Indicators of performance of a calculation (time/memory)

1 Goal

During a simulation with *Code_Aster*, **postings by default trace** in the files message and results (`.mess/.resu`) **certain dimensioning characteristics of calculation** (RAM consumption, time CPU, system and user…). This documentation details postings of these indicators of performance.

One formulates too **some advices to help the user to benefit from these diagnoses**. But it is necessary well to be conscious that there is no universal receipt to optimize the total performances of a calculation. That depends on the type of study, of the aspects software and hardware of the machine, even of its load!

The parameter setting by default and postings/alarms of the code propose a balanced and instrumented operation. But, to be sure as well as possible to have used the capacities of its machine, L'user must remain attentive with the elements described in this document like to the advices present in documentations of the orders. This document shows many elements of user's documentations on the keyword `SOLVEUR` [U4.50.01], [U2.08.06] and on parallelism [U2.08.03].

Contents

1 Goal....1
2 General information....3
3 Characteristics of the linear system....4
4 Ventilation of spent times....6
4.1 Times spent by order....6
4.1.1 Total monitoring of each order....6
4.1.2 Fine monitoring of each order....6
4.2 Total spent times....8
4.3 Typical case of DYNA/STAT_NON_LINE....8
5 Consumption RAM memory....10
5.1 Calibration memory of a calculation....10
5.2 Consumption memory JEVEUX....12
5.3 Consumption memory of external products (MUMPS)....14
5.4 Consumption system....16
6 Some advices to optimize the performances....17
6.1 Concerning the characteristics of the problem....17
6.2 Concerning spent times....17
6.3 Concerning the RAM memory consumed....18
6.4 Concerning parallelism....19

2 General information

During a simulation *Code_Aster*, postings by default trace in the files message and results (.mess/.resu) **certain dimensioning characteristics of calculation**. One finds in particular, for each operator *Aster* :

- *characteristics of the linear systems* to build and solve (many nodes, of equations and Lagranges, size of the matrix…),
- *memories* *JEVEUX* floor (to pass into Out-Of-Core[1]) and optimal (to pass in In-Core),
- *memory out* *JEVEUX* required by some *external products* (e.g. MUMPS),
- *time* CPU, system and "user" (elapsed),
- *ventilation of times* soups according to the stages of calculation (elementary calculation, assembly, resolution of the linear system).

This last description of spent times can be declined according to various reading levels (synthetic impression, detailed and detailed by increment of calculation) *via* the parameter MESURE_TEMPS/NIVE_DETAIL orders DEBUT/POURSUITE. In parallel mode, one adds the median value, on all the processors, of the times spent like their standard deviation.

1 The Outone (OOC) is a way of managing of the memory which consists in discharging on disc certain objects allocated by the code to release from RAM. These unloadings can be automatic (started by the system or the software package JEVEUX) or organized by the programmer. Strategy OOC makes it possible to deal with larger problems but these accesses disc slow down calculation. *A contrario*, the mode In-Core (IC) consists in keeping the objects in RAM. That limits the size of the accessible problems, but privileges speed.

3 Characteristics of the linear system

A large number of operators (STAT/DYNA/THER_NON_LINE, CALC_MODES,…) the construction and the resolution of linear systems require. To solve these systems of equations, one uses called particular algorithms "**linear solveurs**". These linear solveurs are in fact omnipresent in the unfolding of the operators of *Code_Aster* because they are often employed by the digital algorithms: nonlinear diagram, integration in time, analyze modal etc They consume of it the major part of time CPU and the memory.
By default (INFO=1), each order traces, during the construction of its first system linaire, its characteristics: size N, many nonworthless terms NNZ …

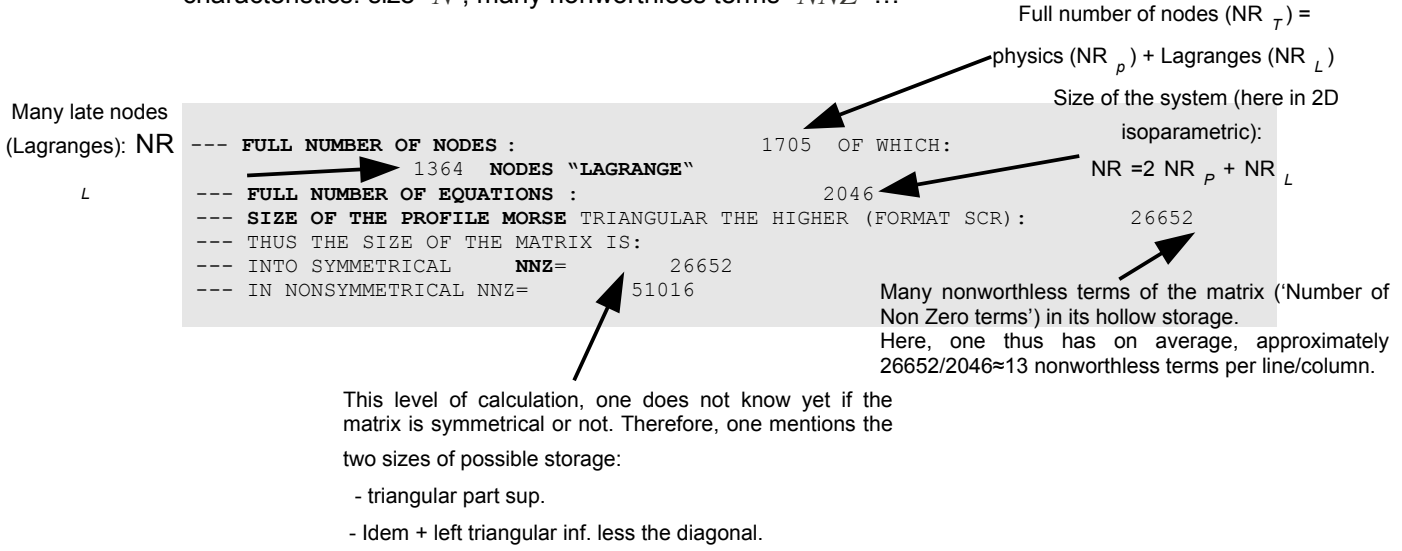

Figure 3-a : Characteristics of the linear system in the course of assembly in a standard order Aster STAT_NON_LINE or THER_LINEAIRE (extracted .mess).

size of the system, **proportion of Lagranges** and it **filling of the matrix** inform indirectly about **consumption mémoires/CPU** calculation and on possible difficulties of resolutions.
Rapidly, the algorithmic complexity of the construction of the linear system is in NNZ while its resolution in $N^{\alpha}.NNZ$ (with $1<\alpha<2$).
The occupation total memory is it of the type[2] $\beta.NNZ$ (with $10<\beta<100$) $\times 8 octets$. This consumption memory gathers several structures of data *Aste*R and/or related to external products (MATR_ASSE, NUME_DDL, factorized matrix…). Fortunately, these structures of data break up into several segments distinct reports and the algorithmic one does not impose their simultaneous presence in RAM memory. One can thus often discharge on disc a good part of these data.
In addition, the introduction of variables of the Lagranges type[3] in the matrices (matrices said then dualized) degrades their digital properties (size, definite-positivity, matric conditioning). That thus implies often more digital processings in the solveurs and degrades their performances. This remark can extend besides to all the mixed finite elements (incompressible modeling, continuous method in contact…).

Note:

2 β is coarsely the factor of filling of factorized. I.e., the surfactor cuts of it memory compared to the initial matrix, which the process of filling of factorization (total factorization in simple implies or double precision of LDLT/MULT_FRONT/MUMPS or factorization single precision of the préconditionneurs GCPC/PETSC, [U4.50.01] and associated reference materials).
3 These variables of Lagrange are introduced "artificially" in the course of calculation in order to take into account the conditions of Dirichlet (simple/blocking or generalized/connection).

The posting of the characteristics of the problem (figure 3.1) is carried out only once, at the beginning of order. In general, these characteristics do not change in the course of calculation. The only exceptions concern the operator STAT_NON_LINE *with method X-FEM in great slip or the method continues in contact. For those, the profile of the matrix changes as iterations.*
This posting is carried out very early in the process, at the conclusion of the creation of the profile of the matrix[4] Aster. The assembly of the elementary terms is not carried out yet[5]. For this reason one cannot rule, a priori, on the symmetrical character or not of the matrix. In practice, it is very often symmetrical and, when it is not the case, one can symmetrize it via the option SOLVEUR/SYME.

4 Procedure which analyzes the unknown factors of the problem (to establish the algebraic link node/ddl/unknown) and dimensions certain structures of data Aster (NUME_DDL, MATR_ASSE.).

5 It is enough that an elementary term is nonsymmetrical so that the assembled matrix becomes it.

4 Ventilation of spent times

4.1 Times spent by order

Code_Aster propose two reading levels to analyze the times spent by each order:
- One **total level** who amalgamates all the stages of calculation of the aforesaid order,
- One **finer level** (skeletal) which makes it possible to dissociate the principal stages.

4.1.1 Total monitoring of each order

At the conclusion of each order, are traced in the file message various consumption in time[6] of the operator: CPU[7]+système (USER+SYST), system (SYST) and elapsed (ELAPS). ***A priori*, any important drift of time SYST and/or of time ELAPSED must question (cf. §6.2).**

```
#  FINE NO ORDERS: 0045   USER+SYST:  520.01s (SYST:  12.17s, ELAPS:    525.12s)
#  --------------------------------------------------------------------------------
```

Figure 4.1.1-a : Traces of total consumption in time of an order Aster (extracted from one .mess).

These times are also recapitulated, for all the orders, in the file result.

```
********************************************************************************
* COMMAND                :       TO USE:     SYSTEM:   USER+SYS:    ELAPSED *
********************************************************************************
* init (jdc)             :       2.62:       0.88:       3.50:       4.30 *
*  . compile             :       0.00:       0.00:       0.00:       0.01 *
*  . exec_compile        :       0.54:       0.03:       0.57:       0.58 *
*  . carryforward        :       0.03:       0.00:       0.03:       0.03 *
*  . build               :       0.00:       0.00:       0.00:       0.00 *
* BEGINNING              :       0.04:       0.05:       0.09:       0.13 *
* PRE_GIBI               :      10.75:       1.89:      12.64:      12.69 *
* LIRE_MAILLAGE          :      27.92:       0.13:      28.05:      28.37 *
* DEFI_MATERIAU          :       0.01:       0.00:       0.01:       0.01 *
* AFFE_MATERIAU          :       0.04:       0.01:       0.05:       0.05 *
* AFFE_MODELE            :       5.48:       0.08:       5.56:       5.57 *
* AFFE_CHAR_MECA         :       0.52:       0.02:       0.54:       0.54 *
                                    ...
* MECA_STATIQUE          :    2249.89:      18.55:    2268.44:    2271.87 *
* TEST_RESU              :       0.01:       0.01:       0.02:       0.01 *
* FINE                   :       0.11:       0.01:       0.12:       0.17 *
*  . share Supervisor    :       3.37:       0.94:       4.31:       5.21 *
*    . sdveri            :       0.68:       0.01:       0.69:       0.73 *
*  . share FORTRAN       :    4600.81:      39.42:    4640.23:    4650.96 *
********************************************************************************
* TOTAL_JOB              :    4604.18:      40.36:    4644.54:    4656.18 *
********************************************************************************
```

Figure 4.1.1-b : Total consumption in times of all the orders Aster (extracted from one .resu).

4.1.2 Fine monitoring of each order

ventilation of spent times (USER+SYST, SYST, ELAPS) according to the various stages of calculation (elementary calculation, assembly, digital factorization…) is carried out at the conclusion of each operator implying the construction and/or the resolution of linear systems (for example STAT_NON_LINE and CALC_CHAMP). By default, one displays synthetic values (niv=1). Indeed, this description of spent times can be declined according to various reading levels *via* the parameter MESURE_TEMPS/NIVE_DETAIL orders DEBUT/POURSUITE.
***A priori*, any important drift of time SYST and/or of time ELAPS must question (cf. §6.2).**

6 Time CPU measures the execution of the sources of code (C, FORTRAN, python). Time system takes into account the calls subjacent systems (access disque/RAM…). Time elapsed includes the two precedents and measurement past real time (“wall clock”).

7 Time CPU is called here TO USE.

Let us detail the levels of impressions of MESURE_TEMPS/NIVE_DETAIL=niv (**défaut=1)**:

/niv =0, No impression relating to monitoring at the end of each order,

/niv =1, Synthetic impressions of the three types of time for elementary calculations/assemblies and the resolution of associated linear systems:

```
#1.Resolution.des.systemes.lineaires..... CPU. (USER+SYST/SYST/ELAPS): … 7.52… 0.79. 11.22
#2.Calculs.elementaires.et.as semblages…   CPU. (USER+SYST/SYST/ELAPS):. .15.07… 0.70… 15.77
```

Figure 4.1.2-a : Consumption in time in an order Aster type STAT_NON_LINE or CALC_CHAMP with niv=1 into sequential (extracted from one .mess).

/niv =2, Detailed impressions of times for the two large classes of calculations: elementary calculations/assemblies and resolution of associated linear systems. This kind of information can inform as for the impact of a modification of the command file or parameters of launching of calculation (memory, parallelism…). For example, knowing that parallelism MPI potentially allows to decrease (cf Doc. U2 dedicated) that stages 1.3/1.4/2, **that is not very useful to parallel a calculation**[8] on tens of processors if the stage 1.2 takes 20% of total time[9] !

```
#1.Resolution.des.systemes.lineaires..... CPU. (USER+SYST/SYST/ELAPS): … 7.72… 0.82… 8.72
#1.1.Numerotation, .connectivity .de.la.ma trice CPU. (USER+SYST/SYST/ELAPS): … 0.21… 0.02…
0.31
#1.2.Factorisation.symbolic ..... CPU. (USER+SYST/SYST/ELAPS): … 0.58… 0.05… 1.28
#1.3.Factorisation.numeric. (ou.precond.)….CPU. (USER+SYST/SYST/ELAPS): … 6.78… 0.73… 7.71
#1.4.Resolution ..... CPU. (USER+SYST/SYST/ELAPS): … 0.15… 0.02… 0.35
# 2.Calculs.elementaires.et.as semblages..... CPU. (USER+SYST/SYST/ELAPS):. .28.87… 0.64…
29.47
#2.1.Routine.calcul ..... CPU. (USER+SYST/SYST/ELAPS):. .26.61… 0.56…
26.61
#2.1.1.Routines.te00ij ..... CPU. (USER+SYST/SYST/ELAPS):. .24.58… 0.07…
25.78
#2.2.Assemblages ..... CPU. (USER+SYST/SYST/ELAPS): … 2.26… 0.08… 3.36
#2.2.1.Assemblage.matrices ..... CPU. (USER+SYST/SYST/ELAPS): … 2.02… 0.06… 3.12
#2.2.2.Assemblage.seconds.membres ..... CPU. (USER+SYST/SYST/ELAPS): … 0.24… 0.02… 0.37
```

Figure 4.1.2-b : Consumption in time in a standard order Aster STAT_NON_LINE or CALC_CHAMP with niv=2 into sequential (extracted from one .mess).

/niv =3, Idem that niv=2 but the impression is made for each step of time or increment of calculation.

In parallel mode MPI (cf Doc. U2 on parallelism or Doc. U4.50.01), one adds the median value, on all the processors, of the times spent like their standard deviation. This information is interesting to locate**possible imbalances of loads**. They can be due to a distribution of data[10] nonhomogeneous of many meshs, in term of complexity of law of behavior, in access report to the structures of data…

```
#1.Résolution.des.systèmes.linéaires ..... CPU. (USER+SYST/SYST/ELAPS): … 0.29… 0.00… 0.35
   ….(average….diff. .procs)..... CPU. (USER+SYST/SYST/ELAPS): … 0.30… 0.00… 0.47
….    (variation-type.diff. .procs)..... CPU. (USER+SYST/SYST/ELAPS): … 0.01… 0.00…
0.05
```

Figure 4.1.2-c : Consumption in time in a standard order Aster STAT_NON_LINE or CALC_CHAMP with niv=1 in parallel mode MPI (extracted from one .mess).

8 That all the same will make it possible to lower consumption in time and the quantity of memory necessary for the construction and the resolution of the linear system. As a result, the other features such contact-friction could be some (a little) accelerated because they would lay out thus of more than place in RAM.

9 The saving of time (speed-up) is limited to a factor 5, even on hundreds of processors!

10 In the flood of distributed data of JEVEUX or of possible parallel external tools (MUMPS, PETSC).

In addition, it happens that the manager memory of *Aster* (JEVEUX) **discharge of the objects on disc** to release from RAM (cf. §5.1). According to the hardware configuration, the load machine and the size of the problem, these unloadings are likely to slow down in a considerable way unfolding of calculation. **When unloadings occurred, during an operator, one traces the office plurality of their spent times** (USER+SYST, SYST, ELAPS) at the end of the operator. That makes it possible thereafter to refine possible diagnoses (cf. §6.2).

```
#4 Unloading of the memory on disc   CPU (USER+SYST/SYST/ELAPS):  0.04   0.04      0.04
```

Figure 4.1.2-d : Overcosts in times which had with unloadings on disc of JEVEUX (extracted from one .mess).

Note:

One recapitulates recovered total volume and the number of occurrences of this mechanism at the end of the file message (cf. §5 .1).

4.2 Total spent times

At the end of the file message are traced systematically **the sum of consumption** CPU, CPU+SYST, SYST and the remainder of time not used (between the time mentioned by the user in *Astk* and time CPU+SYST).

***A priori* , any important drift of time SYSTEM must question (cf. § 6.2).**

```
<I> < INFORMATION TIME EXECUTION > (IN SECOND)
      TOTAL TIME CPU ..............  2160.55
      TIME CPU TO USE TOTAL .........  2099.33
      TIME CPU TOTAL SYSTEM ......    61.22
      TIME CPU REMAINING ............   439.45
```

Figure 4.2-a : Total consumption in time of a calculation (extracted from a .mess).

4.3 Typical case of DYNA/STAT_NON_LINE

In these orders of static dynamics/nonlinear, one traces out of standard (INFO=1) **for each step of time or increment of calculation**, in supplement of the table of the reduction of the residues (the level of details is managed by the keyword POSTING orders):

- Stored fields (selected by the keyword FILING),
- The ventilation of spent times CPU and, possibly, the iteration count associated (e.g. process of Newton),
- Blocks of dedicated postings (e.g. contact-discrete).

filing of the fields can be expensive in time (especially in parallel) and in memory. It is thus to be interesting to analyze the list of the fields filed so if required limiting them.

```
FILING OF THE FIELDS:
   FIELD STORES: CONT_NOEU        MOMENT:  5.00000E+02  SEQUENCE NUMBER:    50
   FIELD STORES: DEPL             MOMENT:  5.00000E+02  SEQUENCE NUMBER:    50
   FIELD STORES: SIEF_ELGA        MOMENT:  5.00000E+02  SEQUENCE NUMBER:    50
   FIELD STORES: VARI_ELGA        MOMENT:  5.00000E+02  SEQUENCE NUMBER:    50
   FIELD STORES: QUICKLY             MOMENT:  5.00000E+02  SEQUENCE NUMBER:    50
   FIELD STORES: ACCE             MOMENT:  5.00000E+02  SEQUENCE NUMBER:    50

TIME CPU CONSUMES IN THIS STEP OF TIME:  0 S
   TIME BY ITERATION OF NEWTON  :  0 S                    - NO. NEWT.:     2
   TIME FILING                  :  0 S
   TIME CLASSIFICATION CREATION    :  0 S                    - DIGITAL NO.:     0
   TIME FACTORIZATION STAMPS    :  0 S                    - NO. FACT.:     0
   TIME INTEGRATION BEHAVIOR:  0 S                    - NO. INTE.:     3
   TIME RESOLUTION K.U = F      :  0 S                    - NO. RESO.:     2
   TIME CONTACT RESOLUTION      :  0 S                    - NO. ITER.:     2
   DIFFERENT TIME OPERATIONS       :  0 S

DISCRETE CONTACT:
   ITERATION COUNT OF CONTACT        :        2
   ITERATION COUNT OF REACTIONARY. GEOM     :        2
```

Filed fields
To limit (especially in parallel)

Ventilation CPU/itérations time

Only these parts can profit from parallelism

Block of postings dedicated (contact…)

```
 FINAL NUMBER OF CONNECTIONS OF CONTACT   :        0
 TOTAL TIME PAIRING              :  0 S
 TOTAL TIME RESOLUTION               :  0 S
```

Figure 4.3-a : Posting with each step of time of DYNA/STAT_NON_LINE with INFO=1 in sequential mode (extracted from one .mess).

At the end of the operator the total statistics on all the transient are summarized. These times CPU find themselves in the total statistics of end of operator mentioned in the preceding paragraph. On the other hand, their granularity finer and is adapted at the various stages of the operator.

```
--------------------------------------------------------------------------------
STATISTICS ON THE TRANSIENT
--------------------------------------------------------------------------------
   MANY STEPS OF TIME                 :    100
   ITERATION COUNT OF NEWTON          :    200
   ITERATION COUNT OF CONTACT (Algorithm)   :    456
   ITERATION COUNT OF CONTACT (GEOM)   :    200
   NUMBER OF CREATION OF CLASSIFICATION      :      1
   NUMBER OF FACTORIZATION OF MATRIX      :      2
   NUMBER OF INTEGRATION OF BEHAVIOR    :    201
   NUMBER OF RESOLUTION K.U = F            :    200
   TIME FOR CLASSIFICATION CREATION        :  10 S
   TIME FOR FACTORIZATION STAMPS        : 100 S
   TIME FOR INTEGRATION BEHAVIOR     :  3 m  8 S
   TIME FOR RESOLUTION K.U = F           :  9 S
   TIME FOR CONTACT (PAIRING)        : 17 S
   TIME FOR CONTACT (ALGORITHM)         :  2 m 30 S
--------------------------------------------------------------------------------
```

Figure 4.3-b : Total statistics with resulting from DYNA/STAT_NON_LINE with INFO=1 (extracted from one .mess).

5 Consumption RAM memory

At every moment, L**RAM memory used has by *Code_Aster*** can be the office plurality of three components.

- **memory allocated directly by the sources of the code.** It is mainly about that allocated by **the software package JEVEUX**[D6.02.01] to manage all the structures of F77 data of the code. This software package also makes it possible to discharge on disc these objects (Out-Of-Core functionality). According to the size of the problem, that of RAM and recommendations of the programmer, JEVEUX carry out data exchanges more or less marked between the disc and RAM.
- **memory used by an external software** solicited by calculation *Aster* (MUMPS/PETSc, METIS/SCOTCH, LOBSTER, MISS3D…). Most external software function in In-Core. I.e. they do not manage directly the possible overflows report and that they sub-contract this task with the operating system. Others, like the linear solvor MUMPS, are potentially OOC and they explicitly manage to them-even unloading on disc of certain large objects (cf keyword SOLVEUR/GESTION_MEMOIRE[U4.50.01]).
- **memory required by the system** (loading of part of achievable, network layer in parallel…) and by the supervisor and the bookstores Python.

When an operator is used *Code_Aster* requiring the construction and the resolution of linear systems (e.g. STAT_NON_LINE or CALC_MODES), **the limitations memories are often imposed by the linear solvor** (cf [U4.50.01] [U2.08.03]). When it acts internal solveurs (LDLT, MULT_FRONT, GCPC), their RAM consumption is found in postings JEVEUX. On the other hand, when one uses an external product (MUMPS or PETSC), it is then necessary to take account of its own consumption (which replaces that mainly of JEVEUX).

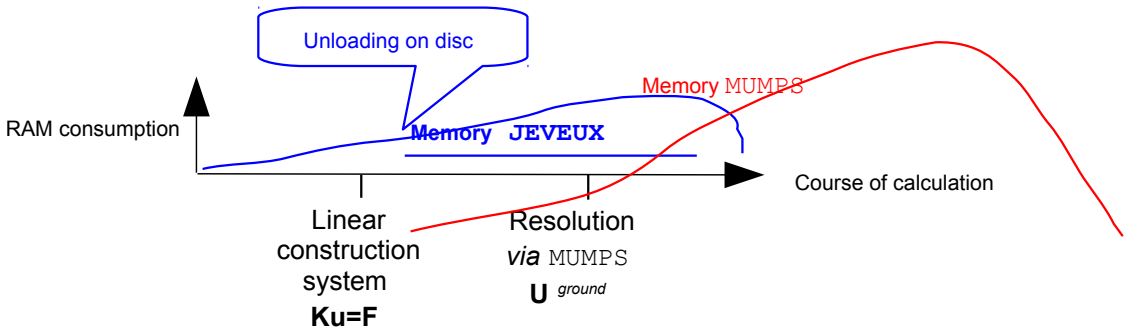

Figure 5-a : Diagram of evolution of RAM consumption to the course of a standard calculation in linear mechanics.

5.1 Calibration memory of a calculation

It is often interesting to gauge on the level RAM report a calculation *Aster.* That can, for example, to make it possible to optimize its placement in a class batch (sequential or parallel) or quite simply to avoid its brutal stop because of a defect of memory. With this intention, one can proceed as follows:

- **Stage n°1** : to locate in the course of calculation the operator who seems more dimensioning in the face of problem (it is often it STAT/DYNA_NON_LINE or it CALC_MODES … treating the largest model).
- **Stage n°2:** if it is not the case, to parameterize the block SOLVEUR of this operator with METHODE=' MUMPS' and GESTION_MEMOIRE=' EVAL'. At the conclusion of the calibration, if required to think of giving the old parameter setting.
- **Stage n°3:** to launch calculation such as it is with parameters modest report and times compared to usual consumption of this kind of calculation. Indeed, with this option,

Code_Aster just will build the first linear system of the operator and will transmit it to MUMPS for analysis. This last will not carry out its stage the expensive one (in time and especially in memory) of digital factorization. Once this completed analysis, estimated memory of MUMPS (MUE_{IC} and MUE_{OOC}), united with that of JEVEUX (JE_{OOC}), are traced in the file message (cf appears 5.1-a). Then calculation stops in FATAL ERROR in order to allow to pass as soon as possible at the following stage.

```
*******************************************************************************
 - Size of the linear system: 500000

 - Minimal RAM memory consumed by Code_Aster                        : 200 Mo
 - Estimate of the Mumps memory with GESTION_MEMOIRE=' IN_CORE'     : 3500 Mo
 - Estimate of the Mumps memory with GESTION_MEMOIRE=' OUT_OF_CORE' : 500 Mo
 - Estimate of the disk space for Mumps with GESTION_MEMOIRE=' OUT_OF_CORE': 2000 Mo

  ===> For this calculation, one thus needs a quantity of RAM memory at least of
        - 3500 Mo if GESTION_MEMOIRE=' IN_CORE',
        -  500 Mo if GESTION_MEMOIRE=' OUT_OF_CORE'.
 In case of doubt, use GESTION_MEMOIRE=' AUTO'.
*******************************************************************************
```

Figure 5.1-a : Posting in the file message in mode 'EVAL.

- **Stage n°4**: exploitation of estimated the reports themselves.

To start again calculation with linear solvor MUMPS, one directly has access to the values "floor" of the RAM memory essential to calculation. One distinguishes two cases according to the memory way of managing from MUMPS chooses: In-Core (value $max(JE_{OOC}, MUE_{IC})$ if GESTION_MEMOIRE= ' IN_CORE ') or Out-Of-Core (value $max(JE_{OOC}, MUE_{OOC})$ if GESTION_MEMOIRE= ' OUT_OF_CORE '). According to the machine/the classes batch which one lays out, it is thus necessary to start again complete calculation by modifying also possibly this parameter of the block SOLVEUR .
These estimated are established for a computer set-up and digital data: material platform, bookstores, many processes MPI, renumerotor, preprocessings…

- **Stage n°4bis**: on the other hand, if one wishes to start again calculation while changing linear solvor or one of the digital parameters, it is more difficult to deduce estimated the adapted memory from it. The combinative one and the variability of the possibilities is too important. One can however propose some empirical rules (in sequential mode). Coarsely, so instead of METHODE=' MUMPS' one chooses 'MULT_FRONT'estimated the memory should remain licit (with RENUM=' METIS' value by default). With 'LDLT', this figure must be significantly re-examined upwards. With 'PETSC'/'GCPC'+' LDLT_SP', it can undoubtedly be reduced to $max(JE_{OOC}, MUE_{IC/OOC}/2)$. With 'PETSC'/'GCPC'+' LDLT_INC' it can undoubtedly be reduced to a factor time JE_{OOC} according to the level of filling of the preconditionnor (keyword NIVE_REMPLISAGE cf [R6.01.02]).

- **Stage n°5:** to start again complete calculation by possibly modifying the parameter setting of the block SOLVEUR and by parameterizing Astk with the value deduced at the stage n°4 (finely "Total memory (Mo)" figure 5.2-b).

Note:

This procedure of calibration memory of a study is available only since the restitution of the keyword SOLVEUR/GESTION_MEMOIRE (starting from the version v11.2 of Code_Aster). For versions more old, if calculation were already launched (and if one has an exploitable file of message), one can, has minimum, to prudently begin again like optimal estimate of the memory required, the greatest value on all the processors of Vmpeak last operator. If not, it is necessary to proceed as mentioned in the previous versions of this document:

- ***Stage n°1:** as above.*

- **Stage n°2:** to parameterize the block SOLVEUR with METHODE=' MUMPS', ' OUT_OF_CORE'=' OUI' and INFO=2 .
- **Stage n°3:** to launch calculation by limiting consumption in time (for example, with little step of time or little research clean modes).
- **Stage n°4:** according to the memory way of managing of necessary MUMPS (IC or OOC), to take the maximum of estimate MUMPS and consumption JEVEUX .

5.2 Consumption memory JEVEUX

elements to quantify consumption memory of JEVEUX are traced at the end of each operator *Aster* (cf appears 5.2-a). Rapidly, one can clarify them in the following way:

- JE_{OOC} provides **minimal size** ('Minimum') **for of which with need JEVEUX to function to this operator**. It will be then completely Out-Of-Core (OOC). In on this side this value, calculation is not possible (in the configuration selected).
- JE_{IC} provides a lower limit ('Optimum'), to this operator, memory JEVEUX necessary to function completely in In-Core (IC): JE'_{IC}. With at least this value of RAM memory parameterized in Astk (cf. MEM_{ASTK}), calculation will proceed in an optimal way: there is no risk of orderly due to a lack of memory and the accesses to the structures of data purely Aster are slowed down little by unloadings on disc.

These two figures are inevitably lower than the total bill of devoted RAM memory with calculation (parameterized in *Astk*), noted MEM_{ASTK} .


```
# Statistics memory (Mo): 15521.78/8920.14/1603.77/248.27 (VmPeak/VmSize/Optimum/Minimum)
```

Figure 5.2-a : Total statistics at the conclusion of each order (extracted from one .mess).

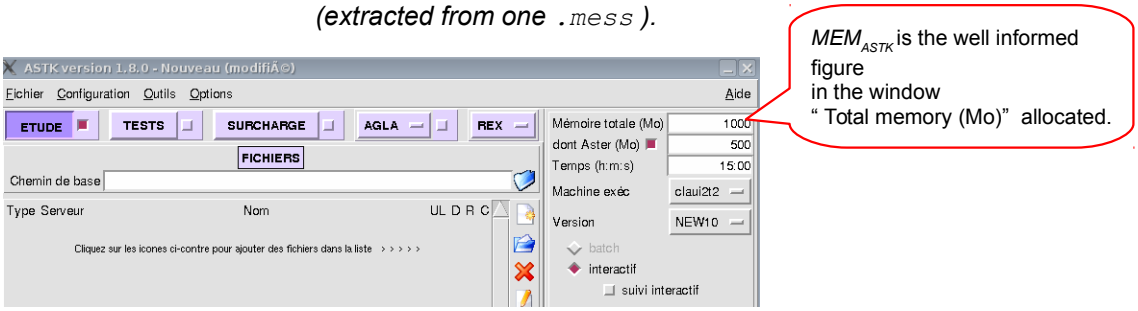

Figure 5.2-b : Parameter setting in Astk of the memory RAM.

Note:

If there were no total mechanism of release (cf paragraph below), JE_{IC} represent truly the memory JEVEUX required to function in IC. If not, this last estimate must be close to the sum[11] *$JE'_{IC} = JE_{IC} + $. "average profit gotten by each release". It is not thus used for nothing (if one does not use an external product) to parameterize calculation with a value much higher than JE'_{IC} .*

During the parameter setting of calculation, it is thus necessary to establish a compromise between its speed and its consumption memory. By treating on a hierarchical basis consumption memories of all the orders, one can locate that which dimensions calculation. With configuration of

11 A strategy to determine the point of IC operation exactly for JEVEUX is to increase the value gradually of MEM_{ASTK} The value of JE_{IC} must then increase. As soon as it does not move any more, it is that one reached the point of optimal operation allowing JEVEUX to function completely in IC: JE'_{IC} .

calculation fixed (many processors), plus the memory capacity reserved for JEVEUX will be large, more calculation will be In-Core (IC) and thus more it will be fast. One can also, according to the contingencies of the queues batch, to cut out his calculation into different CONTINUATION so as to blend and thus optimize the parameter settings “time calculation/memory”.

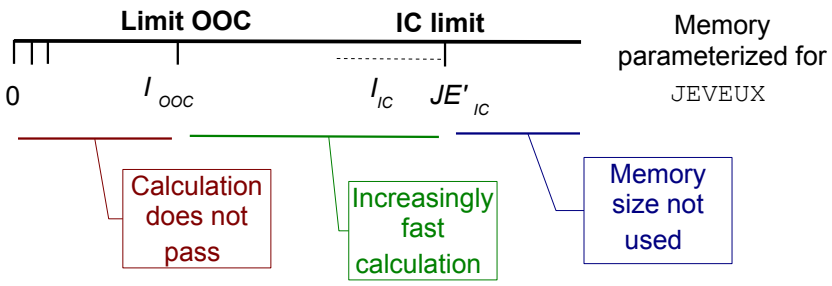

Figure 5.2-c : Significance of postings related to JEVEUX in the file message.

In the course of calculation, **JEVEUX can discharge most of the objects from RAM on the disc**. This mechanism is:

- Automatic (there is no more enough RAM space to create an object or to repatriate it),
- Managed by the programmer (e.g. call to this mechanism of release right before yielding the hand to MUMPS).

The statistics concerning this mechanism are recapitulated at the end of the file message (cf. §5.2-d) for the whole of the orders. Obviously, more this mechanism intervenes (and on large released volumes of memory), more calculation is some slowed down (time system and elapsed which grow and stable time CPU). This phenomenon can be accentuated in parallel mode (applications simultaneous memories) according to the distribution of the processors on the nodes of calculation and according to the characteristics of the machine.

```
STATISTICS CONCERNING THE DYNAMIC ALLOCATION:
     MAXIMUM SIZE CUMULEE            :                 1604  Mo.
     SIZE CUMULEE LIBEREE            :                52117  Mo.
     FULL NUMBER OF ALLOWANCES        :              1088126
     FULL NUMBER OF RELEASES       :               1088126
     CALLS TO THE MECHANISM OF RELEASE:                    1
     SIZE MEMORY CUMULEE RECUPEREE  :                 1226  Mo.
     VOLUME OF THE READINGS               :                0  Mo.
     VOLUME OF THE WRITINGS              :                1352  Mo.

   MEMORY JEVEUX MINIMAL NECESSARY FOR THE EXECUTION:     248.27 Mo  I_OOC
     - IMPOSE MANY ACCESSES DISC
     - THE SPEED OF EXECUTION SLOWS DOWN
   MEMORY JEVEUX OPTIMAL NECESSARY FOR THE EXECUTION:    1603.77 Mo  I_IC
     - LIMIT THE ACCESSES DISC
     - IMPROVE THE SPEED OF EXECUTION
```

Figure 5.2-d : Statistics of the mechanism JEVEUX of release (.mess).

These statistics also recapitulate estimated JE_{IC} and JE_{OOC} detailed previously.

Note:

One traces the possible overcosts in times of these unloadings at the end of the operators concerned (cf. §4.1.1). In the case of a lack of RAM, one can diagnose this one via the parameter ' VmPeak'(cf. §5.4).

5.3 Consumption memory of external products (MUMPS)

For the moment, the only external product of which **consumption RAM memory is really dimensioning, is the product MUMPS** . One can trace his consumption (In-Core and Out-Of-Core) in two manners:

- By **a precalculation** very fast and inexpensive in memory *via* mode GESTION_MEMOIRE=' EVAL' (cf. § 5.1). But they are only estimates (values ME_{IC} and ME_{OOC} of figure 5.1-a). They

can thus be a little pessimistic (especially into Out-Of-Core and/or in parallel). In parallel mode, one traces only estimated the greediest process MPI.

• By **standard calculation** (limited if possible within walking distance of time or with some clean modes) by adding the keyword INFO=2 in the supposed order most expensive. One then recapitulates in the file message (cf 5.3-a), not only estimated the memory ME_{IC} and ME_{OOC} the preceding ones (columns 'ESTIM IN-CORE' and 'ESTIM OUT-OF-CORE'), but also the real consumption of the selected mode $MR_{IC/OOC}$ (here in mode OOC because posting 'RESOL OUT-OF-CORE'). For example, in the figure 5.3-a, it is necessary to read: "MUMPS estimate to need at least $ME_{OOC}=478\text{Mo}$ by processor to function in OOC. To pass in IC (faster calculation), it estimates to need at least $ME_{IC}=964\text{Mo}$; In practice, it consumed exactly $MR_{OOC}=478\text{Mo}$ in OOC (estimated is thus perfect: $ME_{OOC}=MR_{OOC}$!).
In parallel mode, one recapitulates the figures by process MPI, like their minima, maximum and average. It is a posting with vocation of expert testimony which is more detailed than the posting of the mode GESTION_MEMOIRE=' EVAL'. In addition to this information of RAM consumption, he recapitulates also elements dependent on the type of problem, his digital difficulty (errors) and on his parallel distribution (balance of load).

```
*****************************************************************************
 <MONITORING DMUMPS 4.8.4
 SIZE OF THE SYSTEM    210131
 ROW     NO. MESHS     NO. TERMS K     LU FACTORS
 NR    0:            5575          685783        45091154
 NR    1:            8310         1067329        51704129
 NR    2:           11207         1383976        53788943
                                     ...
 NR    7:            8273         1039085         4560659
 ------------------------------------------------------------
TOTAL   :           67200         8377971       256459966

  MEMORY RAM ESTIMEE AND NECESSARY              BY MO MUMPS(FAC_NUM + RESOL)
 ROW ASTER: ESTIM IN-CORE  | ESTIM OUT-OF-CORE | RESOL. OUT-OF-CORE
 NR    0:            869 |              478 |              478
 NR    1:            816 |              439 |              439
 NR    2:            866 |              437 |              437
                                     ...
 NR    7:            754 |              339 |              339
 ------------------------------------------------------------------------------
AVERAGE :        5.92D+02        3.50D+02        3.50D+02
 ------------------------------------------------------------------------------
MINIMUM :        7.54D+02        3.39D+02        3.39D+02
 ------------------------------------------------------------------------------
MAXIMUM :        9.64D+02        4.78D+02        4.78D+02
 ------------------------------------------------------------------------------
*****************************************************************************
```

Figure 5.3-a : Consumption memories by processor (estimates and realized) of the external product MUMPS (posting in .mess if INFO=2).

Note:

According to the type of calculation (sequential, parallel centralized or distributed), the number of processors, the mode of use of JEVEUX (keyword SOLVEUR/MATR_DISTRIBUEE) and of the manager memory MUMPS (keyword SOLVEUR/GESTION_MEMOIRE), the hierarchy memory can however be hustled . In distributed parallel mode, the peak report JEVEUX will drop with the number of processors as soon as MATR_DISTRIBUE will be activated. That also will be the case for MUMPS, some is the mode of parallelism (especially in IC). Moreover, the passage of MUMPS IC mode with mode OOC also drastically will make fall its peak report.
Posting in INFO=2 RAM consumption of MUMPS by processor informs as for possible déséquilibrages of load memory. One can try to limit them by modifying the heuristics of scheduling of the product: parameters SOLVEUR/PRETRAITEMENTS and RENUM or the number of processors.

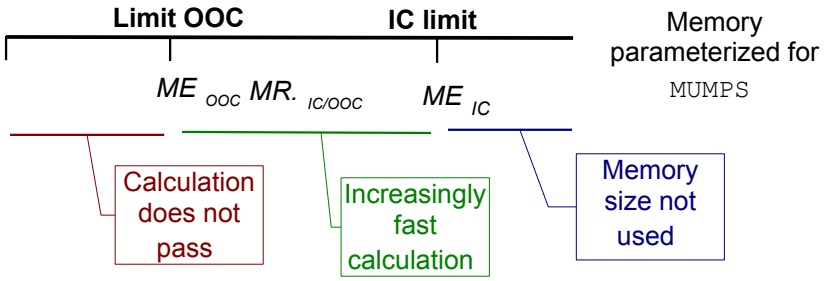

Figure 5.3-b : Significance of postings related to MUMPS in the file message.

5.4 Consumption system

Only consumption systems mentioned in the file message relates to the total memory of the job at this moment of calculation (VmSize) and its peak since the beginning (VmPeak). It is traced at the conclusion of each order (cf figures SYS_1/SYS_2 figure 5.2a). VmPeak is recapitulated at the end of the file message (cf appears 5.4a).

```
MAXIMUM of MEMORY UTILISEE BY the PROCESS          :    15107.89 Mo
   - MEMORY CONSOMMEE BY JEVEUX UNDERSTANDS,
     THE SUPERVISORY PYTHON, EXTERNAL BOOKSTORES
```

SYS 1

Figure 5.4-a : Final Vmpeak of the Unix process.

Rapidly VmPeak inform as for the peak in the face memory caused by all the achievable credits of the job *Aster*. It should be taken care that this figure remains lower than the physical memory available to the job. If not, the system goes, up to a certain point, “swapper” and that will slow down calculation. On the level of the software package JEVEUX that will result in more unloadings (cf. §4.1.2 and §5.2) and, on the level of the external products, by a variation ELAPS/USER growing (traced at the end of the order cf. §4.1.1).

•

Remarks if SOLVEUR=' MUMPS':

- *Since the restitution of the keyword SOLVEUR/GESTION_MEMOIRE (starting from the version v11.2 of Code_Aster), one allows the solvor “to be spread out” in memory, in complement of the allowances JEVEUX /Python/librairies, until occupying the allocated maximum size. That makes it possible the product to be faster and less pernickety on its report needs in term for swivelling. On the other hand, corollary of this strategy of optimum occupation of the resources memory, it VmPeak displayed in the file message becomes depend on the memory allocated with calculation (MEM ASTK or class batch limits). It cannot thus be useful, like before, to gauge consumption minimal memories of the study. With this intention, it is rather necessary to apply the strategy detailed to § 5.1*

6 Some advices to optimize the performances

One formulates here **some advices to help the user to benefit from the traced diagnoses** in the file message. But it is necessary well to be conscious that there is no universal receipt to optimize the total performances of a calculation. That depends on the type of study, of the aspects software and hardware of the machine, even of its load!

The parameter setting by default and postings/alarms of the code propose a balanced and instrumented operation. But, to be sure as well as possible to have used the capacities of its machine, L'user must remain attentive with the elements described in this document as to the advices which swarm in documentations with the orders.

One lists below and, in a nonexhaustive way, several questions which it is interesting to be posed when one seeks to optimize the performances of his calculation. Of course, certain questions (and answers) are cumulative and can thus apply simultaneously.

6.1 Concerning the characteristics of the problem

Within sight of the elements of §3, one can formulate two empirical rules:

- More **size of the problem** (N) and/or **filling of the matrix** (NNZ) increase, more construction, and especially, the resolution of the linear system will be expensive (CPU/RAM).
- Increase in **proportions of Lagrange** (N_L/N) can make more difficult the resolution of the linear system (execution time, quality of the solution).
- Size of the problem **dimension the maximum number of processors** that it is relevant to devote to its parallel calculation: a granularity from at least 20.10^3 degrees of freedom per process MPI is necessary.

6.2 Concerning spent times

To decrease times CPU, the user *Aster* have various tools:

- If **the main part of the costs relates to elementary calculations/assemblies and/or the resolutions of systems linear** (cf. §4.1) one advises to use *Aster* in parallel mode. It is then preferable to use the linear solvor MUMPS as a distributed mode MPI or the solvor MULT_FRONT in OpenMP. The first strategy also makes it possible cause a drop in RAM consumption by processor.
- If one **use already the linear solvor MUMPS,** one can disable his features of OOC[12] (GESTION_MEMOIRE=' IN_CORE') and of improvement of the quality of the solution (RESI_RELA=-1.d0). If the matrix is well conditioned and/or nonsymmetrical, one can also test parameters of relieving of the linear solvor (FILTRAGE_MATRICE, MIXER_PRECISION, SYME).
- If one carries out **a nonlinear calculation** one can test various parameters of relieving of the nonlinear solvor (REAC_INCR, REAC_ITER, SYME).
- If one is carried out **modal calculation**, one advises the use of method IRAM (METHODE=' SORENSEN') and the cutting of the spectrum searched in several frequential bands (*via* the operator CALC_MODES with OPTION=' BANDE' cut out in several sub-bands).

12 That can be interesting when one notes overcosts of I/O in the stages 1.3/1.4 *via* a time SYST considerable and a time ELAPS much higher than time TO USE.

- In a general way, more **operating process of JEVEUX (and MUMPS) is in IC** (cf. §5.1), more calculation is fast. These profits very important are however not compared with those which get the parallelism and the choice of a linear solvor (with the parameter setting) adapted.

For each stage of calculation, one **must normally have a weak time system** (SYST) and an office plurality "time CPU+temps system" (CPU+SYST) very near to the real latency (ELAPS). If it is not the case, two classical cases can arise:

- **Time USER+SYS is much higher than time ELAPS**. Calculation uses certainly OpenMP parallelism of the parallel strategies 1c or 2a described in Doc. U2 on parallelism and with §2.2 of Doc. U4.50.01. This situation is not worrying (and even desired), the principal one being that time back from calculation is low possible!

- **Time ELAPS is much higher than time CPU and/or time SYS is important.** Calculation probably suffers from applications memories (swap system, I/O RAM/disque…).
  ✔ **Track n°1:** This overcost can come from **total unloadings of JEVEUX**. To be convinced some it is enough to go to read, in the end of .mess, statistics concerning the dynamic allocation (cf. §5.2) or consumption in time of various unloadings (cf. §4.1). The more there were calls to the mechanisms of release and/or large released objects, the more times SYST and ELAPS will degrade themselves. **A solution is then to increase the size memory reserved for JEVEUX**.

  ✔ **Track n°2:** Corollary of the preceding report, parallel mode MPI tends has to multiply by ten the overcosts which had with unloadings. Indeed, the distribution of the data induced by parallelism will decrease the size of the objects JEVEUX (if SOLVEUR/MATR_DISTRIBUEE is activated) and thus to limit the impact of unloadings. *A contrario*, those are likely to be carried out at the same time and on processors contiguous. **A palliative solution** can then consist in "wasting" processor, **by interlacing active processors MPI of processors door frames** (e.g. value ncpus of*Astk* initialized with 2).

  ✔ **Track n°3***:* If the linear solvor is used MUMPS in OOC, the problem can come from one **large many descent-increase** product (cf stage 1.4 of §4.1). One can limit them by disconnecting the automatic option of refinement (RESI_RELA=-1.d0) or while passing by again in IC mode (if the resources memories allow it).

6.3 Concerning the RAM memory consumed

- If consumption JE_{IC} and JE_{OOC} is close (a few tens of for hundred), it is that JEVEUX often had to function in mode OOC. There were probably many unloadings report (cf. §4.1/5.2). The computing time can suffer from it (especially in parallel). It is necessary to try to add memory or to increase the number of processor (if the option MATR_DISTRIBUEE is activated).

- **calculation is often dimensioned in memory**, by the maximum, on the greediest operator, between the value floor of JEVEUX (JE_{OOC}) and the value used by the linear solvor MUMPS (if it is used). To decrease this figure one can exploit several factors:
  ✔ **If one uses MUMPS and that this last is prevalent** (it is often the case): to use more processors in MPI (parameters mpi_nbcpu/mpi_nbnoeud of*Astk*), to pass in OOC (keyword SOLVEUR/GESTION_MEMOIRE). If the matrix is well conditioned and/or nonsymmetrical, one can also test parameters of relieving of the linear solvor (FILTRAGE_MATRICE, MIXER_PRECISION, SYME).
  ✔ **If one uses MUMPS and that this last is not prevalent:** to use the distribution of objects JEVEUX option MATR_DISTRIBUEE in parallel mode.
  ✔ If one is used **another linear solvor**: to use MUMPS in parallel even GCPC/PETSC (into sequential/parallel).

6.4 Concerning parallelism

- If **the main part of the costs relates to elementary calculations/assemblies and/or the resolutions of systems linear** (cf. §4.1) one advises to use *Aster* in parallel mode. It is then preferable to use the linear solvor MUMPS as a distributed mode MPI (cf Doc. U2 on parallelism or U4.50.01) or the solvor MULT_FRONT in OpenMP.

- It is preferable to limit the parallelism of calculation only to some expensive operators (in time/memory): STAT/DYNA_NON_LINE, CALC_MODES … And thus, if possible, to cut out this one in a succession of pre/sequential postprocessings and parallel calculations. During long calculations (a few days), this strategy makes it possible moreover to be better guarded against possible stops in error by saving the base associated with each great stage with calculation.

- It is interesting of **to validate**, as a preliminary, **its parallel calculation** by comparing some iterations in sequential mode and parallel mode. Moreover, this approach also allows **to gauge the profits atteignables maxima** (speed-up theoretical) and thus to avoid "wasting processors too much". Thus, if one notes f the parallel portion of the code (given for example *via* run sequential precondition), then theoretical speed-up $\mathbf{S}_p$ maximum accessible on p processors is calculated according to the formula of Amdhal (cf [R6.01.03] §2.4):

  $$\mathbf{S}_p = \frac{1}{1-f+f/p}.$$

  For example, if one uses the parallelism distributed by default (scenarios 1b+2b, cf Doc. U2 on parallelism) and that stages 1.3/1.4 and 2. (cf. §4.1) 90% of sequential time represent ($f=0.90$), the theoretical speed-up is limited to the value $\mathbf{S}_\infty = \frac{1}{1-0.9+0.9/\infty} = 10$! And this, some is the number of allocated processes MPI.

- To optimize its parallel calculation, the possible ones should be supervised **imbalances of loads** flood of data (CPU and memory) and to limit the which had overcosts **with report unloadings** (JEVEUX and MUMPS OOC) (cf. §4.1)and with **filings of fields** (cf. §4.3). To gain in time calculation, it is also necessary to proscribe any procedure of retassage memory (order END keyword RETASSAGE) who is against-productive in parallel.

- **recourse to parallelism MPI** of MUMPS allows to gain in time CPU (on the paralleled stages) and in RAM memory: thanks to the distribution of data of JEVEUX (if the option MATR_DISTRIBUEE is activated) and, especially, thanks to that of the objects MUMPS.

- **Some empirical figures:** one advises to allocate at least 20,000 degrees of freedom per process MPI; A standard thermomechanical calculation generally profits, on 32 processors, of a profit about ten in time elapsed and a factor 4 in RAM memory.