\ u_G \end{bmatrix} = f'$.

3 Examples of command files

3.1 Mechanical calculation with a total order (STAT_NON_LINE):

```
DEPIMP=AFFE_CHAR_CINE(MODELE=MOD,
                      MECA_IMPO=_F (  GROUP_MA = 'LCD1', DY = -2.0))

RESU=STAT_NON_LINE (MODELE=MOD,  CHAM_MATER=CHMAT,
                    EXCIT= _F (  LOAD = DEPIMP,   FONC_MULT = FONC),
                    …)
```

3.2 Loads kinematics for a calculation of clean modes:

```
CHARCINE=AFFE_CHAR_CINE(MODELE=MODEL,
                        MECA_IMPO=_F (GROUP_MA=' GM2', DX=0.0, DY=0.0))


KASS=ASSE_MATRICE(MATR_ELEM=KELEM,
                  NUME_DDL=NUMÉRIQUE,
                  CHAR_CINE=CHARCINE,);

MASS=ASSE_MATRICE(MATR_ELEM=MELEM,
                  NUME_DDL=NUMÉRIQUE,
                  CHAR_CINE=CHARCINE,);

# calculation of the clean modes of the structure
MODES=CALC_MODES (MATR_RIGI=KASS,
                  MATR_MASS=MASS,
                  CALC_FREQ=_F (NMAX_FREQ=10))
```

3.3 Calculation "step by step" by using the orders TO FACTORIZE and TO SOLVE :

```
CHCINE=AFFE_CHAR_CINE(   MODELE=MO,  MECA_IMPO= (
                 _F (GROUP_NO = 'SUPY',   DY = 0.),
                 _F (GROUP_NO = 'LOAD', DX = -1.)))


MEL=CALC_MATR_ELEM (   MODELE=MO, CHAM_MATER=CHMAT,  OPTION=' RIGI_MECA')


NU=NUMÉRIQUE_DDL (  MATR_RIGI=MEL)

MATAS=ASSE_MATRICE(MATR_ELEM=MEL,  NUME_DDL=NU,  CHAR_CINE=CHCINE)


SCMBRE=CRÉA_CHAMP (…)

VCINE=CALC_CHAR_CINE(NUME_DDL=NU, CHAR_CINE=CHCINE )

MATAS=FACTORISER (reuse=MATAS,  MATR_ASSE=MATAS)
```

```
DEP=TO SOLVE(MATR=MATAS, CHAM_NO=SCMBRE,  CHAM_CINE=VCINE)
```