Implementation of STAT_NON_LINE and of DYNA_NON_LINE

Summary:

One describes here implementation the data-processing of the algorithm of resolution of the nonlinear and dynamic problems quasi-static non-linear. The documents describing in detail these algorithms are supposed to be known ([R5.03.01] and [R5.05.05]), one will recall only the principal stages of them. One will find in this document a recall of the notations, the simplified flow chart of the routine op0070, allowing to distinguish the principal logical articulations of the operators STAT_NON_LINE and DYNA_NON_LINE of *Code_Aster*, the tree of call, a description of the data-processing objects and principal routines, and some traps to be avoided during the development in this operator.

Contents

1 Introduction

The operator op0070 consists of several parts:

1. Reading of the data;
2. Initialization of the data;
3. Construction and resolution of a linear system;
4. Put-with-day of the fields;
5. Filing of the results, postprocessings.

The whole being encapsulated in three levels of loop: no time, loops of fixed point (for the contact) and iterations of Newton with management of an event-driven type.

This document proposes the description of these various zones of the operator, while insisting particularly on the side structuring of the SD (§2), events (§ 3.3) and on the algorithm general, buckles by loop (§ 4).

To note that to help with debugging, it is possible to save the fields of displacements in a file MED, with each iteration of Newton. For that, the routine should be overloaded dbgcha by changing the Boolean one dbg=.false. in dbg=.true. The fields of displacements will then be saved in a file with format MED, on the logical unit 80 (not to forget to add a file on this output unit in astk).

2 Structures of data

A SD passes by several phases:
- Creation of the SD;
- Reading of information of the user and storage in the SD;
- Initializations of the SD;
- Destruction of the SD.
  The SD necessarily do not pass by all these phases.

2.1 Types of SD

Three levels of SD are introduced:
1. Concepts coming from other orders or products by the order itself;
2. The SD “means-level” are variables FORTRAN (of the tables of the simple type) having routines of access. In this category, one will find the vector of the activated features `list_func_acti` and the variable-hats which are tables of character strings containing of the more complex objects (matrices, vectors, etc);
3. The SD “high-level” are
   1. Maybe of objects JEVEUX specific to the operator (with routines of access)
   2. Maybe of the standard SD (grid, model,…) of Code_Aster which will remain interns with the operator
   3. Maybe of the SD described using derived types FORTRAN;

2.1.1 SD of type concept

The concepts come from another operator or are built by the operator to be used in other orders. They are built exclusively on standard SD in *Code_Aster* (fields, classifications, etc). Their contents are described in documentations of the D4 type. Here the exhaustive list of these SD in `op0070.F90`:

Description	**Type**	**Name generally used**
Grid	`sd_mailla`	`NETTED`
Model	`sd_modele`	`MODEL`
Result	`sd_resultat`	`RESULT`
Elementary characteristics	`sd_cara_elem`	`CARELE`
Loadings	`sd_l_charges`	`LISCHA`
Field of materials and variables of order	`sd_cham_mater`	`MATT`
Definition of the contacts	`sd_contact`	`DEFICO`
Definition of the unilateral connections	`sd_contact`	`DEFICU`

2.1.2 SD of mean level

The SD of mean level are intermediate between the SD “low-level” and the SD “high-level”. Indeed, these variables are simple objects FORTRAN but have routines of access. They are thus “low-level” from the point of view of storage, but their access is done by specific routines, just like the SD of “high-level”.

Description	Type	Name
Activated features	INTEGER (100)	list_func_acti
Variable-hat – elementary Matrices	CHAR*19 (ZMEELM)	MEELEM
Variable-hat – assembled Matrices	CHAR*19 (ZMEASS)	MEASSE
Variable-hat – elementary Vectors	CHAR*19 (ZVEELM)	VEELEM
Variable-hat – assembled Vectors	CHAR*19 (ZVEASS)	VEASSE
Variable-hat – Fields solutions	CHAR*19 (ZSOLAL)	SOLALG
Variable-hat – Fields incremental solutions	CHAR*19 (ZVALIN)	VALINC

2.1.3 SD of high-level

All these SD are built on objects JEVEUX or derived types and their access is done via dedicated routines (encapsulation of the data) when the SD are specific. One does not return on all, because it are concepts contained in the keywords of the operator or products by the operator (§ 2.1.1).

Description	Type	Name
Map of behavior	sd_carte	COMPOR
Solvor	sd_solvor	SOLVEU
Matrix of pre-packaging	sd_matr_asse	MAPREC
Matrix of assembled resolution	sd_matr_asse	MATASS
Map of the convergence criteria for the behavior	sd_carte	CARCRI
Map of the variables of ordering of reference	sd_carte	COMREF
Map of the code-returns error of the behavior	sd_carte	CODERE
Classification	sd_nume_ddl	NUMDDL
Classification fixes (used for the contact method continues)	sd_nume_ddl	NUMFIX
Management of the impressions	specific to op0070	ds_print
Management of measurements of time and the statistics	specific to op0070	ds_measure
Management of the errors of the algorithm	specific to op0070	SDERRO
Management of filing and the initial state (IN and OUT)	specific to op0070	ds_inout
Management of convergence	specific to op0070	ds_conv
Management of SUIVI_DDL	specific to op0070	SDSUIV
Management of the quality standards	specific to op0070	SDCRIQ
Management of piloting	specific to op0070	SDPILO
Management of classifications	specific to op0070	SDNUME
Management of dynamics	specific to op0070	SDDYNA
Management of the temporal discretization	specific to op0070	SDDISC
Management of the convergence criteria	specific to op0070	SDCRIT
Management of OBSERVATION	specific to op0070	SDOBSE
Management of postprocessing (MODE_VIBR and CRIT_STAB)	specific to op0070	SDPOST
Energy management	specific to op0070	SDENER
Parameters of the algorithm	specific to op0070	will ds_a

		lgop ara

The rest of the page will primarily describe the contents, the use and the access to the SD specific with op0070.

Certain SD are built using type derived FORTRAN. They obey common rules:
- The module describing these types is NonLin_Datastructure_Type.F90 ;
- The types are prefixed by the chain NL_DS_. For example type (NL_DS_Print) for the management of the impressions;
- In general, there is only one variable of this type in the operator, this variable passed in argument in the sub-routines which need some (they are not aggregate variables);
- The management of these SD is standardized:
  - The creation of these SD is made in the routine nmini0. F90 by routines of the form Create**Type**DS.F90 with **Type** the name of the derived type;
  - The reading of the user data starting from the command file is made in nmdata. F90, in general in routines of the form Read**Type**. F90 with **Type** the name of the derived type;
  - The initialization of the data is made in nminit. F90 in general, except for certain types which need a more specific initialization (with each step of time for example). The routines carrying out these initializations are in general form Init**Type**. F90 with **Type** the name of the derived type.

In the long term, the whole of the structures of data of the operator will be of this form.

2.2 Creation of the SD

The following table summarizes the origin of the creation of the SD.

Description	**Type**	**Creation**
Grid	sd_mailla	comes from .comm
Model	sd_modele	comes from .comm
Result	sd_resultat	nmnoli
Elementary characteristics	sd_cara_elem	comes from .comm
Map of behavior	sd_carte	nmdocc
Loadings	sd_l_charges	nmdoch
Coded material	sd_cham_mater	comes from .comm
Definition of the contacts	sd_contact	comes from .comm
Definition of the unilateral connections	sd_contact	comes from .comm
Activated features	INTEGER (100)	op0070
Variable-hat – elementary Matrices	CHAR*19 (ZMEELM)	op0070
Variable-hat – assembled Matrices	CHAR*19 (ZMEASS)	op0070
Variable-hat – elementary Vectors	CHAR*19 (ZVEELM)	op0070
Variable-hat – assembled Vectors	CHAR*19 (ZVEASS)	op0070
Variable-hat – Fields solutions	CHAR*19 (ZSOLAL)	op0070
Variable-hat – Fields incremental solutions	CHAR*19 (ZVALIN)	op0070
Solvor	sd_solvor	nmlect
Matrix of pre-packaging	sd_matr_asse	during the algorithm
Matrix of assembled resolution	sd_matr_asse	during the algorithm

Map of the convergence criteria for the behavior	sd_carte	nmdocr
Map of the variables of ordering of reference	sd_carte	nmvcre
Map of the code-returns error of the behavior	sd_carte	during the algorithm
Classification	sd_nume_ddl	nmprof
Classification fixes (used for the contact method continues)	sd_nume_ddl	nmpro2
Management of the impressions	Derived type	CreatePrintDS
Management of measurements of time	Derived type	nmcrti
Management of the errors of the algorithm	specific	nmcrga
Management of filing and the initial state (IN and OUT)	Derived type	nmetcr ntetcr
Management of the statistics	Derived type	nmcrst
Management of convergence	Derived type	CreateConvDS
Management of SUIVI_DDL	specific	nmcrdd
Management of the quality standards	specific	nmcrer
Management of piloting	specific	nmdopi
Management of classifications	specific	nmnume
Management of dynamics	specific	ndcrdy
Management of the temporal discretization	specific	nmcrli
Management of OBSERVATION	specific	nmcrob
Management of postprocessing (MODE_VIBR and CRIT_STAB)	specific	nmdopo
Energy management	Derived type	eninit
Resolution of CONTACT	Derived type	cfmxsd
Resolution of LIAISON_UNIL	Derived type	cfmxsd
Parameters of the algorithm	Derived type	CreateAlgoParaDS

2.3 Reading of the user data

The reading of the user data is done mainly under the routine nmdata, this routine reads the user data and creates possibly the SD necessary. The routine nmdome bed characteristics given by the keywords MODEL, CHAM_MATER, CARA_ELEM and EXCIT by also treating the case of the resumptions of calculation (reuse) who uses the information stored in the SD result. The table below gathers all the SD which will directly read information in the command file (and thus using the routines getxxx communication enters FORTRAN and the supervisor Python).

Description	**Keywords**	**Routine of reading**
Model	MODEL	nmdome
Result	*concept created*	nmnoli
Elementary characteristics	CARA_ELEM	nmdome
Map of behavior	BEHAVIOR	nmdocc
Loadings	EXCIT	nmdome
Coded material	CHAM_MATER	nmdome
Definition of the contacts	CONTACT	cfmxsd

Definition of the unilateral connections	CONTACT	cfmxsd
Management of convergence	CONVERGENCE	ReadConv
Solvor	SOLVEUR	cresol
Map of the convergence criteria for the behavior	BEHAVIOR	nmdocr
Management of the impressions	IMPRESSION	ReadPrint
Management of filing and the initial state (IN and OUT)	FILING ETAT_INIT	nmetcr nmcrar nmdoet
Management of SUIVI_DDL	SUIVI_DDL	nmcrdd
Management of the quality standards	CRIT_QUALITE	nmcrer
Management of piloting	PILOTING	nmdopi
Management of dynamics	SCHEMA_TEMPS MASS_DIAG PROJ_MODAL MODE_STAT AMOR_MODAL	ndlect
Management of the temporal discretization	DISCRETIZATION	nmcrsu
Management of OBSERVATION	OBSERVATION	nmcrob
Management of postprocessing (MODE_VIBR and CRIT_STAB)	CRIT_STAB MODE_VIBR	nmdopo
Energy management	ENERGY	eninit
Resolution of CONTACT	CONTACT	cfmxsd
Resolution of LIAISON_UNIL	CONTACT	cfmxsd
Parameters of the algorithm	METHOD RECH_LINEAIRE	nmdomt nmdomt_ls

2.4 Description of the SD

2.4.1 SD of mean level

2.4.1.1 Activated features – list_func_acti

This SD makes it possible to know which features are active in the algorithm at any time. The principal idea of this SD is one dismoi very rustic allowing to answer a simple question on the active features in op0070. For example:

- There is contact: isfonc (list_func_acti, 'CONTACT') is true;
- Linear research is activated: isfonc (list_func_acti, 'RECHERCHE_LINEAIRE') is true;

One will not make the list of the questionable features by this mechanism in this document because this vector is often modified, it is enough to read the routine isfonc. Even if this object is low-level (simple tables ofINTEGER), one must reach it via three routines only:

Operation on the vector of the activated features – list_func_acti	**Routine**
Preparation of the activated features	nmfonc
Interrogation of an activated functionality	isfonc
Rules of exclusion of certain features	exfonc

It is **requirement** to modify the vector list_func_acti only in the routine nmfonc (called in nminit) and always to envisage the question corresponding to this functionality in isfonc. In the same way, it is this vector

which one will use in a priority way to test compatibilities between certain features (routine exfonc, called by nmfonc).

2.4.1.2 Variable-hat – Elementary matrices MEELEM

This variable-hat contains the name of all the elementary matrices usable in op0070. It is thus about a list of SD of the type sd_matr_elem (sd_resu_elem). The code of location in the variable hats MEELEM/MEASSE is the same one if it is the same objects.

Variable-hat – elementary Matrices MEELEM	Code of location
Elementary matrices of rigidity	MERIGI
Elementary matrices of the limiting conditions of Dirichlet dualized	MEDIRI
Elementary matrices of mass	MEMASS
Elementary matrices of damping	MEAMOR
Elementary matrices of the following loadings	MESUIV
Elementary matrices of the substructures (macronutrients)	MESSTR
Elementary matrices of geometrical rigidity	MEGEOM
Elementary matrices of contact (method CONTINUOUS and XFEM)	MEELTC
Elementary matrices of friction (method CONTINUOUS and XFEM)	MEELTF

2.4.1.3 Variable-hat – Assembled matrices MEASSE

This variable-hat contains the name of all the matrices assembled usable in op0070. It is thus about a list of SD of the type sd_matr_asse. The code of location in the variable hats MEELEM/MEASSE is the same one if it is the same objects.

Variable-hat – assembled Matrices MEASSE	Code of location
Assembled matrix of rigidity	MERIGI
Assembled matrix of mass	MEMASS
Assembled matrix of damping	MEAMOR
Assembled matrix of the substructures (macronutrients)	MESSTR

2.4.1.4 Variable-hat – Elementary vectors VEELEM

This variable-hat contains the name of all the elementary vectors usable in op0070. It is thus about a list of SD of the type sd_vect_elem (sd_resu_elem). The code of location in the variable hats VEELEM/VEASSE is the same one if it is the same objects.

Variable-hat – elementary Vectors VEELEM	Code of location
Elementary vector of the internal forces	CNFINT
Elementary vector of the reactions of support for the limiting conditions of Dirichlet dualized	CNDIRI
Elementary vector of the limiting conditions of Dirichlet dualized	CNBUDI
Elementary vector of the nodal forces	CNFNOD
Elementary vector of the limiting conditions of Dirichlet given	CNDIDO
Elementary vector of the limiting conditions of Dirichlet controlled	CNDIPI

Elementary vector of the limiting conditions of Neumann given	CNFEDO
Elementary vector of the limiting conditions of Neumann controlled	CNFEPI
Elementary vector of the limiting conditions of type Laplace	CNLAPL
Elementary vector of the limiting conditions of standard plane wave	CNONDP
Elementary vector of the limiting conditions of Neumann following and given	CNFSDO
Elementary vector of the limiting conditions of standard impedance (in prediction)	CNIMPP
Elementary vector of the limiting conditions of Dirichlet differentials	CNDIDI
Elementary vector of the forces on the substructures	CNSSTF
Elementary vector of the forces of contact (method CONTINUOUS and XFEM)	CNELTC
Elementary vector of the forces of friction (method CONTINUOUS and XFEM)	CNELTF
Elementary vector of the forces of reference (RESI_REFE_RELA)	CNREFE
Elementary vector of the variables of order for the initial state	CNVCF1
Elementary vector of the variables of order for convergence	CNVCF0
Elementary vector of the limiting conditions of standard impedance (in correction)	CNIMPC

2.4.1.5 Variable-hat – Assembled vectors VEASSE

This variable-hat contains the name of all the vectors assembled usable in op0070. It is thus about a list of SD of the type sd_cham_no. These vectors are primarily used in the construction of the second members and the evaluation of convergence. The code of location in the variable hats VEELEM/VEASSE is the same one if it is the same objects.

Variable-hat – assembled Vectors 5TH ADZE	Code of location
Assembled vector of the internal forces	CNFINT
Assembled vector of the reactions of support for the limiting conditions of Dirichlet dualized	CNDIRI
Assembled vector of the limiting conditions of Dirichlet dualized	CNBUDI
Assembled vector of the nodal forces	CNFNOD
Assembled vector of the limiting conditions of Dirichlet given	CNDIDO
Assembled vector of the limiting conditions of Dirichlet controlled	CNDIPI
Assembled vector of the limiting conditions of Neumann given	CNFEDO
Assembled vector of the limiting conditions of Neumann controlled	CNFEPI
Assembled vector of the limiting conditions of type Laplace	CNLAPL
Assembled vector of the limiting conditions of standard plane wave	CNONDP
Assembled vector of the limiting conditions of Neumann following and given	CNFSDO
Assembled vector of the limiting conditions of standard impedance (in prediction)	CNIMPP
Assembled vector of the limiting conditions of Dirichlet differentials	CNDIDI
Vector assembled of the forces on the substructures	CNSSTF
Assembled vector of the forces of contact (method CONTINUOUS and XFEM)	CNELTC
Assembled vector of the forces of friction (method CONTINUOUS and XFEM)	CNELTF
Assembled vector of the forces of reference (RESI_REFE_RELA)	CNREFE
Assembled vector of the variables of order for the initial state	CNVCF1

Assembled vector of the variables of order for convergence	CNVCF0
Assembled vector of the limiting conditions of standard impedance (in correction)	CNIMPC
Assembled vector of the limiting conditions of Dirichlet eliminated	CNCINE
Assembled vector of the substructures	CNSSTR
Assembled vector of the forces of friction (method DISCRETE)	CNCTDF
Assembled vector of the variables of order for calculation	CNVCPR
Assembled vector of the dynamic forces	CNDYNA
Assembled vector of modal damping (in prediction)	CNMODP
Assembled vector of modal damping (in correction)	CNMODC
Assembled vector of the forces of contact (method DISCRETE)	CNCTDC
Assembled vector of the unilateral forces (method LIAISON_UNIL)	CNUNIL
Assembled vector of the external forces	CNFEXT
Assembled vector of the limiting conditions of type VECT_ISS	CNVISS

2.4.1.6 Variable-hat – Vectors of the solutions SOLALG

This variable-hat contains the name of the fields to the nodes which are useful in the algorithm to calculate the solution.

Variable-hat – Vectors solutions SOLALG	**Code of location**
Solution in displacement of the current iteration of Newton	DDEPLA
Displacement cumulated since the beginning of the step of time	DEPDEL
Increment of displacement of the step of previous time	DEPOLD
Solution in displacement of the prediction	DEPPR1
Solution in displacement of the prediction (left controlled)	DEPPR2
Solution of speed of the current iteration of Newton	DVITLA
Speed cumulated since the beginning of the step of time	VITDEL
Increment speed of the step of previous time	VITOLD
Solution of speed of the prediction	VITPR1
Solution of speed of the prediction (left controlled)	VITPR2
Solution in acceleration of the current iteration of Newton	DACCLA
Acceleration cumulated since the beginning of the step of time	ACCDEL
Increment of acceleration of the step of previous time	ACCOLD
Solution in acceleration of the prediction	ACCPR1
Solution in acceleration of the prediction (left controlled)	ACCPR2
Solution of the system	DEPSO1
Solution of the system (left controlled)	DEPSO2

2.4.1.7 Variable-hat – Incremental solutions VALINC

This variable-hat contains the name of the fields to the nodes or the fields of the type ELGA who will be the solutions of the non-linear problem.

Variable-hat – incremental Solutions VALINC	Code of location
Displacements at the beginning of the step of time	DEPMOI
Constraints at the beginning of the step of time	SIGMOI
Internal variables at the beginning of the step of time	VARMOI
Speeds at the beginning of the step of time	VITMOI
Accelerations at the beginning of the step of time	ACCMOI
Variables of order at the beginning of the step of time	COMMOI
Variables for the multifibre elements at the beginning of the step of time	STRMOI
Forces external with the beginning of the step of time (for the calculation of energies)	FEXMOI
Forces of damping at the beginning of the step of time (for the calculation of energies)	FAMMOI
Bonding strengths at the beginning of the step of time (for the calculation of energies)	FLIMOI
Nodal forces at the beginning of the step of time (for the calculation of energies)	FNOMOI
Displacements at the end of the step of time	DISPLEASED
Constraints at the end of the step of time	SIGPLU
Internal variables at the end of the step of time	VARPLU
Speeds at the end of the step of time	VITPLU
Accelerations at the end of the step of time	ACCPLU
Variables of order at the end of the step of time	TAKEN PLEASURE
Variables for the multifibre elements at the end of the step of time	STRPLU
Forces external at the end of the step of time (for the calculation of energies)	FEXPLU
Forces of damping at the end of the step of time (for the calculation of energies)	FAMPLU
Bonding strengths at the end of the step of time (for the calculation of energies)	FLIPLU
Nodal forces at the end of the step of time (for the calculation of energies)	FNOPLU
Extrapolated constraints (method IMPLEX)	SIGEXT
Displacements with the current iteration of Newton (management of great rotations)	DEPKM1
Speeds with the current iteration of Newton (management of great rotations)	VITKM1
Accelerations with the current iteration of Newton (management of great rotations)	ACCKM1
Rotations with the preceding iteration of Newton (management of great rotations)	ROMK
Rotations with the current iteration of Newton (management of great rotations)	ROMKM1

2.4.1.8 Access to the variable-hats

The access to the variable-hats is done via five routines.

Operation on the variable-hat	Routine
Creation of a variable-hat	nmcha0
Recovery of the index where the name of the variable in the variable-hat is stored	nmchai
Recopy of a variable-hat	nmchcp
Recovery of the name of the variable in the variable-hat	nmchex

Recopy a variable-hat by possibly changing a name of variable	nmchso
Creation of CHAM_NO for VALINC , SOLAG and VEASSE	nmcrch

The size of these SD is indicated same manner as the SD low-level. Nevertheless, it is advisable to reflect a change of size on the principal routine nmchai. If one wants to modify the contents of a variable-hat (to add, remove or modify the contents of a variable-hat), it is necessary:

- To impact the length if required in op0070, nmini0 (ASSERT of protection) and nmchai ;
- To modify in nmchai ;
- To create the variable-hat (see § 2.2);
- To initialize the contents of the variable-hat if required;

These routines are satisfied to manage the variable-hats as a list of names, the contents itself of these variable-hats does not depend on them. For example, the variable-hat VEASSE contains assembled vectors. None the routines of the preceding table deal with managing the SD CHAM_NO objects contained in the variable-hat, just their name. For the three variable-hats defining of CHAM_NO, the fields are created in the routine nmcrch, by using information on the active features list_func_acti.

2.4.2 SD of high-level

2.4.2.1 Management of the impressions – NL_DS_Print

The impression comprises three types of objects:

- Impressions “standard” containingutmess ;
- Impression of the table of convergence (in the file .mess and possibly in export in a file of the type csv);
- Boundary lines in the file message.
  The main difficulty in the management of this structure of data comes owing to the fact that the table of convergence is dynamic because the posting of the columns depends at the same time on the features activated (linear, standard research of residue, contact, etc) definite via the object list_func_acti (see §2.4.1.1), but also of options defined by the user (INFO_RESIDU and INFO_TEMPS in the keyword factor POSTING but also monitoring of degrees of freedom in the keyword factor SUIVI_DDL).

Only one variable ds_print of type NL_DS_Print manage the whole of the impressions (except utmess standards). It contains various information:

- the information recovered in the keyword POSTING ;
- a flag indicating if one must make of ‘posting in the file message (evaluated starting from the parameter NOT);
- a table of convergence;
- a character string (of the width of the table of convergence) containing the boundary line (with “-”);

Currently, op0070 only one table of convergence has, but the definition of what is a table in generic form (see §2.4.2.2) will in the long term allow to add other tables (like energy for example).

The type NL_DS_Print has the following structure:

Type	Name	Description
aster_logical	l_print	flag to know if one displays or not
type (NL_DS_Table)	table_cvg	table of convergence
aster_logical	l_info_resi	parameter INFO_RESIDU in POSTING
aster_logical	l_info_time	parameter INFO_TEMPS in POSTING
aster_logical	l_tcvg_csv	flag to know if one leaves the table convergence in a file CSV
integer	tcvg_unit	parameter UNIT in POSTING (table of convergence in a file CSV)
integer	reac_print	parameter NOT in POSTING

character (len=255)	sep_line	boundary line (---), width of the table of convergence

The management of the SD is done in three times:

1. Creation of the SD in the routine CreatePrintDS in nmini0. This creation understands in particular the creation of all the possible columns in the table of convergence. Their effective posting will depend on the state of the flag of their activation.
2. Reading of the information given by the user (keyword POSTING). Routine ReadPrint in nmdata. Information stored in ds_print.
3. Initialization of the SD. Is done in two times:
   1. Addition of the columns for SUIVI_DDL in the routine InitPrint in nminit. Indeed, these columns (of which their title) will depend on the fact that the user in the keyword factor asked SUIVI_DDL ;
   2. Activation of the columns according to the features with each step of time in nmimin (routine nmnpas).

Then, on the level of the use in the algorithmy, it is done in two times. The developer assigns the values of the columns to the good moment thanks to the routine SetCol. The algorithm general concatene information, creates the table of convergence and the poster in a regular way to each iteration of Newton. These utility routines of handling of the SD in general have an identifier starting with nmimpX. One does not make the exhaustive list of it.

2.4.2.2 Management of the tables – NL_DS_Table

The structure of data NL_DS_Table is an object which has a double-function:

- it manages one table (with the Code_Aster direction) which is attached to the structure of data result like the east the table of observation (see §2.4.2.13) or of the statistics (see §2.4.2.3);
- it manages a table which will be displayable in the file mess or exportable with the format csv (table of convergence or energies for example)

This table is defined by:

- its columns: number, identifier and type (whole, real, chain or complex);
- its lines: for each line, the affected value and an indicator of assignment;
- its heading: title (three lines to the maximum);
- its graphic elements: boundary line (for example at the time of passage of the fixed loops of points in contact);
- its dimensions: its total width (which is a function amongst columns and of the width of each column).

One of the difficulties to manage this table is to define the columns dynamically (according to the features for example.) For that, a double list is managed:

- the exhaustive list of the possible columns;
- an indicator of the activation of a column.

The standard generic table is defined by NL_DS_Table who has the following structure:

Type	Name	Description
integer	nb_cols	Many columns
integer	nb_cols_maxi	Maximum number of columns of a table
type (NL_DS_Column)	collars (max)	List of the columns
aster_logical	l_cols_acti (max)	Flag indicating that the column is active or not
integer	width	Total width of the table
integer	title_height	Height of the title (3 by the table of convergence)
character (len=255)	sep_line	Boundary line in the table
aster_logical	l_csv	Flag to say that this table is also printed in one external file csv
integer	unit_csv	Logical unit for export with the format csv

type (NL_DS_TableIO)	table_io	Structure of data table_container
integer	indx_vale (max)	Table of indirection enters the columns and the object being attached to it

Objects table_name, will nb_para, will list_para and will type_para allow to directly use the utilities of management of table. For example:

- cal tbajpa (tble%table_name, will tble%nb_para, tble%liste_para, tble %type_para)
- cal tbajli (tble%table_name, will tble%nb_para, tble%liste_para,…)

Without needing to rebuild the parameter lists each time.

For reasons of effectiveness, the maximum number of columns is not evaluated dynamically but is not given by the variable nb_cols_maxi. One uses it for the variables collars (*) and l_cols_acti (*).

The utilities available for the SD are the following:

- CreateTable.F90 : create a table in the SD result
- CreateVoidTable.F90 : create a table empties (initialization of all the objects)
- ComputeTableHead.F90 : create the chains allowing to write the title of the table
- ComputeTableWidth.F90 : calculate the total width of the table (according to the active columns)
- PrepareTableLine.F90 : create the chain corresponding to a line (vacuum) of the table
- PrintTableLine.F90 : create the chain (with the values and the marks) and prints it in a logical unit
- SetTableColumn.F90 : affect a column in a table
- SetTablePara.F90 : prepare the objects will list_para and will type_para

A column is defined by:

- an identifier in the form of chain;
- three chains defining the title of the column (one can use 1.2 or 3 of them according to the definition of the table);
- a flag to say if a value is affected or not (makes it possible nothing to display if the value is not defined);
- four flags to give the type of value contained in the column: entirety, chain, complex or reality;
- four variables (entirety, chain, complex, real) containing the value to be displayed;
    - the possibility of displaying a “mark” beside a value in a column. This mark is used for example to say on which residue one converges (“X”) or if one reached a terminal in the case of piloting. Note: the mark is authorized only with columns of type the whole or real, not with character strings.

A column is defined by the type NL_DS_Column who has the following structure:

Type	Name	Description
aster_logical	l_vale_affe	flag to say that one has affected a value in the column
aster_logical	l_vale_inte	flag to say that the column contains an entirety
aster_logical	l_vale_real	flag to say that the column contains a reality
aster_logical	l_vale_cplx	flag to say that the column contains a complex
aster_logical	l_vale_strg	flag to say that the column contains a chain
integer	vale_inte	value of the column if it is an entirety
real (kind=8)	vale_real	value of the column if it is a reality
complex (kind=8)	vale_cplx	value of the column if it is a complex
character (len=24)	vale_strg	value of the column if it is a chain
character (len=9)	name	name of the column (identifying single)
character (len=16)	title (3)	title of the column (to three lines of title)
character (len=1)	mark	possible mark beside the value in column (X, B, etc)

2.4.2.3 Management of measurements of time and statistics – NL_DS_Measure

The structure of data makes it possible to manage the various measurements carried out during a calculation. It stores times but also various information like the iteration count of Newton or the number of nodes in contact. One bases oneself for that on three structures of data:

- the SD NL_DS_Measure is represented by only one variable of name ds_measure. It is a aggregator of various measurements;
- the SD NL_DS_Timer manage different the timers;
- the SD NL_DS_Device is the object which carries out measurements.

1.

The structure of data NL_DS_Timer is a rather simple object which makes it possible to manage the call to the utilities of the type uttcpu. F90. It has the following structure:

Type	Name	Description
character (len=9)	type	name of the timer (identifying single)
character (len=24)	cpu_name	name for the utilities uttcpu
real (kind=8)	time_init	saved initial time

The routine nmtime. F90 manage these timers (starting, stop, measurement and rebootstrapping).
To launch a stop watch:

```
cal nmtime (ds_measure, 'Launch', 'Time_Step')
```

To stop it (and to measure):

```
cal nmtime (ds_measure, 'Stop', 'Time_Step')
```

Then, the structure of data NL_DS_Device manage measurements. Each *device* allows to measure an operation during calculation. The SD has the following structure:

2.4.2.4

Type	Name	Description
character (len=9)	type	Name of the device (identifying single)
character (len=9)	timer_name	Name of the timer attached to the device
real (kind=8)	time_iter	Time measured for an iteration of Newton
real (kind=8)	time_step	Time measured for a step of calculation
real (kind=8)	time_comp	Time measured for all calculation
integer	time_indi_step	Index in the catalogue of the messages measure.py to display time for this device with each step of time (managed in nmimpr_mess. F90)
integer	time_indi_comp	Index in the catalogue of the messages measure.py to display time for this device at the end of the calculation (managed in nmimpr_mess. F90)
aster_logical	l_count_add	Flag to cumulate the number of occurrences to each stage
integer	count_iter	Meter of occurrences for an iteration of Newton
integer	count_step	Meter of occurrences for a step of calculation
integer	count_comp	Meter of occurrences for all calculation
integer	count_indi_step	Index in the catalogue of the messages measure.py to display the meter for this device with each step of time (managed in nmimpr_mess. F90)
integer	count_indi_comp	Index in the catalogue of the messages measure.py to display the meter for this device at the end of the calculation (managed in nmimpr_mess. F90)

The routine nmrvai. F90 allows to manage the meters.

To increment a meter (many steps of time for example):
```
cal nmrinc (ds_measure, 'Time_Step')
```
To give the value of a meter:
```
cal nmrvai (ds_measure, 'Contact_NumbCont', input_count = nbliac)
```
In fact nmrinc. F90 it is nmrvai. F90 with input_count = 1.

Lastly, the structure of data NL_DS_Measure manage the whole of the SD allowing measurements (time and various statistics). It has the following structure:

Type	Name	Description
aster_logical	l_table	.true when the user wrote TABLE=' OUI' in the keyword MEASUREMENT
type (NL_DS_Table)	table	Table at exit for the statistics
integer	nb_device	Number of device used
integer	nb_device_maxi	Number of device maximum
type (NL_DS_Device)	device (max_{device})	List of device
aster_logical	l_device_acti (max_{device})	List of device credits
integer	indx_cols ($2*max_{device}$)	Reference of the columns of the table towards device
integer	nb_timer	Number of timer used
integer	nb_timer_maxi	Number of timer maximum
type (NL_DS_Timer)	timer($2*max_{timer}$)	List of timer
real (kind=8)	store_mean_time	Average time for filing
real (kind=8)	iter_mean_time	Average time by iteration of Newton
real (kind=8)	step_mean_time	Average time by step of time
real (kind=8)	iter_remain_time	Time remaining for the iteration
real (kind=8)	step_remain_time	Time remaining for the step of time

The whole of the data of this structure is created in the principal routine nmcrti. F90. This routine uses the routine ActivateDevice.F90 who allows to activate one device according to the availability of certain features.

2.4.2.5 Management of the calculation of energies – NL_DS_Energy

To measure energies, one uses a structure of data of the type NL_DS_Energy who has the following structure:

Type	Name	Description
aster_logical	l_comp	.true Quand one wants to measure energies
type (NL_DS_Table)	table	table at exit for energies (see §16)
character (len=16)	command	Appealing order

2.4.2.6 Management of the errors of the algorithm – SDERRO

This SD deals with managing the errors of the algorithm. It contains seven of the same objects cuts, that amongst events (variable ZEVEN) manageable by the algorithm (modifiable in nmcrga , cf § 3.4.1)

SDERRO – Objects	
Name	**Description**

SDERRO (1:19)/'.ENOM'	Name of the event
SDERRO (1:19)/'.ECOV'	Value of the code-return related to the event
SDERRO (1:19)/'.ECON'	Name of the code-return related to the event
SDERRO (1:19)/'.ENIV'	Type and level of release of the event
SDERRO (1:19)/'.EFCT'	Functionality activating an event of type convergence or divergence
SDERRO (1:19)/'.EACT'	State of the event (activated or not)
SDERRO (1:19)/'.EMSG'	Code of the message to be displayed when the event starts

The two other objects make it possible to manage the state of the loop and to store the last started event (will be used for postings).

SDERRO – Objects	
Name	**Description**
SDERRO (1:19)/'.CONV'	State of the loop
SDERRO (1:19)/'.EEVT'	Information on the last started event

SD ERRO be used by using certain contained information in SDDISC (§2.4.2.18), coming from the definitions of the events of the order DEFI_LIST_INST.

Operation on Gestion of the errors of the algorithm – SDERRO	**Routine**
Creation of SDERRO	nmcrga
Recording of an event	nmcrel
Recording of an event starting from a code-return	nmcret
Change of the state of a loop	nmeceb
Reading of the state of a loop	nmleeb
Restoring of the events	nmeraz
Turn over the state of an event (active or not) according to its name	nmerge
Emission of the message of information on the event	nmevim
Evaluation of the state of convergence of a loop	nmevcv
Turn over the state of an event (active or not) according to its type	nmltev

The use of these routines within the framework of the management of the events is detailed in the § 3.4 .

2.4.2.7 Management of the inputs/outputs – NL_DS_InOut

One calls inputs/outputs of the non-linear operators the whole of the operations related to the following features:

- Reading of an initial state (keyword factor ETAT_INIT);
- Extraction of the results in a table during calculation (keyword factor OBSERVATION) and monitoring in real time in the table of convergence (keyword factor SUIVI_DDL);
- Filing of the results in the SD result (keyword factor FILING).

There are two derived types for this functionality: NL_DS_Inout and NL_DS_Field.
Only one variable ds_inout of type NL_DS_Inout manage the whole of the input-outputs. The main object ds_inout contains parameters resulting from the user (keyword ETAT_INIT and FILING) as well as a list of fields and their behavior in input/output.
This SD does not manage yet the case of the _paramètres_ of the SD results (except the list of the loads), but only the list of the fields. It does not manage either the utilities related to the clocking of filing (for the moment

in SDDISC to see §2.4.2.18) nor related information in the manner of carrying out one OBSERVATION (for the moment in SDOBSE to see §2.4.2.13). The type NL_DS_Inout has the following structure:

Type	Name	Description
character (len=8)	result	name of the SD result to file
integer	nb_field	many managed fields
integer	nb_field_maxi	maximum number of manageable fields
type (NL_DS_Field)	field (nb_field_maxi)	List of the fields
character (len=8)	stin_evol	name of the SD result in ETAT_INIT
aster_logical	l_stin_evol	flag for the presence of a SD result in ETAT_INIT
aster_logical	l_field_acti (nb_field_maxi)	flags of the active fields (depends on the features)
aster_logical	l_field_read (nb_field_maxi)	flags to indicate that a field is to be considered in the initial state
aster_logical	l_state_init	flag to indicate the presence of an initial state
aster_logical	l_reuse	flag to indicate that one is in mode reuse
integer	didi_nume	sequence number for the loadings DIDI (NUME_DIDI in ETAT_INIT)
character (len=8)	criterion	value of CRITERION in ETAT_INIT (for selection by one moment)
real (kind=8)	precision	value of PRECISION in ETAT_INIT (for selection by one moment)
real (kind=8)	user_time	value of the initial state given by one moment in ETAT_INIT
aster_logical	l_user_time	flag to say that the value of the initial state is given by one moment in ETAT_INIT
integer	user_nume	value of the initial state given by a sequence number in ETAT_INIT
aster_logical	l_user_nume	flag to say that the value of the initial state is given by a sequence number in ETAT_INIT
real (kind=8)	stin_time	value of the initial state defined by INST_ETAT_INIT in ETAT_INIT
aster_logical	l_stin_time	flag to say that the value of the initial state is defined by INST_ETAT_INIT in ETAT_INIT
real (kind=8)	init_time	value of the initial moment
integer	init_nume	value of initial sequence number
character (len=19)	list_load_resu	name of object JEVEUX storing the list of the loadings in the SD result
aster_logical	l_init_stat	flag to say that the initial state is stationary (for thermics)
aster_logical	l_init_vale	flag to say that the initial state is a value (for thermics)
real (kind=8)	temp_init	temperature given by the initial state when it is a value (for thermics)
type (NL_DS_TableIO)	table_io	Structure of data for table_container PARA_CALC

This object thus gathers at the same time information resulting from the user, but also the definition of the fields of input-output and their behavior, defined by the developer. A field has several states:

- It can define an initial state;
- It can be observed;
- It can be filed.

These three states are not inevitably independent. For example, a field which is read in the initial state is inevitably archivable but the reverse is not true (it exists fields having to be filed but not being in the initial state like BEHAVIOR , CONT_NOEU , CRIT_STAB , etc).
A field is usable (initial state, observation and filing) only so certain features are active (for example, the field speed in dynamics or statutes of contact for the contact)
If a field east defines in the initial state, it could be read in ETAT_INIT (either in an individual way, by field, or in the SD result given in ETAT_INIT), or created (equal to zero) by the order (in this case, it will be necessary to define the name of this null field and to create it).
There is thus a list of fields (variable field) defined by the derived type NL_DS_Field. This object has the following structure:

Type	Name	Description
character (len=16)	type	single identifier of the field. It is also the reference symbol defined in the SD result
character (len=8)	gran_name	Type of size (DEPL_R, SIEF_R, etc…)
character (len=8)	field_read	name of the field given by the user in ETAT_INIT
character (len=4)	disc_type	type of discretization (NOEU, ELGA, ELNO,…) field given by the user in ETAT_INIT
character (len=8)	init_keyw	keyword corresponding to this field for ETAT_INIT
character (len=16)	obsv_keyw	keyword corresponding to this field for OBSERVATION (and SUIVI_DDL)
aster_logical	l_read_init	flag to say that this field must be defined in initial state
aster_logical	l_store	flag to say that this field must be filed in the SD result
aster_logical	l_obsv	flag to say that this field is “observable” (present in OBSERVATION and SUIVI_DDL)
character (len=24)	algo_name	name of object JEVEUX corresponding to this field in the algorithm
character (len=24)	init_name	name of object JEVEUX corresponding to the field initial no one in the algorithm
character (len=4)	init_type	known as how this field was initialized (read in ETAT_INIT, field by field, in the SD result, etc).

The management of the SD is done in three times:

- Creation of the SD in the routine CreateInOutDS . This creation understands in particular the creation of all the possible fields. It is in this routine that the developer can add and define behavior (state initial, archivable, observable, etc.) of a new field;
- Reading of the information given by the user (keyword ETAT_INIT). Routine ReadInOut in nmdata ;
- Initialization of the SD: routine nmetcr for mechanics and ntetcr for thermics. These routines will activate the fields according to the features present

The utility routines of access to the SD are the following ones:

- GetIOField => to recover information of a given field
- SetIOField => to give the information of a given field

These routines uses the type of the field as starter (variable type SD NL_DS_Field).

Context of use: the developer wants to add a field, to make it usable in OBSERVATION, FILING, SUIVI_DDL or ETAT_INIT.

1. It starts by publishing the routine CreateInOutDS_M or CreateInOutDS_M according to whether one is in mechanics or thermics. It is enough exhaustively to supplement information at the beginning of this routine. If a new field is added, it will be advisable to modify the full number (parameter nb_field_maxi);
2. If it must add a field in the SD result, it is necessary in addition to modify rscrsd ;
3. The field is from now on _possiblement_ activable. But one must activate it (for example under condition of a functionality activates) in the routine nmetac ;
4. Lastly, if it is necessary to have a virgin initial state, it is advisable to modify the routine nmetc0 to create this field.

2.4.2.8 Management of the contact – NL_DS_Contact

The contact is managed by two types of structures of data:
- For *definition* contact and unilateral connections[1] (operator DEFI_CONTACT), it is necessary to refer to the documentation [D4.06.14] of sd_contact. This SD is not concerned in this document;
- For *resolution* contact and unilateral connections, a called derived type is introduced NL_DS_Contact.

The main object ds_contact (of type NL_DS_Contact) contains the parameters useful for the resolution of the problem of contact/friction. Only one object ds_contact of type NL_DS_Contact manage the whole of the contact/friction. It has the following structure:

Type	Name	Description
aster_logical	l_contact	flag to say that the contact or the seepage is activated in the operator
aster_logical	l_meca_cont	flag to say that the mechanical contact is activated in the operator
aster_logical	l_meca_unil	flag to say that the seepage is activated in the operator
character (len=8)	sdcont	name of the concept DEFI_CONTACT informed in STAT_NON_LINE/CONTACT
character (len=24)	sdcont_defi	name of object JEVEUX for the definition of the contact
character (len=24)	sdcont_solv	prefix of objects JEVEUX for the resolution of the contact
character (len=24)	sdunil_defi	name of object JEVEUX for the definition of LIAISON_UNIL
character (len=24)	sdunil_solv	prefix of objects JEVEUX for the resolution of LIAISON_UNIL
aster_logical	l_form_cont	flag to say that one is in FORMULATION=' CONTINUE'
aster_logical	l_form_disc	flag to say that one is in FORMULATION=' DISCRET'
aster_logical	l_form_xfem	flag to say that one is in FORMULATION=' XFEM'
aster_logical	l_form_lac	flag to say that one is in FORMULATION=' LAC'
aster_logical	l_elem_slav	flag to say the presence of elements slaves of contact (CONTINUE/LAC/XFEM)
character (len=8)	ligrel_elem_slav	name of <LIGREL> for the elements slaves (created in DEFI_CONTACT)
aster_logical	l_elem_cont	flag to say the presence of elements of contact (CONTINUE/LAC/XFEM)
character (len=19)	ligrel_elem_cont	name of <LIGREL> for the elements of contact (created in STAT_NON_LINE)
aster_logical	l_iden_rela	flag to say the presence of relations of identity to be treated by

1 Unilateral conditions (LIAISON_UNILATER) the conditions of suction and seepage for the problems in THM manage

		modification of the matrix (XFEM with ELIM_ARETE or method LAKE)
character (len=24)	iden_rela	name of the defining SD of the relations of identity
aster_logical	l_dof_rela	flag to say the presence of linear relations between DDL (QUAD8 for the discrete methods or XFEM, created in DEFI_CONTACT)
character (len=8)	ligrel_dof_rela	name of the defining SD linear relations between DDL
character (len=19)	field_input	name of CHAM_ELEM as starter of TE for the methods XFEM/CONTINUE/LAC
character (len=14)	nume_dof_frot	name of NUME_DDL for the matrix of discrete friction
character (len=19)	field_cont_node	name of the field to the nodes CONT_NOEU for postprocessing
character (len=19)	fields_cont_node	name of the simple field to the nodes CONT_NOEU for postprocessing
character (len=19)	field_cont_perc	name of the field to the nodes of the percussions/impacts for postprocessing
integer	nb_loop	effective number of loops of treatment of the contact to be managed
integer	nb_loop_maxi	maximum number of loops of treatment of the contact to be managed
NL_DS_Loop	loop (nb_loop_maxi)	pointers towards loops of treatment of the contact to be managed
aster_logical	l_renumber	flag for renumerotation of the matrix
real (kind=8)	geom_maxi	value to control the loop on the geometry
aster_logical	l_getoff	Flag for the indicator of separation (theta-diagram)
aster_logical	l_first_geom	Flag to say that it is the first iteration of geoemtric loop
aster_logical	to l_pair	Flag to say that it is necessary to pair
aster_logical	l_wait_conv	Flag to say that it is necessary to expect the fixed point of the under-iterations of contact (friction) discrete

To manage the loops of fixed point of the contact (geometry, friction and statutes of contact), one has creates the derived type NL_DS_Loop of which here the structure:

Type	Name	Description
character (len=4)	type	Chain identifying the type of the loop (Geom, Money and Cont)
integer	counter	iterator for the meter of loop
aster_logical	conv	rapeau to say that this loop is converged
aster_logical	error	flag to say that there was an error during the evaluation of the loop
real (kind=8)	vale_calc	value of convergence of the loop
character (len=16)	locus_calc	place of convergence of the loop

The management of the SD is done in three times:

- Creation of the SD in the routine CreateContactDS ;
- Reading of the information given by the user (keyword CONTACT). Routine ReadContact in nmdata ;
- Initializations of the SD:

- Routine InitContact in nminit . This phase understands in particular the reading of the type of contact for LIGREL to manage (see routine nmdoct);
- Routine cfmxsd in nminit. Creation of the objects necessary for the resolution of the contact (primarily discrete). They are dynamic objects (dependent amongst nodes in contact) and it is thus necessary to make objects JEVEUX of them;
- Routine cucrsd in nminit. Creation of the objects necessary for the resolution of the unilateral connections. They are dynamic objects (dependent amongst nodes) and it is thus necessary to make objects JEVEUX of them;
- Routine nmdoct : management of the late LIGREL

The use of the SD is direct by call to its under-objects (use of %) or by already existing utility routines (like the routines cfdis* and mminf*), already used for the access to sdcont coming from DEFI_CONTACT.

2.4.2.9 Management of convergence – NL_DS_Conv

It is a question here of managing information and the algorithm concerning the management of convergence, which understands:

- the management and the calculation of the residues of balance;
- the management of the options coming from the keyword factor CONVERGENCE.

The derived types are created NL_DS_Resi , NL_DS_ResiRefe and NL_DS_Conv. Only one object ds_conv of type NL_DS_Conv manage the whole of convergence. It contains various information:

- the information recovered in the keyword CONVERGENCE ;
- a site to store the value having started the rocker RESI_GLOB_RELA towards RESI_GLOB_MAXI ;
- two variables to manage linear research.

It has the following structure:

Type	Name	Description
integer	nb_resi	many residues
integer	nb_resi_maxi	many residues to the maximum
type (NL_DS_Resi)	list_resi (max)	list of the residues
aster_logical	l_resi_test (max)	flag to say if one must use this residue for convergence
integer	nb_refe	component count of the type RESI_REFE_RELA
integer	nb_refe_maxi	maximum number of components of the type RESI_REFE_RELA
type (NL_DS_RefeResi)	list_refe (max2)	list of the components
aster_logical	l_refe_test (max2)	flag to say if the component is active or not
integer	iter_glob_maxi	parameter ITER_GLOB_MAXI
integer	iter_glob_elas	parameter ITER_GLOB_ELAS
aster_logical	l_stop	parameter ARRET= " NOT"
aster_logical	l_iter_elas	flag to say that the user informed explicitly ITER_GLOB_ELAS
real (kind=8)	swap_trig	value having started the rocker RESI_GLOB_RELA towards RESI_GLOB_MAXI
real (kind=8)	line_sear_coef	coefficient of linear research
integer	line_sear_iter	iteration count of linear research

This object contains the list of the possible residues (currently, there are six residues entering the evaluation of convergence). Each residue is definite itself by the derived type NL_DS_Resi who has the following structure:

Type	Name	Description
character (len=16)	type	identifier of the type of residue
character (len=16)	col_name	identifier of the column of the table of convergence storing the value of the residue
character (len=16)	col_name_locus	identifier of the column of the table of convergence storing the site of the maximum standard of the residue (when INFO_RESIDU=' OUI')
character (len=16)	event_type	identifier of the event of divergence of the residue
real (kind=8)	vale_calc	value of the maximum standard of the residue
character (len=16)	locus_calc	site of the maximum standard of the residue
real (kind=8)	will user_para	value of the criterion given by the user
aster_logical	l_conv	flag to indicate that vale_calc < will user_para

Lastly, the object ds_conv also the list of the objects contains defining the residue of the type RESI_REFE_RELA. Currently, eleven types of components exist. An object of the type NL_DS_ResiRefe has the following structure:

Type	Name	Description
character (len=16)	type	type of the component
real (kind=8)	will user_para	value of the component given by the user
character (len=8)	cmp_name	name of the component in the size for TE

This SD is mainly used for the evaluation of the residues. The routine which evaluates their convergence is nmcore. From now on, it is autonomous in the direction where the addition of a new type of residue on the other hand does not require specific impact for this part (, it will be necessary to carry out the effective calculation of this residue, which is currently, mainly carried out in nmresi).
To facilitate maintenance and legibility, one also extracted the part which deals with the “rockers”:

- Passage de RESI_GLOB_RELA with RESI_GLOB_MAXI when the external loading is null;
- Passage de RESI_COMP_RELA with RESI_GLOB_RELA at the first moment

Note:
It is possible to make a double rocker RESI_COMP_RELA => RESI_GLOB_RELA => RESI_GLOB_MAXI .

The management of the SD is done in three times:

1. Creation of the SD in the routine CreateConvDS in nmini0. This creation understands in particular the creation of all the residues available. Their effective use in the evaluation of convergence will depend on the state of the flag of their activation;
2. Reading of the information given by the user (keyword POSTING). Routine nmdocn in nmdata . Information stored in ds_conv ;
3. Initialization of the SD in InitConv . This routine manages in particular alarms (ARRET=' NON' or too loose residue) and activates the residues specific to the contact (because they are read in nmdoct and not in nmdocn).

To handle the SD, there are the pairs of routines SetResi/GetResi and SetResiRefe.

2.4.2.10 Management of the parameters of the algorithm – NL_DS_AlgoPara

This object stores two types of information:

- data resulting from the options of the keyword factor NEWTON (REAC_ITER, types of matrices, options for linear research, etc);
- intermediate results of the algorithm: search results linear and values of the residues of convergence calculated.
  The derived types are created NL_DS_LineSearch and NL_DS_AlgoPara . An object of the type NL_DS_AlgoPara manage the parameters of the algorithm, it has the following structure:

Type	Name	Description
character (len=16)	method	keyword METHOD (Newton, Newton-Krylov, IMPLEX)
character (len=16)	matrix_pred	type of matrix in prediction
character (len=16)	matrix_corr	type of matrix in correction
integer	reac_incr	value of REAC_INCR
integer	reac_iter	value of REAC_ITER
real (kind=8)	pas_mini_elas	value of PAS_MINI_ELAS
integer	reac_iter_elas	value of REAC_ITER_ELAS
aster_logical	l_line_search	flag to say that linear research is activated
type (NL_DS_LineSearch)	line_search	object to describe the parameters of linear research
character (len=8)	result_prev_disp	SD result for DEPL_CALCULE/EXTRAPOLE
aster_logical	l_matr_rigi_syme	Flag to symmetrize the matrix of rigidity (MATR_RIGI_SYME)

The object line_search is a type derived from type NL_DS_LineSearch of which here the structure:

Type	Name	Description
character (len=16)	method	linear type of research
real (kind=8)	resi_rela	tolerance for linear research
integer	iter_maxi	maximum number of iterations of linear research
real (kind=8)	rho_mini	minimal value of the coeff. of linear research
real (kind=8)	rho_maxi	maximum value of the coeff. of linear research
real (kind=8)	rho_excl	value to be excluded for the coeff. of linear research

Only one variable is defined will ds_algopara of type NL_DS_AlgoPara who will be transmitted as argument in the routines which need some. The management of the SD is done in three times:

1. Creation of the SD in the routine CreateAlgoParaDS in nmini0 ;
2. Reading of the information given by the user (keyword NEWTON). Routine nmdomt ;
3. Reading of the information given by the user (keyword RECH_LINEAIRE). Routine nmdomt_ls ;
4. Initialization of the SD by the routine InitParaAlgo.

2.4.2.11 Management of the extraction of fields – SDEXTR

SUIVI_DDL division withOBSERVATION (see §2.4.2.13) a called common SD SDEXTR, routine of extraction of the values of the fields. One will thus start by describing this last SD. The first three objects are generals and their length is proportional to the number of occurrences of the keyword SUIVI_DDL or OBSERVATION.

SDEXTR – Objects	
Name (attention with the white!)	**Description**

SDEXTR (1:14)/' .INFO'	Various information on the data of extraction
SDEXTR (1:14)/' .EXTR'	Type of extraction • On field (NOEU and ELGA) • On the mesh (if ELGA) • On the components or formulates between the components
SDEXTR (1:14)/' .ACTI'	Active extraction or not

One cannot have more than 99 occurrences of the keywords because one builds the name of certain objects starting from this number of occurrence OCC. These objects are the following:

SDEXTR – Objects	
Name (attention with the white!)	**Description**
SDEXTR (1:14) //OCC// ' .NOEU'	Nodes concerned with the extraction
SDEXTR (1:14) //OCC// ' .MAIL'	Meshs concerned with the extraction
SDEXTR (1:14) //OCC// ' .POIN'	Points of integration concerned with the extraction
SDEXTR (1:14) //OCC// ' .SSPI'	Under-points of integration concerned with the extraction
SDEXTR (1:14) //OCC// ' .CMP'	Components concerned with the extraction

2.4.2.12 Management of SUIVI_DDL – SDSUIV

This SD is used to manage the functionality SUIVI_DDL. Besides the references to the extraction of the fields (SDEXTR, §2.4.2.11), we have a specific object for SUIVI_DDL.

SDSUIV – Objects	
Name (attention with the white!)	**Description**
SD SUIV (1:14)/' . TITR '	Titles of the columns

2.4.2.13 Management ofOBSERVATION – SDOBSE

This SD is used to manage the functionality OBSERVATION. OBSERVATION division with SUIVI_DDL (see §2.4.2.12) a called common SD SDEXTR, already described in (§2.4.2.11).

Then, we have specific objects for OBSERVATION. One which will give the name of the table and one which stores the relative information at the frequency of observation (object utility SDSELI, to see §2.4.2.19), subscripted by the number of occurrence OCC keyword OBSERVATION.

SDOBSE – Objects	
Name (attention with the white!)	**Description**
SDOBSE (1:14)/' .TABL'	Name of the table
SDOBSE (1:14) //OCC// '.LI'	Access to SDSELI, list of the moments to be observed

2.4.2.14 Management of the quality standards – SDCRIQ

This SD is used to evaluate the quality standards (keyword CRIT_QUALITE). It is very simple and comprises only one object.

SDCRIQ – Objects	
Name	**Description**
SDSUIV (1:1 9)/'.ERRT'	Value of errors THM spaces and time, coefficient THETA

2.4.2.15 Management of piloting – SDPILO

This SD is used for piloting (method of continuation). Most these objects are gathered in a logic of the type (realities with realities, chains with chains), rather than in a logic of functionality.

SDPILO – Objects	
Name	**Description**
SDPILO (1:19)/'.PLTK'	Contains the parameters of the piloting (of type chains) or the names of objects necessary to piloting
SDPILO (1:19)/'.PLIR'	Contains the parameters of piloting (of real type)
SDPILO (1:14)/'.PLCR'	List of the coefficients for piloting
SDPILO (1:14)/'.PLSL'	List of the DDL of the type DX, DY and DZ for piloting
SDPILO (1:14)/'.PLCI'	List of the coefficients for piloting – Case XFEM

There are no routines of access of high-level, the SD is created in the routine nmdopi. the SD should be attacked directly.

2.4.2.16 Management of classifications – SDNUME

This SD comes in complement as of two SD NUMDDL and NUMFIX to manage the classification of the equations. NUMDDL is the classification of the equations, which can be variable during the transient, when one uses features like the contact CONTINUOUS or the contact XFEM. NUMFIX is fixed classification, it is useful in the case of the macronutrients, to be able to combine the matrices. SDNUME contains three other objects:

SDDigital – Objects	
Name	**Description**
SDNUME (1:19)/'.NDRO'	Location of the DDL for great rotations
SDNUME (1:19)/'.NUCO'	Location of the DDL for the Lagrangian ones of contact and friction
SDNUME (1:19)/'.ENDO'	Location of the DDL for the damage with the nodes

These three objects have same logic: they are dimensioned with the full number of degrees of freedom of the structure, with same classification as the matrices and them CHAM_NO used in op0070. A particular value (in general 1), is used to locate specific DDL a priori: those corresponding to great rotations, those corresponding to Lagrangian of contact/friction and those corresponding to the damage. One then uses them in certain cases (for example, for the update specific of the fields in the case of great rotations or to filter the components in the evaluation of the residues). There are no specific routines of access.

2.4.2.17 Management of dynamics – SDDYNA

This SD contains all the necessary information with calculation in dynamics.

SDDYNA – Objects	
Name	**Description**

SDDYNA (1:15)/'.PARA_SCH'	Parameters of the diagrams in time
SDDYNA (1:15)/'.INFO_SD'	Parameters of dynamics
SDDYNA (1:15)/'.NOM_SD'	Name of the SD for dynamics (static modes, vectors,…)
SDDYNA (1:15)/'.TYPE_FOR'	Type of formulation (displacement, speed or acceleration)
SDDYNA (1:15)/'.COEF_SCH'	Coefficients to be used in calculation
SDDYNA (1:15)/'.TYPE_CHA'	Relative information with the loading ONDE_PLANE
SDDYNA (1:15)/'.NBRE_CHA'	Relative information with certain loadings specific to dynamics
SDDYNA (1:15)/'.VEEL_OLD'	Elementary vectors of the preceding step for the multi-step diagrams
SDDYNA (1:15)/'.VEAS_OLD'	Assembled vectors of the preceding step for the multi-step diagrams
SDDYNA (1:15)/'.VECENT'	Quantities in trainings (displacements, speeds and acceleration)
SDDYNA (1:15)/'.VECABS'	Absolute quantities (displacements, speeds and acceleration)

The access to this information is done via four routines dedicated each one to a type, with a chain asking the question (about the model of the routine dismoi):

Operation on Gestion of dynamics – SDDYNA	**Routine**
Reading information of the type <booléen>	ndynlo
Reading information of the type <réel>	ndynre
Reading information of the type <entier>	ndynin
Reading information of the type <chaîne>	ndynkk

To note that these questions will involve a fatal error if one is not in dynamics, except in the case of the question NDYNLO (SDDYNA, 'DYNAMICS') who will answer .false. if one is in statics (i.e. which can detect the case where SDDYNA do not exist). Apart from these four routines, the access to SDDYNA is done directly in two routines:

Operation on Gestion of dynamics – SDDYNA	**Routine**
Reading information in the command file	ndlect
Recording value of the various coefficients	ndnpas

One reconsiders the coefficients necessary for calculation in dynamics because they are very numerous. These coefficients are built starting from three information:

- type of diagram;
- Parameters of the diagram (coefficients ALPHA, BETA, KAPPA, etc);
- The increment of time;

The fact of depending on the step of time implies that the coefficients are revalued with each step (in the routine ndnpas).

Coefficient	**Description**
COEF_MATR_RIGI	Coefficient in front of the matrix of rigidity
COEF_MATR_AMOR	Coefficient in front of the matrix of damping
COEF_MATR_MASS	Coefficient in front of the matrix of mass
COEF_DEPL_DEPL	Predictor in displacement: coefficient in front of the displacement of the preceding step
COEF_DEPL_VITE	Predictor in displacement: coefficient in front of the speed of the preceding step
COEF_DEPL_ACCE	Predictor in displacement: coefficient in front of the acceleration of the preceding

	step
COEF_VITE_DEPL	Predictor of speed: coefficient in front of the displacement of the preceding step
COEF_VITE_VITE	Predictor of speed: coefficient in front of the speed of the preceding step
COEF_VITE_ACCE	Predictor of speed: coefficient in front of the acceleration of the preceding step
COEF_VITE_DEPL	Predictor in acceleration: coefficient in front of the displacement of the preceding step
COEF_VITE_VITE	Predictor in acceleration: coefficient in front of the speed of the preceding step
COEF_VITE_ACCE	Predictor in acceleration: coefficient in front of the acceleration of the preceding step
COEF_DEPL	Coefficient in front of the increment of displacement
COEF_VITE	Coefficient in front of the increment speed
COEF_ACCE	Coefficient in front of the increment of acceleration
COEF_MPAS_FEXT_PREC	Coefficient in front of the forces external of the preceding step (multi-step diagram)
COEF_MPAS_FINT_PREC	Coefficient in front of the interior forces of the preceding step (multi-step diagram)
COEF_MPAS_FEXT_COUR	Coefficient in front of the forces external of the step running (multi-step diagram)
COEF_MPAS_EQUI_COUR	Coefficient in front of the other terms of the second member (inertia, damping)
COEF_FDYN_MASSE	Coefficient in front of the back pulling forces dynamics (inertia)
COEF_FDYN_AMORT	Coefficient in front of the back pulling forces dynamics (damping)
COEF_FDYN_RIGID	Coefficient in front of the back pulling forces?????? (is used for Krenk???)
COEF_FORC_INER	Coefficient in front of the inertial forces to be put at the denominator in the residue of balance
INST_PREC	Pas de previous time (only useful for the continuation with, at the same time, the complete diagram HHT and laws of behavior which need this parameter)

2.4.2.18 Management of the temporal discretization – SDDISC

The structure of data SDDISC contains all information coming from the operator DEFI_LIST_INST (creation of the concept SDLIST) and of the relative information to the management of the temporal discretization in op0070. Initially, information coming from the operator DEFI_LIST_INST (see [D4.06.17]) are recopied locally in SDDISC.

SDDISC – Objects	
Name	**Description**
SDDISC (1:19)/'.LINF'	Information on the list of moments (see contained in [D4.06.17]) Recopy of the object SDLIST (1:8)/'.LISTE.INFOR'
SDDISC (1:19)/'.DITR'	List of the moments – This list is dynamic (in the event of cutting or of acceleration of the step of time)
SDDISC (1:19)/'.DINI'	Indicator of the level of under-cutting for each step of time. Initially, the level of cutting is worth 1. There cannot be more than one level of cutting between two successive steps (checking in the routine nmdcin) – This list is dynamic (in the event of cutting or of acceleration of the step of time)
SDDISC (1:19)/'.ITER'	Iteration count of Newton which it was necessary for each step of time – This list is dynamic (in the event of cutting or of acceleration of the step of time)
SDDISC (1:19)/'.EPIL'	Indicator of the choice of the solution of piloting (action

	AUTRE_PILOTAGE)
SDDISC (1:19)/'.LIPO'	List of the obligatory moments of calculation. This object is used in the case of the automatic adaptation of the step as time, it indicates the moments which will be calculated though it arrives (even in the event of enlarging of the step of time). One also speaks about urgent "stakes". Its length is that of the initial list of moments given by the user in DEFI_LIST_INST. It is thus about a list ***not-dynamics***.
SDDISC (1:19)/'.REPC'	Indicator D E management of the reactualization of the pre-packaging (action REAC_PRECOND)
SDDISC (1:19)/'.EEVR'	Recopy of the object SDLIST (1:8)/ '.ECHE.EVENR' See contained in [D4.06.17]
SDDISC (1:19)/'.EEVK'	Recopy of the object SDLIST (1:8)/ '.ECHE.EVENK' See contained in [D4.06.17]
SDDISC (1:19)/'.ESUR'	Recopy of the object SDLIST (1:8)/ '.ECHE.SUBDR' See contained in [D4.06.17]
SDDISC (1:19)/'.AEVR'	Recopy of the object SDLIST (1:8)/ '.ADAP.EVENR' See contained in [D4.06.17]
SDDISC (1:19)/'.AEVK'	Recopy of the object SDLIST (1:8)/ '.ADAP.EVENK' See contained in [D4.06.17]
SDDISC (1:19)/'.ATPR'	Recopy of the object SDLIST (1:8)/ '.ADAP.TPLUR' See contained in [D4.06.17]
SDDISC (1:19)/'.ATPK'	Recopy of the object SDLIST (1:8)/ '.ADAP.TPLUK' See contained in [D4.06.17]
SDDISC (1:19)/'.AEXT'	Object for the prolongation of cutting (treatment of the case of COLLISION)
SDDISC (1:19)/'.IFCV'	Object storing the state of convergence for the adaptation V (1) – Value of MAX (ITER_GLOB_MAXI, ITER_GLOB_ELAS) V (2) – Value of MIN (ITER_GLOB_MAXI, ITER_GLOB_ELAS) V (3) – Maximum number of possible iterations (including the possible additional iterations by ITER_SUPPL) V (4) – Value of 'PAS_MINI_ELAS' V (5) – Value of 'RESI_GLOB_RELA' V (6) – Value of 'RESI_GLOB_MAXI' V (7) – Type of residue requested by the user: =1 for RESI_GLOB_RELA =2 for RESI_GLOB_MAXI =3 for RESI_GLOB_RELA and RESI_GLOB_MAXI V (8) – Value of 'INIT_NEWTON_KRYLOV' V (9) – Value of 'ITER_NEWTON_KRYLOV' V (1 0) – Indicate that one authorizes iterations moreover
SDDISC (1:19)/'.IFRE'	Object storing the value of the residues to each iteration of Newton V (3* (ITER-1) +0) – Value DE RESI_GLOB_RELA V (3* (ITER-1) +1) – Value DE RESI_GLOB_MAXI V (3* (ITER-1) +2) – Value of the loading

To handle the SDDISC, a series of utilities are used.

Operation on Gestion of the temporal discretization – SDDISC	**Routine**
General utility routine which gives access the contents objects .LINF, .EEVR, .EEVK, .ESUR, .AEVR, .AEVK, .ATPR, .ATPK, .REPC and .EPIL. The access can be done in reading or	utdidt

writing.	
Turn over .RUE. if one leaves the list of moments (end of the transient)	didern
Value of the moment according to number of the moment	diinst

2.4.2.19 Management of the one moment selection – SDSELI

The structure of data SDSELI allows to select one moment following a frequency, a list of values, by taking into account a tolerance and a kind of criterion (absolute or relative).

SDSELI – Objects	
Name	**Description**
SDSELI (1:19)/'.INFL'	General information on the selection (parameters of selection)
SDSELI(1:19)/'.LISTE'	List of the moments given by the user

To operate on this SD, mainly two routines are used:
- one which reads the parameters of the user (nmcrpx);
- research if the moment given is selected (nmcrpo);

Operation on Gestion of one moment selection – SDSELI	Routine
Reading of the list of the moments to be selected	nmcrpa
Reading of the precision and the type of criterion (relative or absolute)	nmcrpp
Reading of all information (call to nmcrpa and nmcrpp)	nmcrpx
Search for a reality in a list (with precision and criterion)	utacli
Principal routine of research of the moment	nmcrpo
Research of the index in the SD result right before a given moment	nmttch

This SD is used for the management of the temporal discretization, the initial state, filing and the observation.

2.4.2.20 Management of the convergence criteria – SDCRIT

This SD manages relative information with the convergence criteria, it is used in particular to store the residues in the SD RESULT.

SD CRIT – Objects	
Name	**Description**
SDCRIT (1:19)/'.CRTR'	Values of information (residues, iteration count, etc)
SDCRIT (1:19)/'.CRDE'	Name of information for storage in the SD RESULT (residues, iteration count, etc)

This SD is also used for the order THER_NON_LINE (attention, the objects are not same dimension!). The safeguard of information for this SD is done in the routine nmcore who evaluates convergence with each iteration of Newton.

Operation on Gestion of the convergence criteria – SDCRIT	Routine
Creation of the SD	nmcrcv ntcrcv
Safeguard of information in SDCRIT	nmcore op0186

Safeguard of information in the SD RESULT	nmarc0 ntarc0

2.4.2.21 Management DE the list of selection – NL_DS_SelectList

This structure of data makes it possible to select one moment starting from a list or of a frequency.

Type	Name	Description
integer	nb_value	Many values in the list of the moments given
real (kind=8)	incr_mini	Increment of minimum time between two moments of the list
real (kind=8), to point	list_value	List of the moments
real (kind=8)	precision	Value of the precision to select one moment
aster_logical	l_abso	.true is worth. If the value is selected into absolute
real (kind=8)	tolerance	Calculated tolerance if selection into relative
integer	freq_step	Frequency of selection of the moments
aster_logical	l_by_freq	.true is worth. If one selection by the frequency

The routine selectListRead allows to read the keywords of the user and fills the structure of data.
The routine selectListGet turn over .true. If the moment running is well to select compared to the parameters (frequency, list of moments with precision, etc). Lastly, the routine selectListClean désalloue memory (necessary because one uses a pointer for the list of the moments).

2.4.2.22 Modal management of calculation in the course of non-linear calculation – NL_DS_PostTimeStep, NL_DS_Spectral, NL_DS_Stability and NL_DS_SpectralResults

These structures of data are used to manage the keywords MODE_VIBR and CRIT_STAB.

Structure of data to save Lbe parameters modal analysis: NL_DS_Spectral.

Type	Name	Description
character (len=16)	option	Option of calculation
character (len=16)	type_matr_rigi	Type of matrix of rigidity
aster_logical	l_small	Is worth .true. if one calculates the smallest eigenvalues
aster_logical	l_strip	Is worth .true. if one calculates the eigenvalues in a band
real (kind=8)	strip_bounds (2)	Waveband if l_strip
integer	nb_eigen	Many eigenvalues if l_small
integer	coef_dim_espace	Parameter for the modal solvor
type (NL_DS_SelectList)	selector	Selector of the moments of calculation
character (len=16)	level	Level of calculation: by band, smaller values or mode calibration (only many eigenvalues)

Structure of data to save Lbe parameters analysis of stability (CRIT_STAB): NL_DS_Stability.

Type	Name	Description

aster_logical	l_geom_matr	Is worth .true. to take into account geometrical rigidity
aster_logical	l_modi_rigi	Is worth .true. to take into account the modification of rigidity
integer	nb_dof_excl	Many excluded DDL
character (len=8), to point	list_dof_excl	List of the excluded DDL
integer	nb_dof_stab	Many DDL of stabilization
character (len=8), to point	list_dof_stab	List of the DDL of stabilization
character (len=16)	instab_sign	Value of the sign of instability
real (kind=8)	instab_prec	Absolute value of instability

Structure of data to save Lbe parameters analysis modal (MODE_VIBR and CRIT_STAB): NL_DS_PostTimeStep.

Type	Name	Description
aster_logical	l_crit_stab	If CRIT_STAB is well informed
aster_logical	l_mode_vibr	If MODE_VIBR is well informed
type (NL_DS_Spectral)	crit_stab	Parameters generals of CRIT_STAB
type (NL_DS_Spectral)	mode_vibr	Parameters generals of MODE_VIBR
type (NL_DS_Stability)	will stab_para	Specific parameters of CRIT_STAB
type (NL_DS_TableIO)	table_io	Structure of data table_container ANALYSE_MODALE
aster_logical	l_hpp	Is worth .true. if HP

2.4.2.23 Management DES tables container – NL_DS_TableIO

This structure of data allows to manage them table_container attached to STAT_NON_LINE and DYNA_NON_LINE.

Type	Name	Description
character (len=8)	result	Name of the structure of data results
character (len=19)	table_name	Name JEVEUX of table_container
character (len=24)	table_type	Type of table_container
integer	will nb_para	Full number of parameters in table_container (many columns)
integeR	nb_para_inte	Many parameters of the whole type
integer	nb_para_real	Many parameters of the type reality
integer	nb_para_cplx	Many parameters of the type complex
integer	nb_para_strg	Many parameters of the type chain
character (len=24), to point	will list_para	List of the names of the parameters
character (len=8), to point	will type_para	List of the types of the parameters

The routine nonlinDSTableIOCreate allows to create the structure of data and table_container in the structure of data evol_noli.
The routine will nonlinDSTableIOSetPara allows to give the list of the parameters in the table. One can give them in two manners:

- maybe starting from the structure of data NL_DS_Table (see § 2.4.2.2);
- maybe starting from an exhaustive list of these parameters.

The routine nonlinDSTableIOClean désalloue objects (necessary because of the use of pointers).

2.4.2.24 Management of the behavior CARCRI and COMPOR

The structure of data CARCRI is one sd_carte who contains realities. It currently stores 22 parameters of which here the list.

Index	Contents
1	Keyword COMPORTEMENT/ITER_INTE_MAXI
2	Keyword COMPORTEMENT/TYPE_MATR_TANG
3	Keyword COMPORTEMENT/RESI_INTE_RELA
4	Keyword COMPORTEMENT/PARM_THETA
5	Keyword COMPORTEMENT/ITER_INTE_PAS
6	Keyword COMPORTEMENT/ALGORITHME_INTE
7	Keyword COMPORTEMENT/VALE_PERT_RELA
8	Keyword COMPORTEMENT/RESI_CPLAN_MAXI
9	Keyword COMPORTEMENT/ITER_CPLAN_MAXI
10	Keyword COMPORTEMENT/RESI_RADI_RELA
11	Entirety coded for the presence of variables of orders of the type ExternalStateVariable
12	Keyword SCHEMA_THM/PARM_THETA
13	Keyword COMPORTEMENT/POST_ITER
14	Pointer towards the number of internal variables StateVariable for MFront
15	Pointer towards the names of the internal variables StateVariable for MFront
16	Pointer towards the law of MFront behavior
17	Indicator of symmetry of the tangent matrix of behavior
18	Keyword SCHEMA_THM / PARM_ALPHA
19	Pointer towards the names of the parameters materials for MFront
20	Pointer towards the names of the parameters materials for MFront
21	Keyword COMPORTEMENT/POST_INCR
22	Kinematic type of model for MFront

The structure of data COMPOR is one sd_carte who contains character strings. It currently stores 20 parameters of which here the list.
The index is given in the file Behaviour_type.h.

Name of Indice	Contents
RELA_NAME	Keyword BEHAVIOR RELATION
NVAR	Many internal variables
DEFO	Keyword BEHAVIORDEFORMATION

INCRELAS	Indicator of incremental or total model
PLANESTRESS	Model in plane constraints (Deborst or analytical)
Digital	Index of call of the law of behavior (lcxxxx)
MULTCOMP	Name of the structure of data resulting from DEFI_COMPOR
POSTITER	Contents of the m ot-key BEHAVIOR POST_ITER
KIT1_NAME THMC_NAME CREEP_NAME CABLE_NAME	Name of the first relation for a kit
KIT2_NAME THER_NAME PLAS_NAME SHEATH_NAME	Name of the second relation for a kit
KIT3_NAME COUPL_NAME HYDR_NAME	Name of the third relation for a kit
KIT4_NAME CPLA_NAME MECA_NAME	Name of fourth relation for a kit
KIT1_NUME THMC_NUME PLAS_NUME	Index of call of the first relation for a kit
KIT2_NUME THER_NUME CREEP_NUME	Index of call of the second relation for a kit
KIT3_NUME HYDR_NUME	Index of call of the third relation for a kit
KIT4_NUME MECA_NUME	Index of call of the fourth relation for a kit
KIT1_NVAR THMC_NVAR CREEP_NVAR	Many internal variables of the first relation for a kit
KIT2_NVAR THER_NVAR PLAS_NVAR	Many internal variables of the second relation for a kit
KIT3_NVAR HYDR_NVAR	Many internal variables of the third relation for a kit
KIT4_NVAR MECA_NVAR	Many internal variables fourth relation for a kit

3 Management of the algorithm

We will be interested in management of the algorithm. To manage convergence, the various levels of loop and the errors which have occurred during a calculation, one uses a common formalism, based on the concept **of event**.

3.1 Various loops

In the algorithm, there are five levels of loops <BOUC> :

- <RESI> : the loop on the various residues of balance requested by the user (RESI_GLOB_RELA , RESI_GLOB_MAXI , etc);
- <NEWT> : the loop on the iterations of Newton;
- <FIXE> : the loop on the fixed points (contact);
- <INST> : the loop over the moments of calculation;
- <CALC> : the loop on calculation.

The "loop" <CALC> is not really one. It is used to manage two situations:

1. Calculation finishes normally (not error) but otherwise than when one reached the end of the loop on the steps of time. For the moment, the only case is that where piloting reached its terminals;
2. An event is started outside the loop on the steps of time, i.e. during postprocessing (detection of instability by oscillatory modes);
   The various loops are distributed enters op0070, nmnewt (statics and implicit dynamics) and ndexpl (explicit dynamics). For the loop on the residues of balance, it is necessary to look in the routine nmcore, called by nmconv. For the loop on the fixed points, one uses the routines nmible/nmtble, contained in nmnewt.

3.2 State of the loops

The state of a loop is stored in the object SDERRO (§2.4.2.6), one reaches it by nmleeb and nmeceb. There exist six states:

1. CONT : the continuous loop;
2. CONV : the loop has converged (events of type convergence and divergence);
3. WANDER : an error (event of type error) occurred during the loop, one will treat it;
4. EVEN : an event (event of the informative type) occurred during the loop;
5. STOP : an error (event of type error) occurred during the loop, one stops;
6. CTCD : particular state of the loop of Newton for the discrete contact;

3.3 Events

One can classify the events according to their origin:

- Events defined by the user in DEFI_LIST_INST (for example, DELTA_GRANDEUR);
- Intrinsic events with the algorithm: errors and the convergence of the various loops;

3.3.1 Types of the events

An event is defined by its type which is a character string in two parts . The first channel <XXXX> described the type of the event:

- The events causing an error are prefixed by <ERRC_*> or <ERRI_*> ;
- The events causing a convergence are prefixed by <CONV_*> ;
- The events causing a divergence are prefixed by <DIVE_*> ;
- The informative events are prefixed by <EVEN_*> ;

The second chain <YYYY> described the level of loop of the event:

- Buckle on the residues of balance <*_RESI> ;
- Buckle on the iterations of Newton <*_NEWT> ;
- Buckle of fixed point for the contact <*_FIXE> ;
- Buckle on the steps of time <*_INST> ;

This information describes in which loop the events can potentially start.

3.3.1.1 Events of type error <ERR*_*>

Errors are to be immediately treated (i.e. as of release of the event) because they are blocking for the process, they are noted <ERRI>. The errors to be treated only with convergence of the loop are noted <ERRC>.
For example:

- <ERRI_NEWT is an error to be treated immediately in an iteration of Newton as a defect of integration of the law of behavior prevents or a matrix not factorisable;
- <ERRC_NEWT is an error to be treated with convergence of Newton, for example an exit of a physical criterion out of terminals of its field of definition;
- <ERRI_FIXE is an error to be treated immediately in a loop of fixed point;

Note:

Typified errors <ERRI_CALC> cause the dead halt of calculation because no action can treat them. They are the stops in lack of time CPU and the external stop by user.

3.3.1.2 Events of type convergence <CONV_*

The events of type convergence are treated with each level of loop. This convergence perhaps related to a functionality activated or not (cf §2.4.1.1).

3.3.1.3 Events of type divergence <DIVE_*>

The events of type divergence are treated with each level of loop. This divergence perhaps related to a functionality activated or not (cf §2.4.1.1).
In practical, the state of each loop is rather treated in divergence. Indeed, to avoid having to check that a specific functionality is active (piloting, Deborst, contact, etc) and thus to test the convergence of a loop by checking this functionality, one prefers to emit an event of divergence that an event of convergence: one can have divergence only if the functionality is activated, whereas one could have convergence even if the functionality is not activated.

3.3.1.4 Events of the informative type <EVEN

Events of the informative type are the other events (neither error, nor convergence). By defaults, they are only used to often give indications to the user (in the form of messages of alarms, the passage of RESI_GLOB_RELA with RESI_GLOB_MAXI for example). Some are activables only on request of the user (order DEFI_LIST_INST).
These events could as be treated explicitly (otherwise as by the emission of an alarm or information) via DEFI_LIST_INST.
These events are not related to a level of loop.

3.3.1.5 The case of the code-return

A code-return is a single entirety giving the statute of an operation. It is possible to bind an event to the value of a code-return. Each value of the code-return can generate a different event, of any type. The codes return -1 and 0 always have the same direction:

- -1: nothing was done (not factorization, not integration of law of behavior, etc);
- 0: one did something and very did well;

The other values of codes return have a direction dependent on their type:

Type	Description	Significance of the code	
FAC	Return of factorization	-1	Pas de factorization
		0	Very happened well
		1	The matrix is singular
		2	Factorization failed

		3	One cannot say if the matrix is singular
PIL	Return of piloting	-1	Pas de piloting
		0	Very happened well
		1	Pas de solution of the equation of piloting
		2	One reached a terminal (end of calculation)
LDC	Return of the integration of the law of behavior	-1	Pas d' integration of the behavior
		0	Very happened well
		1	Failure during the integration of the law of behavior
		2	A physical parameter is not in its field of definition
		3	The constraint SIZZ is not worthless (algorithm of Borst)
CTC	Return of the treatment of the discrete contact	-1	Pas de discrete contact
		0	Very happened well
		1	The maximum number of iterations of contact was reached
		2	The matrix of the contact is singular

3.4 Management of the events

3.4.1 Modification, addition or suppression of an event

Most standardized events are envisaged in the structure of data and the utilities which are dependent there. In most case, it is a question of modifying only the routine nmcrga (see §2.4.2.6 by adding to it the characteristic of the event in DATED at the beginning of routine.

Description	**DATED**
Name of the event	NEVEN
Name of the code-return related to the event (XXX if not code-return)	NCRET
Value of the code-return related to the event (99 if not code-return)	VCRET
Type and level of release of the event	TEVEN
Functionality activating an event of type convergence or divergence	FEVEN
Code of the message to be displayed when the event starts	MEVEN

The verification system of the messages makes that the code of the message to be displayed (MEVEN) is not sufficient, it is also necessary to impact nmevim.

3.4.2 Emissions of the events

To start an event the developer must envisage it in the algorithm by calling the routine nmcrel. For example:

Order	**Effect**
CAL NMCREL (SDERRO, 'ERRE_TIMN', .TRUE.)	Lack of time CPU during an iteration of Newton
CAL NMCREL (SDERRO, 'RESI_MAXR', .TRUE.)	Passage de RESI_GLOB_RELA with RESI_GLOB_MAXI

CAL NMCREL (SDERRO, 'DIVE_FIXG'. FALSE .)	Pas de divergence of the loop of point fixes on the geometry

If the event is of type <ERRI_BOUC>, the statute of the loop <GOAT> is automatically modified. It is a precaution to prevent that the developer forgets to treat the event (and thus risk to cause false results). An immediate error of level <ERRI> is reflected on all the levels of loop for the same reasons.
To activate an event related to a code-return, the routine should be used nmcret instead of nmcrel. For example CAL NMCRET (SDERRO, 'LDC', LDCCVG) deal with transforming into event the value of the code return of the law of behavior.

3.4.3 Treatment of the events

The principal idea is to treat the events with all the levels of loop, to change the state of the loops according to the output this.

3.4.3.1 Events of type convergence and divergence

The events of type convergence or divergence are examined on each level of loop, in a called dedicated routine nmevcv . This routine modifies the state of the loop:

1. It is supposed initially that the loop continues (state CONT, cf §3.2)
2. One buckles on the events of type <CONV_*> and <DIVE_*> by taking into account the possible ones features activated;
3. One calculates the state result for this level of buckle: if all events of divergence are forgery and all events of convergence are truths, then this level of loop is in the converged state;
4. One checks the states of convergence of the inner loops on this level. So that the final state of this loop is converged, all the inner loops must be converged. If it is the case, the loop passes in the converged state (state CONV, cf §3.2)
5. One modifies the state of the current loop by calling the routine nmeceb ;
6. The routine nmeceb the state of the loop transmits to the higher loops (avoids the not-treated errors);

Buckle	Code	Routine	Appealing routine
Residues	RESI	nmcvgr	nmconv
Newton	NEWT	nmcvgn	nmnewt
Not fixes	FIXED	nmcvgf	nmtble
Pas de time	INST	nmcvgp	op0070
Calculation	CALC	nmcvgc	op0070

3.4.3.2 Events of the errors type

The events of type error are particularly sensitive and must be treated in a very rigorous way for to avoid the results forgery. Initially, the event of standard error was emitted by the means of the routines standards: nmcrel and nmcret. It is pointed out that in the case of the events error, one systematically modifies the state of the loop in mode WANDER in optics to avoid continuing these loops if ever the error were not treated suitably by the algorithm (lapse of memory of the developer).
In a systematic way, the events of type error are tested in the algorithm via the routine nmltev. If an event of type error is activated, one GOTO is immediately made in the program.

4 Algorithm general

The process is standardized to the maximum, for each level of loop:
- Evaluation of convergence by call to the routines nmcvg* ;
- Action following the situation of the loop by call to the routines nmact* ;

4.1 Total readings and initializations

The routine op0070 manage the total algorithm and cuts out in three parts:
- Reading and initializations of the data;
- Buckle on the steps of time;
- Postprocessings;
- Management of the total errors;
- Filing;

For the reading and the initialization of the data, one will refer in particular to the relative information with the management of the SD (§2). The whole of the initializations made on this level will be valid for all calculation. The principal routine of initialization is nminit. Here operations envisaged in this routine:

Operations of initializations in nminit	**Routine**
Creation of the profile of the matrix and of SDNUME	nmnume
Creation of the variable-hats	nmchap
Creation of the vector of the functionality activated list_func_acti	nmfonc
Creation of the vectors in the variable-hats	nmcrch
Creation of the structure of data piloting	nmdopi
Duplication NUME_DDL for SDNUME	nmpro2
Preparation of the map COMPOR (transformation into CHAM_ELEM_S)	nmdoco
Creation of SDDISC and SDOBSE	diinit
Initialization of the variables of order	nmcrvv
Precalculation of MATR_ELEM constant during calculation	nminmc
Precalculation of VECT_ELEM and VECT_ASSE constant during calculation	nminvc
Creation of SDCRIT	nmcrcv
Initialization of calculation by under-structuring	nmlssv
Creation of the SD for the loading FORCE _SOL	nmexso
Calculation of initial acceleration	accel0
Creation SDCONV	nmcrcg
Initialization of NL_DS_Print	nminim
Precalculation of MATR_ASSE constant during calculation	nminma
Realization of an initial observation	nmobsv
Creation of the SD RESULT (EVOL_NOLI)	nmnoli
Creation of the table of the sizes (for ERRE_THM)	cetule
Calculation of the second members initial for the multi-step diagrams in dynamics	nmihht

The algorithm general of op0070 is summarized on the flow chart (Figure 1).

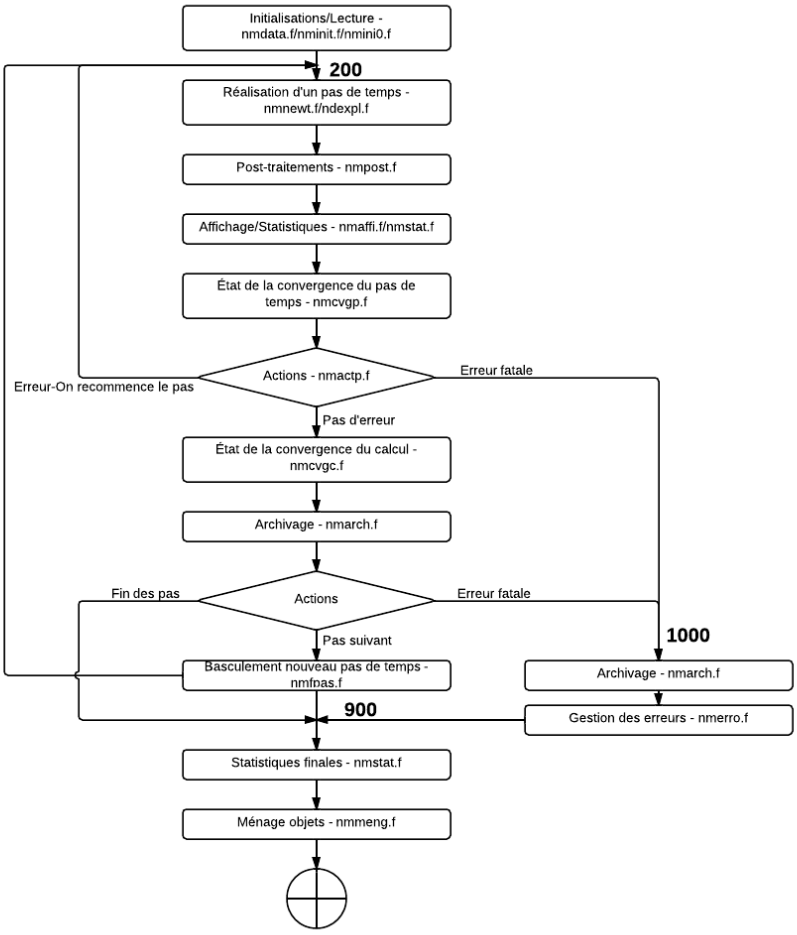

Figure 1: Top-level flowchart of `op0070`

4.2 Realization of a step of time

A step of time uses several overlapping levels of loops:

- Loops of fixed point, used exclusively for the contact (up to three levels of loop);
- A loop on the iterations of Newton. A process of Newton contains one prediction of Euler and the corrections of Newton;
  The algorithm general of `nmnewt` is summarized on the flow chart (Figure 2).

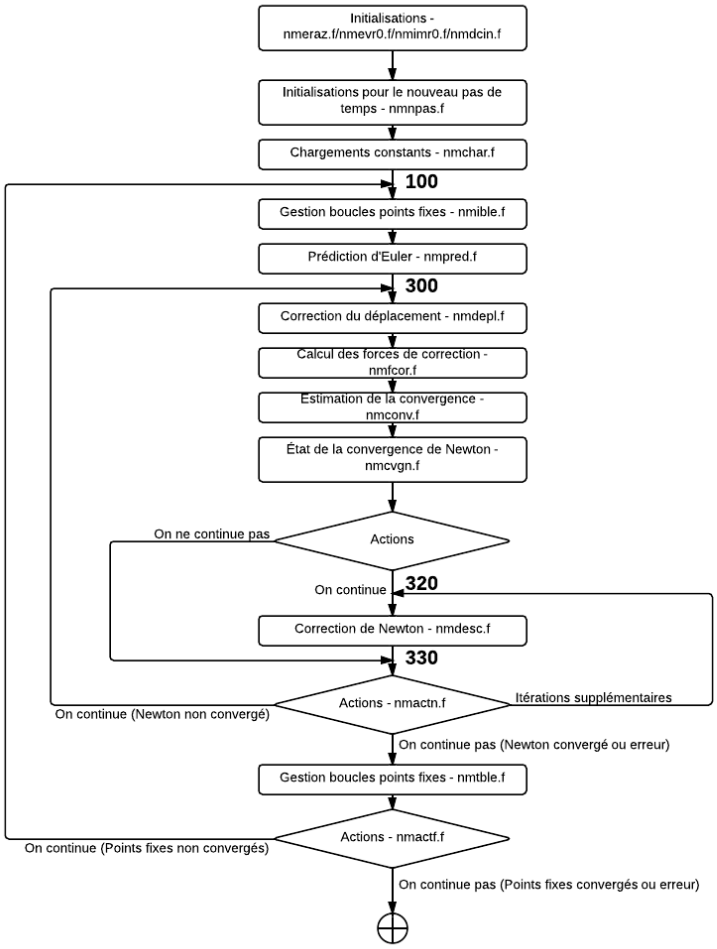

Figure 2: Top-level flowchart of `nmnewt`

4.2.1 Initializations of the step

Each step of time is initialized by a series of routines:

- Routines `nmeraz`, `nmevr0` deal with initializing the management of the events;
- The routine `nmdcin` check that the cutting of the step of time is correct (control amongst level of cutting of a step of time to the other);

The most important routine is the routine `nmnpas` . It carries out the following operations:

- Routines `nmimr0` and `nminin` initialize posting, in particular the table of convergence (what makes it possible to vary the nature of this one of a step of time to the other);
- The routine `nmvcre` prepare the variables of order for the step of current time;
- One initializes to zero the increment of displacement cumulated since the beginning of the step of time (vector `DEPDEL` , to see § 2.4.1.6). Attention initialization is special because of the taking into account of great rotations;
- One initializes all the data for dynamics, in particular the various coefficients (see § 2.4.2.17) in `ndnpas` ;
- One initializes various information for Newton-Krylov and the contact (routines `cfinit` , `mmapin` and `nmnkft`);

4.2.2 Prediction of Euler

The prediction of Euler is an estimate of the increment of displacement by linearizing the problem compared to time. There exist several methods and several types of matrix usable. Calculation itself is carried out in the routine nmpred (and its - pretty girls), by means of the principles established in the §5.

4.2.3 Update of the fields

The update of the fields is carried out in the routine nmdepl. This update consists in modifying the vectors solutions (displacements, speeds and accelerations), while taking into account possibly:

- Linear research without piloting;
- The search for η for piloting with or without linear research;
- The contact (discrete formulation) or unilateral connections;

Here phases:

1. Recalculation of the external efforts, routine nmfext ;
2. Conversion of the increments of solution DEPSO1 and DEPSO2 (see §2.4.1.6), resulting from the resolution of the linear system, towards the increments in displacement/speed/acceleration (taken into account of the coefficients of change of diagram), via the routine nmincr ;
3. Calculation searches linear and piloting, routines nmreli, nmpich and nmrepl ;
4. Update of the direction of descent (taken into account of the coefficients resulting from linear research and piloting), routine nmpild ;
5. Modification of displacements by the contact and the unilateral connections, by the routine nmcoun ;
6. Actualization of the fields of solutions in nmmajc ;

The phase 4.2.3 consist in updating the fields “more” (DISPLEASED, VITPLU, ACCPLU) and cumulated fields (DEPDEL, VITDEL, ACCDEL), by taking of account possible great rotations (put up to date of the quaternions) and of the damage to the nodes.

4.2.4 Forces of correction

Since displacements were modified (see § 4.2.3), it is necessary to revalue the variable forces: either forces external of type “following”, or interior forces and reactions of contact/friction, or forces related to dynamics (inertia, damping). The unit is carried out in the routine nmfcor.

4.2.5 Estimate of convergence

The estimate of the convergence of Newton is one moment critical, which will condition the accuracy of calculation. The whole of the estimate is carried out in the routine nmconv . This routine deals with the calculation of the residues (nmresi), estimate of their convergence (nmcore). It is thus in this routine that one will treat the loop <RESI> on the residues of balance. But one must also take into account the iteration count of Newton, the case of the convergence of the contact (formulations discrete or continues), of the method of Borst or method IMPLEX.

Note:

It is advisable to be very careful in the modification of this routine.

4.2.6 Correction of Newton

If the process did not converge, one carries out the calculation of a correction of Newton. One thus carries out the calculation of a linear system (see §5) in the routine nmdesc (like direction of ***desc***graft).

4.2.7 Buckle on the fixed points

There exist three loops known as “of fixed point” between the loop on the steps of time and the loop on the iterations of Newton. These loops are used for the contact:

1. Buckle of fixed point on the geometry;
2. Buckle of fixed point on the thresholds of friction;
3. Buckle of fixed point on the statutes of contact;

These three loops are managed by the duet of routine nmible/nmtble, via the passage of the variable LEVEL, which indicates in which type of loop of point one fixes is currently.

5 Construction and resolution of the systems

An important part of the non-linear algorithm consists in solving linear systems, which one builds in four times:
- Calculation and assembly of the loadings;
- Calculation and assembly of the second members;
- Calculation and assembly of the matrix;
- Resolution of the linear system;

In this part, we will describe the various phases and the principal routines to be considered.

5.1 Systems to be solved

Currently, there exist several different systems which are solved in the operator op0070. They will always have the following form (except for postprocessing, which uses the operators of search for eigenvalues):

$$\begin{bmatrix} K & B^T \\ B & 0 \end{bmatrix} \cdot \begin{Bmatrix} \delta r \\ \delta s \end{Bmatrix} = \begin{Bmatrix} L_r \\ L_s \end{Bmatrix} \quad (1)$$

$[K]$ is the matrix of rigidity or a linear combination of matrices and $[B]$ is the matrix of Lagrange de Dirichlet. $\{\delta r\}$ is the increment of solution of the unknown nodal values of the system and $\{\delta s\}$ is the increment of the parameters of Lagrange associated with the dualisation with the limiting conditions. Concretely, the “unknown nodal values” of the systems can be displacements, speeds, accelerations, pressures, temperatures, etc the two second members are also nodal values. The detail of the systems is given in the reference materials [R5.03.01] and [R5.05.05].

5.2 The routine merimo

For the calculation of the tangent matrices (options FULL_MECA, FULL_MECA_ELAS, RIGI_MECA_TANG, RIGI_MECA_ELAS, RIGI_MECA, RIGI_MECA_IMPLEX) and interior efforts (option RAPH_MECA), one passes systematically by the routine merimo. This routine prepares the inlet limits (many in this case), it is called at the same time for the calculation of the elementary vectors for the interior efforts (see §5.5.2) but also for the calculation of the tangent elementary matrices of rigidity (see §5.3).

5.3 Calculation ofS matrices

We consider elementary matrices here (MEELEM) and of the assembled matrices (MEASSE). Just like for the loading (§5.5.1) , this calculation uses a system in two times:
- Creation of one **local list** matrices to be calculated or assemble;
- Calculation or assembly of the matrices;

For the phase of creation of the list of the matrices to be calculated or assemble, there are several routines (contrary to the loadings which use only the routine nmchar). These routines are the following ones:

Operations	Routine
Construction of the local list for the matrices used in explicit dynamics	ndxprm
Construction of the local list for the matrices used in correction	nmcoma
Construction of the local list for the matrices used in prediction	nmprma
Construction of the local list for the matrices used in postprocessing (oscillatory modes or modes of buckling)	nmflma
Construction of the local list for the constant matrices during all calculation	nminmc

For the phase of calculation or assembly of the matrices, one finds routines similar to the case of the loadings.

Operations	Routine
Addition of a matrix to be assembled and/or calculate in the local list	nmcmat

Initialization of the local list	nmcmat
Shunting calculation and/or assembly of the matrices	nmxmat
Calculation of the matrices	nmcalm
Assembly of the matrices	nmassm
Calculation and assembly of the matrices of rigidity	nmrigi

There is nevertheless a notable exception: the calculation of the non-linear tangent matrices uses a direct call by nmrigi because that requires additional parameters (call to merimo, to see 5.2) which the other matrices do not need. The type of matrix MERIGI thus use nmrigi instead of nmcalm and nmassm.
If one wishes to modify a matrix, it is thus necessary:

- To add its calculation or its assembly to the good place (ndxprm, nmcoma, nmprma, nmflma or nminmc) according to the case, to even add a new routine;
- To add elementary calculation or the assembly in nmcalm and nmassm ;
  Routines nmxmat and nmcmat of handling of the local list **do not have to be modified**.

5.4 Calculation of the resulting matrix MATASS

According to the type of resulting matrix that one wishes to obtain (in prediction, correction or for initial acceleration), one uses three total routines which will deal with this construction. They are the routines nmprac , nmcoma and nmprma. These three routines also manage posting (option of calculation of the matrix of rigidity) and the statistics (time spent in the operations). They have the same general diagram:

- Calculation or assembly of elementary matrices (MEELEM) in assembled matrices (MEASSE), (see § 5.3) ;
- Management of the parameters of reactualization of the matrices (rigidity, mass, damping), routine nmchrm ;
- Management of the type of the matrix of rigidity (name of the option), routine nmchoi ;
- Construction of the matrix resulting by linear combination from the other matrices. In dynamics, the various contributions in the resulting matrix (matrices of mass, damping and rigidity) combine by using a multiplying coefficient depend on the diagram in time used and step of time, one recovers these coefficients via the routine of access NDYNRE (see § 2.4.2.17). It is the routine nmmatr ;
- Factorization (or not) of the resulting matrix by call to the routine preres ;

Operations	**Routine**
Calculation of the matrix assembled resulting for calculation from initial acceleration	nmprac
Calculation of the matrix assembled resulting for the phase from prediction	nmprma
Calculation of the matrix assembled resulting for the phase from correction	nmcoma
Management of the parameters of reactualization of the matrices (rigidity, mass, damping)	nmchrm
Management of the type of the matrix of rigidity (name of the option)	nmchoi
Choice of re-creation of classification	nmrenu
Calculation of the resulting matrix MATASS	nmmatr
Taking into account of the following loadings with their multiplying function	ascoma
Taking into account of the modification of the resulting matrix by the contact/friction (method DISCRETE)	nmasfr

5.5 Calculation of the second member

The calculation of the second member is the part most different from one linear system to another, it will contain:

- Loadings given (see the § 5.5.1)

- Internal forces coming from the integration of the behavior (§5.5.2);
- Quantities related to the limiting conditions of Dirichlet like the reactions of support (§5.5.3);
- Quantities related to the variables of order (§5.5.4);
- Various quantities related to the contact/friction (not of details in this document);
- Contributions of inertia and damping for dynamics (§5.5.6);

In the case of dynamics, for the multi-step diagrams (complete Newmark with MODI_EQUI=' OUI') one moreover will add the contributions of the steps of previous times.

5.5.1 Calculation of the loadings

There are three modes of evaluation of the loadings and two phases. The modes are the following:

- Mode <ACCI> loadings for the evaluation of initial acceleration in dynamics;
- Mode <FIXE> loadings for the evaluation of the fixed loadings not depending on the solution;
- Mode <VARI> loadings for the evaluation of the variable loadings depending on the solution (following loadings);

There are also two phases. Either one is in correction, or one is in prediction. These phases are useful only for very particular cases: modal damping, method IMPLEX and modal impedance.
The principal routine which calculates the loadings is nmchar. In this routine, according to the activated features (list_func_acti) and according to the mode/phase of calculation, one will cause the calculation of the elementary vectors (variable-hat VEELEM) and their assembly in vectors assembled (variable-hat VEASSE). For that, one uses a certain number of utility routines which manage local lists with nmchar.

Operations	Routine
Addition of a loading to be assembled or calculate, with a possible option Initialization of the list of the loadings (initializations local lists of nmchar)	nmcvec nmcvec
Shunting calculation or assembly of the loadings	nmxvec
Calculation of the loadings	nmcalv
Assembly of the loadings	nmassv

Certain loadings do not have an elementary phase of calculation but only the creation of a directly assembled vector. If one wishes to add a loading, it is thus necessary:

- To define which phase and which mode will activate it;
- To add its calculation or its assembly in nmchar according to the activated functionality;
- To add elementary calculation and/or the assembly in nmcalv and nmassv ;

Routines nmxvec, nmcvec **do not have to be modified**.

5.5.2 Calculation of the quantities related to the interior efforts

The calculation of the interior efforts can be carried out in two manners:

- By integration of the law of behavior, it is the option of calculation RAPH_MECA ;
- By re-use of the constraints calculated in addition, it is the option of calculation FORC_NODA ;

This distinction is important because it does not generate the same costs (the second case is much faster). The integration of the behavior is an expensive operation which implies to modify variable internal and forced, and, often, to solve locally (at each point of Gauss), a non-linear system. As the calculation of the constraints is also necessary during the evaluation of the tangent matrices, the behavior is also integrated during the calculation of the tangent matrices (option FULL_MECA). In statics, the phase of prediction calculates the interior efforts by the option FORC_NODA. In dynamics, the behavior systematically is integrated.

Operations	Routine
Calculation of the elementary vectors for the interior efforts (integration of the behavior)	nmfint
Assembly of the elementary vectors for the interior efforts (integration of the behavior)	nmaint

The calculation of the option FORC_NODA is made by the mechanism of evaluation of the loadings (§5.5.1).

5.5.3 Calculation of the quantities related to the dualized limiting conditions

The dualisation of the limiting conditions of Dirichlet (see [R3.01.01]) involves the calculation of two quantities of type vector assembled:

- Reactions of support, by the product of the matrix of the limiting conditions dualized with Lagrange corresponding to the degrees of freedom $[B].[\lambda]$, it is the vector identified by CNDIRI ;
- The value of the degrees of freedom imposed by dualisation $[B].[u]$. With convergence, this quantity will be equal to the limiting conditions given $[u^d]$, the vector is identified by CNBUDI ;

Operations	Routine
Calculation of the elementary vectors for the reactions of support	nmdiri
Assembly of the elementary vectors for the reactions of support	to nmadir
Calculation and assembly of the quantities imposed by dualisation	nmbudi

5.5.4 Calculation of the quantities related to the variables of order

The calculation of these quantities fact by the mechanism of evaluation of the loadings (§5.5.1), type CNVCF0, CNVCF1 and CNVCPR.

5.5.5 Calculation of the constant quantities during calculation

A certain number of vectors are constant during calculation, they then use a mechanism similar to the loadings (§5.5.1), using the routines nmxvec, nmcvec, nmcalv and nmassv, except that one passes by the routine nminvc instead of nmchar.

Operations	Routine
Calculation and assembly of the constant vectors during calculation	nminvc

5.5.6 Calculation of the quantities related to dynamics – Inertial forces and of damping

The calculation of these quantities fact by the mechanism of evaluation of the loadings (§ 5.5.1), type CNDYNA. These quantities do not exist in an elementary state since they are the result of a product stamps/vector.

Operations	Routine
Calculation of the dynamic quantities for the second member	ndfdyn
Calculation of the inertial forces	nminer
Calculation of the forces of damping	nmhyst

According to the diagrams in time, it is necessary to use specific multiplying coefficients in front of each one of these quantities, one recovers these coefficients via the routine of access NDYNRE (see § 2.4.2.17).

5.5.7 Second resulting members

It remains about it more than to build the various second members by linear combination of all the quantities detailed previously. The principal routines of construction of these second members are with the number of seven:

Operations	Routine
Calculation of the second member for the correction	nmassc
Calculation of the second member for the prediction – Option DEPL_CALCULE or EXTRAPOLATE	nmassd
Calculation of the second member for initial acceleration	nmassi
Calculation of the second member for the prediction	nmassp
Calculation of the second member for the prediction – Case static	nsassp
Calculation of the second member for the prediction – dynamic Case implicit	ndassp
Calculation of the second member for the prediction – dynamic Case explicit	nmassx

All these routines rest on the same principles:

- Recovery of the quantities necessary, already assembled or calculated locally in these routines, via the variable-hat VEASSE (see the §5.5.1 with §5.5.6);
- Recovery of the various multiplying coefficients. In dynamics, according to the diagrams in time, it is necessary to use specific multiplying coefficients in front of each one of these quantities, one recovers these coefficients via the routine of access NDYNRE (see § 2.4.2.17);
- Construction of one or two second members by linear combination (use of the routines standards vtaxpy). There are two second members in the case of piloting;

To improve legibility, one uses utility routines standardized to gather the most frequent cases.

Operations	Routine
Calculation of the components of the vector second member for the loadings of the Dirichlet type, fixed during the step of time, for the normal case and the controlled case	nmasdi
Calculation of the components of the vector second member for the loadings of the Neumann type, fixed during the step of time, for the normal case and the controlled case	nmasfi
Calculation of the components of the vector second member for the loadings of the Neumann type, variables (follower) during the step of time, for the normal case (not of controlled case)	nmasva
Calculation of the components of the vector second member for the specific loadings of dynamics, variables (follower) during the step of time, for the normal case (not of controlled case)	ndasva
Calculation of the components of the vector second member for the specific loadings of dynamics, calculation of initial acceleration, variables (follower) during the step of time, for the normal case (not of controlled case)	nmacva

5.6 Resolution of the system

The resolution is done via the intermediary of the routine nmresd, which takes as starter the resulting matrix assembled (see § 5.4) and second suitable members (two second members if there is piloting).

Operations	Routine
Resolution of the system	nmresd
Resolution of the system on the physical degrees of freedom (call to resoud)	nmreso
Resolution of the system on the generalized degrees of freedom	nmresg

The routine of resolution will thus turn over one or two fields solutions DEPSO1 and DEPSO2, stored in the variable-hat SOLALG.