Implementation of the method multifrontale MULT_FRONT

Summary:

One presents in this note, the data-processing description of the native version in Code_Aster, of the linear solvor multi-frontal, called commonly MULT_FRONT.

This version, developed at EDF R & D is stabilized, with an almost complete perimeter.

This note supplements the reference material of Code_Aster: “Linear Solveur by the multi-frontal method” [R6.02.02].

One points out the principal concepts used by the method: renumerotation, tree of elimination, super-nodes. One describes the data-processing structures created by the algorithm. The data-processing main difficulties and characteristics are announced. The various possible developments are considered.

A report published except for is dedicated in an inventory of fixtures and the prospects for application maintenance: [RB12] “native multi-frontal Method in Code_Aster: prospect and inventory of fixtures”, CR-I23-2012-001.

Contents

Contents

1 Introduction

One describes in this note the data-processing implementation of the solvor multifrontal, in C*ode_Aster*. It is intended for the developers of the code and is not a “monopiece” document.

This method was developed in the shape of prototype within group ISA [RO93], and was in 1993 introduced then into *Code_Aster* in 1994, in a form adapted to this code:
- Factorization of a matrix not necessarily in main memory (“out of core”), based on software package JEVEUX.
- Treatment of the boundary conditions with multipliers of Lagrange.
- One supposed moreover not to need to activate a search for pivot.

Initially, only a sequential version for symmetrical real matrix was implemented in *Code_Aster*.

Thereafter the not-symmetrical and complex versions were introduced.

Complete parallelism was developed in prototype [RO95], it was not retained for the version developed in Code_aster. The reasons are the following ones:
- The top priority at the time was to optimize the memory requirements in order to treat linear systems of big size. However parallelism requires more memory than the sequential one.
- The parallelism of multi-frontal requires a management of the specific memory parallel to this method, which should have cohabited with software package JEVEUX, intrinsically sequential.
- Parallelism was not either a priority at that time where one had only calculators slightly parallel.

One will find a base theoretical on the method in [DU86], [AS87], [RO93] & [RO95], like in the reference material of *Code_Aster* : “Linear Solveur by the Multi-Frontal method”, [R6.02.02].

The version double precision REAL*8 (resp. COMPLEX*16) is carried out by the call to routine FORTRAN MULFR8 (resp. MLFC16).

Thereafter one will describe in detail the version for symmetrical matrix of real variable, of the paragraphs will be devoted to the versions nonsymmetrical and complex. For more details on the structures of data of C*ode_Aster*, one will refer to documentations [D4.06.07] (`sd_ nume_ddl`) and [D4.06.10] (`sd_matr_asse`).

1.1 Principles generals.

One wants to replace the factorization of a hollow matrix, by a continuation of factorization of full matrices. It is the Cartesian principle: to replace a difficult problem by a set of simple problems.

For that one initially has an algorithm of renumerotation of the variables of the system. This algorithm (of minimum degree, for example), makes it possible to minimize the order of the full matrices to factorize, it is the first axis of effectiveness of the method.

Moreover, of this renumerotation, one can extract the two following products:
- A tree of elimination, obtained starting from the “hollow” (sparsity) of the initial matrix and optimum filling resulting from the renumerotation. This graph (in the shape of tree structure) provides the order of elimination of the variables and induces parallelism. (Indeed a node “son” must be eliminated before a node “father”, but all the “sons” of the same “father” can be eliminated independently, i.e. in parallel.) This parallelism is the second axis of effectiveness of the method.
- Concept of super-node. This concept was conceived, initially, to minimize the cost (considerable) of the renumerotation. While simplifying, one calls super-node, a set of nodes (within the meaning of the graph theory), having the same neighbors. The tree structure quoted above, in fact is formed by super-nodes whose size increases when one goes up in the tree structure, this increase is an illustration of the filling due to the method of elimination of Gauss. This regrouping has the following interest: one eliminates the variables by group, and not one by one, thus allowing the use of effective methods per block (using BLAS from level 2 or 3, produced matrice*vector or matrice*matrice). This work per block is the third axis of effectiveness.

In short, the 3 tension fields of the method are:
- A renumerotation,

- A tree of elimination,
- A regrouping of the variables.

Digital factorization proceeds then as follows:

One traverses the tree of elimination, with each stage (elimination of the super-node), one lays out of a matrix full partner with the super-node and with his neighbors, one carries out the elimination of Gauss limited to the variables of the super-node. One thus obtains on the one hand the columns of factorized final (variables of the super-node), and a frontal matrix modified and reduced to the neighbors ("complement of Schur"), who will be assembled in the frontal matrix of the variables of the super-node "father".

1.2 Programming of the method.

In general, the programming of the multi-frontal method is divided into 2 phases:
- a phase of factorization known as "symbolic system", including the renumerotation,
- a phase of digital factorization itself.

In *Code_Aster*, the renumerotation and the first phase are carried out by the call to subroutine MLTPRE (called via MLFC16 and MULFR8, in the routines "hat" TLDLG2 and TLDLG3).

The second phase is carried out via the call to these routines MLFC16 and MULFR8, in the complex cases or realities.

2 Renumérotation and factorization symbolic system: MLTPRE

By convenience, the prefix was omitted NAKED, in front of the names of structure.

Structures of data read.
Structure of data: NUME_DDL, the recordings are extracted:

NUMÉRIQUE.DELG : indicate if the degree of freedom is of "Lagrange" or not.
NUMÉRIQUE.DEEQ ,
NUMÉRIQUE.NUEQ,
NUMÉRIQUE.PRNO : These 3 recordings establish the links between the degrees of freedom subjected to a boundary condition and corresponding Lagranges.
The 3 following recordings describe a matrix "Morse".
SMOS.SMHC : Number of column of the term
SMOS.SMDI : Addresses of the diagonal terms
SMOS.SMDE. : Many unknown factors, and of terms of the matrix

Structures of written data.
(Opposite each structure *Code_Aster*, one wrote the name of the corresponding table FORTRAN, which will be used, thereafter for more conveniences.)
MLTF.ADNT ADINIT : addresses of the initial terms in the factorized matrix
MLTF.LGBL LGBLOC : length of the blocks of the factorized matrix
MLTF.LGSN LGSN : length of the super-nodes
MLTF.LOCL ROOM : used the assembly of the frontal matrices, correspondence enters them local numbers of the super-nodes father and son.
MLTF.ADRE ADRESS
MLTF.SUPN SUPND : definition of the super-nodes
MLTF.FILS SON : son of the super-nodes in the tree of elimination
MLTF.FRER BROTHER : brother of the super-nodes in the tree of elimination
MLTF.LGSN LGSN : length of the super-nodes
MLTF.LFRN LFRONT : order of the frontal matrices (after elimination)
MLTF.NBAS NBASS : pointer related to the assembly of the frontal matrices

MLTF.ADPI ADPILE : addresses in the pile of the frontal matrices
MLTF.ANCI ANC : table reverses renumerotation
MLTF.NBLI NBLIGN : order of the frontal matrices (before elimination)
MLTF.NCBL NCBLOC : many degrees of freedom of each block
MLTF.DECA DECAL : shift between the beginning of column of a super-node and the beginning of the block JEVEUX which contains it
MLTF.SEQU SEQ : order of elimination of the super-nodes (course of the tree structure).
These structures of data will be described in paragraphs following.

Description of the code.
One distinguishes 2 phases, the first consists of the renumerotation, the second in what one calls in the language of the multi-frontal method, factorization symbolic system.

The structure of data NUME_DDL described a classification of NEQ degrees of freedom, including, most of the time, of the degrees of freedom known as of Lagrange. These degrees of freedom must satisfy the following relation of order:
(1) $L1<U<L2$, if $L1$ and $L2$ are the "Lagranges" associated with the boundary condition on U .

In order to preserve this order, one will subject to the algorithm of renumerotation only degrees of freedom not "Lagranges". The degrees of freedom "Lagranges are thus removed and information with regard to them is stored in order to reinstate them in new classification, while satisfying (1).

Routines called (the first brief description of the code).

For the first phase, them routines called are the following ones:
PREML0 : extraction of information on the "Lagranges"
PREMLA : treatment for the "Lagranges" of standard linear relation
PREML1 : pre treatment and renumerotation.
PREMLC : postprocessing of the renumerotation (addition of the "Lagranges").
PREMLD : idem

For the second:
PREML2 : factorization symbolic system.

2.1 PREML0

One stores in the tables $LBD1$, $LBD2$, the "Lagranges" of blocking, and in RL the "Lagranges" of linear relation.

One obtains a new classification of 1 with $N2$ degrees of freedom without Lagrange, defined by:
P of $[1:\mathrm{NEQ}]\Rightarrow[1-N2]$, and Q Opposite of P .

For I of 1 with NEQ , if I is a blocked degree of freedom, $LBD1(I)$, (resp. $LBD2(I)$), is the number of its L1 (resp. $L2$).

In the case of NRL linear relations, one will have:
$RL(1,I)$: $L1$ relation I ,
$RL(2,I)$: $L2$ relation I .

N.B. Programming
In certain routines, the name of variable will be used $N1$ in place of NEQ .

2.2 PREMLA

For all the degrees of freedom $L2$ of linear relation, (second "Lagrange"), its neighbors of lower number are known by extracting information from the structure of data SMOS.SMHC (table COLLAR).

From this information, PREMLA create chained lists (tables DEB., SEE, FOLLOWS) for the degrees of linear freedom of relation $L2$.
That is to say J a degree of freedom included in a linear relation, $j0=DEB(j)$ is the address of the first neighbor of J in the table $VOIS$, $J1=SUIV(J0)$ is the address of the following neighbor, $J2=SUIV(J1)$ is the address of the following and so on.

This chained list will be used thereafter in PREML1, for the manufacturing of $ADJNCY$.

N.B. Memory allocation
The chained list consists of 2 local structures, of names NOMVOI and NOMSUI, (created by WKVECT and destroyed in the end of MLTPRE).
Their length: $LGLIST$, is estimated a priori by PREML0 (parameter of exit LT), if it is insufficient, PREMLA provides an error code, and is called again with structures length 2 times larger.

2.3 PREML1

PREML1 call the following routines:
PRMADJ, GENMMD, AMDBAR, ONMETL.

The crucial role of PREML1 is of launching the renumerotation. This one, carried out with the choice by: MONGREL (method of dissection), GENMMD (minimum dismantles) or AMDBAR (minimum dismantles improved), requires 2 preliminary actions.

2.3.1 Reconstitution of the nodes (within the meaning of the discretization by finite elements).

In a first version of the code, the algorithm of renumerotation worked on $N2$ degrees of freedom (without Lagranges).

One then wanted to preserve the internal order of the unknown factors within the nodes of discretization finite element (for example ux , uy , uz , p).

A development was carried out for this purpose. **One from now on to the algorithm of renumerotation the whole of the nodes subjects and not them $N2$ degrees of freedom**.

One thus forms a new whole of $NBND$ associated nodes and 2 pointers:
$NŒUD[1:N2]\Rightarrow[1:NBND]$, and the pointer opposite $DDL[1:NBND]$

There are thus 3 classifications:
- Initial classification of 1 with NEQ ,
- That of the degrees of freedom without Lagranges 1 with $N2$,
- That of the nodes of interpolation 1 with $NBND$.

Provided with the tables P and Q between the first and the second, $NŒUD$ and DDL between the second and the third.

2.3.2 Creation of the structure ($ADJNCY$, $XADJ$)

This structure is the standard structure of data of the algorithms of renumerotation referred to above. There are 2 tables $ADJNCY$ and $XADJ$ defined as follows:

That is to say a node I , its neighbors are stored in the table $ADJNCY$, between the addresses $XADJ(I)+1$ and $XADJ(I+1)$.

One creates initially a structure $(ADJNCY, XADJ)$ relative to $N2$ degrees of freedom, then the routine PRMADJ transforms into a structure $(ADJNCY, XADJD)$ relative to $NBND$ nodes.
To create $ADJNCY$, the chained list is used $(DEB, VOIS, SUIT)$ created by PREMLA, by forcing the degrees of freedom of a linear relation to being close. Thus these degrees of freedom will be in the same super node, just as them $L2$. The relation of order will be thus respected.

One carries out then the call to GENMMD, AMDBAR or MONGREL. One gets the usual results of this algorithm:
1) Renumérotation,
2) Definition of the super-nodes,
3) Tree structure

The sources of these 3 methods are in the public domain.
Concerning the last 2 points, they should have been adapted. The renumerotation relates to the nodes (1 with $NBND$), it has the form of 2 tables opposite one of the other: $INVPND$, $PERMND$. ($INVPND(I)$ is the new number of the node I).

The tree structure, as for it, relates to the super-nodes created by the renumerotation. The super-node SND is defined as follows: it is made up by the segment of nodes $[SPNDND(SND-1)+1, SPNDND(SND)]$. $SPNDND$ is thus an injection of the segment $[1:NBSND]$ towards $[1:NBND]$, $NBSND$: many super-nodes.

The tree structure is defined as follows: $PARENT(SND)$ "father" of the super-node is the node SND.

N.B. Programming
The tree structure of the super-nodes of multi-frontal, appears under 2 names: $PARENT$ and $PAREND$
The first contains the tree structure of SN except Lagrange, it is provided by the renumerotation (GENMMD/AMDBAR/METIS.)
The second is supplemented first by adding to it Lagranges, it is calculated by PREMLC starting from the first.

In the appealing program MLTPRE, $PARENT$, who is an intermediary, is stored in ZI with the address SEQ, re-used later (space saver).

These preceding tables are defined according to $NBND$ nodes of discretization of *Code_Aster*, (1 with $NBND$). They are rebuilt then in the classification of $N2$ degrees of freedom not "Lagrange".
$INVPND$ becomes thus $INVP$, $PERMND$ becomes $PERM$ and $SPNDND$ becomes $SUPND$. (Cf loops 400,405,406 407 of PREML1).

N.B. Memory allocation

The structure $(XADJ, ADJNCY)$ of PREML1, is called $(XADJ2, ADJNC2)$ in MLTPRE. It is created front PREML1 and destroyed afterwards. The length of $ADJNC2$, $LGADJN$, is estimated at $2\times(lmat-\mathrm{neq}+lglist)$, with $lglist$ = length of the list of the neighbors used in PREML0
$lmat$ = length of the initial matrix (provided by SMOS).
If $LGADJN$ is insufficient, PREML1 provides the length necessary of it codes return, and is started again.

NB. MONGREL

Previously, achievable software was produced MONGREL, written out of C, compiled and linké separately. This achievable was called through APLEXT, layer of call of achievable external by Code_Aster. From now on routines of MONGREL are compiled and called by a call to ONMETL, appealing function C.

2.4 PREMLC

At exit of PREML1, the results are expressed in term of degrees of freedom not "Lagrange" (1 with $N2$), routine PREMLC supplements classification of 1 with $N2$ by adding the "Lagrange" and by creating new super-nodes.

The “Lagrange” are included thus in new classification:
For each $L1$, a new super-node is created SN, constituted of only one degree of freedom ($L1$), and son of the first degrees of freedom it is the “Lagrange” (blocked or linear relation). While being the son of this degree of freedom, it will be eliminated before, which is the rule.

For $L2$, one does not create again SN, one is satisfied to add an additional degree of freedom ($L2$) with the super-node containing the degree of freedom (blocked or R.L.) of which it is the “Lagrange”.

In the event of linear relation, all the degrees of freedom of the relation are put by the algorithm in the same super-node. That is realized in the preceding paragraph (PREML1), during the manufacturing of the structure $ADJNCY$, by forcing the degrees of freedom of the linear relations to being close between them, by means of the chained list (DEB, $VOIS$, $SUIV$).

The new renumérotations are obtained $NOUV$ and ANC on $[1:\mathrm{NEQ}]$, for all the degrees of freedom in new classification, $ANC(I)$ will provide its number in initial classification. $NOUV$ is the opposite table.

$NBSN$ the new number of super-nodes and $SUPND[1:nbsnd+1]$ defines these SN.
The super-node SN is made up by the segment of nodes $[SUPND(SN-1)+1, SUPND(SN)]$.
$SUPND$ is thus an application of the segment $[1:NBSN]$ towards $[1:\mathrm{NEQ}]$.

N.B. Memory allocation

$LGIND$ is the estimated length of the tables $GLOBAL$ AND $LOCAL$, (cf further 2.6) who are allocated front PREML2 ($NOPGLO$ and $NOPLOC$)
$LGIND$ is initially a result of the renumerotation, provided by PREML1 and increased by PREMLC according to the boundary conditions.

$LGIND$ is an over-estimate of $GLOBAL$ and $LOCAL$ one does not want to preserve a too important structure.

Thus, afterwards PREML2, one knows the real length of these tables, one redefines $LGIND$ with this value, and one recopies $GLOBAL$ and $LOCAL$ in structures $NOMGLO$ and $NOMLOC$ of exact length.

2.5 PREMLD

It is a technical stage: this routine manufactures a structure of the type $ADJNCY$ like those used by the algorithms of renumerotation, this time all degrees of freedom of 1 with NEQ are taken into account.

N.B. Memory allocation

The structure $(XADJ, ADJNCY)$ manufactured by PREMLD, is called $(XADJ1, ADJNC1)$ in MLTPRE.
The same length is used $LGADJN$ calculated previously.

2.6 PREML2

It is the second phase of MLTPRE : one carries out inter alia, the factorization symbolic system which is a simulation of real factorization, without floating operation, intended to facilitate the work of real factorization, while manufacturing certain pointers necessary.

PREML2 call the following routines:

- FACSMB: factorization symbolic system itself.
- MLTPOS : manufacturing of the sequence of factorization, i.e., gone up tree structure defined by the table $PAREND$, which thus defines the order of elimination of the super-nodes.

- MLTBLC : One defines here cutting in blocks of the factorized matrix, virtually “out of core”, because cut out in releasable blocks by the software package JEVEUX.
- MLTPAS : The address here is calculated initial coefficients in the factorized matrix.

2.6.1 FACSMB

Data (They result from the renumerotation):
$SUPND$
$PAREND$,
(And of initial topology):
$ADJNCY$.

Results :

$GLOBAL$
$LOCAL$
$ADRESS$
$NBASS$
$DEBFAC$
$DEBFSN$
$LGSN$
$LFRON$
$FILS$
$FRERE$.

These tables are described below.

$FILS$ and $FRERE$ are the tables “opposite” of the table $PAREND$, they make it possible to know all the super-nodes “son” of a given super-node SN , it is $FILS(SN)$, $FRERE(FILS(SN))$, and so on. For the other results, it is necessary to reconsider the basic principles of the multi-frontal method, by means of the following figures:

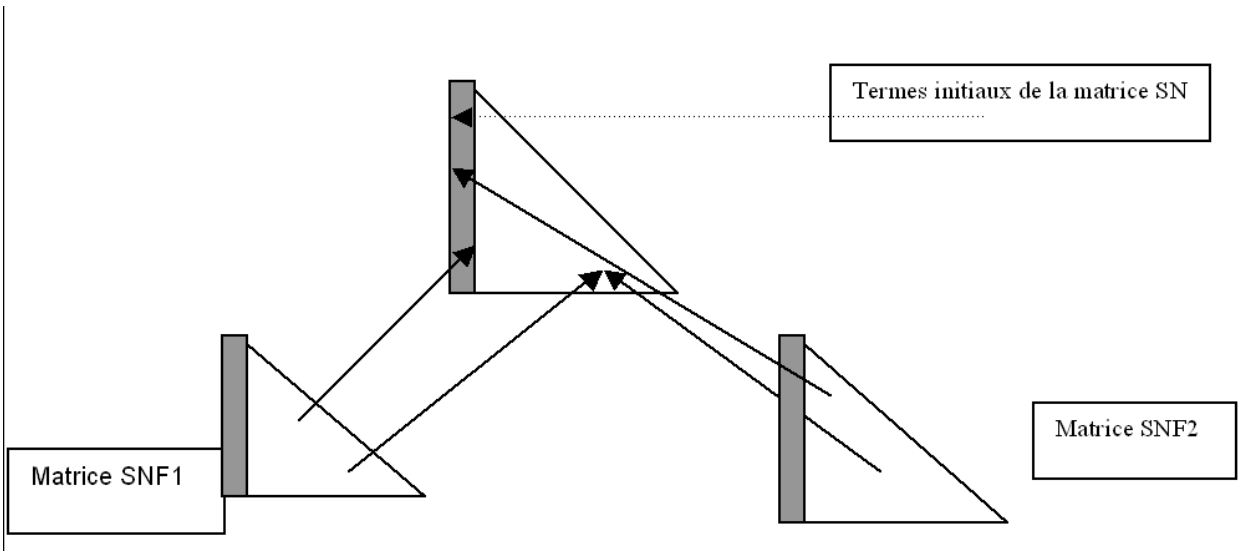


Figure 1 Assembly of 2 matrices girls in a matrix mother

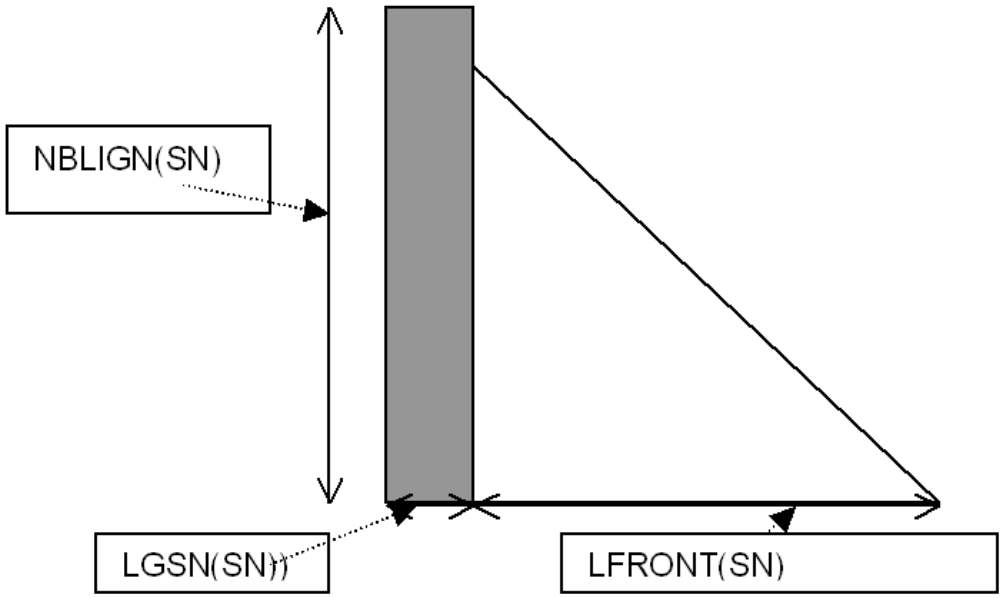


Figure 2 Stamps frontal SN

One sees on Figure 1, the frontal matrix SN (super-node SN), constituted by the terms coming from the initial matrix, as well as terms coming from the matrices girls $SNF1$ and SNF2, after elimination of the super-nodes $SNF1$ and $SNF2$. After elimination, the grayed part of the frontal matrix will constitute the columns of the factorized matrix, and the remaining white part arranged in the pile of the frontal matrices, will agglomerate with the matrix mother during the assembly of the latter.

Factorization symbolic system carried out by FACSMB will simulate this process of assembly/elimination. For each super-node SN, one will manufacture a chained list containing the numbers of unknown factors of the super node itself, as well as the numbers of unknown factors of the neighbors of all the sons of SN. One includes the neighbors of the sons, but not the sons themselves which were eliminated.

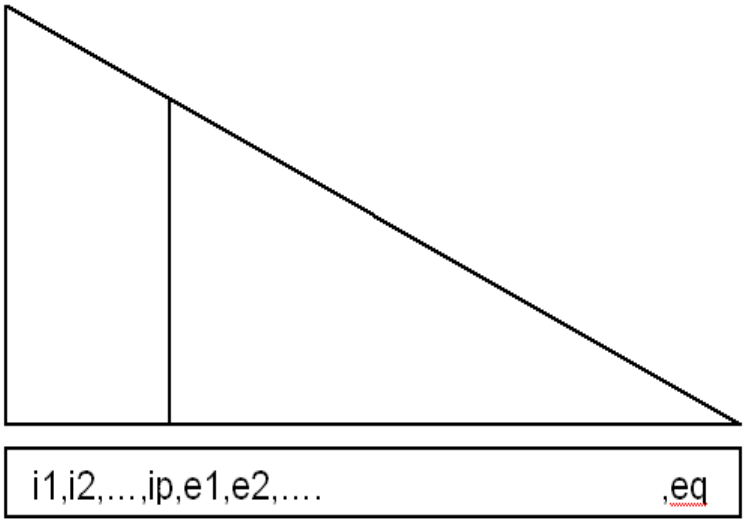
In language FORTRAN, FACSMB product following tables:

$LGSN(SN)$: many unknown factors of the super-node SN (makes some = $supnd(sn+1)-supnd(sn)$),
$NBLIGN(SN)$: total number of unknown factors of the frontal matrix SN
$LFRONT(SN)=NBLIGN(SN)-LGSN(SN)$, order of the frontal matrix after elimination which will be stored in the pile.

FACSMB factory also a table $GLOBAL$, provided with a table of pointers $ADRESS$.
$GLOBAL$ contains the numbers of the unknown factors of the frontal matrix SN (before elimination), between the addresses $ADRESS(SN)$ and $ADRESS(SN+1)-1$.

These unknown factors are with the number of $NBLIGN(SN)=(ADRESS(SN+1)-ADRESS(SN))$.
One reaches the table $LOCAL$ like $GLOBAL$, by means of the table $ADRESS$. That is to say $FSN1$ the super-node "son" of the super-node SN, as on the figure above, and is I an index understood enters $ADRESS(FSN1)+LGSN(FSN1)$ and $ADRESS(FSN1+1)$, $LOCAL(I)$ will be the local number (including enters 1 and $NBLIGN(SN)$) of this unknown factor in the matrix of the super-node "father" SN (one has $PAREND(FSN1)=SN$).

$LOCAL$ will be used in the assembly of the frontal matrix “girl” in the frontal matrix “mother”, it gives the TO address directly. One illustrates this using the following example:



The frontal matrix SN has $(p+q)$ unknown factors: $i1, i2, .., ip, e1, e2, ..., eq$, with

$NBLIGN(SN)=p+q$
$LGSN(SN)=p$
$LFRONT(SN)=q$
$ADRESS(SN+1)-ADRESS(SN)=p+q$,
$GLOBAL=[i1, i2, ... ip, e1, e2, ... eq]$, to the addresses varying enters $ADRESS(SN)+1$ and $ADRESS(SN+1)$,
$LOCAL=[0,0, .., 0, l1, l2, ... lq]$

With the same addresses as above, $l1, l2, lq$ are the local numbers in the frontal matrix of $PAREND(SN)$. (values understood enters 1 and $NBLIGN(PAREND(SN))$).

$NBASS$, meter used also for the assembly, is defined as follows: $NBASS(FSN1)$ is the number of unknown factors INC such as $INC<SUPND(SN+1)$, where $PAREND(FSN1)=SN$. It is thus the number of unknown factors of $FSN1$, assembled in the matrix mother SN and which will be eliminated during elimination of SN .
$DEBFAC$ and $DEBFSN$, these 2 tables give the same values, first is subscripted by unknown factor, the second by super-node, they provide, for $DEBFAC$, addresses of the diagonal coefficients of the unknown factors in the factorized matrix. For $DEBFSN$, it is the address of the diagonal coefficient of the first unknown factor of the super-node.

One saw on Figure 2, that the columns corresponding to the unknown factors of the eliminated super-node, (in grayed), constituted the columns of the factorized matrix which one calls $FACTOR$. Addresses of the diagonal coefficients provided by DEBFAC are addresses in a virtual table $FACTOR$ containing the factorized matrix. In fact a set of blocks JEVEUX will contain the factorized matrix, and one will handle an additional pointer for each block.

N.B. Programming
One sees on the preceding figures that one stores the columns of SN , all whole, (grayed rectangle), without taking into account the symmetry of the diagonal block. One consumes thus a little more memory, but one deals with a regular storage necessary to the use of the routines BLAS who request storages in tables FORTRAN from 2 dimensions.

2.6.2 MLTPOS

It is pointed out that the frontal matrices resulting from eliminations are stored in a pile. All the matrices girls of the same super node SN are stored (piled up), consecutively. Once read (depilated), they are not re-used any more, and the frontal matrix resulting from elimination from SN is stored in their place. The length of this pile decreases or increases with the wire of eliminations, It reaches a maximum which it is necessary to know.

MLTPOS traverses the tree structure, calculates $ESTIM$, maximum length of the pile for its later allowance, built a table $SEQ(1:NBSN)$ who provides the order of elimination of the super-nodes by traversing the tree structure upwards. It provides $ADPILE$ who contains the address in the pile of the frontal matrices of SN.
MLTPOS an algorithm of renumerotation about elimination of the super-nodes contains, intended to minimize the length of the pile of the frontal matrices ([AS87])

2.6.3 MLTBLC

This routine builds the pointers of cutting in blocks of the factorized matrix. Each block will contain the columns of an integer of super-node. One will never meet the configuration where the various columns of the same super-node would belong to 2 blocks.

Data :
$MAXBLOC$: maximum length of the blocks. Each block will contain the columns of a maximum of super nodes. $MAXBLOC$ is calculated before the call to MLTBLC, it is in fact the length (in columns) of largest of the super-nodes, and the latter (in general highest in the tree structure) a block with him will only occupy.

Results :
$NBLOC$: Many blocks.
$LGBLOC(1:NBLOC)$: lengths of each block.
$NCBLOC(1:NBLOC)$: definition of the blocks, as follows the block IB is made up by the columns of the super-nodes $SEQ(NCBLOC(IB-1)+1)$, with $SEQ(NCBLOC(IB))$.
$DECAL(1:NBSN)$ table of shift which gives access the first diagonal term of each super-node in the factorized matrix. One will illustrate that by the following diagram:

```
ISN=0
For IB = 1, NBLOC                                                    !
Buckle on the blocks of factorized
      JEVEUO (JEXNUM (FACTOL, IB), 'IT, IFACL)
                        ! Block IB of collection FACTOL is charged in IFACL
     For NC = 1, NCBLOC (IB)              ! Buckle on the Super-nodes of block IB
           ISN = ISN +1
           SN = SEQ (ISN)
!          The sequence of elimination is followed
           ADFAC= IFACL + DECAL (SN) - 1
!          ADFAC is the address in ZR of the first coefficient of the super-node SN
```

2.6.4 MLTPAS

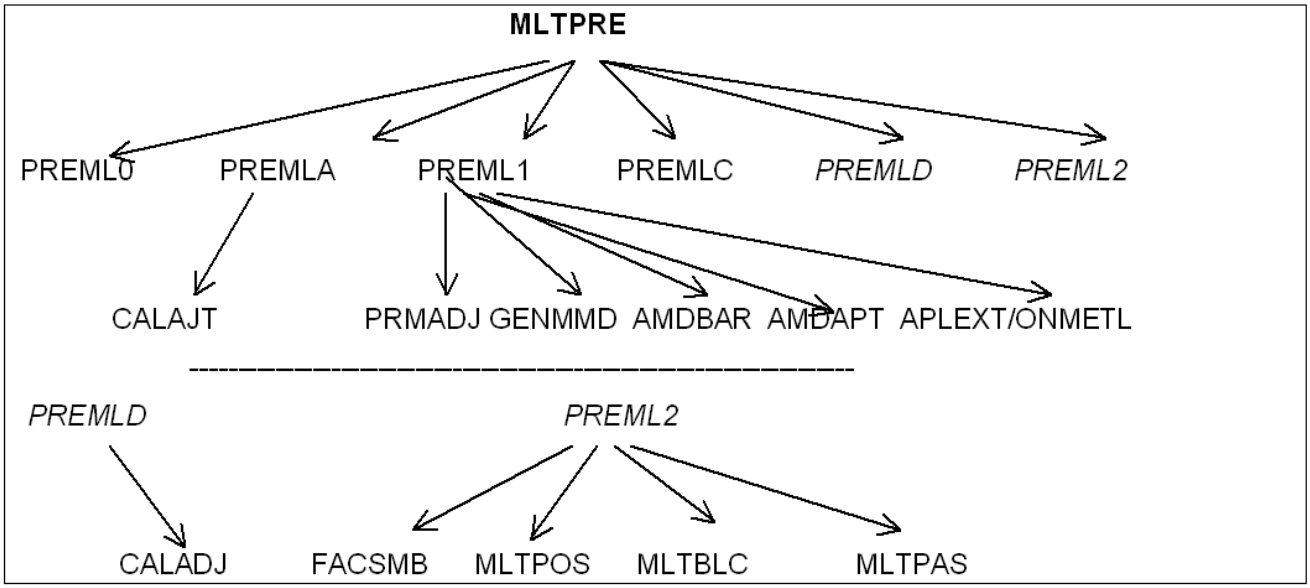
This routine calculates the addresses in the factorized matrix, of the coefficients of the initial matrix.
These addresses are stored in the structure MLTF.ADNT, of name FORTRAN $ADINIT$ or COL of length $LMAT$.

N.B. Characteristic of programming.

Certain addresses generated of ADINIT are negative. Indeed in the not-symmetrical case, there are 2 initial matrices MATI and MATS who have same topology, (same neighbors, even hollow), factorization symbolic system is the same one.
However, it can happen that a coefficient of MATI (lower part of the initial matrix), a destination in the upper part of the factorized matrix has. (In an identical way enters MATS and the lower part of factorized). In this case the address of the initial coefficient is stored negative and the routine of injection of the coefficients of the initial matrix, MLTASA takes into account and addresses the coefficients correctly.

```
IF (CODE.GT.0) THEN
                      ZR (IFACL+ADPROV-DEB) = ZR (MATI+I1-1): MATI injected into IFACL
(ower)
                   ZR (IFACU+ADPROV-DEB) = ZR (MATS+I1-1)
                ELSE
                   ZR (IFACL+ADPROV-DEB) = ZR (MATS+I1-1)
                   ZR (IFACU+ADPROV-DEB) = ZR (MATI+I1-1) MATI injected into IFACU (pper)
                ENDIF
```

2.7 Tree of call of MLTPRE



3 Digital factorization MULFR8

One deals with factorization itself, the routines called are:

1) MLTPRE
2) MLTASA
3) MLTFC1

MLTPRE, considering previously, is called if necessary. One checks that it was not already called upstream, in this case one leaves the routine.

3.1 MLTASA

This routine carries out the injection of the initial terms of the matrix in the factorized matrix. The table ADINIT calculated previously by MLTPRE provides the addresses.

Data :
The recording .VALM structure NOMMAT of type sd_matr_asse provided.
$ADINIT$ and $LGBLOC(1:NBLOC)$ calculated by MLTPRE.

Results :
MLTASA create a dispersed collection of name $FACTOL$ having $NBLOC$ blocks lengths given by $LGBLOC(1:NBLOC)$.

In the case not – symmetrical, the routine reads 2 " .VALM " respective names $MATI$ and $MATS$, lower parts and higher initial matrix, and creates 2 collections $FACTOL(ower)$ and $FACTOU(pper)$.

3.2 MLTFC1

One is if the pile of the frontal matrices is arranged in a named table FORTRAN $PILE$ and allocated by WKVECT in the appealing one MULFR8, this table is destroyed at the end of the routine.

A version with the frontal matrices in dispersed collection was written (MLTFCB), but is not used.
MLTFC1 has the following structure:

In an external loop on the blocks of factorized,
Loading of the memory stack
Buckle on the nodes of the block
SEQ provides the order of elimination
The super-node is thus eliminated SNI in various phases:
Assemblies of the matrices "girls" by call to MLTAFF and MLTAFF
Elimination of the columns of SNI who are the columns of factorized: MLTFLM
Update of the remaining unknown factors in the frontal matrix (complement of Schur): MLTFMJ
End of loop
End of loop

```
For IB = 1, NBLOC                                                              !
Buckle on the blocks of factorized
      JEVEUO (JEXNUM (FACTOL, IB), 'IT, IFACL)
      ! Block IB of collection FACTOL is charged in IFACL
     For NC = 1, NCBLOC (IB)   ! Buckle on the Super-nodes of block IB
           ISN = ISN +1
           SNI= SEQ (ISN)
           For all SN  son of SNI! assembly of the matrices girls
           mltafp
             mltaff
           ! end of the assembly of the matrices girls

            mltflm  ! elimination of SN: factorization on the columns of SN
            mltfmj  ! contribution of the columns eliminated on the other degrees of
freedom from the face, update of the frontal matrix
                      !        end of loop on the super nodes
!        end of loop on the blocks
```

For describing the routines called, it is necessary to specify the way in which the matrix is arranged. The following diagram is looked at:

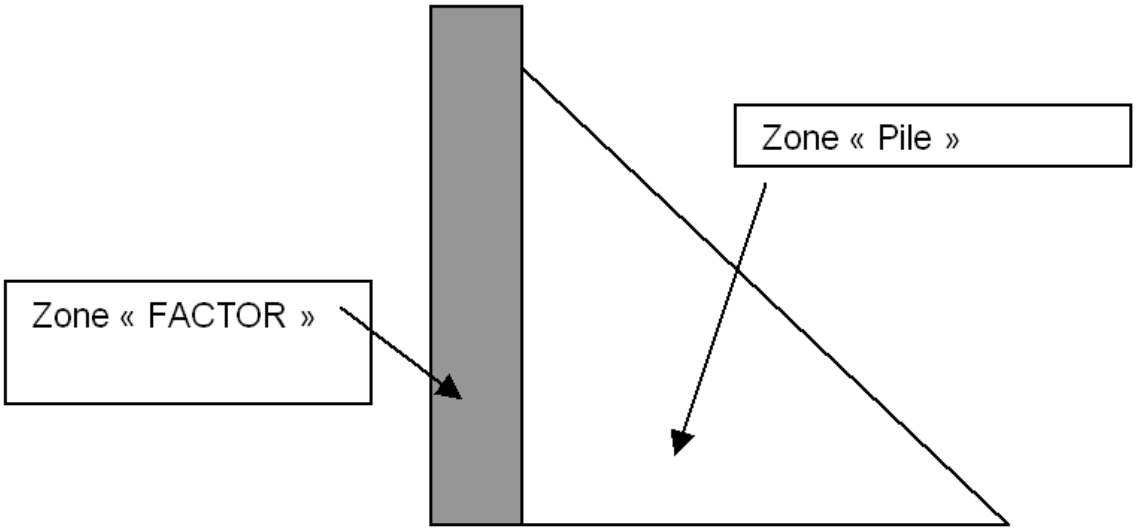


Figure 3 Decomposition of the frontal matrix in 2 storage sections

The grayed rectangular part represents active zone the "of the frontal matrix", they are the columns of SN who will be eliminated, and will constitute the columns of the factorized matrix $FACTOR$. To avoid a copy of memory enters the frontal matrix and $FACTOR$, one stores directly coefficients in $FACTOR$, it will be said that it is the part " $FACTOR$ " frontal matrix. With dimensions, there is the "passive" zone of the frontal matrix, (complement of Schur in fact), which will be stored in the pile, in the same way, to avoid a copy of memory, one directly stores this part of the matrix in the pile.

MLTAFP and MLTAFF are the routines which assemble the frontal matrices "girls" in the frontal matrix "mother".

3.2.1 mltafp

This routine assembles the coefficients of the frontal matrix "girl", in the zone " $FACTOR$ " frontal matrix "mother".

3.2.2 mltaff

This routine copies the coefficients of the frontal matrix "girl", in the zone "crushes" frontal matrix "mother".

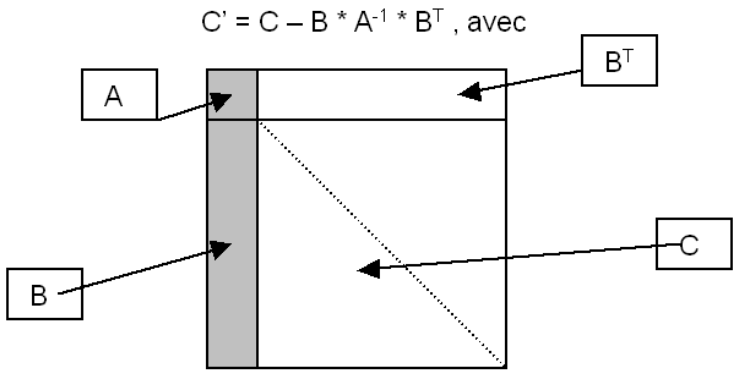
3.2.3 mltflm

Elimination of the super node SN , i.e factorization partial of the frontal matrix, work in the zone " $FACTOR$ ".

MLTFLM call:
DGEMV : Blas routine which carries out the products matrice*vector
MLTFLD (call to DGEMV)
MLTFLJ (call to DGEMM).

3.2.4 mltfmj

Update of the zone " $PILE$ " which is modified by the elimination of SN .
MLTFMJ call:
DGEMM: Blas routine which carries out the products matrix-matrix.
MLTFMJ is the routine most consuming computing time, it carries out the update of the frontal matrix on the lines and the columns of the degrees of freedom close to those eliminated, it is an operation which one can write:



In order to reduce the costs of calculation, the matrices virtually are divided A, B and C under blocks of order NB and the operation (1) above is carried out per blocks by calling the Blas routine DGEMM. This routine optimized by the manufacturer on each machine benefits owing to the fact that the small blocks from order NB thus multiplied hold in the primary mask, thus reducing the costs of access to the memory. NB is currently fixed at 96, this value generally about a few tens depends on the size of the mask $L1$, nearest to the processor and the size of with the dealt problems. It could be D to estimate from time to time according to new acquisitions of machines, on a set of CAS-test significant. The tests carried out previously showed that the performances varied little when NB varied around 96.

MLTFLM use also this division in blocks and uses DGEMM via MLTFLJ. (If the number of columns to be eliminated is lower than a certain threshold ($pmin=10$), the call to these 2 last routines is replaced by the call to MLFT21). This was done by preoccupation with an optimization of the elimination of the “small” super-nodes, it would be appropriate to check that is always justified. Thus, by preoccupation with a simplification, after checking of the absence of degradation of the performances for the small cases, one could remove this threshold and thus eliminate a branch from subprogram call.

To illustrate this, the tree structure of the routines called in the symmetrical case is the following:

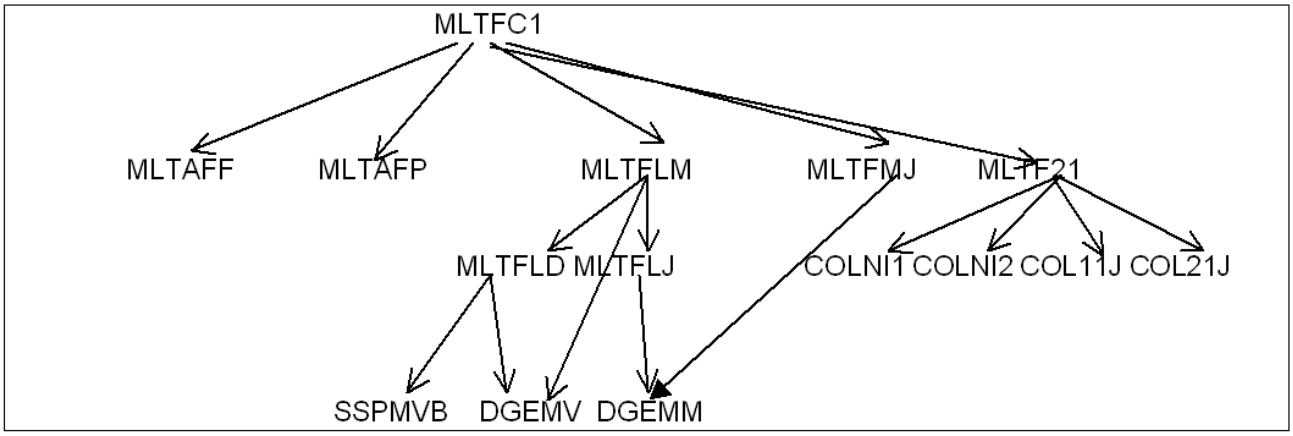


Figure 4 Factorization (symmetrical real case).

4 Increase of the system: RLTFR8

If the number of second members to be solved simultaneously is higher than 4, one calls on a method per blocks implemented by the routine RLBFR8, if not MLTDRA is called for each second member.
Data : they are the pointers resulting from MLTPRE:

$ADRESS$,
$GLOBAL$,
$SUPND$,
$LGSN$,
ANC ,
$NOUV$,
SEQ ,
$LGBLOC$,
$NCBLOC$,
$DECAL$.

And stamps it factorized resulting from MULFR8
$FACTOL$: .VALF (and $FACTOU$: .WALF in nonsymmetrical)

Second members:
$XSOL(1:\neq,1:NBSOL)$

Results : They are found in the table which contained the second members: $XSOL$.

4.1 MLTDRA

This routine successively carries out, for only one second member, the descent, division by the diagonal term, and the increase. In the descent, one calls on the product matrix-vector $DGEMV$ Blas libraries, beyond of a certain threshold. In lower part one optimizes “with the hand” in the routine SSPMVB.
(This optimization is a vectorization written for the CRAY, it could thus be re-examined even removed!)

4.2 RLBFR8

In the presence of several second members, calls to DGEMV quoted in the preceding paragraph, are replaced by calls to DGEMM, product matrice*matrice, on matric blocks of order NB . One thus aims the performance provided by these Blas of level 3, as in digital factorization.
The routines called are:

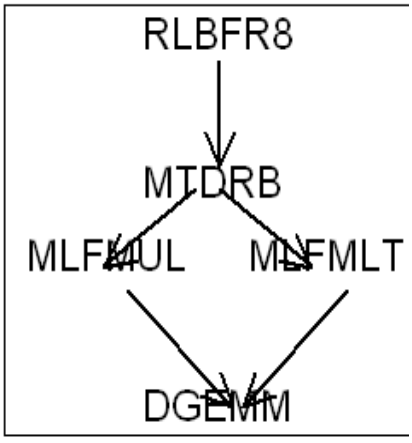


Figure 5 Descent-increase per blocks

5 Not-symmetrical version

The divergence between the versions symmetrical and not-symmetrical is inside the routine MLTFC1, the tree structure of the routines called in the not-symmetrical case is the following.

(MLNFLM, MLNFMJ, MLNFLD, MLNFLJ replace respectively MLTFLM, MLTFMJ, MLTFLD, MLTFLJ).

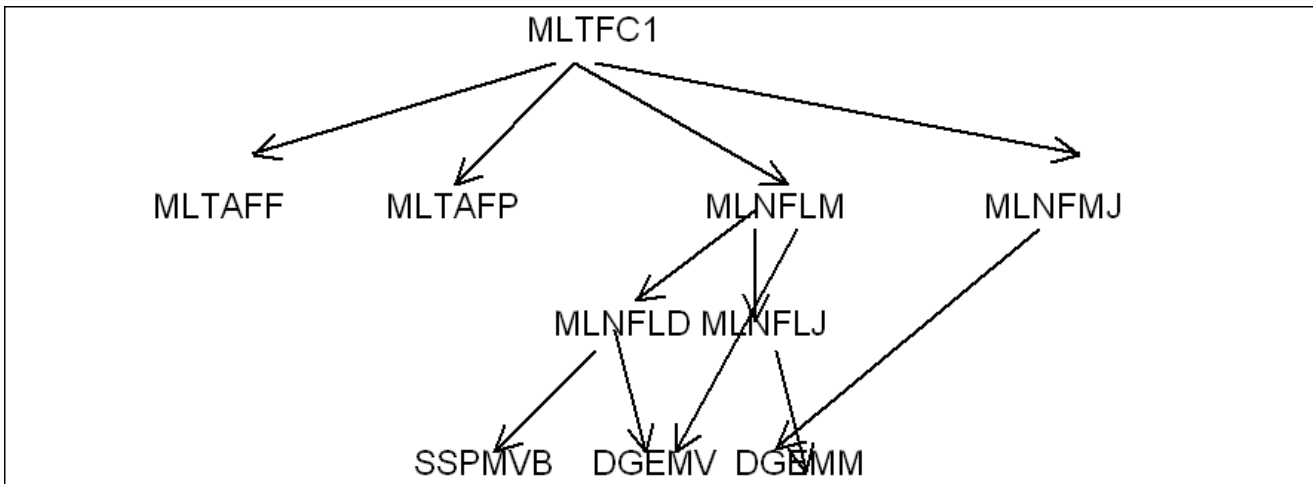
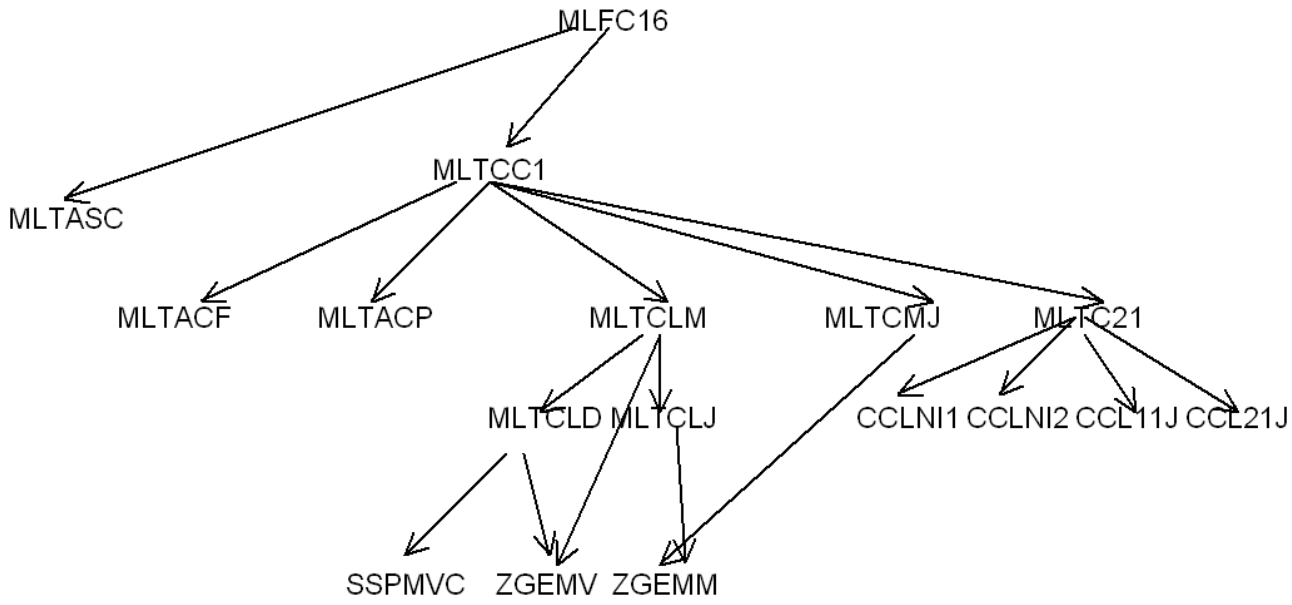


Figure 6 real Tree structure not – symmetrical.

6 Complex version

6.1 Symmetrical complex factorization

Symmetrical complex factorization is represented by the following flow chart, the changes in the names of routines are marked in fat. MLFC16 replace MULFR8 and MLTASC carry out the injection of the initial terms in a way similar to MLTASA.



Symmetrical figure 7 complex Version

6.2 Not-symmetrical complex factorization

Complex factorization nonsymmetrical is represented by the following flow chart, the changes in the names of routines are marked in fat.

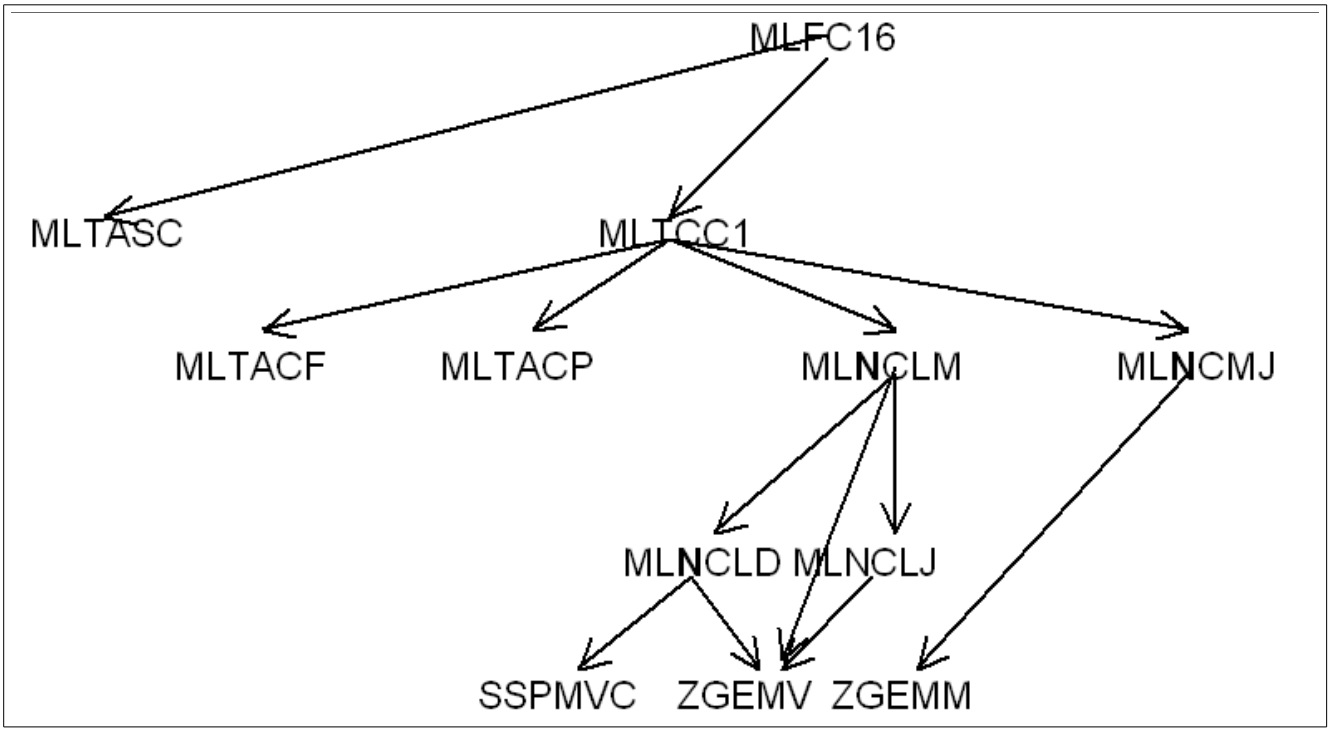


Figure 8 Factorization complexes nonsymmetrical

6.3 complex Descent-increase

The routine RLFC16 is the equivalent complex RLTFR8, and MLTDCA, the equivalent of MLTDRA.
The version per blocks for several second members RLBFR8 in realities does not exist in complex.

MLTDCA call SSPMVC and ZGEMV, the latter, of the Blas library, is the complex version of DGEMV. SSPMVC is the complex version of SSPMVB.

7 Parallelization

Parallelization in MULT_FRONT is realized by Openmp directives. These directives are placed in the following routines:

Symmetrical real case: MLTFMJ, MLTFLJ
Symmetrical complex case: MLTCMJ, MLTCLJ
Real case nonsymmetrical: MLNFMJ, MLNFLJ
Case complexes nonsymmetrical: MLNCMJ, MLNCLJ

Implemented parallelization is “known as” intern, it takes place inside the process of elimination of a super node. The elimination of the super nodes, known as “external” can also be paralleled. But it is not implemented for the following reason: Simultaneous elimination several super nodes requires an occupation of the memory higher than that of the sequential version. However the concern occupation of the memory always prevailed that of the time-saver of restitution at the time as of preceding developments of MULT_FRONT.

Internal parallelization, it, does not require any additional memory with that of the sequential version, it does not require either any development of particular code, only a simple addition of directives.
One will describe the parallelization of the routine MLTFMJ, that of MLTFLJ follows the same principles, as well as the other routines referred to above.

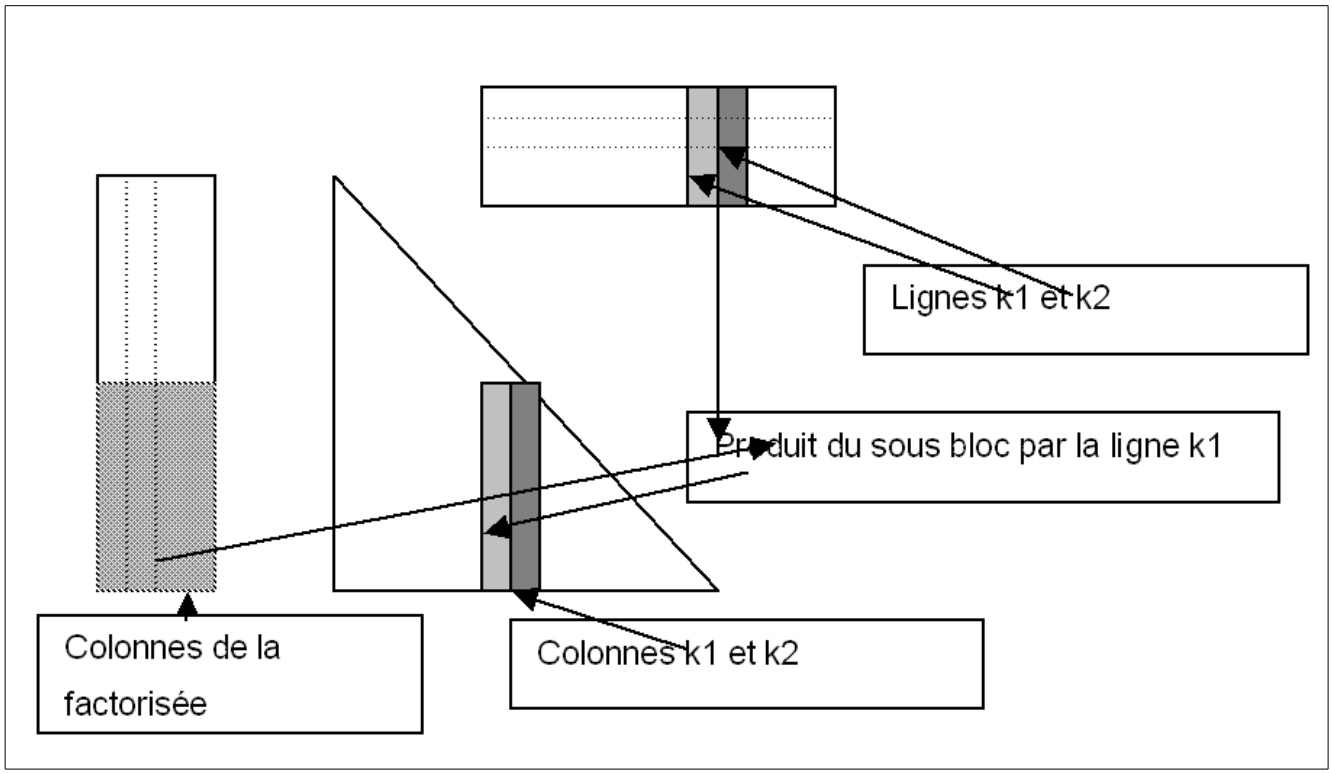


Figure 9 Diagram of the update of the frontal matrix

One sees on the Figure above, a diagram of the update of the frontal matrix during the elimination of a node. The triangle represents the lower part of the frontal matrix, and the 2 rectangles represent the columns of the factorized matrix, columns corresponding only to the super eliminated node. Initially, one will reason by columns. It is known that one updates the column $k1$ frontal matrix by withdrawing to him the product of the under-block of columns (in grayed above) by the line $k1$ (multiplied itself by the diagonal terms). (operation of the Blas2 type). This operation of update is paralleled without problem by a directive OpenMP, the columns of the frontal matrix are adjacent, it does not have no conflict of addresses there. All the variables are shared, except the indices of loops and the local variables. One saw above that by preoccupation with a performance, one works per block of columns, in order to use of Blas3, parallelization evoked above is thus done makes some on the blocks of columns and not on the columns themselves.

Outline of the code:

```
C$OMP PARALLEL C DEFECT (PRIVATE)
C$OMP+SHARED (NR, M, P, NMB, NB, RESTM, FACE, ADPER, DECAL, FRN, WK., C)
C$OMP+SHARED (TRA, TRB, ALPHA, BETA)
C$OMP+SCHEDULE (STATIC, 1)
     C 1000 KB = 1, NMB
     NUMPRO=MLNUMP ()
C     K: INDEX OF COLUMN IN THE FRONTAL MATRIX (ABSOLUTE OF 1 TO NR)
       K = NB* (KB-1) + 1 +P
       C 100 I=1, P
         S = FACE (ADPER (I))
         ADD= N* (I-1) + K
         C 50 J=1, NB
           WK. (I, J, NUMPRO) = FACE (ADD) *S! WK. contains the products: diagonal*ligne term
                  ADD = ADD + 1
 50           CONTINUOUS
 100       CONTINUOUS
```

```
          C 500 IB = KB, NMB
             IA = K + NB* (IB-KB)
             IT=1
             CAL DGEMM (TRA, TRB, NB, NB, P, ALPHA, FACE (IA), NR,
     &                  WK. (IT, 1, NUMPRO), P, BETA, C (1.1, NUMPRO), NB)
C     RECOPY

C
             C 501 I=1, NB
                I1=I-1
                IF (IB.EQ.KB) THEN
                   J1= I
                   IND = ADPER (K + I1) - DECAL
                ELSE
                   J1=1
                   IND = ADPER (K + I1) - DECAL + NB* (IB-KB)  - I1
                ENDIF
                C 502 J=J1, NB
                     FRN (IND) = FRN (IND) +C (J, I, NUMPRO)! Recopy in the frontal
matrix frn
                   IND = IND +1
 502            CONTINUOUS
 501         CONTINUOUS
 500      CONTINUOUS

 1000 CONTINUOUS
C$OMP END PARALLEL C
```

N.B. Parallelization.

The sequential version requires a table of work TRA , writing for each block, it is necessary to duplicate it for parallelization. In the parallel loop, one calls on the function MLNUMP who provides the number of thread treating the iteration of loop; NUMPRO , one works then in the plan NUMPRO table $TRAV$.

8 Various remarks

Some parameters, threshold values, are used in MULT_FRONT. They are intended to optimize the code, in terms of computing time. It is:
NB: length of the blocks for the use of DGEMM. NB is provided by the function LLBLOC () who returns value 96. It would be convenient with each change of processor to check if this value is optimal. $PMIN$ (fixed at 10 in MLTFC1). This value is related to the preceding one. The super nodes of which the size (many degrees of freedom, given by $LGSN$) is lower than $PMIN$, are treated by MLTF21 and not by MLMTFLM, MLTFMJ. That wants to say that the unknown factors are treated column by column and not per block with call to SDGEMM).

This threshold aims at simplifying the treatment of the small CAS-tests and not to activate the machinery per blocks for the small super nodes.

$NBSOL$: It is the number of second members treated at the time of the gone up descent/, if $NBSOL$ is lower than 4, the gone up Descents/are carried out the ones after the others by MLTDRA, if not one works per BLOCK in RLBFR8. $SEUIN$ $SEUIK$: values used in MLTDRA : they determine the use of DGEMV for the product matrix-vector, under these thresholds, one will call SSPMVB “optimized” with the hand. (Being given the little of weight of MLTDRA, these 2 thresholds could be removed).

As one already noticed above, it would from time to time be advisable to check if these thresholds are adequate.

9 Possible conclusions, developments.

report [RB12] was diffused. It serves as detailed conclusion. It offers one state of places and of the more detailed prospects on the possible maintenance and developments for MULT_FRONT, in the context of the availability in *Code_Aster*, of another solvor, MUMPS. One summarizes however in the 2 following paragraphs, the perimeter of execution and the assessment of the performances of MULT_FRONT.

9.1 Perimeter of execution

The perimeter of the multi-frontal method is almost complete since the developments of 2011 which make it possible to factorize the "generalized" matrices known as. Only the numerically unstable systems, requiring a method of pivot (X-FEM for example) are to be excluded from the choice of the solvor MULT_FRONT. A less recent development (renumerotation of the nodes of interpolation and either of the degrees of freedom), makes it possible to preserve the order of these degrees of freedom inside a node and makes it possible to treat the incompressible elements. The recourse to MUMPS is the solution for the systems requiring a method with pivot. However a development of MULT_FRONT is possible: it is possible to carry out, inside each super node, i.e. among the unknown factors eliminated with each stage, a search (obviously partial) for pivot. A more complete research, would then imply the fusion of several super nodes (fusion of matrices girl and mother) and appears not easily possible. This partial swivelling requires a considerable quantity of work and would constitute a risk for the stability of the code.

9.2 Performance

The sequential performances (mono-processor mode) are comparable to those of MUMPS in much of case. Only the cases presenting many linear relations between the degrees of freedom can be very unfavourable with MULT_FRONT who creates matric filling in these cases. Sometimes it was enough to reorder the data of these boundary conditions to solve the problem.

With regard to parallelism (localised as one saw higher, inside the elimination of each super node), it is used little and to little offer a factor of acceleration between 2 and 3 for an execution on four processors (or hearts of processors). This parallelism is interesting for the large case requiring much memory and which must practically only be carried out out of machine. That makes it possible to notably reduce their duration of residence out of machine. The arrival of processors equipped more and more with hearts should provide the occasion of new benchmarks for such large cases. However this parallelization in shared memory does not offer profit in memory. What allows parallelization in memory distributed (MUMPS). Often this problem of memory is a point blocking for the large linear systems.
The multi-frontal method gives the opportunity of another more elaborate parallelization on the various branches of the tree of elimination. It is not possible to program it in MULT_FRONT, it would be "to remake" what is made in MUMPS. The code was not conceived for parallelization by sending of message.

10 References.

1 [DU86] Duff, I.S., REID J.K., "Parallel implementation of multi-frontal designs", Parallel Computing, 3, (1986)

2 [AS87] Ashcraft C., "with vector implementation of the multi-frontal method for broad, sparse symmetric positive definite systems". Boeing Computer Service Technical Carryforward ETA-TR 51, (1987)

3 [RO93] Pink C., "A multi-frontal method for the direct resolution of the linear systems", EDF R & D, note HI-76/93/008

4 [RO95] Pink C., "A parallel multi-frontal method for the direct resolution of the linear systems", EDF R & D, note HI-76/95/021

5 [RB12] Pink C., Boiteau O., native multi-frontal Method in Code_Aster: prospect and inventory of fixtures, CR-I23-2012-001