General information on the direct linear solveurs and use of MUMPS

Summary
Within the framework of thermomechanical simulations with *Code_Aster*, **the main part of the costs calculation often comes from the construction and the resolution of the linear systems**. To carry out these resolutions more effectively, ***Code_Aster* made the choice to integrate the direct method deployed in package MUMPS** ('Multifrontal Massively Parallel sparse direct Solver; P.R.Amestoy, J.Y.L' Excel et al.; CERFACS/IRIT/INRIA/CNRS). This in complement of its multifrontale "house" (C.Rose) and of its other solveurs: 'LDLT', 'GCPC', 'PETSC'.

In distributed parallel mode and Out-Of-Core, coupling *Aster*+MUMPS gets **profits in CPU about the dozen** on 32 processors. And this, for **consumption RAM close relations of those of the native multifrontale** code. This product, *via* the parallelism which it displays and its advanced features (swivelling, pre/postprocessings, quality of the solution…) should thus facilitate the passage of the studies standards. It remains sometimes **only viable alternative** to exploit certain modelings/analyses (quasi-incompressible, X-FEM…) or to pass from very large studies.

In the first part of the document we summarize the general problems of resolution of linear systems, then we approach the large families of hollow direct solveurs and their variations in the bookstores of the public domain. All things which it is necessary to have for the spirit before approaching, **in the second part**, package MUMPS through its main features and of its advanced features. Then we detail the digital, data-processing and functional aspects of its integration in *Code_Aster*. Lastly, we conclude by some digital results.

For more details and advices on the employment of the linear solveurs one will be able to consult the specific notes of use [U4.50.01]/[U2.08.03]. 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].

Contents

1 General information on the direct solveurs....4
1.1 Linear system and associated methods of resolution....4
1.2 Bookstores of linear algebra....6
1.3 Some results and benchmarks....7
1.4 Direct methods: the principle....8
1.5 Direct methods: various approaches....10
1.6 Direct methods: principal stages....12
1.7 Direct methods: difficulties....13
2 Package MUMPS....15
2.1 History....15
2.2 Main features....15
2.3 Zooms on some technical points....17
2.3.1 Swivelling....17
2.3.2 Iterative refinement....18
2.3.3 Reliability of calculations....19
2.3.4 Management memory (In-Core Out-Of-Core versus)....21
2.3.5 Management of the singular matrices....22
3 Establishment in Code_Aster....24
3.1 Context/synthesis....24
3.2 Two types of parallelism: centralized and distributed....24
3.2.1 Principle....24
3.2.2 Various modes of distribution....25
3.2.3 Balancing of load....26
3.2.4 To recut the Code_Aster objects....27
3.3 Management of memory MUMPS and Code_Aster....27
3.4 Particular management of the Lagranges double....28
3.5 Perimeter of use....29
3.6 Parameter setting and examples of use....29
3.6.1 Parameters of use of MUMPS via Code_Aster....29
3.6.2 Monitoring....30
3.6.3 Examples of use....31
4 Conclusion....33
5 Bibliography....34
5.1 Books/articles/proceedings/theses....34
5.2 Account-returned reports/EDF....34
5.3 Resources Internet....34
6 History of the versions of the document....35

1 General information on the direct solveurs

1.1 Linear system and associated methods of resolution

In digital simulation of physical phenomena, **a cost important calculation often comes from the construction and the resolution of linear systems**. The mechanics of the structures does not escape the rule! The cost of the construction of the system depends amongst points on integration and complexity on the laws on behavior, while that of the resolution is related on the number of unknown factors, modelings selected and topology (bandwidth, conditioning). When the number of unknown factors explodes, the second stage becomes dominating[1] and it is thus the latter which mainly will interest us here. Moreover, when it is possible to be more performing on this phase of resolution, thanks to the access to a parallel machine, we will see that this asset will be able to be propagated with the phase of constitution of the system itself (elementary calculations and assemblies).
These **linear inversions of systems in fact omnipresent in the codes and are often hidden with deepest of other digital algorithms**: non-linear diagram, integration in time, analyze modal…. One seeks, for example, the vector of nodal displacements (or of the increments of displacement) $\mathbf{u}$ checking a linear system of the type

$$\mathbf{Ku}=\mathbf{f} \tag{1.1-1}$$

with $\mathbf{K}$ a matrix of rigidity and $\mathbf{f}$ a vector representing the application of forces generalized to the mechanical system.

In a general way, **resolution of this kind of problem requires one (more) broad questioning** that it does not appear to with it:

- Does one have access to the matrix or knows one simply his action on a vector?
- Is this matrix it digs or dense?
- That they are its digital properties (symmetry, definite positive…) and structural (real/complex, by bands, blocks.)?
- Please one solve only one system (1.1-1), several into simultaneous [2] or in a consecutive way [3] ? Even several different and successive systems to which the matrices are very close [4] ?
- In the case of successive resolutions, can one re-use preceding results in order to facilitate the next resolutions (cf technique of restarting, partial factorization)?
- Which is the order of magnitude of the size of the problem, the matrix and of its factorized compared to capacities for treatment of the CPU and the associated memories (RAM, disc)?
- Does one want a very precise solution or simply an estimate (cf encased solveurs)?
- One has access to bookstores of linear algebras (and with their pre-necessary MPI, BLAS, LAPACK…) or does one have to call on products “house”?

In *Code_Aster* , one explicitly builds the matrix and one stores it with the format MORSE[5] . With most modelings, the matrix hollow (because of discretization by finite elements), is potentially badly conditioned[6] and often real, symmetrical and indefinite[7] . Into nonlinear, into modal or during chainings thermomechanical, one

1 For *Code_Aster*, to see the study of ‘profiling’ led by N.Anfaoui [Anf03].
2 Even matrix but several second independent members; Cf construction of a complement of Schur.
3 Problem of the multiples type second members: even matrix but several second successive and interdependent members; Cf algorithm of Newton without reactualization of the tangent matrix.
4 Problem of the multiples type first members: several matrices and second members successive and interdependent, but with close matrices “spectralement”; Cf algorithm of Newton with reactualization of the tangent matrix.
5 Known as still SCR for ‘Symmetric Compressed Row storage’ (makes ‘Column of it’ in *Code_Aster*).
6 In mechanics of the structures conditioning $\eta(\mathbf{K})$ is known to be rather bad. It can vary, typically, of 10^5 to 10^{12} and the fact of refining the grid, of using stretched elements or structural elements has dramatic consequences on this figure (cf B.Smith. *With parallel iterative implementation of year substructuring algorithm for problems in 3D.* SIAM J.Sci.Comput., **14** (1992), pp406-423. §3.1 or I.FRIED. *Condition of finite element matrices generated from nonuniform meshes*. AIAA J., **10** (1972), pp219-221.)
7 The indefinite character rather than definite positive is with the addition of additional variables (known as “of Lagrange”) to impose simple or generalized limiting conditions of Dirichlet [R3.03.01].

often deals with problems of the type “multiple second members”. The discrete methods of contact-friction benefit from faculties of factorization partial and the method by decomposition of fields. In addition, *Code_Aster* use also scenarios of simultaneous resolutions (complements of Schur of the contact and under-structuring…).
As for the sizes of the problems, even if they swell year by year, they remain modest compared to the CFD: about the million unknown factors but for hundreds of step of time or iterations of Newton.
In addition, of one **point of view “middleware and hardware”** , the code is pressed from now on on many optimized and perennial bookstores (MPI, BLAS, LAPACK, MONGREL, SCOTCH TAPE, PETSc, MUMPS…) and is used mainly on clusters of SMP (fast networks, great RAM storage capacity and disc). One thus seeks especially to optimize the use of the linear solveurs accordingly.

For 60 years, two types of techniques have disputed supremacy in the field, the solveurs direct and the iterative solveurs (cf [Che05] [Dav03] [Duf06] [Gol96] [Las98] [Liu89] [Meu99] [Saa03]).
First are robust and lead in a finished number of operations (theoretically) known by advance. Their theory is relatively well completed and their variation according to moults standard of matrices and software architectures is very complete. In particular, their algorithmic multilevel is well adapted to the hierarchies memories of the current machines. However, they require storage capacities which grow quickly with the size of the problem what limits the extensibility of their parallelism[8]. Even if this parallelism can break up into several independent layers, thus gearing down the performances.
On the other hand, them **iterative methods** are more “scalables” when one increases the number of processors. Their theory abounds in many “opened problems”, especially into arithmetic finished. In practice, their convergence in a “reasonable” number of iterations, is not always acquired, it depends on the structure of the matrix, the starting point, the criterion of stop… This kind of solveurs has more difficulty boring in mechanics of the industrial structures where one often cumulates heterogeneities, non-linearities and junctions of models which cause to become gangrenous the conditioning of the operator of work. In addition, they are not adapted to solve the problems of the type effectively “multiple second members”. Out those are very frequent in algorithmic mechanical simulations.
Contrary to their direct counterparts, it is not possible to propose the iterative solvor who will solve any linear system. The adequacy of the type of algorithm to a class of problems is done on a case-by-case basis. They present, nevertheless, other advantages which historically gave them established among for certain applications. With management equivalent memory, they require some less than the direct solveurs, because one has right need to know the action of the matrix on an unspecified vector, without having truly to store it. In addition, one is not subjected to the “diktat” of the phenomenon of filling which deteriorates the profile of the matrices, one can effectively exploit the hollow character of the operators and control the precision of the results[9] . **In short, the use of direct solveurs concerns the area of the technology rather whereas the choice of the good couple iterative method/preconditionnor is rather an art!** In spite of its biblical simplicity on paper, the resolution of a system linear, even symmetrical definite positive, is not “a long quiet river”. Between two evils, filling/swivelling and pre-packaging, it is necessary to choose!

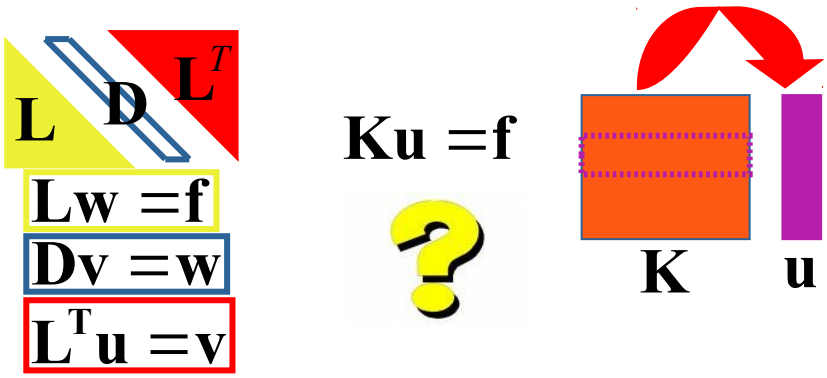


Figure 1.1-1. _Two classes of methods to solve a linear system: the direct ones and the iterative ones.

8 One also speaks about “scalability” or passage on the scale.
9 What can be very interesting within the framework of encased solveurs (e.g. Newton+GCPC), cf V.Frayssé. *The power of backward error analysis*. HDR of the Institut National Polytechnique of Toulouse (2000).

Note:

• A third class of methods tries to draw part of the respective advantages from direct and the iterative ones: methods of Decomposition of Field (DD) [R6.01.03].
• The two large families of methods must more be seen like complementary that like competitors. One often seeks to mix them: methods DD, preconditionnor by incomplete factorization (cf [R6.01.02] § 4.2) or of multigrille type, procedure of iterative refinement at the end of the direct solvor…

1.2 Bookstores of linear algebra

To effectively carry out the resolution of a linear system, **the question of resorts to a bookstore or with an external product is from now on impossible to circumvent**. Why? Because this strategy allows:

- Less technical, less invasive developments and much faster in the code host.
- To acquire, with less expenses, a broad perimeter of use while outsourcing good numbers of the contingencies associated (typology with the problem cf. §1.1, representation of the data, structures of the machine targets…).
- To profit from the experience feedback from a community of users varied and competences (very) pointed international teams.

These bookstores indeed often combine effectiveness, reliability, performance and portability:

- Effectiveness because they exploit the space and temporal locality data and exploit the hierarchy memory (example of the various categories of BLAS).
- Reliability because they propose tools to consider the mistake made on the solution (estimate of conditioning and the 'backward/forward errors') to even improve it (for the direct ones, balancing of the matrix and iterative refinement).

Since **emergence in years 70/80 of the first bookstores public**[10] **and private/manufacturers**[11] and their communities of users, the offer was geared down. The tendency being of course to suggest powerful solutions (vectorial, distributed parallelism with memory centralized then, parallelism multiniveau *via* threads) as well as "toolkits" of handling of algorithms of linear algebra and structures of associated data. Let us quote in a nonexhaustive way: ScaLAPACK (Dongarra & Demmel 1997), SparseKIT (Saad 1988), PETSc (Argonne 1991), HyPre (LL 2000), TRILINOS (Sandia 2000)…

Figure 1.2-1. _Some "logos" of libraries of linear algebra.

Note:

• To structure their use more effectively and to suggest solutions "black box", macro-bookstores recently came out. They gather a panel of these products to which they add solutions "houses": Numerical Plato (CEA-DEN), Mystery (CEA-DAM)…

Concerning more specifically them **direct methods of resolution of linear systems** , about fifty packages are available. One distinguishes the "autonomous" products from those incorporated in a bookstore, the public ones of commercial, those dealing with the dense problems and others of the hollows. Some function only in sequential mode, others support a parallelism with shared and/or distributed memory. Lastly, certain products are general practitioners (symmetrical, nonsymmetrical, SPD, reality/complex…) others adapted to a quite precise need/scenario.

10 EISPACK (1974), LINPACK (1976), BLAS (1978) then LAPACK (1992)…
11 NAG (1971), IMSL/ESSL (IBM 1971), ASL/MathKeisan (NEC), SciLib (CRAY), MKL (Intel), HSL (Harwell)…

One can find a list rather exhaustive of all these products on the site of one of the founding fathers of LAPACK/BLAS: Jack Dongarra [Gift]. The table below (table 1.2-1) is an expurgated version. It takes again only the direct solveurs of the public domain and forgets to mention: CHOLMOD, CSPARSE, DMF, Oblio, PARASPAR, PARDISO, PaStiX (the other French direct solvor with MUMPS), S+, SPRSBLKKT and WSMP. This resource Internet counts also packages implementing of the iterative solveurs, the préconditionneurs, the modal solveurs as well as many products support (BLAS, LAPACK, ATLAS…).

DIRECT SOLVERS	License	Support	Real	Complex	F77	C	Seq	Dist	SPD	Gen
DENSE										
FLAME	LGPL	yes	X	X	X	X	X			
LAPACK	BSD	yes	X	X	X	X	X			
LAPACK95	BSD	yes	X	X	95		X			
NAPACK	BSD	yes	X		X		X			
PLAPACK	?	yes	X	X	X	X		M		
PRISM	?	not	X		X		X	M		
ScaLAPACK	BSD	yes	X	X	X	X		M/P		
Trilinos/Pliris	LGPL	yes	X	X		X and C++		M		
SPARSE										
DSCPACK	?	yes	X			X	X	M	X	
HSL	?	yes	X	X	X		X		X	X
MFACT	?	yes	X			X	X	M	X	
MUMPS	PD	yes	X	X	X	X	X	M	X	X
PSPASES	?	yes	X		X	X		M	X	
SPARSE	?	?	X	X		X	X		X	X
SPOOLES	PD	?	X	X		X	X	M		X
SuperLU	Own	yes	X	X	X	X	X	M		X
TAUCS	Own	yes	X	X		X	X		X	X
Trilinos/Amesos	LGPL	yes	X				X	M	X	X
UMFPACK	LGPL	yes	X	X		X	X			X
Y12M	?	yes	X		X		X		X	X

Table 1.2-1. _Extracts from the Web page of Jack Dongarra [Gift] on the free products implementing a direct method; 'Seq' for sequential, 'Dist' for parallel ('M' OpenMP and 'P' MPI), 'SPD' for symmetrical definite positive and 'Gen' for unspecified matrix.

Note:

• A resource Internet more detailed but focused on the hollow direct solveurs is maintained by another great name of the digital one: T.A.Davis [Dav], one of the contributors of Matlab.

1.3 Some results and benchmarks

To attest founded good of its approach, each product/bookstore gets on its Web site of the sequential and parallel results of runs. They are often based on "matrices of test" drawn from public collections (MatrixMarket [MaMa], University of Florida, SPARSEKIT, Harwell…). Taking into account, in particular, difficulty of the exercise and sound strong investment in time and means (human and machine), one finds relatively little comparative on the linear solveurs.
Among these **benchmarks**, **four held our attention**. They compare good number of the products quoted with the preceding paragraph at the time of the resolution of various types of hollow systems (real/complex, symmetrical or not…). The matrices, resulting from specialized libraries, are variable sizes (of 10^4 to 5.10^7 unknown factors) and model various scopes of application (mechanics of structures, electromagnetism, CFD, electronics…).

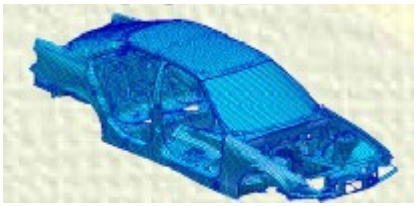
- *Nonsymmetrical into sequential*[Gup01] treated by UFMPACK, SuperLU, MUMPS, SPOOLES, UFMPACK and WSMP; Study very excavated into sequential on 40 matrices; *Results*: repeated failures of certain products and Net favours with **MUMPS** and especially with **WSMP**.

- *Symmetrical definite positive or indefinite into sequential*[GHS05] treated by BCSLIB, MA57, MUMPS, Oblio, PARDISO, SPOOLES, SPRSBLKKT, TAUCS, UFMPACK and WSMP; Study on 150 matrices; *Results in SPD*: few failures, all the products are *ex aequo* with a light advantage with **WSMP** and **PARDISO**; *Results into indefinite*: up to 25% of failures with a Net favours for **PARDISO** and **MA57**.

- *Symmetrical definite positive into sequential*[GGS08] treated by PETSc, HyPre, TRILINOS, ILUPACK and WSMP; Very excavated and exhaustive study (807 runs X 30 matrices!) which tests the iterative solveurs/préconditionneurs of the principal bookstores (CG/GMRES + incomplete factorization/multigrille/SAI) and compare them to direct solvor WSMP; *Results*: the direct solvor carries it "high the hand" in term of robustness, speed and consumption memory until consequent sizes of problems (10^7 degrees of freedom); The iterative solveurs containing direct (AMG and incomplete factorization) mark time: in second and third position one finds preconditionnor AMG of HyPre and incomplete Cholesky IC (0) of PETSc.

- *Nonsymmetrical in parallel*[ADEL00] treated by MUMPS and SuperLU; Very algorithmic/data-processing study carried out by the core-teams of the two products; *Results*: light advantage for MUMPS which displays of speed-UPS sometimes very interesting (90 on the 128 processors of a CRAY T3E cf table 1.3-1).

The codes are used, at least initially, in "black box" by the testers. From where often of the failures what can appear surprising for supposed direct solveurs (much) more robust than the iterative ones. In fact, these products are often victims of their sophistications. These "orderlies" are with a badly adapted parameter setting by default or a lack of RAM memory.
In these benchmarks direct solvor MUMPS is often in good position. It is one of the reasons which led to its integration in *Code_Aster* . In addition, even into sequential, its performances (RAM/CPU) are comparable to those of the solvor of reference of the code, the multifrontale [R6.02.02] [BHV06] [BD08]. Its faculties of swivelling, that the multifrontale does not have, makes it even essential to treat certain modelings/methods of analysis: incompressible elements, X-FEM…



Matrix	Ops x10⁹	Factorization time in seconds on 1 proc	Factorization time in seconds on 64 proc	Factorization time in seconds on 128 proc
AUDIKW_1	5682	3262.8	S/U 90 54.6	35.9
BRGM	31010	-	283.9	-
CONESHL_mod	1640	1099.0	S/U 90 19.6	12.1
CONV3D64	23880	-	207.5	146.5
ULTRASOUND80	3915	1542.2	S/U 52 37.1	29.5

Table 1.3-1. _Extracted the official site of MUMPS [Mum]: example of grid of car (1.5 10 [5] degrees of freedom) and parallel computation results on matrices of public tests.
Speed-UPS (noted S/U) up to 90 out of 128 processors on the digital stage of factorization (cf. § 1.6).

Note:

• Another French direct solvor, PaStiX [Not] gets of speed-UPS interesting to a hundred processors. In particular thanks to its parallelism on two levels (MPI+threads) very adapted to the machines cluster type of SMP. It seems to hold the record of the largest linear system solved by direct (in 2008): 83 million unknown factors in complex double precision which deals with a problem of Maxwell (5h on 768 processors of machine TERA-10 of the CEA-DAM).

1.4 Direct methods: the principle

The basic idea of the direct methods is of **to break up the matrix of the problem $\mathbf{K}$ in a product of particular matrices** (triangular lower and higher, diagonal) easier "to reverse". It is what is called **factorization**[12] matrix of work:

- *If $\mathbf{K}$ is SPD*, she admits the single "factorization of Cholesky": $\mathbf{K}=\mathbf{L}\mathbf{L}^{\mathbf{T}}$ with $\mathbf{L}$ triangular lower;

12 By analogy with polynomial factorizations of the lower school…

- *If* $\mathbf{K}$ *is symmetrical unspecified and regular*, she admits at least a "factorization $\mathbf{LDL^T}$": $\mathbf{PK=LDL^T}$ with $\mathbf{L}$ triangular lower than diagonal coefficients equal to the unit, $\mathbf{D}$ a diagonal matrix[13] and $\mathbf{P}$ a matrix of permutation;
- *If* $\mathbf{K}$ *is unspecified and regular*, she admits at least a "factorization $\mathbf{LU}$": $\mathbf{PK=LU}$ with $\mathbf{L}$ triangular lower than diagonal the unit, $\mathbf{U}$ triangular higher and $\mathbf{P}$ a matrix of permutation;

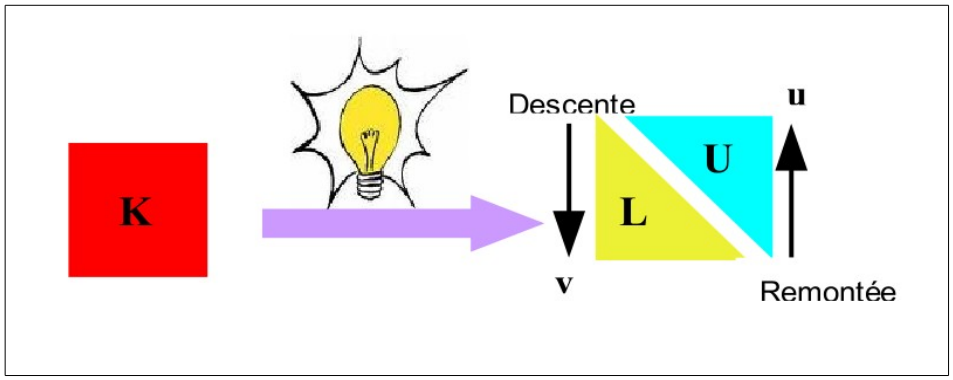


Figure 1.4-1. _Principle of the direct methods.

Note:

• For example, the symmetrical and regular matrix $\mathbf{K}$ *below breaks up in the form* $\mathbf{LDL^T}$ *following (without needing here permutation* $\mathbf{P=Id}$ *)*

$$\mathbf{K}:=\begin{bmatrix}10 & & sym\\ 20 & 45 & \\ 30 & 80 & 171\end{bmatrix}=\underbrace{\begin{bmatrix}1 & 0 & 0\\ 2 & 1 & 0\\ 3 & 4 & 1\end{bmatrix}}_{\mathbf{L}}\underbrace{\begin{bmatrix}10 & 0 & 0\\ 0 & 5 & 0\\ 0 & 0 & 1\end{bmatrix}}_{\mathbf{D}}\underbrace{\begin{bmatrix}1 & 2 & 3\\ 0 & 1 & 4\\ 0 & 0 & 1\end{bmatrix}}_{\mathbf{L^T}} \qquad (1.4\text{-}1)$$

Once this decomposition carried out, the resolution of the problem is largely facilitated. It is not expressed any more but in the form of the linear resolutions simplest which are: containing triangular or diagonal matrices. They are the famous ones "**descent-increase"** (' **forward/backward algorithms** '). For example in the case of a factorization $\mathbf{LU}$, the system (1.1-1) will be solved by

$$\left.\begin{array}{l}\mathbf{Ku=f}\\ \mathbf{PK=LU}\end{array}\right\rangle\Rightarrow\begin{array}{l}\mathbf{Lv=Pf}\ \left(\text{descente}\right)\\ \mathbf{Uu=v}\ \left(\text{remontée}\right)\end{array} \qquad (1.4\text{-}2)$$

In the first lower diagonal system (descent), one determines the intermediate vector solution $\mathbf{v}$. This last serves then as second member with the higher diagonal system (gone up) whose the vector is solution $\mathbf{u}$ who interests us.

This phase is inexpensive (into dense, about N^2 against N^3 for factorization[14] with N the size of the problem) and can thus be repeated of many factorized time by preserving the same one. What is very useful when a problem of the type is solved **multiple second members** or when one wishes to carry out **simultaneous resolutions**.

In **first scenario** , the matrix $\mathbf{K}$ is fixed and one changes second member successively $\mathbf{f_i}$ to calculate as much solution $\mathbf{u_i}$ (the resolutions are interdependent). That makes it possible to pool and thus to amortize these initial costs of factorization. This strategy is abundantly used, in particular in *Code_Aster*: buckle nonlinear with periodic reactualization (or not of reactualization) of the matrix tangent (e.g. operator *Aste*R

13 It can also comprise diagonal blocks 2×2 .

14 Into dense, Coppersmith and Winograd (1982) showed that one could, as well as possible, to decrease this algorithmic complexity with CN^{α} with $\alpha=2,49$ and C constant (for N large).

STAT_NON_LINE), methods of subspaces or the power reverses (without acceleration of Rayleigh) in modal calculation (CALC_MODES), thermomechanical chaining with characteristic materials independent of the temperature (MECA_STATIQUE),…

In **second scenario**, one is aware of all them $\mathbf{f}_i$ at the same time and one organizes, by blocks, the phases of descent-increase, to calculate the solutions simultaneously $\mathbf{u}_i$ independent. One can thus use more effective routines of high level linear algebra, and even to exploit consumption memory by storing the vector $\mathbf{f}_i$ in hollow. This strategy (partially) is used in *Code_Aster* , for example, in the construction of the complements of Schur of the algorithms of contact-friction or for the under-structuring.

Note:

• *Products MUMPS envisages these two types of strategy and proposes even features to facilitate the construction and the resolution additional of Schur.*

Let us examine it now **process of factorization in itself**. It is already clearly clarified in other theoretical documentations of *Code_Aster* on the subject [R6.02.01] [R6.02.02], like in the already quoted bibliographical references [Duf06] [Gol96] [Las98]. Also we will not detail it. Let us specify just that it is an iterative process organized schematically around three loops: one “known as in i ” (on the lines of the matrix of work), the second “in j ” (resp. columns) and the third “in k ” (resp. stages of factorization). They build a new matrix repeatedly $\tilde{\mathbf{A}}_{k+1}$ starting from certain data of the preceding one, $\tilde{\mathbf{A}}_k$, *via* the classical formula of factorization which is written formally:

$$\begin{array}{c}\textbf{Boucles en } i,j,k\\ \tilde{\mathbf{A}}_{k+1}(i,j):=\tilde{\mathbf{A}}_k(i,j)-\dfrac{\tilde{\mathbf{A}}_k(i,k)\tilde{\mathbf{A}}_k(k,j)}{\tilde{\mathbf{A}}_k(k,k)}\end{array} \qquad (1.4\text{-}3)$$

Initially the process is activated with $\tilde{\mathbf{A}}_0=\mathbf{K}$ and at the last stage, one recovers in the square matrix $\tilde{\mathbf{A}}_N$ triangular parts ($\mathbf{L}$ and/or $\mathbf{U}$) even diagonal ($\mathbf{D}$) who interest us. For example, in the case $\mathbf{L}\,\mathbf{D}\,\mathbf{L}^{\mathbf{T}}$:

$$\begin{array}{l}\textbf{Boucles en } i,j\\ \text{si } i<j:\ \mathbf{L}(i,j)=\tilde{\mathbf{A}}_N(i,j)\\ \text{si } i=j:\ \mathbf{D}(i,j)=\tilde{\mathbf{A}}_N(i,j)\end{array} \qquad (1.4\text{-}4)$$

Note:

• *The formula* (1.4-3) *contains in germ the problems inherent in the direct methods: in hollow storage, the fact that the term* $\tilde{\mathbf{A}}_{k+1}(i,j)$ *can become nonnull whereas* $\tilde{\mathbf{A}}_k(i,j)$ *is (notion of filling of factorized, ‘rope’, thus implying a renumerotation or ‘ordering’); propagation of rounding error or the divide check via the term* $\tilde{\mathbf{A}}_k(k,k)$ *(notion of swivelling and balancing of the terms of the matrix or ‘scaling’).*

1.5 Direct methods: various approaches

The order of the loops i , j and k is not fixed. One can permute them and carry out the same operations but in a different order. That defines six alternatives thus $kij, kji, ikj\ldots$ who will handle various zones of the current matrix $\tilde{\mathbf{A}}_k$: “zone of the new calculated terms” *via* (1.4-2), “zone already calculated and used” in (1.4-2), “already calculated and unutilised zone” and “zone not yet calculated”. For example, in the alternative jik , there is the diagram of operation following for j fixed

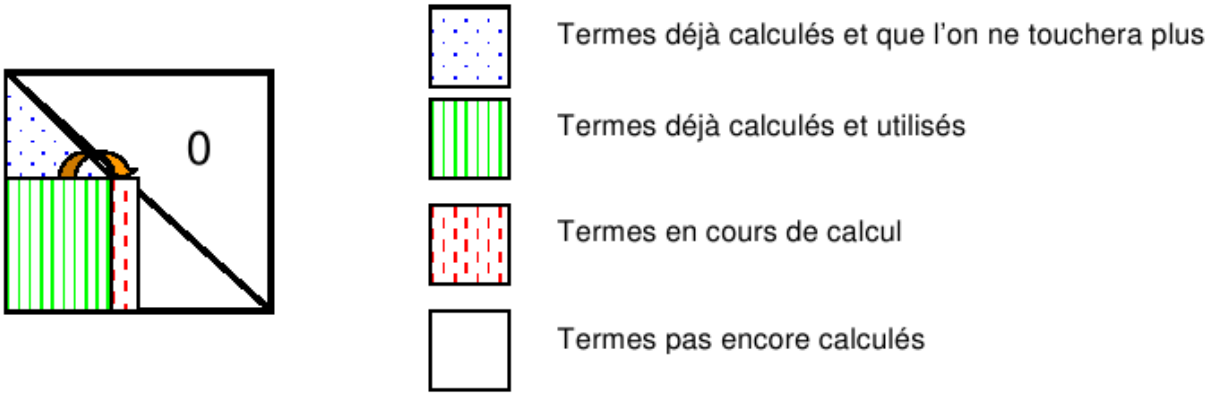


Figure 1.5-1. _Diagram of construction of a factorization " jik " ('right looking').

Note:

• Method $\mathbf{LDL^T}$ *of Code_Aster (*`SOLVEUR/METHODE=' LDLT' )` *is a factorization "* ijk *", multifrontales of C.Rose (* `… = ' MULT_FRONT' )` *and of MUMPS (* `… = ' MUMPS')` *directed columns are they ("* kji *").*

• Certain alternatives bear particular names: algorithm of Crout (" jki *") and that of Doolitle ("* ikj *").*

• In papers one often uses the Anglo-Saxon terminology indicating the orientation of matric handling rather than the order of the loops: 'looking forward method', 'looking backward method', 'up-looking', 'left-looking', '' right-looking', 'left-right-looking'…

All **these alternatives are declined** still according to:

- That one exploits certain properties of the matrix (symmetry, definite-positivity, band…) or that one seeks the broadest perimeter of the use;
- Whether one carries out scalar treatments or by blocks;
- That the decomposition in blocks is determined by aspects reports (cf method $\mathbf{LDL^T}$ paginated *Code_Aster*) or rather related to independences of the later tasks (*via* a graph of elimination cf multifrontale [R6.06.02] § 1.3);
- That one reintroduces worthless terms in the blocks to facilitate the access to the data[15] and to cause very effective algebraic operations, often *via* BLAS3[16] (cf multifrontale of C.Rose and MUMPS);
- That one groups the contributions affecting a block of lines/columns (approach 'fan-in', cf PaStiX) or that they are applied as soon as possible ('fan-out');
- That in parallelism, one seeks to spare various levels of sequences of independent tasks, that they are scheduled statically or dynamically, that one recovers calculation by communication…
- That one applies the pre ones and postprocessings to reduce the filling and to improve quality of the results: renumerotation of the unknown factors, put at the level of the terms of the matrix, partial swivelling (line) or total (line and column), scalar or per diagonal blocks, iterative refinement…

For **to gather four categories are often distinguished**:

- **classical algorithms**: Gauss, Crout, Cholesky, Markowitz (*Matlab, Mathematica, Y12M* …) ;
- Methods **frontal** (*MA62* …);
- Methods **multifrontales** (*MULT_ FRONT Aster, MUMPS, SPOOLES, TAUCS, UFMPACK, WSMP* …)
- **supernodales** (*SuperLU, PaStiX, CHOLMOD, PARDISO* …).

15 This hollow/dense compromise allows to decrease indirect addressings with the data and thus to better use the hierarchy memory of the current machines.

16 The ratio "calculation/access report" of Blas level 3 (produced matrix/matrix) is N time better (with N size of the problem) that of the other levels of Blas. It is also often higher than that of routines "made with the hand" not optimized on these aspects "locality of the data/hierarchy memory".

1.6 Direct methods: principal stages

When hollow systems are treated, the digital phase of factorization (1.4-3) does not apply directly to the initial matrix $\mathbf{K}$, but with **a matrix of work** $\mathbf{K}_{\text{travail}}$ resulting from one **phase of pretreatments** . And it **in order to reduce the filling, to improve the precision of calculations** and thus to optimize the later costs in CPU and memory. Coarsely this matrix of work can be written in the shape of the following matric product

$$\mathbf{K}_{travail} := \mathbf{P}_o \, \mathbf{D}_r \, \mathbf{K} \, \mathbf{Q}_c \, \mathbf{D}_c \, \mathbf{P}_o^T \qquad (1.6\text{-}1)$$

we will describe the various elements thereafter. One can thus break up the operation of a direct solvor into four stages:

1) *Pretreatments and factorization symbolic system*: it inverted the order of the columns of the matrix of work (*via* a matrix of permutation $\mathbf{Q_c}$) in order to avoid the divide checks of term $\tilde{\mathbf{A}}_k(k,k)$ and to reduce the filling. Moreover it rebalances the terms in order to limit the errors rounding (*via* matrices of scaling $\mathbf{D_r}/\mathbf{D_c}$). This phase can be too **crucial for the algorithmic effectiveness** (profit of a sometimes noted factor 10) and the quality of the results (profit of 4 or 5 decimals).
In this phase, one also created the structures of storage of the hollow factorized matrix and auxiliaries (dynamic swivelling, communication…) required by the following phases. Moreover, one estimates the tree of dependence of the tasks, their initial distribution according to the processors and consumption total memories envisaged.

2) *The stage of renumeroration*: it inverted the order of the unknown factors of the matrix (*via* the matrix of permutation $\mathbf{P_o}$) in order to reduce the filling which factorization implies. Indeed, in the formula (1.4-3) it is seen that factorized ($\tilde{\mathbf{A}}_{k+1}(i,j) \neq 0$) a new term not no one in its profile can contain whereas the initial matrix did not comprise any ($\tilde{\mathbf{A}}_k(i,j) = 0$) . Because of the term $\dfrac{\tilde{\mathbf{A}}_k(i,k)\tilde{\mathbf{A}}_k(k,j)}{\tilde{\mathbf{A}}_k(k,k)}$ not necessarily no one . In particular, it is nonnull when one can find terms nonworthless of the matrix initial of the type $\tilde{\mathbf{A}}_k(i,l) \text{ ou } \tilde{\mathbf{A}}_k(l,j) \ (l<i \text{ et } l<j)$. This phenomenon can lead to overcosts very important report and calculation (factorized can be 100 times larger than the initial hollow matrix!).
From where the idea to renumber the unknown factors (and thus to permute the lines and the columns of $\mathbf{K}$) in order to slow down this phenomenon which is it **true "Achilles' heel" of the direct ones** . With this intention, one often calls on external products (MONGREL, SCOTCH TAPE, CHACO, JOSTLE, PARTY…) or with the heuristic ones embarked with solveurs (AMD, RCMK…). Of course, these products display different performances according to the treated matrices, the number of processors… Among them, **MONGRELS and SCOTCH TAPE are very widespread and "often leave the batch"** (profit up to 50%).

3) *The digital phase of factorization* : it implements the formula (1.4-3) *via* the methods interviews in the paragraph § 1.5 precedent. It is the phase, by far, **most expensive** who will build hollow factorizations explicitly $\mathbf{L}\mathbf{L}^{\mathbf{T}}, \mathbf{L}\mathbf{D}\mathbf{L}^{\mathbf{T}}$ or $\mathbf{L}\mathbf{U}$.

4) *The phase of resolution*: it carries out the descent-increase (1.4-2) whose (finally!) the solution "spouts out" $\mathbf{u}$. It is **not very expensive** and **pools possibly a later digital factorization** (multiple second members, simultaneous resolutions, restarting of calculation…).

Note:

• Stages 1 and 2 require only the knowledge of the connectivity and the graph of the initial matrix. Thus finally that data storable and easy to handle in the form of entireties. Only the two last stages use realities on the effective terms of the matrix. They require for the terms of the matrix only if the stages of scaling are engaged (calculation of $\mathbf{D}_r/\mathbf{D}_c$ *).*

• Stages 1 and 4 are independent while the 1.2 and 3, a contrario, are dependent. According to the algorithmic products/approaches, one agglomerates them differently: 1 and 2 is dependent in MUMPS, 1 and 3 in SuperLu and 1.2 and 3 in UFMPACK… MUMPS makes it possible to carry out separately but successively stages 1+2, 3 and 4, to even pool their results to carry out various sequences. For the moment, in Code_Aster, one alternates sequences 1+2+3 and 4,4,4… and again 1+2+… (cf following chapter).

• Certain products propose to test several strategies in one or more stages and choissent the most adapted: SPOOLES and WSMP for stage 1, TAUCS for the stage 3 etc.

• The tools of renumerotation of the first phase are based on very varied concepts: methods of geometrical engineers, techniques or of optimization, graph theory, spectral theory, methods taboo, algorithms évolutionnaires, those mémétics, those based known as of "colonies of ants", neural networks… All the blows are allowed to improve the local optimum in the form of which the problems of the renumerotor are expressed. These tools are also often used for partitionner/to distribute grids (cf [R6.01.03] §6). For the moment, Code_Aster uses renuméroteurs METIS/AM/AMD (for `METHODE=' MULT_FRONT'` *and* `' MUMPS'`*), AMF/QAMD/PORD (for* `' MUMPS'` *) and RCMK[17] (for* `'GCPC', 'LDLT'` *and* `'PESTC'`*).*

• Some is the linear solvor[18] used, Code_Aster carries out a preliminary phase of factorization (stage 0) to describe the unknown factors of the problem (link degree of physical or late freedom/number of line of the matrix via the structure of data `NUME_DDL`*) and to envisage the ad hoc storage of the profile MORSE of the matrix.*

1.7 Direct methods: difficulties

Among the difficulties which the "hollow direct methods must overcome" one finds:

- **handling of structures of data complex** who optimize the storage (cf profile of the matrix) but which complexes the algorithmic one (cf swivelling, OOC…). That contributes has to lower the ratio "calculation/access to the data".
- **effective management of the data with respect to the hierarchy memory** and rocker IC/OOC[19] . A this recurring question with much of problems, but which here is prégnante because of strong consumption calculation.
- The management of **hollow/dense compromise** (for the methods by faces) with respect to consumption memory, of the accessibility to the data and the effectiveness of building blocs of linear algebra.
- The choice of **good renumerotation**: it is a Np-complete problem! For the problems of big sizes, one cannot find in a "reasonable" time the optimal renumerotation. One must be satisfied with a "local" solution.
- The effective management of **propagation of the rounding errors** *via* the scaling, swivelling and analyses error on solution (the direct/opposite error[20] and conditioning).
- **size of factorized which is often the "bottleneck" n°1** . Its distribution between processors (*via* distributed parallelism) and/or the OOC always do not make it possible to surmount this shelf (cf figure 1.7-1).

17 For minilmiser the filling, incomplete factorization is used as preconditionnor of these iterative solveurs.

18 Among *'*`MULT_FRONT'/'LDLT'/'MUMPS'/'GCPC'/'PESTC'`.

19 IC for In-Core (all the structures of data are in RAM) and OOC for Out-Of-Core (some are rocked on disc).

20 Often referred under the Anglo-Saxon term: 'forward/backward errors'.

Figure 1.7-1. _The "ball" of the hollow direct solveurs: size of factorized.

Figure 1.7-1 shows two examples: a canonical CAS-test (cubic) and an industrial study (pump LAUGH). With the following notations: M for million terms, N size of the problem, nnz the number of nonworthless terms of the matrix and $\mathbf{K}^{-1}$ that of its factorized renumbered *via* MONGREL. The surfactor when one passes from the one to the other is noted between brackets.

2 Package MUMPS

2.1 History

Package MUMPS implements a parallel multifrontale "massively" ('**MUltifrontal Massively Parallel sparse direct Solver** '[Mum]) developed during the European project PARASOL (1996-1999) by the teams of three laboratories: CERFACS, ENSEEIHT-IRIT and RAL (I.S.Duff, P.R.Amestoy, J.Koster and J.Y.L' Excel…). Since this finalized version (MUMPS 4.04 9/22/99) and public (free of right), about thirty other versions were delivered (3/4 a year). These developments correct anomalies, extend the perimeter of use, improve ergonomics and especially, enrich the features. MUMPS is thus a perennial product, developable and maintained by teams of **IRIT, CNRS**, **CERFACS** and of **INRIA** (half a dozen people).

Figure 2.1-1. _Logos of the principal contributors with MUMPS [Mum].

The product is public and downloadable on its Web site: **http://graal.ens-lyon.fr/MUMPS** . One counts approximately **1000 direct users** (including 1/3 Europe + the 1/3 USA) not counting those which use it *via* the bookstores which it reference: PETSc, TRILINOS, Matlab and Scilab. Its site proposes documentation (theoretical and of use), links, examples of application, as well as a newsgroup (in English) tracing the experience feedback on the product (bugs, problems of installation, advices…).
Each year about ten algorithmic/data-processing work leads to improvements of the package (thesis, post-Doc., research tasks…). In addition it is used regularly for studies industrial (EADS, ECA, BOEING, GéoSciences Azure, SAMTECH, *Code_Aster…*).

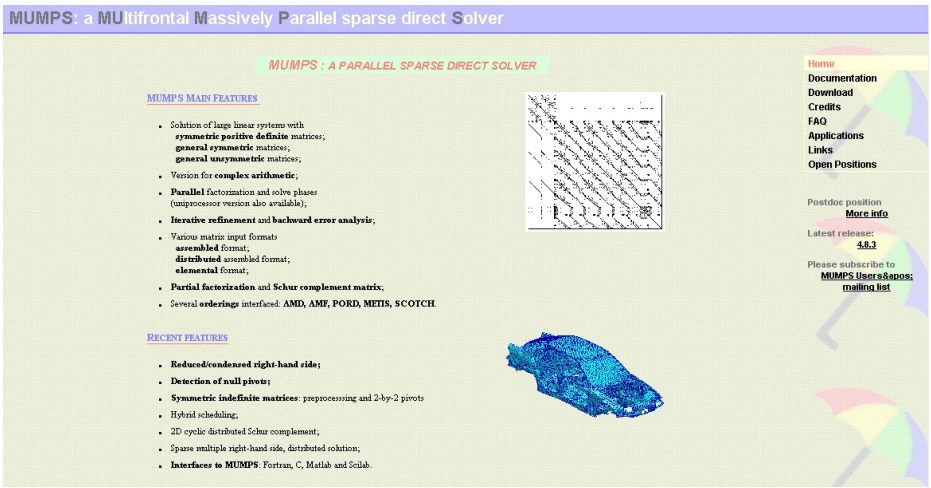
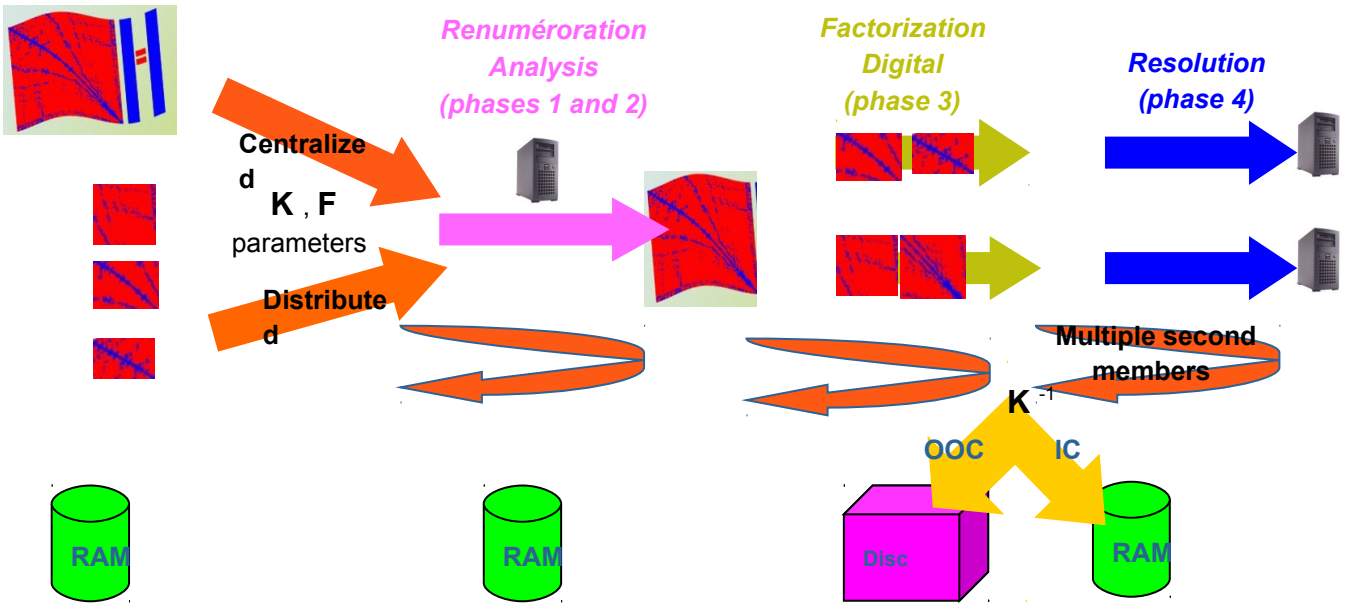


Figure 2.1-2. _The homepage of the Web site of MUMPS [Mum].

2.2 Main features

MUMPS implements one **multifrontale** [ADE00] [ADKE01] [AGES06] carrying out a factorization LU or LDL^T (cf. § 1.4). Its main features are:

- **Broad perimeter of use**: SPD, symmetrical unspecified, nonsymmetrical, real/complex, simple/double precision, stamps regular/singular (all this perimeter is exploited in *Code_Aster*);
- Admits **three modes of distribution of the data**: by element, centralized or distributed (the last two modes are exploited in *Code_Aster*);
- **Interfacing** in FORTRAN (exploited), C, matlab and scilab.
- **Parameter setting by default** and possibility of letting the package choose some of its options according to the type of problem, its nature and amongst processors (exploited).
- **Modularity** (3 distinct phases interchangeable cf. § 1.6 and appears 2.2.1) and opening of certain digital mysteries of MUMPS. The user (very) advanced can thus leave the product the result of certain pretreatments (scaling, swivelling, renumerotation), modify them or replace them by others and reintegrate them in the computation channel specific to the tool;
- Different **strategies of resolutions**: multiple second members, simultaneous resolutions and complements of Schur (only the two first are exploited and into dense);
- Different **renumeroteurs** embarked or external: MONGREL, AMD, QAMD, MFA, PORD, SCOTCH TAPE, 'provided by L' utilisateur' (exploited except both the last);
- **Related features**: detection of small pivots, calculation of row and cores (exploited soon), analyzes error on the solution (exploited);
- **Pre and posttraitements**: scaling, permutation line/column and scalar/block 2x2, iterative refinement (exploited);
- **Parallelism** : potentially on 2 levels (MPI + OpenMP of the BLAS3), asynchronous management of the floods of tasks/given and their dynamic regrouping, covering calculation/communication; Distribution of the data associated with the distribution of the tasks; This parallelism starts, for the moment, only on the level of the phase of factorization [21] (exploited).
- **Memory** : unloading on disc or not of factorized (modes In-Core or Out-Of-Core) with preliminary estimate of RAM consumption by processor in both cases; Mode OOC starts, for the moment, only on the level of the phase of factorization (exploited).



21 Within the framework of the ANR SOLSTICE [GROUND], this parallelism will be wide with the initial phase of analysis.

*Figure 2.2-1. _Functional Flow chart of MUMPS:
its three stages in parallel centralized/distributed and IC/OOC.*

Note:

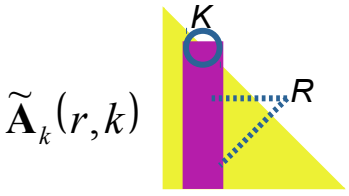
• In term of parallelism, MUMPS exploits two levels (cf [R6.01.03] § 2.6.1): one external related to the concurrent elimination of faces (via MPI), the other interns, within each face (via "threadées" BLAS). The native method multifrontale of Code_Aster exploits only the second level and in parallelism with shared memory (via OpenMP). Thus without covering, dynamic regrouping and distribution of the data between the processors. On the other hand, by its fine connections with Code_Aster, it exploits all the facilities of the manager memory JEVEUX (OOC, restarting, diagnosis…) and specificities of modeling of the code (elements of structures, Lagranges).

2.3 Zooms on some technical points

2.3.1 Swivelling

The technique of swivelling consists in choosing a term $\tilde{\mathbf{A}}_k(k,k)$ adapted (in formula 1.4-3) to avoid dividing by a too small term (what would amplify the propagation of the errors rounding during the calculation of the terms $\tilde{\mathbf{A}}_{k+1}(i,j)$ following). With this intention, one permutes lines (partial swivelling) and/or columns (resp. total) to find the denominator of (1.4-3) adapted. For example, in the case of a partial swivelling, one chooses like "pivot" the term $\tilde{\mathbf{A}}_k(r,k)$ such as

$$\tilde{\mathbf{A}}_k(r,k) \geq u \max_i |\tilde{\mathbf{A}}_k(i,k)| \quad \text{avec } u \in]0,1] \tag{2.3-1}$$



$\tilde{\mathbf{A}}_k(r,k)$

***F** igure 2.3-1. _Choice of the partial pivot at the stage K .*

From where an amplification of the errors rounding to the maximum of ($1+1/u$) with this stage. **What is important here is not so much to choose the largest possible term in value absolute** (U=1) **to avoid choosing smallest!** The reverse of these pivots also intervenes at the time of the phase of descent-increase, therefore it is necessary to spare these two sources of amplification of errors by choosing one U median. MUMPS, like much of package, proposes by default U =0.01 (parameter MUMPS CNTL (1)).
To swivel one generally uses scalar diagonal terms but also of the blocks of terms (of the diagonal blocks 2x2).
In MUMPS, two types of swivelling are implemented, one says 'static' (at the time of the phase of analysis), the other known as 'digital' (resp. digital factorization). They are skeletal and activables separately (cf parameters MUMPS CNTL (1), CNTL (4) and ICNTL (6)). For matrices SPD or with dominant diagonal, these faculties of swivelling can be disabled without risk (calculation will gain there in speed), on the other hand, in the other cases, they should be initialized to manage the possible very small or worthless pivots. That in general implies an addition of filling of factorized but increases digital stability.

Note:

• This functionality of swivelling makes MUMPS essential to treat certain modelings of Code_Aster (quasi-incompressible elements, mixed formulations, X-FEM…). At least as long as other direct solveurs including of the swivelling will not be available in the code.

• The user Aster does not have access directly to the fine parameter setting of these faculties of swivelling. They are activated with the values by default. He can just partially disconnect them while posing SOLVEUR/PRETRAITEMENTS=' SANS' (by défaut='CAR').

• The addition of filling of to the digital swivelling must be scheduled as soon as possible in MUMPS (as of the phase of analysis). And this, by envisaging arbitrarily a percentage of overconsumption memory compared to the profile envisaged. This figure must be indicated in for hundred in the parameter MUMPS ICNTL (14) (accessible to the user Aster via the keyword SOLVEUR/PCENT_PIVOT initialized by default with 20%). Thereafter, if this evaluation proves to be insufficient, according to the type of management selected memory (keyword SOLVEUR/GESTION_MEMOIRE), that is to say calculation stops in ERREUR_FATALE, is one retente several times a digital factorization by doubling each time the size of this reserved space to the swivelling.

• Certain products restrict their perimeter/robustness by not proposing strategy of swivelling (SPRSBLKKT, MULT_FRONT_Aster…), others are limited to scalar pivots (CHOLMOD, PaStiX, TAUCS, WSMP…) or propose particular strategies (method of perturbation+correction of Bunch-Kaufmann for PARDISO, Bunch-Parlett for SPOOLES…).

2.3.2 Iterative refinement

At the end of the resolution, having obtained the solution $\mathbf{u}$ problem, one can evaluate his residue easily $\mathbf{r} := \mathbf{Ku} - \mathbf{f}$. Knowing factorized matrix already, this residue can then feed with few expenses **the iterative process of improvement according to** (in the nonsymmetrical case general):

$$\begin{aligned} &\text{Boucle en } i \\ &(1)\quad \mathbf{r}^i = \mathbf{f} - \mathbf{K}\mathbf{u}^i \\ &(2)\quad \mathbf{LU}\ \delta\mathbf{u}^i = \mathbf{r}^i \\ &(3)\quad \mathbf{u}^{i+1} \Leftarrow \mathbf{u}^i + \delta\mathbf{u}^i \end{aligned} \qquad (2.3\text{-}2)$$

This process is "painless"[22] since it costs mainly only the price of the descent-increase of the stage (2). It can be thus reiterated until a certain threshold or a maximum iteration count. If the calculation of residue does not comprise too much rounding error, i.e. if the algorithm of resolution is rather reliable (cf following paragraph) and that the conditioning of the matric system is good, this process of iterative refinement[23] is very beneficial on the quality of the solution.

22 It is true when MUMPS functions in memory way of managing In-Core and into sequential. On the other hand, when the data are distributed between the processors and the memories RAM and disc (parallelism and Out-Of-Core is activated), this stage can be a little expensive.

23 One also speaks "about iterative improvement" ('iterative refinement').

In MUMPS this process is activable or not (parameter ICNTL (10)<0) and limited by a maximum iteration count N_{err} (ICNTL (10)). The process (2.3-2) continues as much as the "balanced residue" $\mathbf{B}_{err}$ is higher than a skeletal threshold *threshold* (CNTL (2), fixed by default at $\sqrt{\varepsilon}$ with ε precision machine)

$$\mathbf{B}_{err} := \max_j \frac{|\mathbf{r}_j^i|}{(|\mathbf{K}||\mathbf{u}^i|+|\mathbf{f}|)_j} < seuil \qquad (2.3\text{-}3)$$

or that it does not decrease a factor at least 5 (nonskeletal). In general, one or two iterations is enough. If it is not the case, it is often revealing other problems: bad conditioning or opposite error (cf following paragraph).

Note:

• For the Code_Aster user these parameters MUMPS are not directly accessible. The functionality is activable or not via the keyword POSTTRAITEMENTS .
• This functionality is present in many packages: Oblio, PARDISO, UFMPACK, WSMP…

2.3.3 Reliability of calculations

To estimate the quality of the solution of a linear system [ADD89] [Hig02], MUMPS proposes digital tools deduced from the theory from **the analysis reverses rounding errors initiated by Wilkinson** (1960). In this theory, the rounding errors due to several factors (truncation, operation into arithmetic finished.) are comparable to disturbances on the initial data. That makes it possible to compare them with other sources of errors (measurement, discretization…) and to handle them more easily *via* three indicators obtained in postprocessing:

- *Conditioning* $cond(\mathbf{K},\mathbf{f})$: it measures **sensitivity of the problem to the data** (unstable problem, badly formulated/discretized…). I.e., the multiplicative factor that the handling of the data will operate on the result. To improve it, one can try to change the formulation of the problem or to balance the terms of the matrix, apart from MUMPS or *via* MUMPS (SOLVEUR/PRETRAITEMENTS=' OUI' in *Code_Aster*).
- *The opposite error* $be(\mathbf{K},\mathbf{f})$ ('backward error'): it measures **propensity of the algorithm of resolution to transmit/amplify the rounding errors**. A tool is known as "reliable" when this figure is close to the precision machine. To improve it, one can try to change algorithm of resolution or to modify one or more his stages (in *Code_Aster* one can exploit the parameters SOLVEUR/TYPE_RESOL, PRETREATMENTS and RENUM).
- *The direct error* $fe(\mathbf{K},\mathbf{f})$ ('forward error'): it is the product of the two preceding digits and provides one **raising relative error on the solution**.

$$\frac{\|\delta\mathbf{u}\|}{\|\mathbf{u}\|} < \underbrace{cond(\mathbf{K},\mathbf{f})\times be(\mathbf{K},\mathbf{f})}_{fe(\mathbf{K},\mathbf{f})} \qquad (2.3\text{-}4)$$

One can give a chart (cf figure 2.3-2) of these concepts while expressing **the opposite error** like the variation enters "the initial data and the disturbed data", while **the direct error** measurement the variation enters "the exact solution and the solution really obtained" (that of the problem disturbed by the errors rounding).

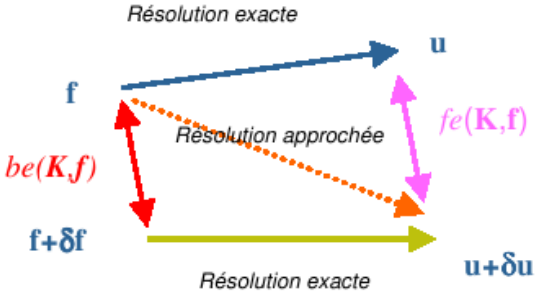


Figure 2.3-2. _Chart of the concept of errors direct and opposite.

Within the framework of the linear systems, the error reverses is measured *via* **balanced residue**

$$be(\mathbf{K},\mathbf{f}) := \max_{j\in J} \frac{|\mathbf{f}-\mathbf{K}\mathbf{u}|_j}{(|\mathbf{K}||\mathbf{u}|+|\mathbf{f}|)_j} \tag{2.3-5}$$

One cannot always evaluate it on all the indices ($J \neq [1,N]_N$). In particular when the denominator is very small (and the numerator not no one), one prefers the formulation to him (with J^* such as $J \cup J^* = [1,N]_N$)

$$be^*(\mathbf{K},\mathbf{f}) := \max_{j\in J^*} \frac{|\mathbf{f}-\mathbf{K}\mathbf{u}|_j}{(|\mathbf{K}||\mathbf{u}|)_j + \|\mathbf{K}_{j.}\|_\infty \|\mathbf{u}\|_\infty} \tag{2.3-6}$$

where $\boldsymbol{K}_{j.}$ represent $j^{ième}$ line of the matrix $\mathbf{K}$. With these two indicators, one associates two estimates of conditioning matricel (one related to the lines retained as a whole J and the other with its complementary J^*): $cond(\mathbf{K},\mathbf{f})$ and $cond^*(\mathbf{K},\mathbf{f})$. The theory then provides us the results according to:

- **The approximate solution U is the exact solution of the disturbed problem**

$$\begin{aligned} &(\mathbf{K}+\delta\mathbf{K})\,\mathbf{u} = (\mathbf{f}+\delta\mathbf{f}) \\ &\text{avec } \delta\mathbf{K}_{ij} \leq \max(be, be^*)|\mathbf{K}_{ij}| \\ &\text{et } \delta\mathbf{f}_i \leq \max(be.|\mathbf{f}_i|, be^*.\|\mathbf{K}_{i.}\|_\infty \|\mathbf{u}\|_\infty) \end{aligned} \tag{2.3-7}$$

- There is following increase (*via* the direct error $fe(\mathbf{K},\mathbf{f})$) on **the relative error in solution**

$$\frac{\|\delta\mathbf{u}\|}{\|\mathbf{u}\|} < \underbrace{cond\times be + cond^*\times be^*}_{fe(\mathbf{K},\mathbf{f})} \tag{2.3-8}$$

In practice, one scans especially this last estimate $fe(\mathbf{K},\mathbf{f})$ and its components. Its order of magnitude $n(10^{-n})$ indicate *grosso-modo* the number of "true" decimals of the solution calculated. 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.

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.

Note:

• This analysis of the quality of the solution is not limited to the linear solveurs. It is also declined, for example, for the modal solveurs [Hig02].

• In MUMPS, the estimators fe, be, be^* , $cond$ and $cond^*$ are accessible via, respectively, the variables `RINFO (7/9/8/10` and `11)`. These postprocessings are a little expensive (~jusqu'à 10% of time calculation) and thus désactivables (via `ICNTL (11)`).

• For the Code_Aster user these parameters MUMPS are not directly accessible. They are displayed in a specific insert of the file of message (made out "monitoring MUMPS") if the keyword is informed `INFO=2` in the operator . In addition, this functionality is activated only if it chooses to estimate and to test the quality of its solution via the parameter `SOLVEUR/RESI_RELA` . According to the operators

Aster, this parameter is by default disconnected (negative value) or fixed at 10^{-6}. When it is activated (positive value), one tests if the direct error $fe(\mathbf{K},\mathbf{f})$ is quite lower than RESI_RELA. If that is not the case, calculation stops in ERREUR_FATALE by specifying the nature of the problem and the values accused.
• The activation of this functionality is not essential (but often useful) when the required solution itself is corrected by another algorithmic process (algorithm of Newton, diagram of Newmark): in short, in the linear operators THER_LINEAIRE, MECA_STATIQUE, STAT_NON_LINE, DYNA_NON_LINE ...
• This kind of functionality seems not very present in the bookstores: LAPACK, Nag, HSL...

2.3.4 Management memory (In-Core Out-Of-Core versus)

It was seen that the major drawback (cf. §1.7) direct methods lies in the face of factorized. **To make it possible to pass in random access memory larger systems, MUMPS proposes to discharge this object on disc**: it is the mode **Out-Of-Core** (keyword GESTION_MEMOIRE=' OUT_OF_CORE') in opposition to the mode **In-Core** (keyword GESTION_MEMOIRE=' IN_CORE') where all the structures of data reside in RAM (cf figure 2.2-1 and 2.3-2). This mode of economy of RAM is complementary to distribution of data which parallelism induces naturally. The appreciation of the OOC is thus especially prégnante for numbers of moderate processors (<32 processors).
In addition, team MUMPS was very attentive with overcost CPU generated by this practice. By working over again in algorithmic code handling of the discharged entities, they could limit to the strict minimum these overcosts (some for hundred and especially in the phase of resolution).

Figure 2.3-2. _Two types of management memory standards:
entirely in RAM ('IN_CORE') and RAM/disque ('OUT_OF_CORE').

These two memory ways of managing are "without net". No correction will be operated later on in the event of problem. If one does not know *a priori* not which of these two modes to choose and if one wants to limit, as far as possible, the problems due to defects of place memory, one can choose the automatic mode: GESTION_MEMOIRE=' AUTO'. Heuristic interns with *Code_Aster* manage then only the contingencies memory of MUMPS according to the computer set-up (machine, parallelism) and of the digital difficulties of the problem.
In the same order of idea, an option of the same keyword, GESTION_MEMOIRE=' EVAL', allows to gauge the needs for a calculation by displaying in the file message the resources memories required by calculation *Code_Aster*+MUMPS.

```
************************************************************************
 - 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 2.3-3. _Extrait of file of message with GESTION_MEMOIRE=' EVAL'.

Note:

• Parameters MUMPS ICNTL (22) /ICNTL (23) allow to manage these options memory. The user Aster indirectly activates them via the keyword SOLVEUR/GESTION_MEMOIRE.
• Unloading on disc is entirely controlled by MUMPS (many files, frequency unloading/recharging…). One informs just the site report: it is quite naturally the repertoire of work of the achievable specific Aster to each processor (%OOC_TMPDIR='. '). These files are automatically erased by MUMPS when associated occurrence MUMPS is destroyed. That thus avoids a clogging of the disc when various systems are factorized in the same resolution.
• Other strategies of OOC would be possible even are already coded in certain packages (PaStiX, Oblio, TAUCS…). One thinks in particular well aware of being able to modulate the perimeter of the discharged objects (cf phase of sometimes expensive analysis in RAM) and of being able to re-use them on disc during another execution (cf. CONTINUATION with the direction Aster or partial factorization).

2.3.5 Management of the singular matrices

One of the large strong point of the product is its **management of the singularities**. It is not only able of **to detect the digital singularities**[24] of a matrix and to synthesize information for an external use of it (calculation of row, warning to the user, posting of expert testimony.), but moreover, in spite of this difficulty, it calculates one **regular solution "**[25]**"** even whole or part of **associated core**.
These new developments were one of deliverable of the ANR SOLSTICE [GROUND]. We had asked them to team MUMPS (in partnership with the Algorithm team of the CERFACS) to make this product Iso-functional compared to the other direct solveurs of *Code_Aster*.
IN EDF, this functionality finds a second field of application with numerically singular modelings of code_Carmel3D. Except the iterative solutions already integrated into the code, MUMPS is probably one of the rare products armed to solve this kind of difficulties. Indeed, systems FCARMEL present systematically (because not measured modelings) of very large singularities: about 15% of the size of the problem.

On this point, the Carmel needs/Aster are complementary:

- for *Code_Carmel3D* it is a question of finding a solution possible of the problem,
- for *Code_Aster*, this situation is often regarded as pathological. One then wishes to inform the user of a problem in his setting in data (limiting condition, contact.) or to return a signal to algorithmic (refinement of the step of time…).

And in practice, how does MUMPS proceed you it?
With large features, during the construction of the factorized matrix, it detects the lines comprising of the pivots[26] very small (compared to criterion CNTL (3)[27]). It indexes them in the vector PIVNUL_LIST (1: INFOG (28)) and, according to the case, it is replaces them by a prefixed value (*via* CNTL (5)[28]), it is stores them except for. The block thus made up (moreover small) will undergo an algorithm QR later on *ad hoc*.

24 The singularities known as digital are given except for a digital precision, contrary to the singularity known as exact or true.

25 It is a possible solution of the problem since the second member $\mathbf{f} \in ker((\mathbf{K}^T)^T)$. What in our symmetrical case returns to $\mathbf{f}$ element of space image.

26 It acts, in any rigour, of the infinite standard of the line of the matrix of work comprising the pivot.

27 By default one fixes it at 10^{-8} (in double precision) and 10^{-4} (into simple) because these figures represent (empirically) a loss of at least half of the level of precision if factorization nevertheless is continued.

28 This value must be enough large to limit the impact of this modification on the rest of factorization. In Code_Aster/Code_Carmel3D, one fixes it at $10^6 \|\mathbf{K}_{travail}\|$.

And to finish, the iterative iterations of refinement come to supplement this hank. As they use this factorized "improved" only as preconditionnor, and that they profit, on the other hand, of the exact information of the product matrix-vector, they bring back[29] the solution "skewed" in the good way!

Note:

• Parameters MUMPS `ICNTL (13) /ICNTL (24) /ICNTL (25)` *and* `CNTL (3) /CNTL (5)` *allow to parameterize these features. They are not modifiable by the user. By prudence, one keeps the functionality activated permanently.*

• This functionality can also prove to be useful in modal calculation (filtering of the rigid modes).

29 It is the same mechanism as for the static swivelling.

3 Establishment in *Code_Aster*

3.1 Context/synthesis

To improve the performances of calculations carried out, **strategy retained by *Code_Aster*** [Dur08], as by most great codes general practitioners in mechanics of the structures, in particular consists in diversifying its panel of linear solveurs[30] in order to better target the needs and the constraints of the users: local machine, computer cluster or centre; obstruction memory and disc; time CPU; industrial or more exploratory study…
Dimensioned parallelism and linear solvor, a way is particularly prospected[31]:

- "**digital parallelism**" external bookstores of solveurs such that MUMPS and PETSc, possibly supplemented by a "data-processing parallelism" (intern with the code) for elementary calculations and the matric/vectorial assemblies;

We are interested here in the first scenario through MUMPS. **This external solvor "is plugé" in *Code_Aster* and accessible to the users since the v8.0.14**. It thus enables us to profit, "with less expenses", of Rex from a broad community of users and very pointed competences international teams. The whole while combining effectiveness, performance, reliability and broad perimeter of use.
This work was initially completed by exploiting the sequential mode In-Core product. In particular, thanks to its faculties of swivelling, he does invaluable favours by treating new modelings (quasi-incompressible elements, X-FEM…) who can prove to be problematic for the other linear solveurs.
Since, **MUMPS is daily used on studies** [GM08] [Tar07] [GS11]. Our Rex of course packed itself and we maintain one **partnership relation activates with the development team of MUMPS** (in particular *via* the ANR SOLSTICE [GROUND] and a thesis in progress). In addition, its integration in *Code_Aster* profit from a continuous enrichment: parallelism centralized IC [Des07] (since the v9.1.13), parallelism distributed IC [Boi07] (since the v9.1.16) then IC mode and OOC [BD08] (since the v9.3.14) .
In distributed parallel mode, the use of MUMPS gets **profits in CPU** (compared to the method by default of the code) **about the dozen on 32 processors** machine *Aster*. On very favorable cases this result can be much better and, for "studies borders", MUMPS remains sometimes the only viable alternative (cf interns of tank [Boi07]).
As for RAM consumption, one saw in the preceding chapters that it is the principal weakness of the direct solveurs. Even in parallel mode, where one however has naturally a distribution of the data between the processors, this factor can prove handicapping. To overcome this problem it possible to activate in *Code_Aster*, a recent functionality of MUMPS (developed within the framework of the above mentioned ANR): the "Outone" (OOC), during "In-Core" (IC) by default. It makes it possible to reduce this bottleneck by discharging on disc a good amount of data. Thanks to the OOC, one can thus **to approach RAM consumption of the native multifrontale** of *Code_Aster* (even into sequential), to even go down in lower part by combining the efforts from parallelism and this unloading on disc. The first tests show a profit in RAM between the OOC and the IC from at least 50% (even more on successful outcomes) for a overcost in limited CPU (<10%).
Solvor MUMPS thus allows, not only to solve numerically difficult problems, but, inserted in a computing process Aster already partially parallel, it gears down the performances of them. It gets for the code a framework parallel performing, credits, robust and general public. It facilitates thus the passage of the studies standards (< million degrees of freedom) and makes available to the greatest number the treatment of large cases (| several million degrees of freedom).

3.2 Two types of parallelism: centralized and distributed

3.2.1 Principle

MUMPS is a paralleled linear solvor. This parallelism can be activated several manners in particular according to the sequential or parallel aspects of the code which uses it. Thus, it **parallelism can be limited to floods of internal data/MUMPS treatments**, or, *a contrario* **to be integrated into a flood of data/parallel treatments**

30 This research continues improvement of the performances is obviously not reduced only to the only linear solveurs. The code proposes a good amount of tools to answer the same objectives: distribution of independent calculations, X-FEM, improvement of contact-friction, the EDO/modaux/non-linear solveurs, adaptive grid, finite elements of structure…

31 Cf [R6.01.03] for a detailed vision of the potential strategies of parallelism and those actually put in work in the code.

already organized upstream solvor, as of the elementary phases of calculations of *Code_Aster*. First mode ('CENTRALIZE') has for him the robustness and a broader perimeter D' use, the second ('GROUP_ELEM'/'MAIL_ ***' and 'SOUS_DOMAINE') is generic but more effective.
Because the most expensive phases often in time CPU of a simulation are: the construction of the linear system (purely *Code_Aster* , cut out in three stations: factorization symbolic system, calculations elementary and matric/vectorial assemblies) and its resolution (in MUMPS, cf §1.6: renumerotation + analyzes, digital factorization and descent-increase). The first mode of parallelization benefits only from the parallelism of stages 2 and 3 of MUMPS, whereas the three others parallel also elementary calculations and the assemblies of *Code_Aster (*cf figure 2.2-1 and 3.2-1*)* .

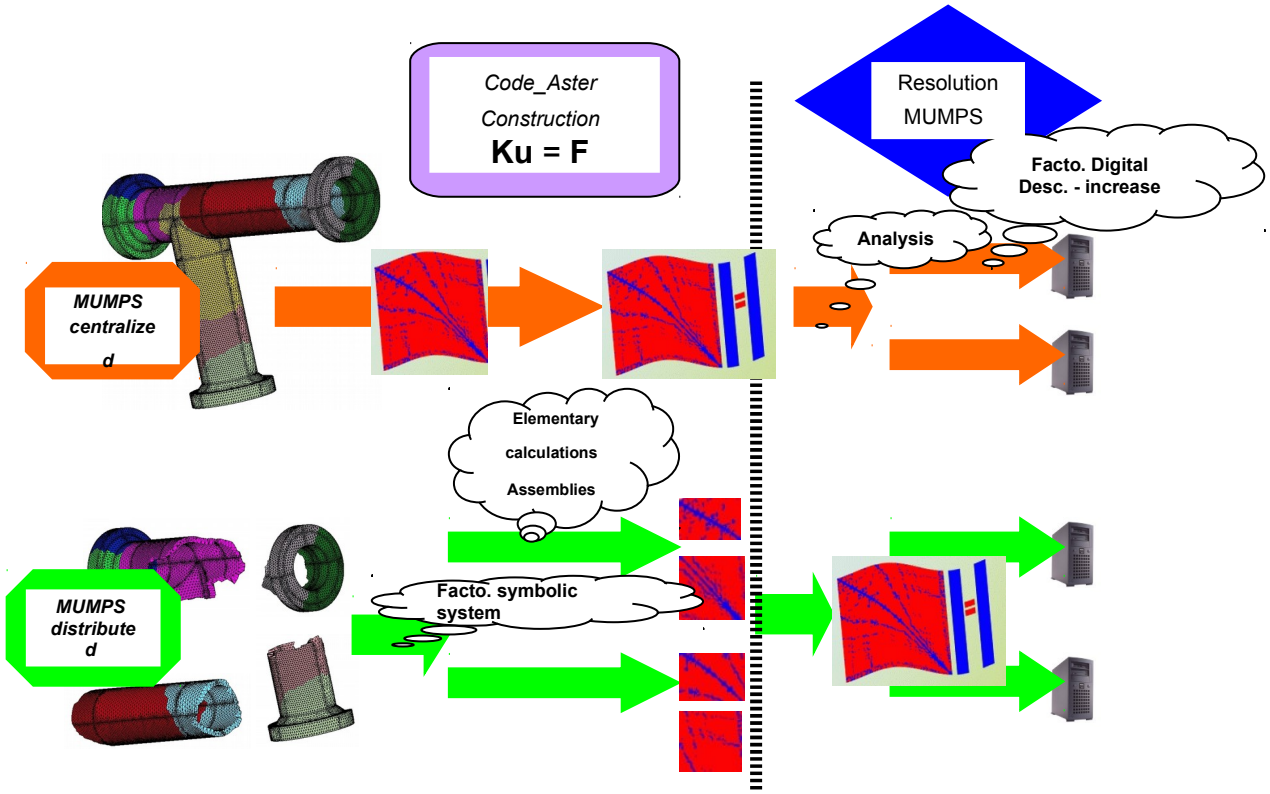


Figure 3.2-1. _ Floods of parallel data/treatments of centralized/distributed MUMPS.

3.2.2 Various modes of distribution

The four strategies of distribution need to organize a flood of "data/treatments" parallel by distributing the initial data. *Code_Aster* being a code finite elements, the natural distribution of data is that by group of meshs. This distribution parameter in the operator AFFE_MODELE. It can change in the course of calculation *via* the order MODI_MODELE.
With each assignment or modification of the model, these strategies thus distribute the meshs to the various processors. Maybe by packages of meshs ('GROUP_ELEM'/'MAIL_ ***'), that is to say by under-fields ('SOUS_MODELE') *via* a decomposition of the model under-fields built upstream of the operator (cf. DEFI_PARTITION).

In both cases, the processors fill only the pieces with matrices and of second members who their correspond then transmit them to MUMPS. This last assembles them in-house before carrying out the resolution itself. Let us detail each mode:

- **CENTRALIZE : The meshs are not distributed** (as into sequential). Each processor knows all the grid. The parallelism of elementary calculations/assemblies is thus not implemented. It starts only on the level of MUMPS. Each processor built and provides to the linear solvor the entirety of the system to be solved. This mode of use is useful for the tests of not-regression. In

all the cases where elementary calculations represent a weak share of total time (e.g. in linear elasticity), this option can be sufficient.

- '**GROUP_ELEM (defect)/MAIL_DISPERSE/MAIL_CONTIGU**': **parallelism begins, upstream of MUMPS**, as of the elementary phase of calculation of *Code_Aster*. Each processor allocates the whole matrix[32] (and structures of data *Aster* related NUME_DDL , MATR_ELEM…), calculates and fills only the terms of them *AD HoC* and provides the nonworthless values to MUMPS. This last then will gather them (for its phase of analyse+renumerotation) before carrying out the paralleled resolution of the system.
  The distribution of the meshs of the model is initiated, that is to say according to **type of mesh** ('GROUP_ELEM'), that is to say by **packages of contiguous meshs** ('MAIL_CONTIGU'), that is to say by **cyclic distribution** ('MAIL_DISPERSE'). For example, with a model comprising 8 meshs and for a calculation on 4 processors, there are the following distributions of load:

Mode of distribution	*Mesh 1*	*Mesh 2*	*Mesh 3*	*Mesh 4*	*Mesh 5*	*Mesh 6*	*Mesh 7*	*Mesh 8*
MAIL_CONTIGU	Proc. 0	Proc. 0	Proc. 1	Proc. 1	Proc. 2	Proc. 2	Proc. 3	Proc. 3
MAIL_DISPERSE	Proc. 0	Proc. 1	Proc. 2	Proc. 3	Proc. 0	Proc. 1	Proc. 2	Proc. 3

- 'SOUS_DOMAINE': parallelism similar to the preceding options, but this time the initial distribution of the meshs is based on one **decomposition under-fields** built upstream (*via* operators DEFI_PARTITION). For example, with a structure of data SD_PARTIT comprising 5 under-fields and a calculation on 2 processors: the meshs of under-fields 1 and 2 are assigned with the first processor, the meshs of the under-fields remaining with the second.

3.2.3 Balancing of load

The distribution by meshs is very simple but can lead to imbalances of load because she does not take explicitly account of the meshs spectators, of the meshs of skin (cf figure 3.2-2 on an example comprising 8 voluminal meshs and 4 meshs of skin), of particular zones (not linearities…). The distribution by under-fields is more flexible and can prove more effective while making it possible to adapt its flood of data to its simulation.
Another cause of déséquilibrage can come from the conditions of Dirichlet by dualisation (DDL_IMPO, LIAISON_ ***…). By preoccupations with a robustness, their treatment is affected only with the main processor. This extra work, often negligible, however introduced into certain cases, a déséquilibrage more marked. The user can compensate for it by informing one of the keyword CHARGE_PROC0_MA/SD. This differentiated treatment relates to in fact all the implying cases of the meshs known as "late" (Dirichlet *via* of Lagranges but also nodal force, contact method continues…).
For more details on the data-processing specifications and the functional implications of this mode of parallelism one will be able to consult documentations [U2.08.03] and [U4.50.01].

Note:

• Without the option MATR_DISTRIBUEE (cf following paragraph), different the strategies are equivalent in term of occupation memory. One dries up as soon as possible the flood of data and instructions. It is a question of treating matric/vectorial blocks selectively total problem, that MUMPS will gather.
• For the moment, the distribution by under-fields is based on a partitioning generated by the operator DEFI_PARTITION.
• In distributed mode, each processor handles only matrices partially filled. On the other hand, in order to avoid introducing too many communications MPI into the code (criteria of stop, residue…), this scenario was not retained for the second members. Their construction is well paralleled, but at the end of the assembly, the contributions of all the processors are summoned and sent to all. Thus all the processors entirely know the vectors implied in calculation.
• In the same way, the matrix for the moment is duplicated: in space JEVEUX (RAM or disc) and in F90 space of MUMPS (RAM). In the long term, because of unloading on disc of factorized, it will become a dimensioning object of RAM. It will thus have to be built directly via MUMPS.

32 Except if the option SOLVEUR/MATR_DISTRIBUEE (cf §3.2.4) is activated.

3.2.4 To recut the Code_Aster objects

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 3.2-2 illustrate this profit in parallel on two studies: a Pump LAUGH and the Epicure tank.

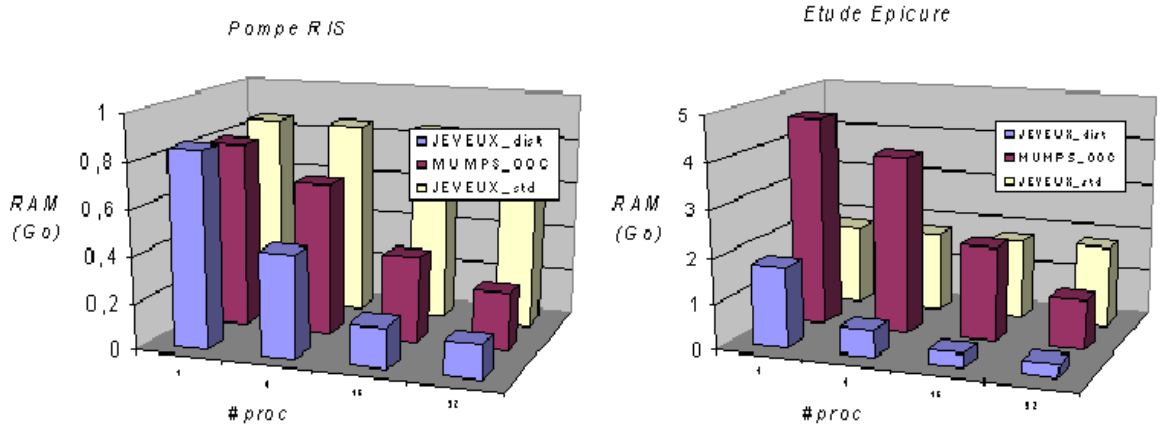


Figure 3.2-2. _ Evolution of RAM consumption (in Go) according to the number of processors, Code_Aster (JEVEUX standard MATR_DISTRIBUE=' NON' and distributed, resp. 'YES') and of MUMPS OOC. Results carried out on a Pump LAUGH and the tank of the Epicure study.

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.*

3.3 Management of memory MUMPS and Code_Aster

To activate or disable faculties OOC of MUMPS (cf figure 3.3-1)**,** the user informs the keyword SOLVEUR/GESTION_MEMOIRE=' IN_CORE'/'OUT_OF_CORE'/'CAR' (defect). This functionality is of course cumulable with parallelism from where a larger variety of operation for if required adapting to the contingencies of execution: “sequential IC or OOC”, “parallelism centralized IC or OCC”, “parallelism distributed by IC under-fields or OOC”…

For a small linear case, sequential mode the “IC” is enough; for a larger case always into linear, parallel mode the “centralized IC” (or better OOC) brings truly a profit in CPU and RAM; into nonlinear, with frequent reactualization of the tangent matrix, parallel mode the “distributed OOC” is advised.
For more details on the data-processing specifications and the functional implications of this way of managing of memory MUMPS one will be able to consult documentations [BD08] and [U 2.08.03/U4.50.01].

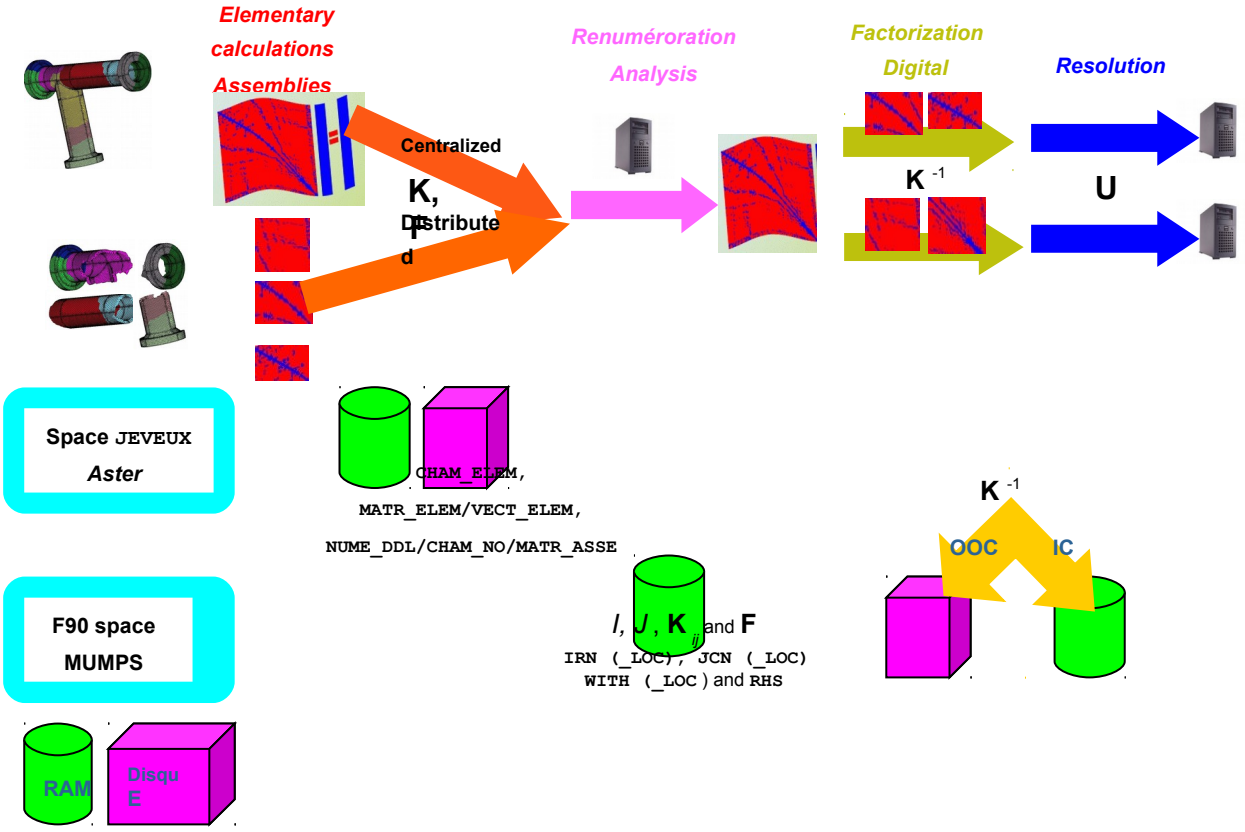


Figure 3.3-1. _Diagram functional of the Code_Aster/MUMPS coupling
with respect to the principal structures of data and occupation memory (RAM and disc).

3.4 Particular management of the Lagranges double

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 $\mathbf{K}_0$

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

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

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

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 the keyword ELIM_LAGR with 'LAGR2', **one does not take any more account but of one Lagrange, the other being spectator**[33]. From where a matrix of work 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 the number of late swivellings that MUMPS must carry out. To impose ELIM_LAGR=' NON' can be then very interesting (profit of 40% in CPU on the CAS-test mac3c01).

3.5 Perimeter of use

A priori , all operators/features using the resolution of a linear system except the options [34] of calculation ' SEPARATED 'and' ADJUST 'of CALC_MODES . For more details one will be able to consult the user's documentations [U4.50.01].

3.6 Parameter setting and examples of use

Let us recapitulate the principal parameter setting allowing to control MUMPS in *Code_Aster* and let us illustrate its use *via* an official CAS-test (mumps05b) and a geometry of study (pump LAUGH). For more information one will be able to consult the associated user's documentations [U 2.08.03/U4.50.01], the notes EDF [BD08] [Boi07] [Des07] or CAS-tests using MUMPS.

3.6.1 Parameters of use of MUMPS *via Code_Aster*

Operand	Keyword	Value by default	Details/advices	Ref.
SOLVEUR/ METHODE= 'MUMPS'				
Functional parameters	TYPE_RESOL	'CAR' ('NONSYM' if the matrix is nonsymmetrical, 'SYMGEN' if not)	'AUTO', 'NONSYM', 'SYMGEN' and 'SYMDEF'. Parameter allowing to specify the nature of the problem to be treated.	§1
	PCENT_PIVOT	10%	Overcost report planned for the swivellings.	§2.3
	ELIM_LAGR	'LAGR2'	'LAGR2' / 'NOT'.	§3.4

33 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 K_2 with K_1, rather than with the optimal scenario K_0.

34 Because MUMPS does not provide yet the calculation of the determinant of a matrix.

Operand	Keyword	Value by default	Details/advices	Ref.
	RESI_RELA	-1 (nonlinear) 10^{-6} (linear)	If this parameter is positive, MUMPS carries out iterations of iterative refinement and examines the quality of the solution. If the relative error in solution is lower than this value, Aster stops in ERREUR_FATALE.	§2.3
Digital parameters	PRETREATMENTS	'CAR'	'CAR' and 'WITHOUT'.	§ 1.6 § 2.3
	RENUM	'CAR'	'CAR', 'AMD', 'MFA', 'QAMD', 'PORD', 'SCOTCH TAPE' and 'MONGREL'. In first case MUMPS chooses the best renumerotor available, in others, one imposes to him. If this renumerotor is not available: ERREUR_FATALE.	§1.6
	FILTRAGE_MATRICE /MIXER_PRECISION		Options "to release" the resolutions carried out *via* MUMPS.	[U4.50.01]
	POSTTRAITEMENTS	'CAR'	'CAR','FORCE' and 'WITHOUT'.	§2.3
Memory	GESTION_MEMOIRE	'CAR'	'IN_CORE', 'OUT_OF_CORE', 'CAR' or 'EVAL'.	§ 3.3
	MATR_DISTRIBUEE	'NOT'	'YES' or 'NOT'.	§ 3.2

Table 3.6-1. _Summary of the parameter setting of MUMPS in Code_Aster.

3.6.2 Monitoring

By positioning it **keyword INFORMATION to 2 and by using solvor MUMPS** , the user can make display in the file of message a synthetic monitoring of the various phases of construction and resolution of the linear system: distribution by processor amongst meshs, of the terms of the matrix and of its factorized, the analysis of error (if requested) and an assessment of their possible déséquilibrage. With **this monitoring directed CPU, one adds some information on RAM consumption of MUMPS**: by processor, estimate (according to the phase of analysis) of the requirements in RAM in IC, OOC and the value actually used with recall of the strategy chosen by the user. The times spent for each stage of calculation following the processors can appear too. They are managed by a more total mechanism which is not specific to MUMPS (cf. §4.1.2 [U1.03.03] or the user's documentation of the operator DEBUT/POURSUITE).

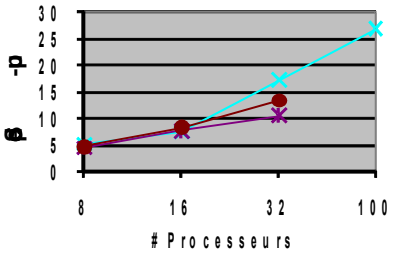
```
****************************************************************
  <MONITORING MUMPS >
  SIZE OF THE SYSTEM    803352
 CONDITIONNEMENT/ERREUR ALGORITHM  2.2331D+07 3.3642D-15
 ERROR ON THE SOLUTION  7.5127D-08
  ROW      NO. MESHS      NO. TERMS K      LU FACTORS
  NR    0:           54684          7117247        117787366
  NR    1:           55483          7152211         90855351
…
   IN %: RELATIVE VALUE AND DESEQUILIBRAGE MAX
         :  1.45D+01   2.47D+00   2.38D+00         1.50D+01   4.00D+01   2.57D+01
         :  1.40D-01  -1.09D+00  -5.11D+00        -9.00D-02   1.56D+00  -4.16D-01

   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:             1854              512              512
  NR    1:             1493              482              482
…
#1 Resolution of the linear systems           CPU (USER+SYST/SYST/ELAPS):    105.68      3.67     59.31
#1.1 Classification, connectivity of the matrix    CPU (USER+SYST/SYST/ELAPS):        3.26        0.04
3.26
#1.2 Factorization symbolic system                    CPU (USER+SYST/SYST/ELAPS):       3.13       1.20
4.11
#1.3 digital Factorization (or precond.)      CPU (USER+SYST/SYST/ELAPS):     45.22      0.83     23.48
#1.4 Resolution                                 CPU (USER+SYST/SYST/ELAPS):     54.07      1.60     28.46
#2 Elementary calculations and assemblies            CPU (USER+SYST/SYST/ELAPS):       3.44        0.03
3.42
```

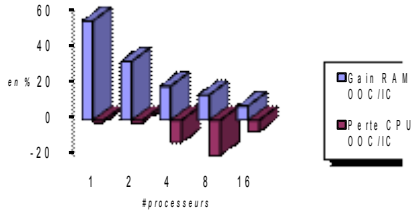
```
#2.1 Routine calculation                              CPU (USER+SYST/SYST/ELAPS):       2.20       0.01
2.20
#2.1.1 Routines te00ij                          CPU (USER+SYST/SYST/ELAPS):      2.07      0.00      2.06
#2.2 Assemblies                                 CPU (USER+SYST/SYST/ELAPS):      1.24      0.02      1.22
#2.2.1 Assembly matrices                        CPU (USER+SYST/SYST/ELAPS):      1.22      0.02      1.21
#2.2.2 Assembly second members                  CPU (USER+SYST/SYST/ELAPS):      0.02      0.00      0.01
```

Figure 3.5-1. _Extrait of file of message in INFORMATION =2.

3.6.3 Examples of use

Let us conclude this chapter by two series of tests illustrating them **variations of performance according to the case and the criterion observed** (cf figures 3.5-2/3). The canonical CAS-test of the cube into linear is paralleled very well. In centralized (resp. in distributed), more than 96% (resp. 98%) of the phases of construction and inversion of the system linear are paralleled. That is to say a theoretical speed-up near to 25 (resp. 50). In practice, on the parallel nodes of the centralized machine *Aster*, rather good accelerations are obtained: effective speed-up of 14 out of 32 processors instead of the 17 theoretical ones.
On the nonlinear study of the pump, the profits which one can hope for are weaker. Taking into account the phase of sequential analysis of MUMPS, only 82% of calculations are parallel. From where of appreciable but more modest speed-UPS theoretical and effective. From a point of view RAM report, management OOC of MUMPS gets profits interesting in the two cases, but more marked for the pump: into sequential, profit IC vs OOC of about 85%, against 50% for the cube. By increasing the number of processors, distribution of data which parallelism bad temper induces gradually this profit. But it remains prégnant on the pump to 16 processors and disappears almost with the cube.





$N = 0.7\,M$ / $nnz = 27\,M$
%parallèle centralized/distributed: 96/98

theoretical Speed-UPS hundred. /dist. <25/50
32 proc (x1): ~3min
Consumption. RAM IC: 4Go (1 proc)/1.3Go (16)
Consumption. RAM OOC: 2Go (1 proc)/1.2Go (16)

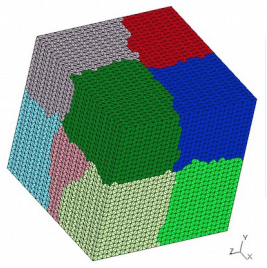


Figure 3.5-2. _Linear mechanical Calculation (op. MECA_STATIQUE) on the official CAS-test of the cube (mumps05b). And solved only one linear system is built. Simulation carried out on the centralized machine Aster (Bull). Consumption measured RAM Aster +MUMPS.

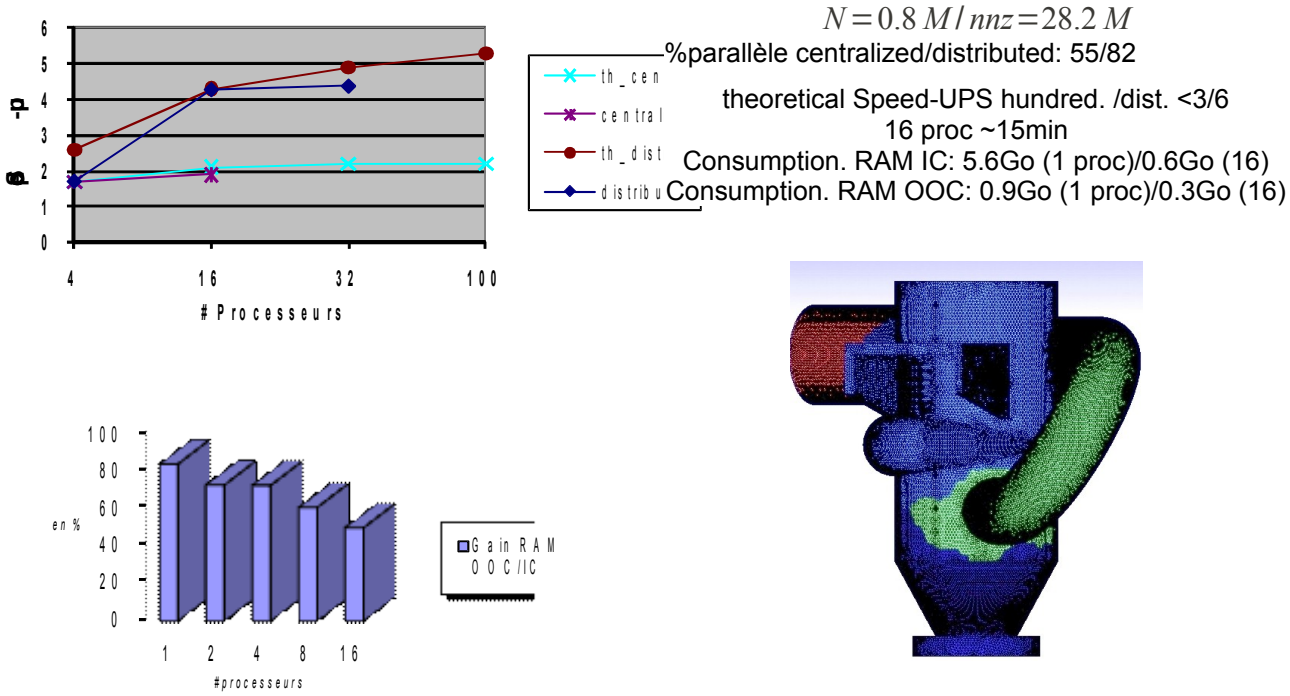


Figure 3.5-3. _Nonlinear mechanical Calculation (op. STAT_NON_LINE) on a geometry more industrial (pump LAUGH). And solved 12 linear systems are built (3 pas de time X 4 pas de Newton). Simulation carried out on the centralized machine Aster (Bull). Consumption estimated RAM MUMPS.

4 Conclusion

Within the framework of thermomechanical simulations with *Code_Aster*, **the main part of the costs calculation often comes from the construction and the resolution of the linear systems**. For 60 years, two types of techniques have disputed supremacy in the field, the direct solveurs and those iterative. *Code_Aster*, like a good amount of codes general practitioners, made the choice of an offer diversified in the field. With however **an orientation rather hollow direct solveurs**. Those are adapted to its needs than one can summarize under the triptych "robustness/problems of the multiple type second members/moderate parallelism". The code resting from now on on many "middlewares" optimized and perennial (MPI, BLAS, LAPACK, MONGREL…) and being used mainly on clusters of SMP (fast networks, great RAM storage capacity and disc), one seeks to optimize the linear solveurs salary accordingly.

Taking into account necessary technicality[35] and of a plethoric international offer[36], **for to effectively carry out these resolutions, the question of resorts to an external product is from now on impossible to circumvent**. That makes it possible to acquire, with less expenses, a functionality often effective, reliable, powerful and profiting from a broad perimeter of use. One can thus profit from the experience feedback from a broad community of users and competences (very) pointed international teams.
Thus ***Code_Aster* made the choice to integrate the parallel multifrontale of package MUMPS**. This in complement, in particular, of its multifrontale "house". But if this one profits from a long-term adaptation to modelings *Aster,* it remains less rich in features (swivelling, pre/postprocessings, quality of the solution…) and less powerful in parallel (for RAM consumption of the same order). To exploit certain modelings (quasi-incompressible elements, X-FEM…) or to pass from the "studies borders" (cf interns of tanks), this coupling "*Code_Aster*+MUMPS" becomes sometimes the only viable alternative.

Since, its integration in *Code_Aster* profit from a continuous enrichment and **MUMPS** (`SOLVEUR/METHODE='MUMPS'`) **is daily used on studies** . Our Rex of course packed itself and we maintain one **partnership relation activates with the "core-TEAM" of MUMPS** (in particular *via* the ANR SOLSTICE and a thesis).
In mode **parallel**, the use of MUMPS gets **profits in CPU** (compared to the method by default of the code, the multifrontale "house") **about the dozen on 32 processors** machine *Aster*. On more favorable cases or by exploiting a second level of parallelism *via* the BLAS, this profit CPU perhaps much better.

Solvor MUMPS thus allows, not only to solve numerically difficult problems, but, inserted in a computing process *Aster* already partially parallel, it gears down the performances of them. **It gets for the code a framework parallel performing, credits, robust and general public** . It facilitates thus the passage of the studies standards (< million degrees of freedom) and makes available to the greatest number the treatment of large cases (| several million degrees of freedom).

35 To give an order of magnitude, package MUMPS makes more than 10^5 lines (F90/C).
36 Only in the public domain, one counts tens of packages, bookstores, "macro-bookstores"…

5 Bibliography

5.1 Books/articles/proceedings/theses…

[ADD89] M.Arioli, J.Demmel and I.S . Duff. Solving sparse linear systems with sparse backward error. SIAM newspaper one matrix analysis and applications . 10,165:190 (1989).
[ADEL00] P.R.Amestoy, I.S.Duff, Excellent J.Y.L' and X.S.Li. General Analysis and comparison of two sparse solvers for distributed memory computers . Report CERFACS TR/PA/00/90 (2000).
[ADE00] P.R.Amestoy, I.S.Duff and J.Y.L' Excel. Multifrontal parallel distributed symmetric and unsymmetric solvers . Comput. Methods in Appl. Mech. Eng. 184,501:520 (2000).
[ADKE01] P.R.Amestoy, I.S.Duff, J.Koster and J.Y.L' Excel. With fully asynchronous multifrontal solvor using distributed dynamic scheduling . SIAM newspaper of matrix analysis and applications, 23,15:41 (2001).
[AGES06] P.R.Amestoy, A.Guermouche, Excellent J.Y.L' and S.Pralet. Hybrid scheduling for the parallel solution of linear systems . Parallel computing. 32,136:156 (2006).
[Che05] K.Chen. Matrix preconditioning technical and applications . ED. Cambridge University Close (2005).
[Dav06] T.A.Davis. Direct methods for sparse linear systems . ED. SIAM (2006).
[Duf06] I.S.Duff et al. Direct methods for sparse matrices . ED. Clarendon Close (2006).
[GGS08] T.George, A.Gupta and V.Sarin. Experimental year iterative of evaluation solvers for broad SPD systems of linear equations. Report of the research centre IBM T.J.Watson (2008).
[GHS05] N.Gould, Y.Hu and J.A.Scott. With numerical evaluation of sparse direct solvers for the broad solution of sparse, symmetric linear systems of equations . Report of Rutherford Appleton Laboratory (2005).
[Gol96] G.Golub & C. Van Loan. Matrix computations . ED. Johns Hopkins University Close (1996).
[Gup01] A.Gupta. Recent advances in direct methods for solving unsymmetric sparse systems of linear equations. Report of the research centre IBM T.J.Watson (2001).
[Hig02] N.J.Higham. Accuracy and stability of numerical algorithms . ED. SIAM (2002).
[Las98] P.Lascaux & R.Théodor. Matric digital analysis applied to the art of the engineer. ED. Masson (1998).
[Liu89] J.W.H.Liu. Broad computer solution of sparse positive definite systems . Prentice Hall (1981).
[Meu99] G.Meurant. Broad computer solution of linear systems . ED. Elsevier (1999).
[Saa03] Y.Saad. Iterative methods for sparse matrices . ED. PWS (2003).

5.2 Account-returned reports/EDF

[Anf03] N.Anfaoui. A study of the performances of Code_Aster: proposal for an optimization. Internship of mathematics applied of PARIS VI (2003).
[BD08] O.Boiteau and C.Denis. Activation of the “Out-Of-Core” features of MUMPS in Code_Aster. Report EDF R & D CRY 8/23/047 (2008).
[Boi07] O.Boiteau. Integration of parallel MUMPS distributed in Code_Aster. Note EDF R & D HI-I 7/23/03167 (2007).
[BHV06] O.Boiteau, F.Hulsemann and X.Vasseur. Comparison of linear solvor MUMPS and the multifrontale of Code_Aster. Note EDF R & D HI- 6/23/004 (2006).
[Des07] T.DeSoza. Evaluation and development of parallelism in Code_Aster. Internship of Master degree ENPC (2007) and notes EDF R & D HT-62/08/01464 (2008).
[Dur08] C.Durand and al. HPC with Code_Aster: prospect and inventories of fixtures. Note EDF R & D HT-62/08/0139 (2008).
[GM08] S.Géniaut and F.Meissonnier. Feasibility of a study of harmfulness of crack in a valve MP by method X-FEM and with the platform SALOMÉ. Report EDF R & D CR-AMA/08/0255 (2008).
[GS11] V.Godard and N.Sellenet. Calculation HPC with Code_Aster: prospect and inventory of fixtures. CR-AMA-11.042 (2011).
[Tar07] N.Tardieu. GCP+MUMPS, a simple solution for the resolution of problems with contact in parallel. Report EDF R & D CR-AMA/07/0257 (2007).
[GROUND] O.Boiteau. Follow-up of the ANR SOLSTICE. Report EDF R & D, slides… on the Notes basis dedicated of department SINETICS.

5.3 Resources Internet

[Dav] T.A.Davis. Pointer on the packages of direct hollow solveurs: http://www. cise.ufl.edu/research/sparse/codes/.
[Gift] J.Dongarra. Pointer on the packages of solveurs: http://www.netlib.org/utk/people/JackDongarra/la-sw.html.
[MaMa] Web site of MatrixMarket: http://math.nist.gov/MatrixMarket/index.html.
[Mum] Official Web site of MUMPS: http://graal.ens-lyon.fr/MUMPS.
[Not] Official Web site of PaStiX: http://pastix.gforge.inria.fr/files/README-txt.html.

6 History of the versions of the document

Version Aster	Author (S) or contributor (S), organization	Description of the modifications
9.4	O.BOITEAU EDF R & D SINETICS	Initial text
V10.4	O.BOITEAU EDF R & D SINETICS	Formal corrections due to the bearing .doc/.odt; Update on parallelism; **Taking into account of the rqs of team MUMPS**; Addition of new keyword (ELIM_LAGR2, LIBERE_MEMOIRE, MATR_DISTRIBUEE); **Slimming of the council part/perimeter of use now reserved for the U2.08.03 note**.
V11.3	O.BOITEAU EDF R & D SINETICS	Addition of the new keyword GESTION_MEMOIRE instead of OUT_OF_CORE and LIBERE_MEMOIRE. Addition of the paragraph on the taking into account of singular systems.