Note of use of the linear solveurs

1 Goal

linear solveurs are in fact omnipresent in the unfolding of the operators of *Code_Aster* because they are often hidden with deepest of other digital algorithms: nonlinear diagram, integration in time, analyze modal etc They often consume of it the major part of time CPU and the memory. The choice and the complete parameter setting of the necessary linear solvor are carried out *via* the keyword factor SOLVEUR. It is present in most orders of calculation (STAT_NON_LINE, THER_LINEAIRE, CALC_MODES etc.).
This **note of use is complementary to user's documentation** keyword SOLVEUR[U4.50.01]. She wants to be an intermediate link between “the simple” theoretical documentation of keyword and that (with each solvor a reference material is leaned).

The first part of this note gives one **overall vision** various linear solveurs available, of their perimeter of use and their average performances in term of robustness and CPU/mémoire consumption.
The following chapter gathers some **advices** to use the parameters as well as possible of SOLVEUR according to the cases: size of the system, its digital properties, consumption CPU and memory… To finish, one details certain parameters of SOLVEUR in order to help the user in one **advanced use** of this functionality of the code.
related problems of improvement of performances (RAM/CPU) of a calculation and use of parallelism are also briefly approached. They are the object of specific notes, respectively, [U1.03.03] and [U2.08.06].

Contents

2 Various linear solveurs available

These **linear solveurs are in fact omnipresent in the unfolding of the operators** of *Code_Aster* because they are often hidden with deepest of other digital algorithms: nonlinear diagram, integration in time, analyze modal etc They often consume of it the major part of time CPU and the memory. The choice and the complete parameter setting of the necessary linear solvor are carried out *via* the keyword factor SOLVEUR. It is present in most orders of calculation (STAT_NON_LINE, THER_LINEAIRE, CALC_MODES...).

This keyword makes it possible to choose between the two classes of solveurs: the direct ones and the iterative ones. Concerning **direct**, one has the classical algorithm of "Gauss" (SOLVEUR/METHODE=' LDLT'), of a factorization multifrontale ('MULT_FRONT') and of external ('MUMPS'). For **iterative**, it is possible to call on a combined gradient ('GCPC') or with certain tools of the public bookstore PETSc ('PETSC').

Only MULT_FRONT, MUMPS, PETSC are **paralleled**. The first in OpenMP, two others in MPI and OpenMP (hybrid parallelism). But all the solveurs are compatible with a parallel treatment (*via* MPI) of elementary calculations and the assemblies. And this, that these treatments are initiated right before the use of the linear solvor itself, or, in another operator (for example the pre one/postprocessings).

Let us detail a little the operation of each one of them:

Direct Solveurs

/'MUMPS'

Direct Solvor of multifrontale type with swivelling. This solvor is obtained by calling it **external product MUMPS** developed by CERFACS/IRIT/INRIA/CNRS (cf Copyright §4). Matric storage except solvor is MORSE. As starter of the solvor, one makes conversion with the internal format of MUMPS: i, j, K_{ij}, centralized or distributed.

For *Code_Aster*, **its principal interest lies in its capacity to swivel** lines and/or columns of the matrix during factorization in the event of small pivot. This possibility is useful (even essential) for the models leading to positive nondefinite matrices (except boundary conditions); For example, "mixed" elements having degrees of freedom of the type "Lagrange" (incompressible elements…).

This **method is paralleled in distributed memory** (MPI) and in **shared memory** (OpenMP). It can be carried out on several processors (*via* small the Astk interface `Options/Optionsdelancement/ncpus&mpi_nbcpu&mpi_nbnoeud`) .

In parallel MPI (`mpi_nbcpu&mpi_nbnoeud`)**, MUMPS distributes its data naturally** (matrix, factorized…) between the hearts of the various allocated nodes of calculation. What accelerates calculations largely and makes it possible to reduce the occupation memory necessary, by process MPI, to launch calculation. This RAM consumption can be reduced even more *via* parameters `GESTION_MEMOIRE` or `RENUM` MUMPS.

In term of memory, the bottleneck can then be on the level of space `JEVEUX`. To reduce this last, one can distribute the matrix *Aster* (`MATR_ASSE`) *via* the option `MATR_DISTRIBUEE`.

MUMPS also rests on one **second level of overlapping parallelism** within parallelism MPI (**hybrid parallelism**) and based on OpenMP. This second level of parallelism is mainly activated in calls to the subjacent mathematical bookstores (BLAS, LAPACK). Contrary to the MPI, it is a parallelism with shared memory and thus limited to the only hearts of nodes of calculation (`ncpus`).

To accelerate the large studies (NR at least > 2.10^6 ddls) one can moreover activate options of acceleration/compression of MUMPS (starting from the v5.1.0) *via* keywords `ACCELERATION/LOW_RANK_SEUIL`.

/'MULT_FRONT'

Direct Solvor of multifrontale type developed in-house EDF R & D. Matric storage is MORSE (or 'CSC' for 'Compressed Sparse Column') and thus proscribed any swivelling. This method is paralleled in shared memory (OpenMP) and can be carried out on several processors (*via* small the Astk interface `Options/Options of launching/ncpus`). The initial matrix is stored in only one object `JEVEUX` and its factorized is distributed on several, therefore it can be discharged partially and automatically on disc.

/'LDLT' **Direct Solvor with factorization of Crout** by blocks (without swivelling) **developed in-house EDF R & D**. Matric storage except solvor is MORSE. As starter of the solvor, one makes conversion with the internal format of LDLT: 'line of sky' ('SKYLINE'). There is a pagination of the completely skeletal memory (the matrix broken up into blocks is managed in memory and is independently discharged on disc progressively) which makes it possible to pass from large case but which is paid by accesses expensive discs.
Moreover, this solvor allows to factorize only partially the matrix. This possibility is "historical". It makes it possible to factorize the matrix in several "times" (several work) to even modify with the flight the last lines of this factorized. Today, one does not imagine well the interest of this functionality except for certain methods (known as discrete) of contact-friction where one has, intentionally, placed in the last lines of the matrix the terms concerning the nodes likely to be in contact. Thus, as iterations of pairing, the relations between these nodes changing, one erases then recomputes that these last contributions of factorized. It is a typical example where the astute use of an algorithm enough frustrates little to bring major profits (in time).

Iterative Solveurs

/'GCPC' **Iterative Solvor of standard gradient combined with pre-packaging $ILU(k)$ or based on one factorized single precision (*via* MUMPS).**
The storage of the matrix is then MORSE. With incomplete Cholesky, the factorized initial matrix and its incomplete are stored, each one, in only one object JEVEUX.
With factorized the single precision, the preconditionnor is much more expensive (in CPU/RAM), but it is carried out in single precision and its calculation can be pooled during several resolutions (problem of the type multiples second members, e.g. STAT_NON_LINE).

/'PETSC' **Iterative Solveurs resulting from the external bookstore PETSc** (Argonne Laboratory). Matric storage except solvor is MORSE. As starter of the solvor, one makes conversion with the internal format of PETSc: 'CSR' for 'Compressed Sparse Row'. PETSc allocates contiguous blocks of lines step processor. This **method is paralleled in distributed memory** (MPI) and can be carried out on several processors (*via* small the Astk interface Options Options of launching /mpi_nbcpu &mpi_nbnoeud).
When PETSc uses a preconditionnor based on MUMPS (PRECOND='LDLT_SP'), it also profits from the potentially hybrid parallelism of the direct solvor. But best accelerations of PETSc occur rather by privileging the first level of parallelism, MPI.

Caution: solveurs PETSC and MUMPS being incompatible into sequential, one generally privileges MUMPS. To use PETSC, it is thus necessary often to launch a parallel version of *Code_Aster* (since it is necessary to requesting one processor).

	Solveur perimeter	Robustness	CPU	Memory	Details
Direct					
MULT_FRONT	Universal Solvor. To disadvise for modelings requiring swivelling (mixed EF of	+++	Séq: ++ //: ++ (speed-up~2)	+ OOC[1]	On 4/8 hearts.

	Solveur perimeter	Robustness	CPU	Memory	Details
	X-FEM, incompressible…).				
MUMPS	Universal Solvor, solvor of reference	+++	Séq: ++ //: +++ (sp~30)	-- IC + OOC	To 512 hearts
LDLT	Solvor universal (but very slow on large case). To disadvise for modelings requiring swivelling (mixed EF of X-FEM, incompressible…) . Possible partial factorization (discrete contact).	+++	Séq: +	+ OOC	Rather small cases or cases of mean size which can draw part of partial factorization.
Iterative					
GCPC	Real symmetrical problems except those requiring obligatorily a detection of singularity (modal calculation, buckling).	- (LDLT_INC) + (LDLT_SP)	Séq: ++ +	+++ (LDLT_INC) ++ (LDLT_SP)	With LDLT_INC not too much not to increase it level of pre-packaging. Sometimes very effective (thermics…) especially into nonlinear with LDLT_SP.
PETSC	Idem GCPC but compatible with the nonsymmetrical one.	- (LDLT_INC) + (LDLT_SP)	Séq: ++ + //: +++ (sp~4)	++ IC	Rather robust algorithms: GMRES. Often very effective into nonlinear with LDLT_SP.

Figure 2. - 1: Synoptic of the linear solveurs available in Code_Aster.

Note:

- *To be completely exhaustive, one can specify that some (rare) digital operations are operated with a parameter setting "fixed" solvor. This one is thus inaccessible to the users (without overload software). Among those, one finds certain resolutions of the discrete methods of contact-friction (reserved for LDLT), the search for rigid modes into modal, calculations of modes of interfaces (reserved for MUMPS or with LDLT)… Each time functional reasons or of performance explain this not very transparent choice.*

1 OOC for 'Out-Of-Core'. I.e. one will release from the RAM memory by discharging on disc part of the objects. That makes it possible to compensate for the swap system and to deal with problems much larger. According to the algorithmic one, these accesses additional discs can be penalizing. The opposite way of managing is "it In-Core" (IC). All the data-processing objects remain in RAM. That limits the size of the accessible problems (with the swap system near) but privileges speed.

3 Some advices of use

If one wishes[2] to change linear solvor or to adapt his parameter setting, several questions must be approached: Which is the type of problem which one wishes to solve? What are they the digital properties of the linear systems met? etc.
One lists below and, in a nonexhaustive way, several questions which it is interesting to be posed when one seeks to optimize the aspects linear solveurs. Of course, certain questions (and answers) are cumulative and can thus apply simultaneously.

In short:

The method by default remains the internal multifrontale MULT_FRONT. But for fully profiting from profits CPU and RAM which gets it **parallelism**, or to solve one **problem numerically often difficult** (X-FEM, incompressibility, THM), one recommends the use of the external product **MUMPS**. The more the problem is of big size, the more one advises to use parallelism MPI and the technique of acceleration/compression (ACCELERATION/LOW_RANK_SEUIL).
If, despite everything, on a given data-processing platform, the problem does not pass in memory, resorts it to the iterative solveurs of PETSC can be a solution (if their functional perimeters allows it). Let us note that by default they use in "writing pad" MUMPS as a preconditionnor (option PRE_COND=' LDLT_SP').
To go further in **savings of it space memory** , one can also degrade the pre-packaging (calculation will be probably longer) by choosing one of the other préconditionneurs of PETSC ('LDLT_INC'…).

In **non-linear, to gain in time** , one can also exploit several parameters of relieving (SYME in nonsymmetrical) or interactions "nonlinear solvor/linear solvor" (REAC_PRECOND, NEWTON_KRYLOV.).
For nonlinear problems **conditioned well** (thermics…), the recourse to one MUMPS "released" (MIXER_PRECISION/FILTRAGE_MATRICE) can bring very significant profits report. In the same way, into linear as into nonlinear, with PETSC without preconditionnor (PRE_COND=' SANS').

For more details and advices on the employment of the linear solveurs one will be able to consult the note of use [U4.50.01] and the associated reference materials [R6…]. The related problems of improvement of performances (RAM/CPU) of a calculation and, use of parallelism, are also the object of detailed notes: [U1.03.03] and [U2.08.06].

Which is the type of problem to be solved?

- **Resolution of many systems with the same matrix** (problem of the multiples type second members[3]) ⇒ solveurs MULT_FRONT or MUMPS (if possible while disabling RESI_RELA and with GESTION_MEMOIRE=' IN_CORE').
- **Standard parametric calculation into linear** ⇒ a first test with MUMPS by leaving the parameters by default, then all others runs while disconnecting RESI_RELA or with POSTTRAITEMENTS=' MINI'.
- **Parametric calculation into nonlinear with a well conditioned matrix** ⇒ a first test with MUMPS while exploiting the parameters of relieving (FILTRAGE_MATRICE/MIXER_PRECISION or SYME if one is in nonsymmetrical). Then, if a point of optimized operation (consumption CPU/RAM) were elucidated, to use it for all the others runs.
  One can also test MUMPS but, this time, as preconditionnor single precision of PETSC/GCPC (LDLT_SP). The interest is then to reactualize it only periodically (REAC_PRECOND).

Which are its digital properties?

2 Or if, quite simply, it is necessary to exploit this parameter because calculation does not pass on the selected software platform, or with consumption in times and memory incompatible with the constraints of the study.
3 It is the case, for example, in thermomechanical chaining when the characteristic materials do not depend on the temperature or, into nonlinear, when the tangent matrix is seldom reactualized.

- **Linear system conditioned good**[4]($<10^4$) ⇒ solveurs MUMPS+MIXER_PRECISION or GCPC/PETSC.
- **Difficult linear system** (bad conditioning, mixed finite elements, prevalences of Lagranges…) ⇒ solvor MUMPS.

Does one need a very precise solution?

- One can be satisfied with one **very approximated solution**[5] ⇒ solveurs GCPC or PETSC with one RESI_RELA=10^{-3} or MUMPS with LOW_RANK_SEUIL<10^{-9} and POSTTRAITEMENTS='SANS'.
- One is wanted **precise solution** or, at least, a diagnosis on its quality and the digital difficulties of the system to be solved ⇒ solvor MUMPS while activating NPREC and RESI_RELA (in INFO=2).

How to optimize consumption time/RAM memory of the linear solvor?

- **Time** ⇒ solvor MUMPS by disabling the OOC (GESTION_MEMORE=' IN_CORE') even RESI_RELA. Parallel calculation hybrid MPI/OpenMP (cf Doc. U2 on parallelism). On the problems of big size use of compressions low_rank and accelerations *via* keywords ACCELERATION/LOW_RANK_SEUIL.
- **Memory** ⇒ solvor MUMPS while activating GESTION_MEMOIRE='OUT_OF_CORE'/MATR_DISTRIBUE and in distributed parallel mode. Or iterative solvor of Krylov type: GCPC/PETSC+LDLT_SP.
  Into nonlinear, if the matrix is well conditioned and/or nonsymmetrical, one can also exploit the parameters of relieving of MUMPS (FILTRAGE_MATRICE, MIXER_PRECISION and SYME) or to use an iterative solvor by reducing the cost of the pre-packaging. One will find more details in the discussion of §7.2.6.
- **Memory** ⇒ solvor PETSC while activating MATR_DISTRIBUE and in distributed parallel mode.

This is a problem border of very big size (> 5.10^6 degrees of freedom) ?

- **Robust Solvor** ⇒ solvor MUMPS with optimizations preceding memories and compressions low-rank (ACCELERATION/LOW_RANK_SEUIL).
- **Solveurs of the "last chance"** ⇒ iterative solveurs (with a low level of pre-packaging).

How to optimize the total performances of my calculation?

- Note of use [U1.03.03].

How to carry out, gauge and optimize a parallel calculation?

- Note of use [U2.08.06].

4 Either the typology of calculation informs about this information (it is known for example that it is often the case of thermics), or it in addition is obtained (for example, by making a preliminary test with MUMPS and by activating the keyword RESI_RELA+INFO=2).

5 Nonlinear calculations convex, prospective calculation or calculation whose quality of the mechanical solutions of interest is controlled in addition.

4 Recommendations on the external products

To solve the many linear systems that it produces, *Code_Aster* can be pressed on external products. This paper documents the use of these products within the framework of *Code_Aster* while referring to a use and a classical installation.

The recommended versions are:

- **Renuméroteurs/partitionneurs** MONGREL 5.1.0/PARMETIS 4.0.3 and (Pt)SCOTCH TAPE 6.0.4.
- **Direct Solvor and preconditionnor:** MUMPS 5.0.2/5.1.0 (public versions) and 5.0.2/5.1.0consortium (versions in restricted access).
  versions consortium in restricted access are installed by default on the machines of calculation *Aster* (with possible corrective patchs and dedicated procedures of optimization). Their uses are limited here to the internal studies of engineering at EDF. Their source codes cannot be redistributed outside EDF. They potentially get an access to exploratory features in phase lead compared to the public versions (cf keywords ACCELERATION/LOW_RANK_SEUIL).
- **Iterative Solveurs and préconditionneurs:** PETSc 3.7.3.

For MUMPS 5.1.0 Copyright is the following (licence CeCILL-C V1):

```
 Copyright 1991-2017 CERFACS, CNRS, ENS Lyon, INP Toulouse, Inria,
 University of Bordeaux.

 This version of MUMPS is provided to you free of load. It is
 released under the CeCILL-C license,
 http://www.cecill.info/licences/Licence_CeCILL-C_V1-en.html,
 except for the external and optional ordering PORD,
 in separate directory PORD, which is public domain (see PORD/README).

 You edge acknowledge (using references [1] and [2]) the contribution of
 this package in any scientific publication depends upon the uses of
 the package. Please uses reasonable endeavours to notify the authors
 of the package of this publication.

  [1] P.R. Amestoy, I.S. Duff, J. Koster and  J. there. The Excellent one,
  With fully asynchronous multifrontal solver using distributed dynamic
  scheduling, SIAM Journal of Matrix Analysis and Applications,
  Flight 23, No 1, pp 15-41 (2001).

  [2] P.R. Amestoy, A. Guermouche, J. there. Excellent and
  S. Pralet, Hybrid scheduling for the parallel solution of linear
  systems. Parallel Computing Flight 32 (2), pp 136-156 (2006).

 Ace has counterpart to the access to the source codes and rights to Copy,
 modify and redistribute granted by the license, users are provided only
 with has limited warranty  and the software' S author,  the holder of the
 economic rights,  and the successive licensors  cut only  limited
 liability.

 In this respect, the user' S attention is drawn to the risks associated
 with loading,  using,  modifying and/or developing gold reproducing the
 software by to use in light of its specific status of free software,
 that may mean  that it is complicated to manipulate,  and  that  also
 therefore means  that it is reserved for developers  and  experienced
 professionals having in-depth computer knowledge. Users are therefore
  encouraged to load and test the software' S suitability ace glances to
their
 requirements in conditions enabling the security of their systems and/or
 to Be ensured and dated,  more generally, to uses and operate it in the
 same conditions ace glances security.

 The fact that you are presently reading this means that you cuts had
 knowledge of the CeCILL-C license and that you accept its terms.
```

For MUMPS 5.1.0consortium, Copyright is the following:

```
 Copyright 1991-2017 CERFACS, CNRS, ENS Lyon, INP Toulouse, Inria,
University of Bordeaux.

This version of MUMPS (the software) is not public and should Be
considered confidential. It is reserved to the members of the MUMPS
consortium. You should suppress it and any Copy you might cuts
obtained yew you are not has member of the MUMPS consortium.

Ace defined in the membership agreement, you, ace has member, are
granted access to this version of the MUMPS software with has free
not-exclusive license limited to the length of your membership
and with limited redistribution conditions: should you wish to
redistribute this not-public version, it shall only Be allowed
to C so in Object Codes forms.

THIS MATERIAL IS PROVIDED ACE IS, WITH ABSOLUTELY NO WARRANTY
EXPRESSED GOLD IMPLIED.  ANY USES IS AT YOUR OWN RISK.
```

For PETSc 3.7.3, Copyright is the following (licence BSD clause n°2)

```
Copyright (c) 1991-2016, UChicago Argonne, LLC and the PETSc TEAM Development
All rights reserved.

Redistribution and uses in source and binary forms, with gold without
modification,
are permitted provided that the following conditions are puts:

* Redistributions of source codes must retain the above copyright note, this
  list of conditions and the following disclaimer.
* Redistributions in binary forms must reproduce the above copyright note,
this
  list of conditions and the following disclaimer in the documentation and/or
  other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “ACE IS”
AND
FAST ANY GOLD IMPLIED WARRANTIES, INCLUDING, GOAL NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR WITH PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO VENT SHALL THE COPYRIGHT HOLDER GOLD CONTRIBUTORS BE LIABLE
FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, GOLD CONSEQUENTIAL
RAMMINGS
(INCLUDING, GOAL NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS GOLD
SERVICES;
LOSS OF USES, DATED, BUT PROFITS; HOWEVER BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ONE
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, GOLD WRONG
(INCLUDING NEGLIGENCE GOLD OTHERWISE) ARISING IN OUT ANY WAY OF THE USES OF
THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH RAMMING.
```

MONGREL is distributed under Apache licence v2.0 available under http://www.apache.org.

SCOTCH TAPE is distributed under licence CeCILL-C V1 available under http://www.cecill.info .

5 Links with parallelism

Foot-note: for more information one will be able to consult the note of dedicated use to parallelism [U2.08.06].

5.1 General information

Often a simulation *Code_Aster* can **to profit from important profits of performance** by distributing its calculations on several hearts of a PC or one or more nodes of a centralized machine.
One can **to gain in time** (with parallelism MPI and OpenMP parallelism) as in **memory** (only *via* MPI). These profits are variable according to the requested features, their parameter settings, the data file and the software platform used: cf figure 5.1.1.

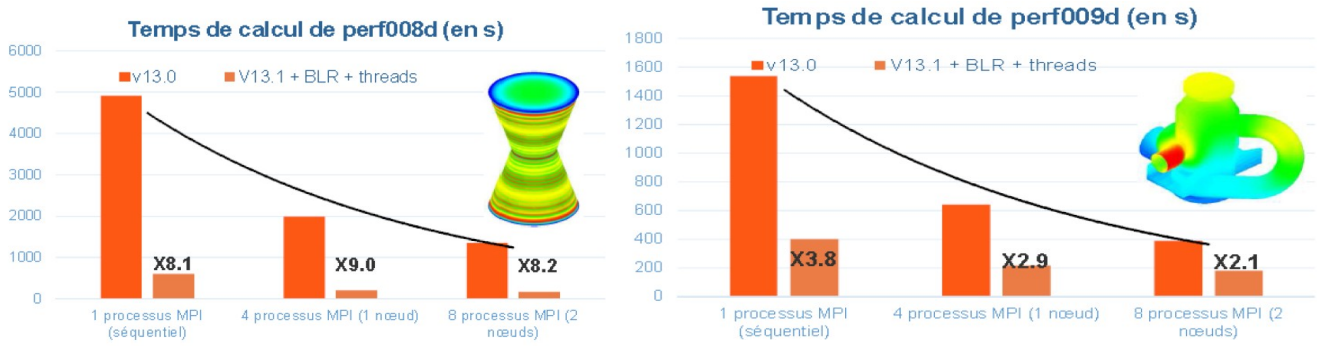


Figure 5.1.1. _Example of savings of time gotten by parallelism MPI of Code_Aster v13.0, and with that hybrid, MPI+OpenMP (+ compressions low-rank cf [U4.50.01]) of Code_Aster v13.1. Comparisons carried out on the CAS-tests of performance perf008d and perf009d and on the centralized machine Aster5.

In *Code_Aster,* by default, calculation is sequential. But one can activate **various strategies of parallelization**. Those depend on the stage of calculation considered and the selected parameter setting. They are often cumulative or chaînables.
There are three big classes of problems parallélisables, the second being most current:

- that is to say simulation can be organized in several **independent subcalculations** (cf. § 5.2),
- either it is not the case but:
  - this one remains **dominated by or not linear linear calculations** (operators `STAT/DYNA/THER_NON_LINE`, `MECA_STATIQUE`... cf. §5.3),
  - this one remains **dominated by modal calculations** divisible underfrequential bands (`INFO_MODE/CALC_MODES+' BANDE'`, cf. §5.4).

To have one **estimate of the time spent by an operator** and thus of the prevalent stages of a calculation, one can activate the keyword `MESURE_TEMPS` orders `DEBUT/POURSUITE` (cf [U1.03.03]) on a standard study (possibly shortened or edulcorated).
In all the cases, one will advise **to divide largest calculations into various stages** in order to separate those purely **calculative**[6], those concerning of **postings**, **postprocessings** and of **handling of fields**[7].

5.2 Independent calculations

6 Possibly of various types (n°2 category or n°3 quoted previously) and which will gain with being carried out in parallel.
7 Who will be often faster into sequential because of the risks of cloggings at the time of the accesses report.

When simulation can be organized in various subcalculations *Aster* independent (cf figure 5.2.1), the tool Astk [U1.04.00] proposes an adapted functionality (cf [U2.08.07]). This one distributes these subcalculations on various resources machine and recovers their results. It is a parallel diagram "data-processing" completion.
Limit being, for the moment, that all these unit calculations must be able sequentially to be carried out each one on the selected machine (what can sometimes pose problems of resources memory, cf [U4.50.01]).

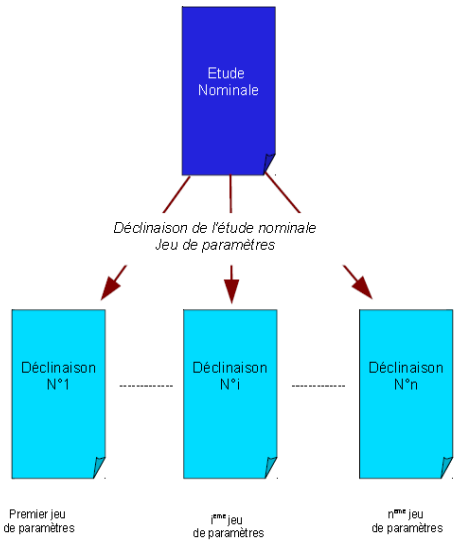


Figure 5.2.1. _: Parallelism of independent calculations.

5.3 Parallelization of the linear systems

When this simulation cannot break up into subcalculations *Aster* similar and independent, but that it remains **dominated** nevertheless **by linear calculations** or **nonlinear** (operators STAT/DYNA/THER_NON_LINE , MECA_STATIQUE ... cf. § 5.3), one can organize a specific parallel diagram.
It is founded on **distribution of the tasks** and of **structures of data** implied in handling of **linear systems** . Because they are these stages of construction and resolution of linear systems which are often more soliciting in computing times and resources memory. They are present in most operators because they are hidden with deepest other algorithms "plus trades": nonlinear solvor, modal and vibratory calculation, diagram in time…

The first stage of the parallel diagram relates to the distribution of the finite elements of the model on all processes MPI. Each process MPI will thus manage only the treatments and the data associated with the elements of which it has the load. The construction of the linear systems in *Code_Aster* (elementary calculations, assemblies) is some then accelerated. One often speaks about " **parallelism in space** ". It is a parallel diagram rather of a "data-processing" nature.

Once these built portions of linear system (cf figure 5.3.1), two cases arise:

- that is to say it **following treatment is naturally sequential** and thus all processes MPI must have access to total information. With this intention these ends of linear systems are gathered and thus the following stage neither will be accelerated, nor will not see lowering its consumption memory. It is generally of an end of operator, a postprocessing or a linear solvor not paralleled in MPI (MULT_FRONT, LDLT, GCPC).
- that is to say it **following treatment accepts parallelism MPI**, it acts then mainly linear solveurs HPC MUMPS and PETSC. The parallel flood of data builds upstream is then transmitted to them (after some adaptations). These packages of linear algebra reorganize then, in-house, their own parallel diagrams (with a algebraic vision). One speaks then about parallel diagram of a "digital" nature rather. This combination "data-processing parallelism", on the level of assembly of the linear system, and, "digital parallelism", on the level from its resolution, the 2 *via* MPI, is the most current combination.

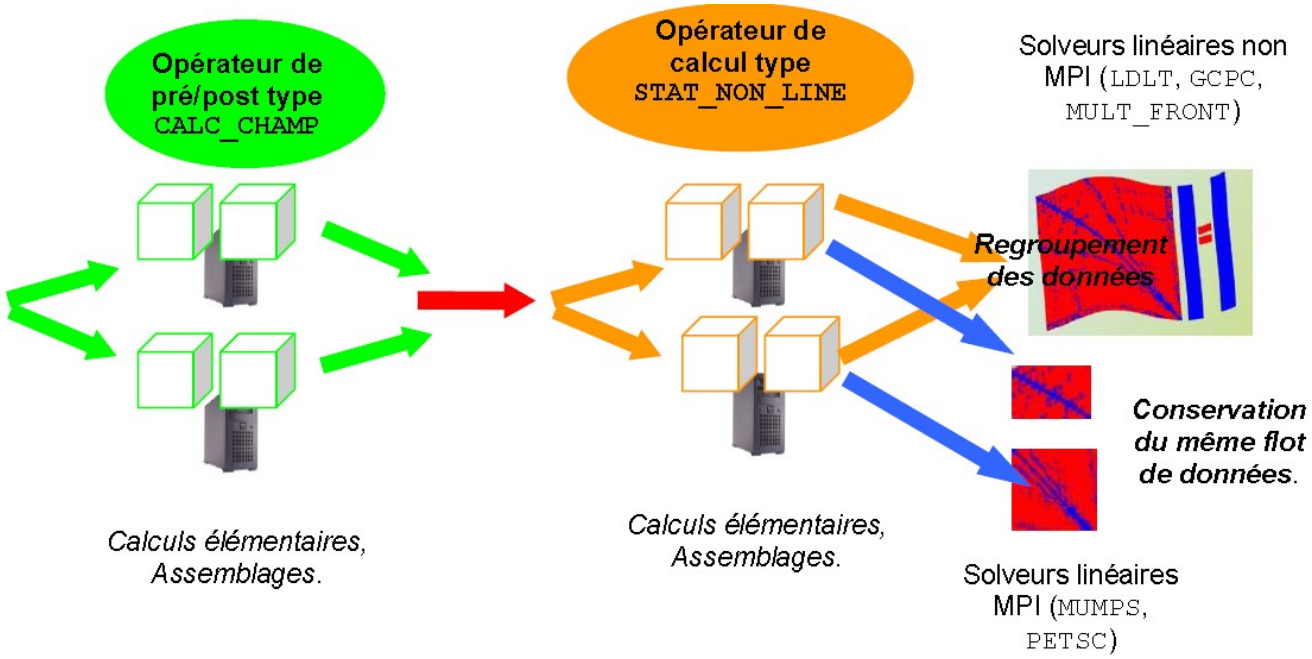


Figure 5.3.1. _Organization of parallel diagram MPI of construction and resolution of the linear systems.

Note:

- *Let us note that at the conclusion of the cycle "construction of system linear – resolution of this one", some is the scenario implemented (sequential linear solvor or parallel MPI), the vector solution is then transmitted, in entirety, with all processes MPI. The cycle can thus continue some is the following configuration.*

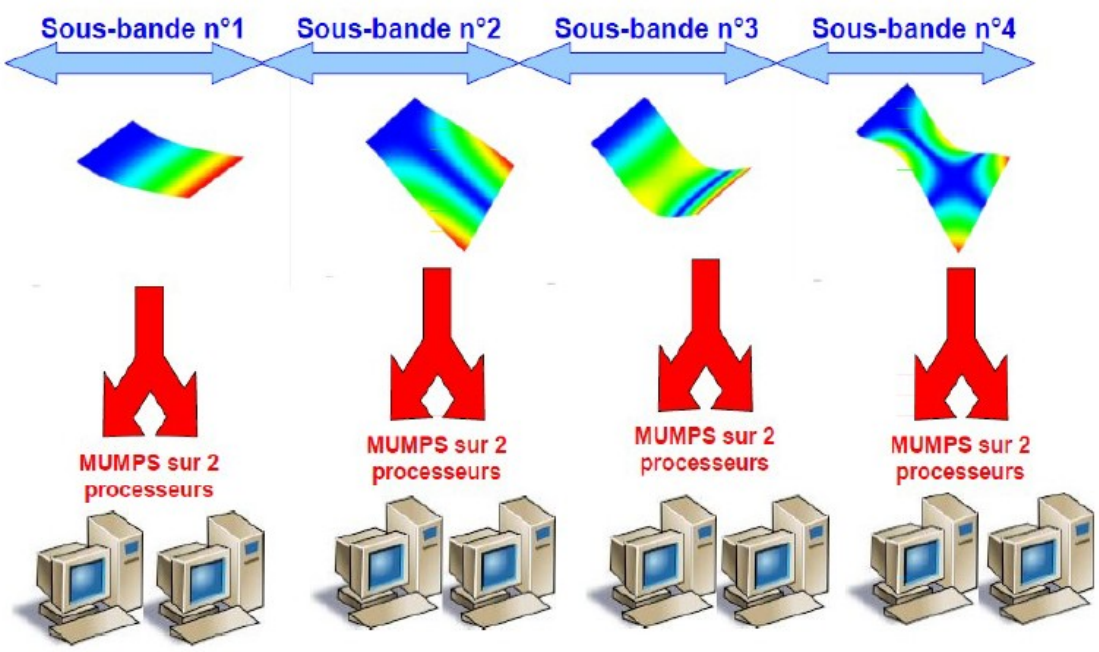
Moreover, one can **to superimpose or substitute for this parallelism MPI** (which functions on all the platforms), another level of parallelism managed this time by **OpenMP language**. This one is however limited to the fractions of machine sharing the same memory physically (PC multi-hearts or nodes of waiter of calculation).
It does not make it possible to lower consumption memory but on the other hand it accelerates certain types of calculation and this, with a granularity lower than that of the MPI: it gets a better acceleration even if the flood of data/treatments is not very important. It is a parallel diagram of a "data-processing" nature which intervenes mainly in the basic operations of algorithms of linear algebra (*via* for example bookstore BLAS and certain stages of solvor MUMPS).

This parallelism can be:

- that is to say cumulated with parallelism MPI of MUMPS by accelerating calculations within each process MPI. One obtains a hybrid diagram then parallel with 2 levels.
- that is to say to replace parallelism MPI by accelerating the resolution of system linear with MULT_FRONT.

5.4 Distribution of modal calculations

When simulation cannot break up into calculations *Aster* independent, but that it remains **dominated** nevertheless **by generalized modal calculations** (operators INFO_MODE and CALC_MODES), one can organize a specific parallel diagram (cf 5.4.1).



FigRUE 5.4.1. _Organization of parallel diagram MPI of distribution of modal calculations and resolutions of the associated linear systems.

It is founded on **distribution of modal calculations** independent: each one being in charge of a sub-band frequential .
This parallel diagram of a “data-processing” nature purely gets only savings of time (except if one it office plurality with MUMPS). It can however be mixed with the preceding parallel diagrams:

- **chaining** between various operators: parallelism MPI of construction of linear matrices (in for example CALC_MATR_ELEM) and modal resolution in CALC_MODES.
- **office plurality** , within CALC_MODES , by activating parallelism MPI (even OpenMP) of the direct solvor MUMPS . One obtains a hybrid diagram then parallel to 2 or 3 levels.

6 Indicators of performance of a calculation

For more information one will be able to consult the note of use [U1.03.03]: 'Indicating of performance of a calculation (time/memory)'.

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

- Characteristics of calculation (many nodes, of equations, Lagrange, size of the matrix…),
- Memories JEVEUX floor (to pass into Out-Of-Core) and optimal (to pass in In-Core),
- Memory required by certain external products (e.g. MUMPS),
- Times CPU, system and "user" (elapsed),
- The ventilation of spent time following the stages of calculation (elementary calculation, assembly, resolution of the linear system, unloading on disc).

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.

7 Additional information on the keyword

Significance of the various parameters of the keyword SOLVEUR are the object of the user's documentation [U4.50.01]. This one must be synthetic to help the user in a standard use of the code. For a more advanced use some additional information can prove extremely useful. This chapter recapitulates these elements.

7.1 Detection of singularity and keyword NPREC/STOP_SINGULIER/RESI_RELA

The crucial step of the direct solveurs (SOLVEUR=_F (METHODE=' LDLT'/'MULT_FRONT'/'MUMPS')) in term of consumption. However it can stumble in two cases: problem of construction of factorized (structurally or numerically singular matrix) and digital detection of a singularity (more significant approximated process). The behavior of the code will depend on the case, of the parameter setting of NPREC/STOP_SINGULIER/RESI_RELA and of the solvor used. The combinative one of the cases is described in the table attached.

Standard type of solvor/of problem	**Construction of factorized** *In the event of problem that does occur?*	**Digital detection of singularity (S).** *In the event of singularity that does occur?*
LDLT/MULT_FRONT	Stop in ERREUR_FATALE	**Case n°1:** One factorizes a dynamic matrix in an operator of dynamics (cf R5.01.01 § 3.8, CALC_MODES …): Stop in ERREUR_FATALE if it is the matrix of work of the algorithm. Emission of one ALARM if it is about a stage of the test of Sturm. In these two cases STOP_SINGULIER no incidence has on the process and NPREC must be positive. **Case n°2:** If STOP_SINGULIER=' OUI', Stop in ERREUR_FATALE in the phase of postprocessing of the linear solvor. **Case n°3:** If STOP_SINGULIER=' NOT', Emission of one ALARM in the phase of postprocessing of the linear solvor. Spotentially vague olution step detected by the linear solvor. Except a possible including process (Newton…) there is no digital parapet to guarantee the quality of the solution. **Case n°4:** If STOP_SINGULIER=' CUTS OUT', □ Launching of processus of division step of time. One rebuilds a new problem for another increment of time/loading.
MUMPS	Stop in ERREUR_FATALE	**Case n°1/4:** If NPREC> 0, even behavior that LDLT / MULT_FRONT (one finds the cases n°1 to 4) .

Standard type of solvor/of problem	Construction of factorized *In the event of problem that does occur?*	Digital detection of singularity (S). *In the event of singularity that does occur?*
		Case n°5: If NPREC<0 and RESI_RELA>0, Detection of disabled singularity but one measures the quality of the solution in the linear solvor. If the system is singular, its conditioning will be very high and the quality of resolution will be very bad. If this quality of resolution is higher than the value parameterized in RESI_RELA: stop in ERREUR_FATALE. **Case n°6:** If NPREC<0 and RESI_RELA<0, Detection of singularity and measurement of the quality of the solution both disabled. Except a possible including process (Newton…) there is no digital parapet to guarantee the quality of the solution.

Table 7.1-1. Behavior of the code, according to the parameter setting, when digital factorization detects problems (bad setting in data, digital instabilities, strong conditioning…).

One compares some digital details of the two types of criteria of detection (LDLT/MULT_FRONT *versus* MUMPS) of singularity in the table below:

Characteristics/Standard of solvor	LDLT/MULT_FRONT	MUMPS
Criterion	Room with each degree of freedom (relative value)	Total for all them degrees of freedom
Term tested	Absolute value of the diagonal term of each line	Infinite standard of the line/column of the line corresponding to the pivot
Detection of the number of line (ISINGU in the messages)	Always	Yes except at the time of problems with construction of factorized
Supply amongst lost decimals	Yes, except at the time of problems with construction of factorized	Not
Désactivable	Not	Yes

Table 7.1-2. Differences in the processes of detection of singularity according to the solveurs.

However, beyond the differences in implementations and of the error messages (such solvor points a degree of freedom, such other solvor another degree of freedom), two classes of direct solveurs, LDLT/MULT_FRONT and MUMPS, generally conclude with the same type from diagnoses in the event of problems[8].
They point a defective setting in data: blockings redundant or, on the contrary, absent; superabundant linear relations due to contact-friction; very heterogeneous numerical data (term of penalization too large) or illicit (negative Young modulus…).

Note:

8 Cf CAS-tests erreu03a and erreu04a.

- *In the worst case, it is necessary to adjust the value of NPREC (to increase or decrease by 1) to lead to the same report. In general, the singularities are so obvious, that the parameter setting by default is appropriate completely.*
- *Contrary to the two other solveurs, MUMPS do not specify the number of lost decimals, on the other hand the activation of its quality standard (for example RESI_RELA=1.10 -6) constitute an ultimate effective parapet against this kind of hitch.*
- *It normal and is assumed that it degree of freedom detected either sometimes different when a digital parameter is changed (solvor, renumerotor, pretreatments…) or data processing (parallelism…). All depends on the order in which each processor treats the unknown factors of which it has the load and of the techniques of swivelling/balancing possibly put in work. In general, even different, the results contribute to the same diagnosis: to re-examine blockings of its setting in data.*
- *To obtain matric conditioning[9] of his "rough" operator (i.e. without the possible pretreatments operated by the linear solvor), one can use the combination MUMPS+PRETRAITEMENTS=' NON'+INFO=2+NPREC<0+RESI_RELA>0 . A value very important (> 10 12) the presence of at least a singularity betrays then. On the other hand, if the overcost calculation of the detection of singularity is painless, that of the estimate of the conditioning and the quality of the solution is less (up to 40% in time elapsed; cf. § 7.2.5).*

To be more precise, there are 9 distinct cases listed in the table below. They are based on exact or approached nullity[10] terms "pivots" (cf [R6.02.03] §2.3) selected by the digital phase of factorization of the direct solvor considered. The first case appears during digital factorization itself (numerically or structurally singular matrix). The second case comes from the phase of postprocessing activated at the conclusion of this digital factorization.

Firstly the user must take the advice lavished by the message of alarm (listed in the table below). If that is not really enough, the advanced user can try to exploit the digital parameters of the solvor (renumerotor…), even on the criterion NPREC when it is about a pivot almost no one.

Type of problem	Information available	The Councils
Matrix not being based on a grid. Null pivot	Number of line (if MF/LDLT).	Setting in data (limiting conditions, characteristic materials…). To test MUMPS (if MF/LDLT).
Matrix not being based on a grid. Pivot almost no one.	Number of line, Many lost decimals (if MF/LDLT).	Idem.
The pivot is one degree of freedom physics except X-FEM. Null pivot.	Number of component line/node/(if MF/LDLT).	Setting in data (limiting conditions, characteristic materials…). To test MUMPS (if MF/LDLT).
The pivot is one degree of freedom physics except X-FEM. Pivot almost no one.	Number of component line/node/, Many lost decimals (if MF/LDLT)	Rigid mode of body blocked evil (defect of blocking). If calculation comprises contact it is not necessary that the structure "holds" only by the relations of contact.
The pivot is one degree of freedom physics X-FEM. Pivot almost no one.	Number of component line/node/(if MF/LDLT).	The level-set (crack) passes very close to the node considered (increase NPREC until 10).
The pivot is Lagrange related to a linear relation enters degrees of	Number of line (if MF/LDLT).	Linear relations enters degrees of freedom superabundant (CONNECTION, contact…).

9 With the direction resolution of hollow linear system, cf papers of M.Arioli/J.Demmel/I.Duff.
10 Controlled by NPREC (cf [U4.50.01] §3.2).

Type of problem	Information available	The Councils
freedom. Null pivot.		Setting in data (limiting conditions, characteristic materials…). To test MUMPS (if MF/LDLT).
The pivot is dependent Lagrange with a linear relation between degrees of freedom. Pivot almost no one	Number of line, Many lost decimals (if MF/LDLT).	Linear relations enters degrees of freedom superabundant (CONNECTION, contact…).
The pivot is one degree of freedom of Lagrange related to one blocking of a degree of freedom. Null pivot	Number of line (if MF/LDLT). Blocking concerned.	Superabundant blocking. Setting in data (limiting conditions, characteristic materials…). To test MUMPS (if MF/LDLT).
The pivot is a degree of freedom of Lagrange related to a blocking of a degree of freedom. Pivot almost no one	Number of line. Blocking concerned. Many lost decimals (if MF/LDLT).	Superabundant blocking.

Table 7.1-3. Various cases of detection of singularity and associated advices.

7.2 Solveur MUMPS (METHODE=' MUMPS')

7.2.1 General information

solvor MUMPS currently developed by **CNRS/INPT-IRIT/INRIA/CERFACS** is one **direct solvor of multifrontal type, paralleled** (in MPI) and robust, because it makes it possible to swivel the lines and columns of the matrix during digital factorization.
MUMPS provides an estimate of the quality of the solution $\mathbf{u}$ (cf keyword RESI_RELA) matric problem $\mathbf{K}\,\mathbf{u}=\mathbf{f}$ *via* concepts of*direct error relative* ('relative forward error') and of error *opposite* ('backward error'). This 'backward error', $\eta(\mathbf{K},\mathbf{f})$, the behavior of the algorithm of resolution measures (when all is well, this reality is close to the precision machine, that is to say 10^{-15} in double precision). MUMPS calculates also an estimate of the conditioning of the matrix, $\kappa(\mathbf{K})$, which translates the good behavior of the problem to be solved (real ranging between 10^4 for a problem conditioned well up to 10^{20} for very badly conditioned). The product of both is one raising of the relative error on the solution ('relative forward error'):

$$\frac{\|\delta u\|}{\|u\|} < C^{st}\cdot\kappa(\mathbf{K})\cdot\eta(\mathbf{K},\mathbf{f})$$

By specifying a strictly positive value with the keyword RESI_RELA (e.g. 10^{-6}), the user indicates that it wishes to test the validity of the solution of each linear system solved by MUMPS with the ell of this value. If the product $\kappa(\mathbf{K})\cdot\eta(\mathbf{K},\mathbf{f})$ is higher than RESI_RELA the code stops in ERREUR_FATALE, by specifying the nature of the problem and the values accused. With posting INFO=2, one details each term of the product: $\eta(\mathbf{K},\mathbf{f})$ and $\kappa(\mathbf{K})$.

To continue calculation, one can then:

- **To increase the tolerance of RESI_RELA**. For the badly conditioned problems, a tolerance of 10^{-3} is not rare. But it must be taken with serious because this kind of pathology can seriously disturb a calculation (cf notices following on conditioning and §3.4).
- If **it is the 'backward error'** who is too important: it is advised to modify the algorithm of resolution. I.e., in our case, to exploit the parameters of launching of MUMPS (TYPE_RESOL, PRETREATMENTS…).
- If **it is the conditioning of the operator** who is in question, it is advised to balance the terms of the matrix, apart from MUMPS or *via* MUMPS (PRETREATMENTS= 'YES'), or to change the formulation of the problem.

Note:

- *Even within the very precise framework of the resolution of system linear, there exists in many ways to define the sensitivity to the rounding errors of the problem considered (i.e. its conditioning). That retained by MUMPS and, which refers in the field (cf Arioli, Demmel and Duff 1989), is indissociable 'backward error' of the problem. The definition of the one does not have a direction without that of the other. One thus should not confuse this kind of conditioning with the concept of matric conditioning classical.*
  In addition, conditioning provided not MUMPS takes into account the SECOND MEMBER of the system as well as the HOLLOW CHARACTER of the matrix. Indeed, it is not worthwhile to take account of possible rounding errors on worthless matric terms and thus not provided to the solvor! The degrees of freedom corresponding "do not speak each other" (seen spyglass finite element). Thus, this conditioning MUMPS respects the physique of the discretized problem. It does not dip back the problem in the too rich space of the full matrices.
  ***Thus, the figure of conditioning displayed by MUMPS is much less pessimistic than the standard calculation which another product can provide** (Matlab, Python…). But let us hammer, that it is only its product with the 'backward error', called 'forward error', which has an interest. And only, within the framework of a resolution of system linear via MUMPS.*

7.2.2 Perimeter of use

It is a universal linear solvor. He is deployed for all the features of *Code_Aster.*
In addition, the use of solvor suffers from small not very frequent limitations and which one can circumvent easily if necessary.

- In mode CONTINUATION one saves on file only objects FORTRAN77 of*Aster* and thus not external occurrences of products (MUMPS, PETSc). Thus attention with the use of the orders burst within this framework (NUME_DDL/FACTORISER/RESOUDRE). With MUMPS, it is not possible of TO FACTORIZE or of TO SOLVE a linear system built at the time of a run preceding Aster (with NUME_DDL).
- In the same way one limits the number of occurrence simultaneous of MUMPS (and PETSc) to NMXINS=5. During the construction of its problem *via* burst orders, the user must take care not to exceed this figure.

7.2.3 Parameter RENUM

This keyword makes it possible to control the tool used to renumber the linear system[11]. The user *Aster* can choose various tools divided into two families: the tools "frustrate" dedicated to a use and provided with MUMPS ('AMD','MFA','QAMD','PORD'), and, "rich' libraries more and more "sophisticated" that it is necessary to install separately (MONGREL 'PARMETIS'/'', 'PTSCOTCH'/'SCOTCH TAPE').

choice of the renumerotor a great importance has on consumption memory and time of the linear solvor. If one seeks with **to optimize/regulate the digital parameters** bound to the linear solvor, this parameter must be **one of the first to be tested**.

Product MUMPS breaks up its calculations into three stages (cf [R6.02.03] §1.6): phase of analysis, digital factorization and descent-increase. In certain cases, **the stage of analysis can prove to be prevalent**. Shears because the problem is numerically difficult (many blockings, connections or zones of contact, incompressible elements…), that is to say because the two other stages were very reduced thanks to parallelism (cf. [U2.08.06]). It can be then interesting to parallel this stage of analysis (*via* MPI). That makes it possible to accelerate it and lower its consumption memory. This is carried out by choosing one of the parallel renuméroteurs proposed: 'PARMETIS' or 'PTSCOTCH'.
The choice of such a parallel renumerotor, if it often improves the performances of the stage of analysis of MUMPS, can nevertheless degrade those of two other following stages MUMPS. Nevertheless, if the principal problem were to reduce the concommation memory of this stage of analysis or if the following stages of MUMPS profit from sufficient parallelism (or compression, cf parameters ACCELERATION/LOW_RANK_SEUIL), the assessment can be overall positive.

In addition, at the time of **parallel modal calculations** (operator CALC_MODES), one sometimes observed of speed-UPS disappointing because of an inappropriate choice of renumerotor. In this case there one noted that the choice of a "sophisticated" renumerotor was against-performing. It is to better impose on MUMPS simple 'MFA'or'QAMD', rather than'MONGREL'or'PARMETIS'(often taken automatically in mode'CAR').

7.2.4 Parameter ELIM_LAGR2

Historically, direct linear solveurs of *Code_Aster* ('MULT_FRONT' and 'LDLT') did not have D' algorithm swivelling (which seeks to avoid accumulations of rounding errors per division by very small terms). To circumvent this problem, the taking into account of the limiting conditions by of Lagranges (AFFE_CHAR_MECA/THER…) was modified by introducing Lagranges doubles. Formally, one does not work with the initial matrix $\mathbf{K}_0$

$$\mathbf{K}_0=\begin{bmatrix} \mathbf{K} & \text{blocage} \\ \text{blocage} & \mathbf{0} \end{bmatrix}_{\text{lagr}}^{\mathbf{u}}$$

but with its doubly dualized form $\mathbf{K}_2$

11 So, in particular, limiting the phenomenon of filling of factorized, cf [R6.02.03] §1.7.

$$\mathbf{K}_2=\begin{bmatrix} \mathbf{K} & \text{blocage} & \text{blocage} \\ \text{blocage} & \mathbf{-1} & \mathbf{1} \\ \text{blocage} & \mathbf{1} & \mathbf{-1} \end{bmatrix} \begin{matrix} \mathbf{u} \\ \text{lagr}_1 \\ \text{lagr}_2 \end{matrix}$$

From where a overcost report and calculation.
Like MUMPS have faculties of swivelling, this choice of dualisation of the limiting conditions can be called into question. By initializing this keyword with 'YES', **one does not take any more account but of one Lagrange, the other being spectator**[12]. From where a matrix of work $\mathbf{K}_1$ simply dualized

$$\mathbf{K}_1=\begin{bmatrix} \mathbf{K} & \text{blocage} & \mathbf{0} \\ \text{blocage} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{-1} \end{bmatrix} \begin{matrix} \mathbf{u} \\ \text{lagr}_1 \\ \text{lagr}_2 \end{matrix}$$

smaller because the extra-diagonal terms of the lines and the columns associated with these Lagranges spectators are then initialized to zero. *A contrario*, with the value 'NOT', MUMPS receives the usual dualized matrices.

For **problems comprising of many Lagranges (up to 20% of the numbers of total unknown factors)**, the activation of this parameter is often paying (smaller matrix). But when **this number explodes (>20%)**, this perhaps against-productive process. The profits carried out on the matrix are cancelled by the size of factorized and especially by the number of late swivellings that MUMPS must carry out. To impose ELIM_LAGR2=' NON' can be then very interesting (for example: profit of 40% in CPU on the CAS-test mac3c01).
One **disconnect** also temporarily this parameter when one wishes to calculate **the determinant of the matrix**, because if not its value is distorted by these modifications of the terms of blocking. The user is informed of this automatic modification of parameter setting by a message dedicated (visible in INFO=2 only).

7.2.5 Parameter RESI_RELA

Value by default=-1.d0 into nonlinear and modal, 1.d-6 into linear.
This parameter is disabled by a negative value.

By specifying a strictly positive value with this keyword (e.g. 10^{-6}), the user indicates that it wishes to test the validity of the solution of each linear system solved by MUMPS with the ell of this value.
This careful approach is advised when the solution is not it not even corrected by another algorithmic process (algorithm of Newton, detection of singularity…) in short in the linear operators THER_LINEAIRE and MECA_STATIQUE. Into nonlinear, the criterion of detection of singularity and the correction of Newton are sufficient parapets. One can thus disconnect this process of control (it is what is made by default *via* the value -1). Into modal, this detection of singularity is an algorithmic tool to capture the clean modes. This detection is the object of a specific parameter setting dedicated to each method.
If the relative error on the solution estimated by MUMPS is higher than resi the code stops in ERREUR_FATALE, by specifying the nature of the problem and the values accused.
The activation of this keyword initiates also a process of iterative refinement of which the objective is to improve the solution obtained. This postprocessing profits from a particular parameter setting (keyword POSTTRAITEMENTS). It is the solution resulting from this process of iterative improvement which is tested by RESI_RELA.

Note:

- *This process of control implies the estimate of matric conditioning and some descent-increase of the postprocessing of iterative refinement. It can thus be enough expensive, in particular in OOC, because of the I/O RAM/disque at the time of the descent-increase*

12 To maintain the coherence of the structures of data and to keep a certain legibility/data-processing maintainability, it is preferable "to bluff" the usual process while passing of $\mathbf{K}_2$ with $\mathbf{K}_1$, rather than with the optimal scenario $\mathbf{K}_0$.

(up to 40%). When sufficient parapets is put in work one can disconnect it while initializing `resi` with a negative value.

7.2.6 Parameters to optimize management memory (MUMPS and/or JEVEUX)

In general most of times calculation and peaks RAM reports of a simulation *Code_Aster* are ascribable to the linear solveurs. The linear solvor MUMPS do not escape the rule but the wealth from its internal parameter setting and its fine coupling with *Code_Aster* spare a certain flexibility with the user. In particular with regard to the management of consumption in RAM memory.

In fact, **at the time of a peak report** occurring for a resolution of system linear *via* MUMPS, the RAM memory can break up into 5 parts:

- Objects JEVEUX except the matrix,
- Objects JEVEUX associated with matrix (generally SD MATR_ASSE and NUME_DDL),
- Objects MUMPS allowing to store the matrix,
- Objects MUMPS allowing to store factorized,
- Objects MUMPS auxiliaries (pointer, vectors, buffers of communication…).

Coarsely, part 4 is most cumbersome. In particular, it is much larger than parts 2 and 3 (factor of at least 30 due to the phenomenon of filling cf [R6.02.03]). These last are equivalent cuts some but one is exerted in space reserved for JEVEUX, while the other, lives in the space complementary allocated by the manager of task to the job. As for the two other parts, 1 and 5, they often play a marginal part[13].

To decrease this consumption memory, it**user *Aster* have several arms of lever** (most of the time cumulable):

- **Parallel calculation centralized** on N hearts (part 4 divided by n) or distributed (parts 3 and 4 divided by n).
- **Distributed parallel calculation** + **MATR_DISTRIBUEE** (perimeter of use limited): parts 3 and 4 divided by n , part 2 divided by a little less n .
- The activation of**OOC of MUMPS** (keyword GESTION_MEMOIRE='OUT-OF-CORE'): part 4 decreases by the 2/3.
  Knowing that before each call to the cycle 'digital analyse+factorisation' of MUMPS, one systematically discharges on disc the largest objects JEVEUX part 2.

The Councils:
Within sight of the elements precedent, an obvious tactic consists in rather passing calculation in parallel mode distributed (value by default) than into sequential. Centralized parallel mode not bringing anything from this point of view[14], it is not to consider. If however, one remains dependent on a particular mode, here the advices associated with each one:

- **In sequential mode:** the principal profits will come from activation from GESTION_MEMOIRE=' OUT_OF_CORE' (on problems of reasonable size).
- **In distributed parallel mode:** passed about ten processors, itOUT_OF_CORE do not get much any more of profit. MATR_DISTRIBUEE can then help in certain situations.

If these strategies do not bring profits sufficient, one can also try into nonlinear (at the cost of possible losses of precision and/or time) to release the resolution of the linear system (FILTRAGE_MATRICE, MIXER_PRECISION) even those of the process including (tangent matrix rubber band, reduction of the space of projection into modal…). If the matrix is well conditioned and if one does not need to detect the singularities of the problem (thus not modal calculation, buckling…), one can also try an iterative solvor (GCPC/PETSC+LDLT_SP).

13 But not always. Thus part 1 can end up being expensive when one calculates much step of time, when one charges with the projected fields or when a law of behavior handles many internal variables. In the same way, part 5 can become considerable during a parallel calculation on a large number of processor.
14 It is mainly used to test and validate new developments as well as parallel studies.

Solution	Profit in RAM memory	Overcost in time	Loss of precision
Parallelism	+++	On the contrary, saving of time	None
GESTION_MEMOIRE= 'OUT_OF_CORE'	++	Weak except so many descent-increase	None
MATR_DISTRIBUEE (limited perimeter)	+	None	None
Relieving of the resolutions FILTRAGE_MATRICE, MIXER_PRECISION (limited perimeter)	++	Variable	Variable. Possibility of nonconvergence.
To change solvor: GCPC/PETSC+LDLT_SP (limited perimeter)	+++	Variable	Variable. Possibility of nonconvergence.

Table 7.2-1. Synoptic of the various solutions allowing to optimize the memory during a calculation with MUMPS.

Keyword GESTION_MEMOIRE= 'OUT_OF_CORE'
To activate or disable faculties OOC of MUMPS who then will discharge entirely on disc the real part of the blocks of factorized managed by each processor. This functionality is of course cumulable with parallelism, from where a larger variety of operation to adapt to the contingencies of execution. The OOC, just like parallelism, contribute to reduce the RAM memory required by processor. But of course (a little) to the detriment of time CPU: price to be paid for the I/O for one, communications MPI for the other.

Caution: During a parallel calculation, if the number of processors is important, the size of objects MUMPS déchargeables on disc becomes low. The passage in OOC can then prove to be against-productive (weak profit in RAM and overcost in time) compared to the IC mode parameterized by default.
This phenomenon occurs all the more precociously as the size of the problem is low and it is all the more sensitive as many descent-increase are carried out[15] in the solvor.
Roughly speaking[16], below 50.10^3 degrees of freedom by processors and if one has sufficient RAM (3 or 4 Go) by processor, one can undoubtedly rock in IC.

Note:

- *For the moment, during an execution MUMPS in OOC, only the vectors of real containing factorized (entirely) are discharged on disc. The vectors of entireties accompanying this structure of data (of size quite as important) do not profit yet from this mechanism. In addition, this unloading does not take place that after the phase of analysis of MUMPS. In short, on very large case (several million degrees of freedom), even with this OOC, of the contingencies memories can prevent calculation. One leaves in theory with one ERREUR_FATALE documented.*
- *In MUMPS, so as to optimize the occupation memory, the main part of the entireties is coded in INTEGER*4. Only the entireties corresponding at an address memory are transcribed in INTEGER*8. That makes it possible to address problems of bigger size, on architectures 64 bits. this cohabitation short/long entireties to optimize the place memory was extended to certain large objects JEVEUX.*
- *When Code_Aster finished assembling the matrix of work, before passing "the relay" with MUMPS, it discharges on disc the largest objects JEVEUX dependent on the resolution of the linear system (SMDI/HC, DEEQ, NUEAQ, VALM..). And this in order to leave the most RAM possible place to MUMPS. In mode GESTION_MEMOIRE='*

15 For example, in one STAT_NON_LINE with much of iterations of Newton and/or iterations of iterative refinement.
16 That depends much on the study.

AUTO', if this place memory is not sufficient so that MUMPS functions in IC, one supplements this release partial of objects JEVEUX by a general release of all the releasable objects (i.e. nonopen in read/write). This operation can get much profits when one "trails" in memory much objects JEVEUX peripherals (projection of fields, transitory length…). On the other hand, this massive unloading can make waste time. In particular in parallel mode because of cloggings of the access coeurs/RAM.

Keyword MATR_DISTRIBUEE

This parameter is usable in the operators MECA_STATIQUE, STAT_NON_LINE, DYNA_NON_LINE, THER_LINEAIRE and THER_NON_LINE with AFFE_CHAR_MECA or AFFE_CHAR_CINE. It active only in parallel is not distributed (AFFE_MODELE/ DISTRIBUTION METHOD equal other that **CENTRALIZE)**. This functionality is of course cumulable with GESTION_MEMOIRE='OUT_OF_CORE', from where a larger variety of operation to adapt to the contingencies of execution.

In parallel mode, when the data are distributed JEVEUX upstream of MUMPS, one redécoupe not inevitably structures of data concerned. With the option MATR_DISTRIBUEE=' NON', all the distributed objects are allocated and initialized with the same size (the same value as into sequential). On the other hand, each processor will modify only the parts of objects JEVEUX it has the load. This scenario is particularly adapted to the distributed parallel mode of MUMPS (by default mode) because this product gathers in-house these incomplete floods of data. Parallelism allows then, in addition to savings of time calculation, to reduce the place memory required by the resolution MUMPS but not that necessary to the construction of the problem in JEVEUX.

This is not awkward as long as RAM space for JEVEUX remain much lower than that necessary by MUMPS. Like JEVEUX store mainly the matrix and MUMPS, its factorized (generally of tens of larger time), the RAM bottleneck of calculation is theoretically on MUMPS. But as soon as one uses a few tens of processors in MPI and/or that the OOC is activated, as MUMPS distributes this factorized by processor and discharge these pieces on disc, the "ball returns in the camp of JEVEUX".

From where the option MATR_DISTRIBUEE who recuts the matrix, with just of the nonworthless terms for which the responsibility the processor has. Space JEVEUX required falls then with the number of processors and goes down below RAM necessary to MUMPS. The results of figure 7.2-1 illustrate this profit in parallel on two studies: a Pump LAUGH and the Epicure tank.

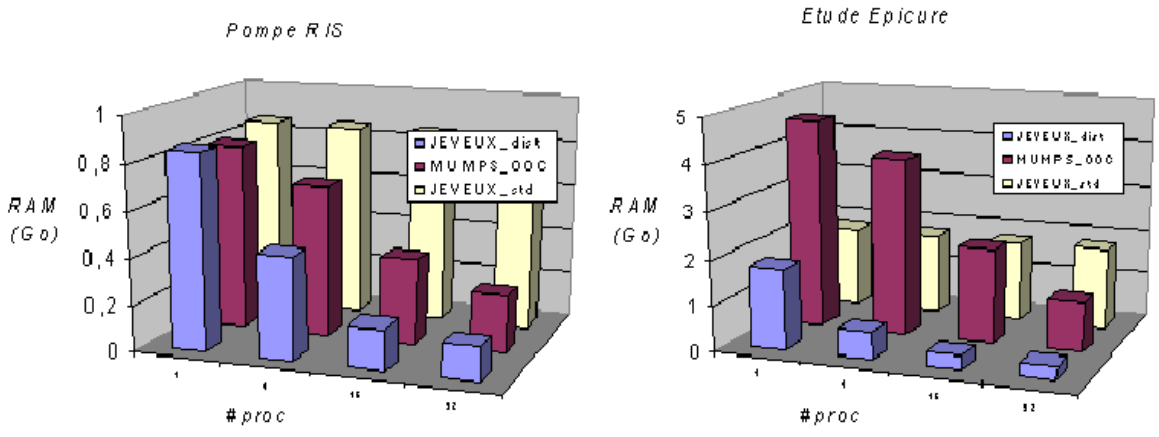


Figure 7.2-1: Evolution of RAM consumption (in Go) according to the number of processors, Code_Aster v11.0 (JEVEUX standard MATR_DISTRIBUE=' NON' and distributed, resp. 'YES') and of MUMPS OOC. Calculations carried out on a Pump LAUGH (perf009) and on the tank of the study Epicure (perf011).

Note:

- *One treats here data resulting from an elementary calculation (RESU_ELEM and CHAM_ELEM) or of a matric assembly (MATR_ASSE). Assembled vectors (CHAM_NO) are not distributed because the induced profits report would be weak and, in addition, as they intervene in the evaluation of many algorithmic criteria, that would imply too many additional communications.*
- *In mode MATR_DISTRIBUE, to make the joint the end enters of MATR_ASSE room with the processor and MATR_ASSE total (that one does not build), one adds a vector of indirection in the form of one NUME_DDL room.*

7.2.7 Parameters to reduce time calculation *via* various techniques of acceleration/compression

Keywords ACCELERATION and LOW_RANK_SEUIL

These two keywords activate and of techniques of acceleration/compression of MUMPS control . Those can significantly reduce the computing time of large studies, and this, without restriction of the perimeter of use, and, with potentially little or not of impact on the precision, the robustness and the total behavior of simulation.

They are generally interesting only on problems of big sizes (NR at least > 2.10 ^{6} ddls) . **noted profits** on some CAS-tests of performance and studies Aster vary **20% to 80%** (examples of figures 7.2-2/7.2-3). They increase with the size of the problem, its massive character and they are complementary to those gotten by parallelism and the renumerotor.

One supplements here the description of these features of the document [U4.50.01].

When one chose an acceleration based on compressions 'low-rank' (values 'LR'and'LR+'keyword ACCELERATION), the rate should be defined of these last. This rate is indicated by the keyword LOW_RANK_SEUIL . It controls the criterion of truncation of the digital algorithm of compression[17]. *Roughly speaking* , more this figure is large, for example 10 $^{-12}$ or 10 $^{-9}$, more compression will be important and thus more the savings of time can be interesting.
However, if this value is too large (for example >10 $^{-9}$), the factorized matrix can be too approximated and the vector solution to prove thus too vague! In a nonlinear process it is not always so serious because the algorithm of including Newton can correct the shooting!
On the other hand, into linear or to deal with numerically difficult problems (incompressible finite elements, X-FEM…), it should then be made sure that the resolution included well the procedure of post-treatments corrective (cf iterative refinement, keyword POSTTRAITEMENTS). In fact, in this case where when one seeks to establish a compromise between performance and precision, the value POSTTRAITEMENTS=' MINI' (+ RESI_RELA<0) is often to privilege compared to the by default choice (POSTTRAITEMENTS=' AUTO' + RESI_RELA>0).

To be exhaustive, let us note that the actual value of this keyword can also be worthless, compression will be done then except for the precision machine, or to become negative, compression will use the relative threshold then

$$\|\mathbf{K}\| \times |\text{lr\_seuil}|$$

The first value makes it possible to profit from a little compression without any impact on the precision (for functional tests or expert testimonies).

17 Alternative with truncation of a factorization QR: RRQR for 'Rank Revealing QR'.

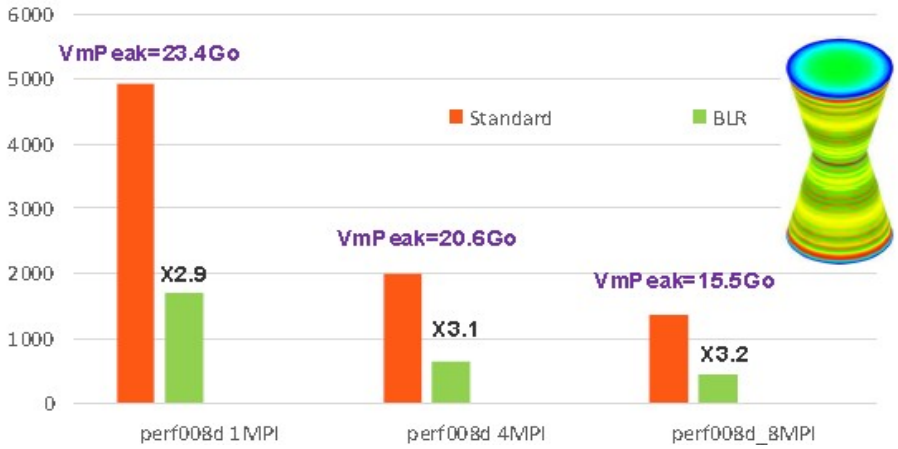


Figure 7.2-2: Example of profits gotten by compressions low-rank on the case benchmark perf008d (parameters by default, management memory in OOC, N=2M, NNZ=80M, Facto_METIS4=7495M, conditionnement=10^7). One traces, according to the number of activated processes MPI, the times elapsed spent by all the stage of resolution of system linear in Code_Aster v13.1, his peak RAM report, as well as the factor of acceleration gotten by BLR.

In short, the criterion of approximation of the algorithms of compression of MUMPS is fixed according to the following rule:

If `LOW_RANK_SEUIL=0.D0`:
truncation except for the precision machine.

If `LOW_RANK_SEUIL>0.D0`:
the truncation of the algorithms of compression uses directly `lr_seuil`.

If `LOW_RANK_SEUIL<0.D0`:
the truncation of the algorithms of compression is based on the relative threshold pointed out above.

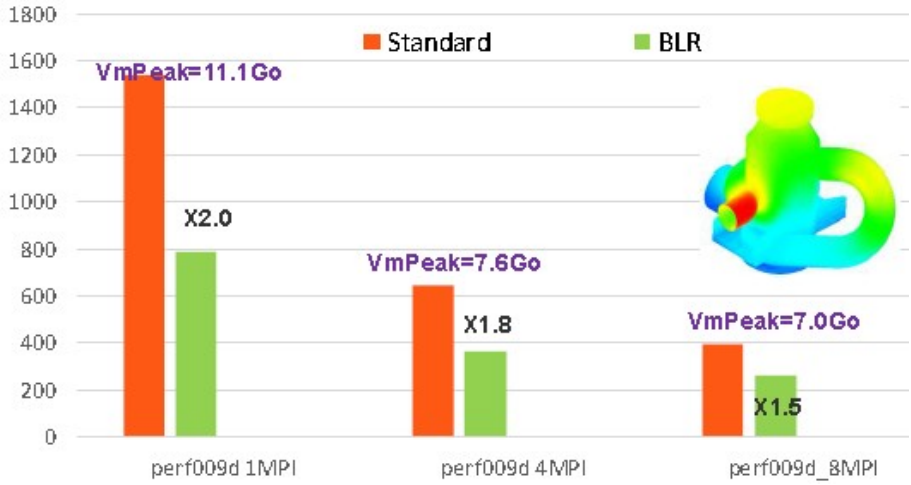


Figure 7.2-3: Example of profits gotten by compressions low-rank on the case benchmark perf009d (parameters by default, management memory in OOC, N=5.4M, NNZ=209M, Facto_METIS4=5247M, conditionnement=10^8). One traces, according to the number of activated processes MPI, the times elapsed spent by all the stage of resolution of system linear in Code_Aster v13.1, his peak RAM report, as well as the factor of acceleration gotten by BLR.

Note:

• For the moment, the profits of compressions low-rank relate to only the second phase of calculation of MUMPS, that of digital factorization, which is often most expensive. These profits thus depend on the importance of this stage compared to the other stages of the linear solvor (construction on `NUME_DDL`, analyzes and descent-increase). Let us recall that one can easily

trace the cost of this stage of factorization (item #1.3) by activating the detailed monitoring of the costs in times of each stage of calculation (keyword DEBUT/MESURE_TEMPS [U1.03.03]).

• Apart from the tools for compressions in them-even, the strategy low-rank implies two overcosts, one in the stage of analysis[18] and the other at the digital end of the factorization[19]. But, on the big problems, those are generally quickly compensated by the profits gotten by this technique.

• For the moment these profits relate to only the computing time, consumption memories remain similar even slightly higher (auxiliary vectors for compression) than those of a calculation 'full-rank' standard.

• For reasonable thresholds of compressions (<10^{-9}) impact on the quality of the results and the related 'outputs' (detection of singularity, calculation of determinant and the criterion of Sturm…) are often negligible[20] . Beyond that, it is not completely any more guaranteed. The good behavior and the robustness of calculation can suffer from it. This parameter setting is to be limited to a use "released" direct solvor or preconditionnor[21] for an iterative solvor of Krylov.

18 Preselection of the eligible dense blocks and construction of the structures of data related.

19 The compressed dense blocks are decompressed after use before approaching the last stage of descent-increase, because this one is still treated in 'full-rank'. With terms, it will be it-also treated in 'low-rank' and this overcost will disappear.

20 Concerning, the quality of the result it is all the more true, if the procedure of postprocessing were activated.

21 Features to come: for example for a use in time that direct solvor "released" (cf keywords FILTRAGE_MATRICE/MIXER_PRECISION) or as a preconditionnor (cf methods PETSc/GCPC + LDLT_SP) .