Implementation of STAT_NON_LINE and DYNA_NON_LINE

Summarized:

One describes here implementation the data-processing of the algorithm of resolution of the nonlinear quasi-static problems and dynamics nonlinear. The documents describing in detail these algorithms are supposed to be known ([R5.03.01] and [R5.05.05]), one will recall only the main steps 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 pinned ends of operator STAT_NON_LINE and DYNA_NON_LINE of *Code_Aster*, his shaft of call, a description of the data-processing objects and principal routines, and some traps to during avoid the development in this operator.

Summary

1

the operator op0070.f consists of several parts: Reading
1) of the data; Initialization
2) of the data; Construction
3) and resolution of a linear system Setting
4) - with-day of the fields; Archivage
5) of the results, postprocessings;

The whole being encapsulated in three levels of loop: time step, 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 with dimensions one structuring SD (§ 2) 6 events (§ 3.3 38 and on the general algorithm, buckles by loop (§ 4 42 It should be noted that

to help with the débugage, it is possible med file to save the fields of displacements in one, with each iteration of Newton. For that, it is necessary to overload the routine dbgcha.f `by changing` the Boolean DBG=. `FALSE. in` DBG=. `FALSE. The fields` of displacements will be then saved in a file with med format, on the logical unit 80 (not to forget to add a file on this output unit in astk) Data structures

2 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. The types

2.1 of SD four

levels of SD are introduced: Concepts
1.coming from other commands or products by the command itself; The SD
2.“low-level” are variables FORTRAN (of the tables of the simple type), without routine of access; The SD
3.“layer-level” are variables FORTRAN (of the tables of the simple type) having routines of access. In this category, one will find the vector of activated features FONACT and the variable-hats which are tables of character strings containing of the more complex objects (matrixes, vectors, etc); The SD
4.“high-level” are JEVEUX objects specific with the operator (with routines of access) or to the SD standards of Code_Aster who will remain interns with the operator; The SD
low-level, are, in general, a remainder of the old versions of op0070.f. Their total disappearance and their resorption with the profit of the SD of high-level remain to be made. NB:
In the description of the SD, column # is a sequence number making it possible to locate them. SD of

2.1.1 type concept the concepts

come from another operator or are built by the operator to be used in other commands. They are built exclusively on SD standards 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 .f: # Standard

	Description	Name generally	used 1 Mesh
1	sd_mailla	M AILLA	2 Models
2	sd_modele	MODELE	3 Result
3	sd_resultat	RESULT	4 Characteristics
4	elementary will sd_cara	_elem CARELE	6 Loadings
6	sd_l_	loads LISCHA	7 Material field
7	and command variables sd_cham	_mater SUBDUE	8 Definition
8	of the contacts sd_contact	DEFICO	9 Definition
9	of unilateral connections sd_contact	DEFICU	SD of

2.1.2 low-level These SD

are tables, their length is thus important. To avoid the errors (bad recopy lengths), they pre-are dimensioned in op0070 .f, via a PARAMETER , the size is then (*) in the routines called. # Description

	Standard	Name 11	Parameters
11	of resolution REAL (	ZPMET) PARMET	12 Convergence

			criteria
12	REAL (	ZPCRI) PARCRI	13 Information
13	on convergence of computation REAL (	ZCONV) CONV	14 Parameters
14	of the convergence criterion in stress REAL (	ZPCON) PARCON	15 Method
15	of resolution CHAR*	16 (ZNMETH) METHOD	SD of

2.1.3 mean level the SD

of mean level is 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

	Standard	**Name 10**	**Functionalities**
10	activated INTEGER	(100) FONACT	16 Variable
16	- hat – elementary Matrixes CHAR*	19 (ZMEELM) MEELEM	17 Variable
17	- hat – assembled Matrixes CHAR*	19 (ZMEASS) MEASSE	18 Variable
18	- hat – elementary Vectors CHAR*	19 (ZVEELM) VEELEM	19 Variable
19	- hat – assembled Vectors CHAR*	19 (ZVEASS) VEASSE	20 Variable
20	- hat – Fields solutions CHAR*	19 (ZSOLAL) SOLALG	21 Variable
21	- hat – Fields incremental solutions CHAR*	19 (ZVALIN) VALINC	SD of

2.1.4 high-level All

these SD is built on JEVEUX objects and their access is done via dedicated routines (encapsulation of the data) when the SD are specific. One does not reconsider the nine first, which are concepts contained in the keywords of the operator or products by the operator (§ 2.1.1 6 Standard

	Description	**Name 5**	**Card**
5	of behavior sd_carte	COMPOR	22 Solver
22	sd_solvor	SOLVEU	23 Stamps
23	preconditionning sd_matr	_asse MAPREC	24 Stamps
24	assembled resolution sd_matr	_asse MATASS	25 Card
25	of the convergence criteria for the behavior sd_carte	CARCRI	26 Card
26	of the command variables of reference sd_carte	COMREF	27 Card
27	of the code-returns error of the behavior sd_carte	CODERE	28 Classif

			ication
28	sd_nume	_ddl NUMDDL	29 Classification
29	fixes (used for the contact continuous method) sd_nume	_ddl NUMFIX	30 Management
30	of the printings specific	to op0070 .f SDIMPR	31 Management
31	of measurements of time specific	to op0070 .f SDTIME	32 Management
32	of the errors of the algorithm specific	to op0070 .f SDERRO	33 Management
33	of the archivage and the initial state (IN and OUT) specific	to op0070 .f SDIETO	34 Management
34	of the statistics specific	to op0070 .f SDSTAT	35 Ma

			nagement
35	of convergence specific	to op0070 .f SDCONV	36 Management
36	of the SUIVI_DDL specific	to op0070 .f SDSUIV	37 Management
37	of the quality standards specific	to op0070 .f SDCRIQ	38 Management
38	of control specific	to op0070 .f SDPILO	39 Management
39	of classifications specific	to op0070 .f SDNUME	40 Man

			agement
40	of the dynamics specific	to op0070 .f SDDYNA	41 Management
41	of the temporal discretization specific	to op0070 .f SDDISC	42 Management
42	of the convergence criteria specific	to op0070 .f SDCRIT	43 Management
43	of the OBSERVATION specific	to op0070 .f SDOBSE	44 Management
44	of postprocessing (MODE_VIBR and CRIT _ STAB) specific	to op0070 .f SDPOST	45 Enter

			gy management
45	specific	to op0070 .f SDENER	46 Resolution
46	of the contact specific	to op0070 .f RESOCO	47 Resolution
47	of LIAISON_UNIL specific	to op0070 .f RESOCU	the rest

of the document will primarily describe the contents, the use and the access to the SD specific to op 0070 .f. Creation

2.2 of the SD the following

table summarizes the origin of the creation of the SD. # Description

	Standard	Creation	1 Mesh
1	sd_mailla	comes	from the .comm 2 Models
2	sd_modele	comes	from the .comm 3 Result

3	sd_resultat	nmnoli.f	4 Characteristics
4	elementary sd_cara_	elem comes from	the .comm 5 Card
5	behavior sd_carte	nmdocc.f 6	Loadings
6	sd_l_charges	nmdoch.f 7	coded Material
7	sd_cham_mater	comes from the .comm	8 Definition
8	the contacts sd_contact comes	from the .comm	9 Definition
9	unilateral connections sd_contact comes	from the .comm	10 Features
10	activated INTEGER (100)	op0070.f 11	Parameters
11	resolution REAL (ZPMET) op	0070.f 12	Convergence criteria
12	REAL (ZPCRI) op	0070.f 13	Information
13	convergence of computation REAL (ZCONV) op	0070.f 14	Parameters
14	the convergence criterion in stress REAL (ZPCON) op	0070.f 15 Method	of
15	CHAR*16 (ZNMETH	) op0070.f 16	- hat Variable
16	– elementary Matrixes CHAR*19 (ZMEELM	) op0070.f 17	- hat Variable
17	– assembled Matrixes CHAR*19 (ZMEASS	) op0070.f 18	- hat Variable
18	– elementary Vectors CHAR*19 (ZVEELM	) op0070.f 19	- hat Variable
19	– assembled Vectors CHAR*19 (ZVEASS	) op0070.f 20	- hat Variable
20	– Fields solutions CHAR*19 (ZSOLAL	) op0070.f 21	- hat Variable
21	– Fields incremental solutions CHAR*19 (ZVALIN	) op0070.f 22	sd_ Solver
22	nmlect.f	23 Matrix	of preconditionning
23	sd_matr_asse during	the algorithm 24	Stamps assembled
24	resolution sd_matr_asse during	algorithm 25	Card of the convergence criteria
25	for the behavior sd_carte nmdocr.f 26	Card of	the command variables
26	of reference sd_carte nmvcre.f 27 Card	of the codes	- returns

27	error of the behavior sd_carte during algorithm	28 Classification	sd_nume
28	d.o.f. nmprof.f 29 Classification		fixes (used
29	for the contact continuous method) sd_nume_ddl nmpro2.f 30 specific	Management of	
30	nminim.f 31 Management	of measurements	of specific
31	time nmcrti.f 32 Management	of the errors	of the specific
32	algorithm nmcrga.f 33 Management of	the archivage	and the initial
33	state (IN and OUT) specific nmetcr.f ntetcr.f 34 specific	Management of	the statistics
34	35 Management of specific	convergence	nmcrcg.f
35	36 Management of the specific SUIVI_DDL	nmcrdd.f	37 Management
36	of the specific quality standards		nmcrer.f
37	38 Management of specific control	nmdopi.f 39	specific Management
38	of classifications	nmnume.f	40 Management
39	of the specific dynamics	ndcrdy.f 41 Management	of
40	the discretization temporal	specific	nmcrli.f
41	specific Management of the convergence criteria	nmcrcv.f	ntcrcv.f 43
42	of the specific OBSERVATION	nmcrob.f 44 Management	of postprocessing
43	(MODE_VIBR and CRIT_STAB	) specific	nmdopo.f
44	45 specific Energy management eninit.f 46 Resolution	of the contact	specific
45	cfmxsd.f 47 Resolution	of specific	LIAISON_UNIL
46	cfmxsd.f Reading of	the user data	
47	the reading of the user data	is done	mainly

2.3 under the routine nmdata.f, this

routine reads the user data and creates possibly the SD necessary . The routine nmdome.f bed the characteristics given by the keywords MODELS, CHAM_MATER, CARA_ELEM and EXCIT by also treating the case of the resumptions of computation (reuse) which result uses the information stored in the SD. The table below gathers all the SD which will directly read information in the command file (and thus using the routines getxxx communication between FORTRAN and the supervisor Python). # Description Keywords Routine of reading 2 Models MODELS nmdome.f 3 Result concept created

	nmnoli.f 4 Characteristics		**elementary CARA_ELEM**
2	nmdome.f	5	Card of
3		*COMP_* nmdocc.f*	Loadings

		6	
4	EXCIT nmdome.f 7 Material coded CHAM_MATER		nmdome.f 8 Definition
5	of contacts	CONTACT	cfmxsd.f 9
6	of	unilateral	connections
7	CONTACT	cfmxsd.f 11 Parameters	
8	resolution METHODE RECH_LINEAIRE		nmdomt.f
9	Convergence criteria CONVERGENCE nmdocn.f	14	Parameters
11	of the convergence criterion	in stress CONVERGENCE	nmdocn.f
12	Method of resolution	METHODE RECH_LINEAIRE	
14	22 Solver solvers cresol.f 25 Card of the convergence criteria		for
15	COMP_* nmdocr.f	30 Management of the printings	PRINTING
22	nmdoim.f	33	Management of
25	the archivage and initial state (IN and OUT) ARCHIVAGE ETAT_INIT		nmetcr.f nmcrar.f
30	nmdoet.f 36 Management of SUIVI_DDL	SUIVI_DDL	nmcrdd.f
33	Management of the quality standards CRIT_QUALITE nmcrer.f 38	Management of control	CONTROL nmdopi.f 40 Management
36	dynamics SCHEMA_TEMPS	MASS_DIAG	PROJ_MODAL
37	MODE_STAT AMOR_MODAL ndlect.f 41 Management	of	the temporal
38	discretization DISCRETIZATION		nmcrsu.f 43
40	of OBSERVATION	OBSERVATION nmcrob.f 44 Management of postprocessing (MODE_VIBR	and
41	_STAB) CRIT_STAB MODE_VIBR nmdopo.f 45	Energy management	ENERGIE
43	eninit.f 46 Resolution of the contact	CONTACT	cfmxsd.f
44	Resolution of LIAISON_UNIL CONTACT cfmxsd.f Description	of the SD SD of low-level	Parameters
45	of resolution –	PARMET	the parameters
46	of resolution	are contained	in
47	PARMET. SD PARMET 1	Value of	REAC_INCR

2.4 2 Value of REAC_ITER

2.4.1 3 Value

2.4.1.1 of PAS_MINI_ELAS 4 Value of REAC_ITER_ELAS

5 Value of ITER_LINE_MAXI 6 Value of R ESI_LINE_RELA 7 Value

	of RHO_MIN
1	8 Value of RHO_MAX
2	9 Value of RHO_EXCL
3	Convergence criteria
4	PARCRI the convergence criteria
5	are contained in
6	vector PARCRI . SD PAR
7	CRY 1 Value of ITER_GLOB_MAXI
8	2 Value
9	RESI_GLOB_RELA 3

2.4.1.2 Value of RESI_GLOB_MAXI 4 Value of

ARRET 5 Value of ITER_GLOB_ELAS 6 Value of REAC_ REFE_ RELA 7 Value

	of CONVERGENCE
1	/TYPE 8 Value of PLATEAU_ITER
2	9 Value of PLATEAU_RELA
3	1 0 Value of RESI_COMP_RELA
4	Information
5	on the convergence of
6	– CONV information
7	on convergence
8	D years vector
9	. SD CONV 1 Nombre of iterations
	of linear search

2.4.1.3 2 Coefficient of linear search 3 Value

of RESI_GLOB_RELA 4 Value of RESI_GLOB_MAXI Parameters of the convergence criterion

1	in stress – PARCON the parameters of
2	in stress
3	(RESI_REFE_RELA) are contained
4	in vector PARCON

2.4.1.4 . SD PARCON 1 Value of SIGM_REFE 2 Value of EPSI_REFE 3 Value

of FLUX_THER_REFE 4 Value of FLUX_HYD1_REFE 5 Value of FLUX_HYD2_REFE 6 Value of VARI_REFE 7 Value of

	FORC_REFE (

1	8 Value of FORC_REFE
2	(2) 9 Value of DEPL_REFE
3	10 Value of LAGR_REFE
4	Method of resolution -
5	METHOD the methods of resolution
6	are contained
7	in vector METHODE
8	. SD METHOD 1 Method
9	resolution (Newton
10	, Newton - Krylov)

2.4.1.5 2 Type of matrix in correction 5 Type

of prediction 6 Name of SD RESULTAT for PREDICTION= "DEPL_CALCULE

	" 7 Method
1	of linear search SD of mean level activated
2	Functionalities – FONACT This
5	SD is very useful because
6	it makes it possible to know which functionalities are active
7	in the algorithm at any

2.4.2 time. The principal

2.4.2.1 idea of this SD is a very

rustic DISMOI making it possible to answer a simple question on the active features in op0070.f. For example: There is of the contact: ISFONC (FONACT, "CONTACT ") is true; The linear search is activated: ISFONC (FONACT, "RECH_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.f

. Even if this object is low-level (simple tables of INTEGER), one must reach it via three routines only: Operation on the vector of the activated features – FONACT Routine Preparation of the activated features nmfonc.f Interrogation of an activated functionality isfonc.f Rules

of exclusion of certain features exfonc.f It is imperative	
to modify vector FONACT only in	the routine
nmfonc.f (called in nminit.f) and always	to envisage
question the corresponding with this functionality	in

isfonc.f. ***In the same way, it*** is this vector which one will use in a priority way to test compatibilities between certain features (routine exfonc.f, called by nmfonc.f). Variable-hat - elementary Matrixes MEELEM This variable-hat contains the name of all the elementary matrixes usable in op0070.f. It is thus about a list of SD of the type sd_

2.4.2.2 matr_elem (sd_resu _elem). The code of location in

variable hats MEELEM/MEASSE is the same one if it is the same objects. Variable-hat - elementary Matrixes MEELEM Codes location elementary Matrixes of elementary stiffness MERIGI Matrixes of the limiting conditions of elementary Dirichlet dualized MEDIRI

Matrixes of mass MEMASS elementary Matrixes	of elementary damping
MEAMOR Matrixes of following	
loadings MESUIV elementary Matrixes of the substructures (macro-elements	
) MESSTR Matrixes elementary	of geometrical
stiffness MEGEOM elementary Matrixes	
of contact (continuous method and XFEM) MEELTC	Matrixes
elementary of friction (continuous method and XFEM	) MEELTF
Variable-hat – assembled Matrixes MEASSE	This
variable-hat contains the name of all the matrixes assembled	usable
in op0070.f. It is thus about a list of SD of the sd type	_matr_asse

2.4.2.3 . The code of location in variable hats

MEELEM/MEASSE is the same one if it is the same objects. Variable-hat – assembled Matrixes MEASSE Codes location Stamps assembled stiffness MERIGI Stamps assembled of mass MEMASS Stamps assembled damping MEAMOR Stamps assembled

substructures (macro-elements) MESSTR Variable	- hat –
elementary Vectors VEELEM	This variable
- hat contains	the name of
all the elementary vectors usable	
in op0070.f. It is thus about a list of SD of the type	sd_vect

2.4.2.4 _elem (sd_resu _elem). The code of location in

variable hats VEELEM/VEASSE is the same one if it is the same objects. Variable-hat - elementary Vectors VEELEM Codes location elementary Vector of elementary internal forces CNFINT Vector of the reactions of bearing for the limiting conditions of elementary

Dirichlet dualized CNDIRI Vector of the limiting	conditions of elementary
Dirichlet dualized CNBUDI Vector	of
nodal forces CNFNOD elementary Vector of the limiting conditions of elementary Dirichlet given	CNDIDO
Vector of the limiting conditions of elementary Dirichlet controlled	CNDIPI
Vector of the limiting conditions of	Neumann
given CNFEDO Vector elementary of the limiting conditions of controlled	
Neumann CNFEPI Vector elementary of the limiting conditions of type	elementary
Laplace CNLAPL Vector of the limiting conditions of standard elementary	

plane wave CNONDP Vector of the limiting conditions of Neumann following	
and given CNFSDO Vector elementary of the limiting conditions	of
standard impedance (in prediction) Elementary CNIMPP Vector of	the limiting
conditions of Dirichlet elementary differentials CNDIDI Vector of	the forces
on substructures CNSSTF elementary Vector of the forces of contact (continuous method	
and XFEM) CNELTC Vector elementary of frictional forces (continuous method	
and XFEM) CNELTF elementary Vector of the forces	of
reference (RESI_REFE_RELA) CNREFE Vector elementary of the command variables	
for initial state CNVCF1 elementary Vector of the command variables	for
elementary convergence CNVCF0 Vector of the limiting conditions	of standard
impedance (in correction) CNIMPC Variable-hat – assembled Vectors	
VEASSE This variable-hat contains the name of all the vectors	assembled
usable in op0070.f. It is thus about a list of SD of the type sd_cham	_no. These

2.4.2.5 vectors are primarily used in

the construction of the second members and the evaluating of convergence. The code of location in variable hats VEELEM/VEASSE is the same one if it is the same objects. Variable-hat – assembled Vectors 5TH ASSE Codes location Vector assembled of internal forces CNFINT Vector assembled of the reactions of bearing for the limiting conditions of Dirichlet dualized

CNDIRI Vector assembled of the limiting conditions	**of Dirichlet dualized**
CNBUDI Vector assembled of nodal	forces
CNFNOD Vector assembled of the limiting conditions of Dirichlet given CNDIDO Vector assembled	
of the limiting conditions of Dirichlet controlled CNDIPI Vector assembled	
of the limiting conditions of Neumann given	CNFEDO
Vector assembled of the limiting conditions of Neumann controlled	CNFEPI
Vector assembled of the limiting conditions of type Laplace CNLAPL	Vector
assembled of the limiting conditions of standard plane wave CNONDP	Vector
assembled of the limiting conditions of Neumann following and	given
CNFSDO Vector assembled of the limiting conditions of standard	impedance
(in prediction) CNIMPP Vector assembled of the limiting	conditions
of Dirichlet differentials CNDIDI Vector assembled of the forces on	substructures
CNSSTF assembled Vector of the forces of contact (continuous method	and XFEM
) CNELTC Vector assembled of frictional forces (continuous method	and

XFEM) CNELTF Vector assembled of the forces of reference	
(RESI_REFE_RELA) CNREFE Vector assembled of the command variables for	initial
state CNVCF1 Vector assembled of the command variables for convergence	
CNVCF0 Vector assembled of the limiting conditions of standard impedance	
(in correction) CNIMPC Vector assembled of the limiting conditions	of Dirichlet
eliminated CNCINE Vector assembled from substructures	CNSSTR
assembled Vector of the frictional forces (method DISCRETE) CNCTDF Vector	assembled
of command variables for computation CNVCPR Vector assembled	
of dynamic forces CNDYNA Vector assembled	
of modal damping (in prediction) CNMODP Vector assembled	of
modal damping (in correction) CNMODC Vector assembled	of
the forces of contact (method DISCRETE)	CNCTDC
Vector assembled of unilateral forces (method LIAISON_UNIL	)
CNUNIL Vector assembled of external forces CNFEXT Vector	
assembled of the limiting conditions of Variable type VECT_ISS CNVISS	- hat
– Vectors of solutions SOLALG This variable - hat contains	
the name of the fields at nodes which to compute: serve	in
the algorithm the solution. Variable-hat - Vectors	

2.4.2.6 of solutions SOLALG Codes location Solution in

displacement of the current iteration of Newton DDEPLA Displacement cumulated since the beginning of time step the DEPDEL Displacement increment

of time step the preceding	**DEPOLD Solution in**
displacement of prediction DEPPR1 Solution in displacement	of
prediction (left controlled) DEPPR2 Solution of velocity	
of the current iteration of Newton DVITLA Velocity cumulated	since the beginning of
time step the VITDEL Increment	
velocity of time step the preceding VITOLD Solution of velocity	of
prediction VITPR1 Solution of velocity of prediction	(left
controlled) VITPR2 Solution in acceleration of	the current
iteration of Newton DACCLA Acceleration cumulated	
since the beginning of time step the ACCDEL	Increment
of acceleration of time step the preceding ACCOLD	Solution
in acceleration of prediction ACCPR1 Solution in acceleration	

of prediction (left controlled) ACCPR2 Solution	of system
DEPSO1 Solution of system (left controlled) DEPSO	2 Variable
- hat – incremental Solutions VALINC	This
variable-hat contains the name of the fields at nodes or	the fields
of the type ELGA which will be	
the solutions of the nonlinear problem.	Variable

2.4.2.7 - hat – incremental Solutions V ALINC Codes

time step location Displacements at the beginning of the DEPMOI Forced with the beginning of time step the SIGMOI Local variables with the beginning of time step

the VARMOI Velocities with the beginning of time step the VITMOI	**Accelerations with**
the beginning of time step the ACCMOI Command variables	
with the beginning of time step the Variable	COMMOI
for the multifibre elements with the beginning	of time step
the STRMOI External forces	with the beginning
of time step (for the computation of	energies
) the FEXMOI Damping forces with the beginning of time step	
(for the computation of energies) the FAMMOI Strengths bonding with the beginning	of time step
(for computation energies) FLIMOI nodal Forces at the beginning of	time step
(for the computation of energies) the FNOMOI Displacements at the end of time step	DISPLEASED
Forced at the end of time step the SIGPLU Local variables at the end of	
time step the VARPLU Velocities at the end of time step the VITPLU Accelerations	at the end of
time step the ACCPLU Command variables	
at the end of time step TAKEN PLEASURE	Variable
for the multifibre elements at the end of	time step
the STRPLU External forces	at the end of
time step (for the computation of	energies
) the FEXPLU Damping forces at the end of time step	
(for the computation of energies) the FAMPLU Strengths bonding at the end of	time step
(for the computation of energies) the FLIPLU nodal Forces at the end of	time step

(for the computation of energies) extrapolated Forced FNOPLU (method	IMPLEX)
SIGEXT Displacements with the current iteration of Newton (management of large rotations	
) DEPKM1 Velocities with the current iteration of Newton (management of large rotations	
) VITKM1 Accelerations with the current	
iteration of Newton (management of large rotations) ACCKM1 Rotations with the iteration	of Newton
preceding (management of large rotations) ROMK Rotations with the current	iteratio n
of Newton (management of large rotations) ROMKM1 Access to the variable-hats	the access
to the variable-hats are done via five routines. Operation	
on the variable-hat Routine Creation of a variable-hat nmcha0.f Recovery	

2.4.2.8 of the index where is stored

the name of the variable in the variable-hat nmchai.f Recopy of a variable

- hat nmchcp.f Recovery of the name	**of**
the variable in the variable-hat	nmchex.f Recopy
a variable-hat by possibly changing a name of variable nmchso.f Creation	of the CHAM_NO
for VALINC, SOLAG and	VEASSE nmcrch.f
the size of these SD is indicated same way as	the SD
low-level. Nevertheless, it is advisable to reflect a change of size on	the principal
routine nmchai.f. 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 if required impact the length in op0070.f, nmini0.f (ASSERT of protection) and nmchai.f; To modify in nmchai.f; To create the variable-hat (see § 2.2); To initialize the contents of the variable if required
- - hat; These routines are satisfied to manage the variables - hats as a list of names ,
- the contents itself of
- these variable-hats do not depend 11 them
- . For example, variable-hat VEASSE contains assembled vectors

. None the routines of the preceding table deal with managing SD CHAM_NO of the objects contained in the variable-hat, just their name. For the three variable-hats defining of the CHAM_NO, the fields are created in the routine nmcrch.f, by means of information on active functionalities FONACT. SD of high-level Management of the printings – SDIMPR L be information of L has SDIMPR allow to manage the display, in particular that of the table of convergence . The SDIMPR is built starting from the concepts called "

2.4.3 generic SD objects

2.4.3.1 of type STRUCT" (see § 6

). The STRUCT of the type <AFFICHAGE > contains various information: Parameters of SD <AFFICHAGE> Standard Name Description TABL_CONV_CSV . True. If one leaves the table convergence on an external file (AFFICHAGE/UNITE) 54 <Booléen > INFO_RESIDU . True. If one adds columns of information on

the residues (DISPLAY /INFO_RESID)

<Booléen	> INFO_TEMPS	
. True. If one	adds columns of information over time (AFFICHAGE/INFO_TEMPS) <Booléen> PRINT	. True.
if one displays	information in the file .mess for time step running <Booléen> UNITE	Output unit
of the table	of convergence on an external file (AFFICHAGE/UNITE) <Entier> REAC_AFFICHAGE	Frequency
of	reactualsiation of the display in the file .mess (DISPLAY /PAS) <Entier> TABLEAU	_CONV Table
of	convergence <TABLEAU> One notes the presence of object TABLEAU_ CONV, of type <TABLEAU	> which
is the table	of convergence displayed during computation. The STRUCT of the type <TABLEAU > makes it possible to describe	
a table to be displayed	or leave on	a file

. SD <TABLEAU> – Parameters Standard Name Description OUTPUT _CSV. True . If one leaves the table convergence on an external file <Booléen > LARGEUR_LIGNE Width line of the table of convergence logical <Entier> UNITE_CSV Unit

to leave the table		
convergence	<Entier	>
HAUTEUR_TITRE	Many lines of title of the table of convergence <Entier>	COLONNES_DISPONIBLES
Lists	columns available in the table of convergence	<
SDLIST> <TABLEAU	_COLONNE> a <TABLEAU> consists of columns	COLUMNS
_DISPONIBLES, which	are of the SDLIST of the type <TABLEAU_COLONNE>.	The SDLIST of
the type <TABLEAU_ COLUMN	> makes it possible to describe a column in a table. Each	column is described by

the type of data which it contains (character string , integer, reality), its width, its title (contained and height), a possible additional mark (only for the columns of type the whole and real) and the fact that it is affected indeed of a value. The notion of assignment is important because a column can p ace contain value with each iteration of Newton. In certain case, that results by a null string, message “SANS_OBJET” or an error (the value should have been affected). SD <TABLEAU_COLONNE> – Parameters Standard Name Description VALE_AFFE .true if the value were affected <Booléen > WHOLE .true. If the column contains an integer <Booléen> REEL .true. If the column contains a reality <Booléen>

CHAINE .true. If the column contains		
	a character string <Booléen	
> NON_AFFE	_ERREUR .true. If an affected value	causes
an error	<Booléen> NON_AFFE_VIDE .true. If	an affected value
	causes the display of a blank <Booléen	> NON_
AFFE_SANSOBJ	.true. If an affected value causes	display
SANS_OBJET	<Booléen> WIDTH Width D E the column <Entier	> HAUTEUR_
TITER Many	lines of title of the column <Entier> VALE_I Value of	the column if

it is an integer <	Entier> VALE_R Value of the column if it is a reality <Réel>	VALE_K Value
of	the column if it is a character string	Standard
<Chaîne >	TYPE_COLONNE of the column (reference for <SDLIST	>) <Chaîne
>	TITRE_LIGN_1 First line of the title of	the column
<Chaîne	> TITRE_LIGN_2 Second line of the title	of
the column	<Chaîne> TITRE_LIGN_3 Third line	of the title of
the column	<Chaîne> MARK Marking of the column <Chaîne	>
the SDIMPR	is built starting from the STRUCT, one can	thus reach
it via	the routines of access of STRUCT (see	§6.5).
Nevertheless, most	operations on the SDIMPR	are encapsulated
	in routines moreover	high-level

. Operation on the management of the printings – SDIMPR Routine Printing of line of the table of convergence nmimpr.f55 Printing of boundary line nmimpx.f Printing of the recapitulation of the residues of equilibrium at the end of

the step nmimps.f Assignment of a value in a column of	**the type <**
Réel> nmimcr.f Assignment of a value in a column	of
the type <Entier> nmimci.f Assignment of	a value in
a column of the type <Chaîne> nmimck.f Activation/deactivation of	a column nmimca.f
Deactivation of all the columns nmimr0.f Activation	or deactivation
of the display for time step running nmimpa.f	In the same way
, the STRUCT of the type <TABLEAU> can be handled with	
more practical routines: Operation on	the STRUCT
of the type <TABLEAU> Routine Creation of	one blank line
with the separators of columns obtlig.f Assignment of a mark in	a column

obtsdm.f Creation of the heading of table obttit.f Computation of the width of one line obtcla.f Creation

of boundary line in table	**implis.f**
Printing of line of table impsdl.f There exist four low	routines
level dealing with the formatting of the character string	of information
to display according to its type	. Operation
of formatting Routine Formatting	of a reality
impfor.f Formatting of an integer impfoi.f Formatting of	a character string

impfok.f Formatting of a time (transformation	

second in hours/min/s) impfot.f Management of measurements of time – SDTIME This SD is used to make the various

mesures of time in	**the nonlinear**
phases of computation	. SDTIME
– Objects Name Description	SDTIME
(1:19)/" .TDEB" Storage	of initial
time SDTIME (1:19)/" .TIMN' Storage of measurements for	the iteration

2.4.3.2 of Newton SDTIME (1:19)/".TIMP " Storage

of measurements for time step the SDTIME (1:19)/".TIMT" Storage of measurements for complete transient

SDTIME (1:19)/	
/".TMP	**1" temporary Timer**
1 SDTIME (1:19	)/".TMP2" temporary Timer
2 SDTIME (1:19)/".TPAS	" Timer for time step the SDTIME (1:19)/".TITE
" Timer for the iteration	of Newton SDTIME (1:19)/".TARC" Timer
for archivage SDTIME	(1:19)/".TPSD" Timer for postprocessing
Four timers are reserved	for recurring
measurements (time step	, iteration
of Newton, archivage and	postprocessing) and two
others timers are useful	for other measurements (factorization
time in the contact	, etc). The SD
is conceived to store	the times cumulated on the phases

(on all time step, on all the transient), and this, in an automatic way. It is thus preferable to use the SDTIME rather than the timers directly (routines UTTCPU) because the statistics are made
automatically, as well as the displays. The principal routines of handling it SD are the following ones: Operation on the management of
measurements of time – SDTIME Routine Creation of the SDTIME – It is in this routine that one will give the number of measurable things (variable NBMESU) nmcrti.f Given to zero of the statistics and of the timers nmrini.f Management of the timers (initialization

, measurement of time) nmtime.f Safeguard of a time given in	**the statistics**
of the timer indicated nmrtim.f Measurement of time remaining at the end of an iteration of Newton or one time step nmtima.f Reading	of
the information stored in SDTIME nmtimr.f Measurement	of "
wasted" time in cuttings of time step during	computation
nmlost.f Display of the statistics and times nmstat.f the two	most

	important
routines is nmcrti.f with which one will be able to add a new measurement and nmtime.f	which
carries out measurement. Example of use : CAL	NMTIME
(SDTIME, “INI”, “ASSE_MATR”) CAL NMTIME (SDTIME, “RUN”, “ASSE_MATR”) /.OPERATION	. /CAL
NMTIME (SDTIME, “END”, “ASSE_MATR”) the routine	nmstat.f

makes it possible to make the display of information (also used by the SDSTAT, to see § 2.4.3.5). Management of the errors of the algorithm
– SD ERRO 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 algorithm (modifiable in nmcrga.f, cf § 3.4.1) SDERRO – Objects Name Description SDERRO 27)/

2.4.3.3 “.ENOM” Name of event SDERRO (1:19) /“ .ECOV

” Value of the code-return related to event SDERRO (1:19)/“.ECON” Name of the code-return related to the event SDERRO (1:19)/Standard “.ENIV” and level of release of event SDERRO (1:19)/“.EFCT ” Functionality 40

activating an event	
	of type convergence
or divergence	SDERRO (1:19)/“.EACT
” State of event	(activated or not) SDERRO (1:19)/“.EMSG
” Codes 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 the displays). SDERRO – Objects Name Description
SDERRO (1:19)/“.CONV	” State of loop SDERRO (1:19)/“.EEVT
” Information on	the last event started SD ERRO is by means of used

certain contained information in the SDDISC (§ 2.4.3.14), coming from the definitions of events of the command DEFI_LIST_INST

. Operation on	
	the management of
the errors of the algorithm	– SDERRO Routine
Creation of	SD ERRO nmcrga.f Record of an event nmcrel.f

Record of an event from a code-return nmcret.f Change of the state 33 nmeceb.f Reading of the state of a loop nmleeb.f Given to zero of the events

nmeraz.f Turns over the state of an event (active or not) according to its	**name nmerge.f**
Emission of the message	of

	information
on the event nmevim.f Evaluating	of
the state of convergence of a loop nmevcv.f Turns over the state	of an event
(active or not) according to its	nmltev.f type
the use of these routines	in
the frame of the management of the events	is
detailed in the § 3.4. Management of the archivage and the initial	state
(IN and OUT) – SDIETO This SD is used to carry out	the following
operations: Initialization of the fields (keywords	ETAT_INIT
, taking into account a possible enrichment of the SD	result

by the use of reuse); Archivage of the fields: puts rhythm into of archivage, exclusion of certain fields 40 ;

2.4.3.4 This SD has only two objects. One which gives general information

, and a second who gathers the necessary information
•for each field. This last object is proportional to the number of fields (variable NBMAX) manageable. SDIETO – Objects Name Description SDIETO
•(1:19)/" .LCHA' Information on fields SDIETO (1:19)/".INFO" Information

general (many fields, many fields as starter, many fields in output) information at once are contained in DATED in the routine nmetcr.f, they are the following ones: Description of a field for SDIETO

Description A	
DATED	**result**
to inform Name of the field in	SD NOMCHS YES Name of
the field being used to initialize	to modify nmetc0.f Statute of the field after initialization (no one, given individually by

the user, given in a SD RESULTAT, resulting from a steady , constant computation with a given value), NON

Name of the quantity support of field		
NOMGD YES Keyword		**for ETAT_INIT**
MOTCEI Not compulsory Localization		of
field (NOEU/ELGA/ELEM) LOCCHA YES Known as	if the field must	
be in an initial state (it null or would be read by ETAT_INIT) LETIN YES Known as if the field must be filed LARCH YES Keyword for OBSERVATION MOTCOB Not compulsory Name field corresponding		
during the algorithm (for example		, in
variable hat SOLALG	)	to modify nmetcc.f
all the current operations	is envisaged	
in routines in general prefixed by nmet*. When one wants to add a field	in	management

ARCHIVAGE/OBSERVATION/ETAT_INIT	: It	is necessary
that the developer active	or not	the field following
the activated features (for example). The routine nmetac.f should be impacted; the developer specifies It is necessary that		

the initial field (if field in recovery). The routine nmetc0.f should be impacted; It is necessary that the developer specifies the name of the field in the operator . The routine nmetcr.f should
•be impacted; When a field is filed in SD RESULTAT, rscrsd.f should be impacted. Operation on the management of the archivage and the initial
•state (IN and OUT) – SDIETO Routine Creation of the SDIETO nmetcr.f Addition of a field in the SDIETO nmetci.f Addition of
•the name of the field which corresponds in the algorithm of op0070.f nmetcc.f Name and creation of the null field nmetc0 .f Lecture
•of the initial state nmdoet.f Reading of a field – Case of the SD result in

ETAT_INIT nmetl 1. F Reading of a field – Case field by field in ETAT_INIT nmetl 2.f	Reading
of a field – various	Checks
: Check the quantity and	the localization
Treats the case of prestressing (nmsigi.f) Checks the coherence	of
the field of the local variables	(vrcomp.f) nmetl
3.f Turns over the name of	the field used
in op0070.f (by index of field) nmetnc.f Turns over the index	of the observable
field result (by name of the field) nmetob.f Transformation	of the fields
of the variable-hats type of the type “ME” (urgent •previous) in type “RAINED” (urgent •according to) nmetpl.f Conversion of the localization of •a field nmetcv.f Writing of the field in the SD nmeteo.f Management	of the statistics
– SDSTAT This SD makes it possible to make statistics (collection	and display
). It contains objects which all are dimensioned	with the number
of things to be measured (variable NBSTAT in nmcrst.f). SDCONV – Objects Name Description SDSTAT (1:19)/Statistical “.VLIT”	for
transient SDSTAT (1:19)/Statistical “.VLIP”	
for time step the Statistical SDSTAT (1:19)/“	.VLIN″

2.4.3.5 for the iteration of Newton If

one wishes to modify statistics, only the first three following routines are to be modified. The routine nmstat.f makes it possible so much to make the display of information (also used

by the SDTIME	
,	to see § 2.4.3.2
. Description Routine	Creation of the SDSTAT nmcrst.f Read/write
in	the SDSTAT nmrvai.f Display of the statistics
and times	nmstat.f Rebootstrapping of the statistics

nmrini.f Management of convergence – SDCONV This SD manages the collection and the display of the relative information to the residues of equilibrium (<RESI> buckles, to see §3.1). It contains objects which all are dimensioned24

of residues	to be considered
(variable NRESI	in nmcrcg.f
). SDCONV – Objects Name Description	SDCONV
(1:19)/Standard ".TYPE" of residues (	RESI_GLOB_RELA
, RESI_GLOB_MAXI, etc) SDCONV	(1:19)/

2.4.3.6 ".LIEU" Place where the residue is maximum

SDCONV (1:19)/".VALE" Value of residue SDCONV (1:19)/".ACTI". True. if the residue is activated (is used for convergence38) SDCONV (1:19)/".NCOL" Name of the columns in the SDIMPR the activation of a residue for convergence is not necessarily

constant during	
	computation. It
happens sometimes that one	rocks automatically of RESI_GLOB_RELA with RESI_GLOB_MAXI
for example). For	the management of convergence, one
will modify primarily	the routines
nmcrcg.f, nmimre.f and nmcore.f	. The rocker from one residue to another for example, will be done
in nmcore.f. The routine	of nmimps.f printing is "car

- bearing": the contained information in SDCONV is enough for him. It should not thus be modified. Operation on the management of convergence – SDCONV Routine Creation of the SDCONV nmcrcg.f

Checking of the stopping criteria on the residues nmcore.f Printing of the residues summaries at the end of time step the nmimps.f Printing of information on the residues in the table of convergence nmimre.f Management of the extraction of fields – SDEXTR the SUIVI_DDL divides with the OBSERVATION ***(see § 2.4.3.9) a SD commune called***

SDEXTR, routine of extraction of the values of the fields . One	will thus
start by describing	the latter

SD. The first three objects are general	and their length
is proportional to the number of occurrences of keyword	SUIVI_DDL
or OBSERVATION. SDEXTR – Objects Name (attention with the blanks!) Description	SDEXTR (

2.4.3.7 1:14)/". INFO" various Information on

the data of extraction SDEXTR (1:14)/". EXTR" Type29 On field (NOEU and ELGA) On the mesh (if ELGA) On the components or formulates between components SDEXTR (1:14)/".ACTI" Extraction activates or not One cannot have more than 99 occurrences of the keywords because one builds the name of certain objects from this number

of occurrence	
OCC. These objects are the following	**:**
SDEXTR – Objects Name (	attention with the blanks!) Description SDEXTR (1:14) //OCC

/".NOEU" Nodes concerned	with extraction
	•SDEXTR (1:14) //OCC// ".
	•MAIL" Meshes concerned
	•with extraction SDEXTR (1:14) //OCC// ".POIN" Points

of integration concerned	with extraction SDEXTR

(1:14) //OCC// ".SSPI" Subpoints of integration concerned with extraction SDEXTR (1:14) //OCC// "Component .CMP" concerned with the extraction Management of the SUIVI_DDL –

SDSUIV This SD is used	
to manage functionality	**SUIVI_DDL**
. Besides the references	to the extraction of the fields (SDEXTR
, § 2.4.3.7), we have	a specific object for the SUIVI_DDL.
SDSUIV – Objects Name (attention	with the blanks!) Description SD SUIV (1:14)/"
. TITR" Titles of the columns	Management of the OBSERVATION – SDOBSE This SD is used
to manage functionality	OBSERVATION. The OBSERVATION divides with

2.4.3.8 the SUIVI_DDL (see § 2.4.3.8)

a SD commune called SDEXTR, already described in (§ 2.4.3.7). Then, we have specific objects for the OBSERVATION29 One which will give the name of the array and one which stores

information relating	
to the frequency of observation	**(utility**
object SDSELI , to see	§ 2.4.3.15), subscripted

2.4.3.9 by the number of occurrence OCC of

keyword OBSERVATION. SDOBSE – Objects Name (attention with the blanks!) Description SDOBSE (1:14)/ ".TABL"29 array SDOBSE (1:14)/ OCC//".LI " Access to the SDSELI,29

times to observe 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 –34 Name Description SDSUIV (1:1 9)/".ERRT " Value of errors THM spaces

and time , coefficient	
THETA Management of control	**– SDPILO**
This SD is used for control	(method of
continuation). Most	these objects are gathered in a logic of the type

2.4.3.10 *(realities with realities, character strings with character strings) ,*

rather than in a logic of functionality. SDPILO – Objects Name Description SDPILO (1:19)/".PLTK" Contains the parameters

of the control (of	
type	**chains) or**
the names of objects necessary	to control SDPILO (1:19)/".PLIR" Contains the parameters

2.4.3.11 *of control (of real type*

) SDPILO (1:14)/".PLCR" Lists coefficients for control SDPILO (1:14)/".PLSL" Lists DDL of the type, and for control SDPILO (1:14)/".PLCI" Lists coefficients for control – Case XFEM It

does not have there routines	
of access	**of high-level**
, the SD is created	in the routine nmdopi.f. the SD should be attacked directly. Management of classifications – SDNUME
This SD comes in complement from	two SD NUMDDL and NUMFIX to manage
the numbers of the equations	. NUMDDL is the numbers of the equations
, which	can be variable during DX DY DZ the transient, when
one CONTINUE uses	functionalities like the contact or contact

XFEM. NUMFIX is fixed classification, it is useful in the case of the macro-elements , to be able to combine the matrixes.

2.4.3.12 *The SDNUME contains three other objects :*

SDNUME – Objects Name Description SDNUME (1:19)/ ".NDRO " Location of the DDL for large 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 nombre total of degrees of freedom of structure, with same classification as matrixes and the CHAM_NO used

in op 0070.f	
.	**A particular value**
(in general	1), is used to locate specific DDL a priori

: those corresponding	to large 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 large rotations or to filter the components in the evaluating of the residues). There are no specific routines of access. Management of the dynamics – SDDYNA This SD contains all the necessary information with computation in dynamics. SDDYNA – Objects Name Description SDDYNA (1:15)/".PARA_SCH" Parameters of time schemes SDDYNA (1:15)/".INFO_SD" Parameters of dynamics SDDYNA (1:15)/".NOM_SD" Name of the SD for the dynamics (static modes, vectors,…) SDDYNA (1:15)/Standard ".TYPE_FOR" of formulation (displacement, velocity or acceleration) SDDYNA

2.4.3.13 *(1:15)/".COEF _SCH" Coefficients*

to be used in computation SDDYNA (1:15)/".TYPE_CHA" Information relating to loading

ONDE_PLANE	
SDDYNA	**(1:15)/"**
.NBRE_CHA" Information relative	to certain loadings specific
to dynamics	SDDYNA (1:15)/".VEEL_OLD" elementary
Vectors of	the preceding step for multi-step diagrams SDDYNA (1:15)/".VEAS
_OLD" assembled Vectors	of the preceding step for multi-step diagrams SDDYNA (1:15)
/".VECENT" Quantities in	trainings (displacements, velocities and
acceleration) SDDYNA (1:	15)/".VECABS" absolute Quantities (displacements
, velocities and acceleration	) the access to this information is done via four routines dedicated each one
to a type, with a character string	posing question (on the model of routine DISMOI): Operation
on the management of	the dynamics – SDDYNA Routine Reading information of the <booléen type
> ndynlo.f Reading information	of the type <réel> ndynre.f Reading information of the type <entier> ndynin.f
Reading information of the type	<chaîne> ndynkk.f It should be noted that these questions will involve

a fatal error if one is not in dynamics, except in the case of question the NDYNLO (SDDYNA, "DYNAMIQUE") which will answer .false if one is in static (i.e.

which can detect the case where SDDYNA does not exist) .	**Apart from**
these four routines, the access to	the SDDYNA
is done directly in two routines	: Operation
on the management of the dynamics	–
SDDYNA Routine Reading information in	the command file

ndlect.f Record value of the various coefficients ndnpas.f One reconsiders the coefficients necessary for computation in dynamics because they are very numerous . These coefficients are built from three information : The type of diagram; Parameters of the diagrams (coefficients ALPHA , BETA, KAPPA, etc); The increment

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

in front of stiffness matrix COEF_MATR_AMOR Coefficient in front of damping matrix COEF_MATR_MASS Coefficient in front of mass matrix COEF_DEPL_DEPL Predictor
- in displacement
- : coefficient in front of the displacement of step preceding COEF_ DEPL_VITE
- Predictor in displacement

: coefficient in front of the velocity of step preceding COEF_DEPL_ACCE Predictor in displacement: coefficient in front of the acceleration

of step	preceding
COEF_VITE_DEPL	Predictor of velocity: coefficient in front of
the displacement	of step preceding COEF_VITE_VITE Predictor
of velocity	: coefficient in front of the velocity of step
preceding	COEF_VITE_ACCE Predictor of velocity: coefficient in front of the acceleration of preceding
step COEF	_VITE_DEPL Predictor in acceleration: coefficient in front of the displacement
of step preceding	COEF_VITE_VITE Predictor in acceleration: coefficient in front of the velocity of preceding
step COEF	_VITE_ACCE Predictor in acceleration: coefficient in front of the acceleration
of step preceding	COEF_DEPL Coefficient in front of displacement increment COEF_VITE
Coefficient in front of	the increment velocity COEF_ACCE Coefficient in front of the increment of acceleration
COEF_MPAS_FEXT	_PREC Coefficient in front of the external forces of step preceding (multi-step diagram
) COEF_MPAS_FINT	_PREC Coefficient in front of the internal forces of step preceding (multi-step diagram
) COEF_MPAS_	FEXT_COUR Coefficient in front of the external forces 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 recoil forces dynamics (inertia
) COEF_FDYN_AMORT	Coefficient in front of the recoil forces dynamics (damping) COEF_FDYN_RIGID
Coefficient in front of	the recoil forces????? ? (is used for Krenk???) COEF_FORC_INER Coefficient in front of
the inertia forces	to put at the denominator in the residue of equilibrium INST_PREC Time step
preceding	(only useful for the poursuite with, at the same time, complete diagram HHT
and the constitutive laws	which needs this parameter) Management of

the temporal	discretization – SDDISC data structure SDDISC contains all
information	coming from operator DEFI_LIST_INST (creation of concept
SDLIST) and from	information relating to the management of the temporal discretization in op0070.f. Initially
	, information coming from operator DEFI_LIST_INST (see [D4.06.17]) is recopied locally in the SDDISC. SDDISC – Objects Name Description SDDISC

2.4.3.14 *(1:19)/“ .LINF ” Information on the list of* `times`

(see contained in [D4.06 .17]) Recopy of object SDLIST (1:8)/“.LISTE.INFOR” SDDISC (1:19)/“.DITR” Lists times – This list is dynamic (in the event of cutting or of acceleration of time step) SDDISC (1:19)/Indicating “.DINI” of the level of under-cutting for each time step . 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.f) – This list is dynamic (in the event of cutting or of acceleration of time step) SDDISC
(1:19)/“.ITER” Nombre of iterations	of Newton which it was necessary for each time step – This list is dynamic (in the event of
cutting or of acceleration	of time step) SDDISC (1:19)/Indicating “ .EPIL” of the choice of the solution of control (action AUTRE_PILOTAGE) SDDISC (1:19)/“.LIPO” Lists compulsory times of computation. This object is useful in the case of the automatic adaptation of time step, it indicates times which will be calculated though it arrives
(even in the event of enlarging	of time step). One also speaks about urgent “stakes”. Its length is that of the initial list of times given by the user in
DEFI_LISTE_INST. It is	thus about a list NON-dynamics. SDDISC (1:19)/“.REPC ” Indicating D
E management of the reactualization	of preconditionning (action REAC_PRECOND) SDDISC (1:19)/“.EEVR” Recopies object SDLIST (1:8)/“.ECHE.EVENR” See contained in [D4.06.17] SDDISC (1:19)/“.EEVK” Recopies object SDLIST (1:8)/“.ECHE.EVENK” See contained in [D4.06.17] SDDISC (1:19)/“.ESUR” Recopies object SDLIST (1:8)/“.ECHE.SUBDR” See contained in [D4.06.17] SDDISC (1:19)/“.AEVR” Recopies object ***SDLIST (1:8)/***
“.ADAP.EVENR” See contained	in [D4.06.17] SDDISC (1:19)/“.AEVK” Recopies object SDLIST (1:8)/ “.ADAP.EVENK” See
contained in [D4.06	.17] SDDISC (1:19)/ /“.ATPR” Recopies object SDLIST (1:8)/“.ADAP .TPLUR”
See contained in [D4.06	.17] SDDISC (1:19) /“.ATPK” Recopies object SDLIST (1:8)/“.ADAP .TPLUK”
See contained in [D4.06	.17] SDDISC (1:19) /“.AEXT” Object for the prolongation of cutting (processing
of the case of the 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) – Standard 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) – Indicates that one authorizes iterations in more SDDISC (1:19)/".IFRE" Object storing the value of the residues to each iteration of Newton V (3* (ITER-1) +0) – Value of RESI_GLOB_RELA V (3* (ITER - 1) +1) – Value of RESI_GLOB_MAXI V (3* (ITER - 1) +2) - Value of the loading to handle the SDDISC , one uses a series of utilities . Operation on the management 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 writing. utdidt.f Turns over .RUE if one leaves the list of times (end of the transient) didern.f Value of time according to number

of time diinst.f Management of selection the one time –

SDSELI data structure SDSELI allows selects one time	**following**
a frequency, a list of values, in taking into account a tolerance and a kind of criterion (absolute or relative). SDSELI – Objects Name Description SDSELI (1:19)/".INFL" general Information on selection	
(parameters of selection) SDSELI (1:19)/".LISTE" Lists the times given	by
the user to operate on this SD, one uses	mainly

2.4.3.15 *two routines: one which reads the parameters*

of the user (nmcrpx.f); search if time given is selected (nmcrpo.f); Operation on the management of the one time selection – SDSELI Routine Reading of the list of

times to select	
	nmcrpa.f
Reading of the accuracy	and the type of criterion (relative or absolute) nmcrpp.f Reading of all
information (call	to nmcrpa.f and nmcrpp.f) nmcrpx.f Search for

a reality in a list (with accuracy and criterion) utacli.f principal Routine
- of search of time nmcrpo.f Search of the index
- in the SD result right before a time given nmttch.f This

SD is used for the management of the temporal discretization ,	**the initial**
state, the archivage and the observation. Management	of S
convergence criteria – SDCRIT This SD manages information	relating

to the convergence criteria, it is used in particular to store	the residues
in SD RESULTAT. SD CRIT – Objects Name Description SDCRIT (	1:19)/". CRTR
" Values of information (residues, nombre of iterations	
, etc) SDCRIT (1:19)/".CRDE" Name of information for storage	in

SD RESULTAT (residues, nombre of iterations, etc) This SD is also used for command THER_NON_LINE (attention,

2.4.3.16 *the objects are not same dimension !) .*

The safeguard of information for this SD is done in the routine nmcore.f which evaluates convergence with each iteration of Newton. Operation

on the management of	
the convergence criteria	
– SDCRIT	Routine Creation of the SD nmcrcv.f ntcrcv.f Safeguard of information
in the SDCRIT nmcore.f	op0186.f Safeguard of information in SD RESULTAT nmarc0.f ntarc0.f Management of postprocessing

– SDPOST This SD manages information relative so that one indicates like "postprocessing". Postprocessings on the spectral analyzes (oscillatory modes or modes of stability) which one can make with each time step. The SD consists

of seven objects . Three of them are simple vectors	**gathering**
information	and the parameters
necessary, classified according to their type.	SDCRIT – Objects
Name (attention with the length!) Description SD	POST (1:19)/". INFI

2.4.3.17 *" Parameters of the type <entier > SD*

POST (1:19)/". INFR" Parameters of the type <réel> SD POST (1:19)/". INFK" Parameters of the type <chaîne> SD POST (1:1 4)/". VIBR" Name of the SD of the type SDSELI (§ 2.4.3.15) for oscillatory modes SDPOST (1:14)/".EXCL" Name of the SD storing the names of the degrees of freedom excluded for the modes from stability SDPOST (1:14)/".STAB" Name of the SD storing the names of the degrees of freedom taken

into account for	
the modes of stability SD POST	**(1:1 4)/**
". FLAM" Name of the SD of	the type SDSELI (§ 2.4.3.15) for
the modes of stability or	buckling Here the list
of the parameters: Standard	parameter Description CRIT_STAB

<entier> Is worth 1 if one does	a calculation of the modes type of stability 34 buckling modes MODE_VIBR
<entier> Is worth 1 if	one does a calculation of the type oscillatory modes OPTION_CALCUL_FLAMB <chaîne> Type of computation
of stability: static	or dynamic OPTION_CALCUL_VIBR <chaîne> Type of computation of oscillatory modes : inevitably dynamic
Standard TYPE_MATR_VIBR	<chaîne> of stiffness matrix (elastic34, secant or tangent) in oscillatory modes OPTION

_EXTR_VIBR <chaîne> Standard of search

(“		**smaller”**
or “bandages	”)	for oscillatory modes NB_FREQ_VIBR <entier> Many frequencies to calculating
for oscillatory modes		BANDE_VIBR_1 <réel> First limit if search
“bandages” for	oscillatory modes	BANDE_VIBR_2 <réel> Second limit if search
“bandages	” for	oscillatory modes TYPE_MATR_FLAMB <chaîne> Standard of stiffness matrix
	(elastic	, secant or tangent) in buckling modes (uses information given in
keyword PREDICTION	of NEWTON	). NB_FREQ_FLAMB <entier> Many frequencies to calculating for the modes of
stability or	buckling	BANDE_FLAMB_1 <réel> First limit if search “
bandages” for		the modes of stability or buckling BANDE_FLAMB_2 <réel> Second limit
if search	“	bandages” for the modes of stability or buckling RIGI_GEOM_FLAMB
<chaîne> Computation	of	the modes of stability or buckling with or without taking into account of geometrical stiffness matrix OPTION_EXTR_FLAM <chaîne> Standard of search (“
smaller” or	“bandages	”) for the modes of stability or buckling NB_DDL_EXCLUS <entier> Many
degrees of freedom		excluded for the modes from stability SOLU_FREQ_VIBR <réel> Value of the frequency for
oscillatory mode		SOLU_FREQ_FLAM <réel> Value of the frequency for the mode of buckling SOLU_NUME_VIBR
<entier> Index	of oscillatory mode	SOLU_NUME_FLAM <entier> Index of the mode of buckling SOLU_MODE_VIBR <chaîne> Oscillatory mode (field of displacement
) SOLU_MODE	_FLAM <chaîne	> Mode of buckling (field of displacement) NOM_DDL_EXCLUS <chaîne> SDPOST (1:14)/“.EXCL” NOM_DDL
_STAB <chaîne	> SDPOST (	1:14)/“.STAB” NB_DDL_STAB <entier> Many degrees of freedom
taken into account	for	the modes of stability SOLU_FREQ_STAB <réel> Value
of the frequency	for	the mode of stability SOLU_NUME_STAB <entier> Index
of the mode of	stability	SOLU_MODE_STAB <chaîne

> Mode of stability	(field	of displacement) COEF_DIM
_FLAMB <entier	> Coefficient	of the subspace (see modal analysis
) for the modes	of stability	or buckling modes COEF_DIM_
VIBR <entier> Coefficient		of the subspace (see
modal analysis	) for	oscillatory modes MODI
_RIGI <chaîne	> Modificatio n	of stiffness SIGN_INSTAB <chaîne> Sign to detect an instability
PREC	_INSTAB	<réel> Accuracy to detect an instability
Management of the algorithm		We will be interested
in management of	the algorithm	. To manage convergence, different
the levels	from loop	and the errors which have occurred during a computation, one uses a common formalism, based on the notion of event
. The various		loops In the algorithm, there are five levels of loops <BOUC>: <RESI
>: the loop	on	the various residues of equilibrium
requested	by	user (RESI_GLOB_RELA, RESI_GLOB_MAXI
, etc	…) ;	<NEWT>: the loop on the iterations of

3 Newton; <FIXE>: the loop

on the fixed points (contact); <INST>: the loop over times of computation; < CALC >: the loop on computation. “Loop” <CALC> is not really one. It is used to manage two situations: The computation **finishes** normally

3.1 (not error)

but otherwise than when one reached the end of the loop on
• time step. For time, the only case is that where control reached its limits; An event is started outside
• the loop on time step, i.e.
• during postprocessing (detection of instability
• by oscillatory modes); The various
•loops are distributed between op 0070
.f, nmnewt.f (static and dynamic implicit) and ndexpl.f (dynamic explicit). For
1.the loop on the residues of equilibrium, it is necessary to look in the routine nmcore.f, called by nmconv.f. For the loop on the fixed points, one uses the routines nmible.f/nmtble.f, contained in nmnewt.f. State of
2.the loops the state of a loop is stored in object SDERRO (§ 2.4.3.3), one reaches it by nmleeb.f and nmeceb.f. There exist six states: CONT: the continuous loop; CONV:
the loop converged (events of type convergence and divergence); WANDER: an error (event of type error) occurred during the loop, one will treat it; EVEN: an event (event of the informative type) occurred during the loop ; STOP: an error (event of type error) occurred during the loop, one stops ; CTCD

3.2 : particular state

of the loop of Newton for the discrete contact ; The events25 One can classify the events according to their origin: Events
1. defined by the user
2. in DEFI_LIST _INST (for example, DELTA_GRANDEUR); Intrinsic events
3. with the algorithm: errors and the convergence of the various loops; Types of the events
4. an event is defined by its type which is a character string in two
5.parts . First channel describes the type of the event:
6.The events causing an error are prefixed by <ERRC_*> or

3.3 <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_*>; Second character string <YYYY> describes the level

3.3.1 of loop of the event

: Buckle on the residues of equilibrium <*_RESI>; Buckle on the iterations of Newton <*_NEWT>; Buckle of fixed point for contact <*_FIXE>; Buckle on
•time step the <*_INST>; This information describes in which loop the events
•can potentially start . Events of type error <ERR*_*> Of
•the errors are with being immediately treated (i.e. as of release of the event
•) because they are blocking for the process , they
are noted <ERRI >. The errors with treating only with convergence of
•the loop are noted <ERRC>. For example: <ERRI
•_NEWT> is an error with treating immediately
•in an iteration of Newton as a default
•of integration of the constitutive law
prevents or a matrix not factorisable; <ERRC_NEWT> is an error with treating with convergence of

3.3.1.1 Newton, for example an output of a physical

criterion out of the limits of its field of definition; <ERRI_FIXE> is an error with treating immediately in a loop of fixed point ; Important: typified errors <ERRI_CALC> causes the dead halt of computation because no action
can treat them
•. They are the stops in lack of time CPU and the external stop by user . Events of type convergence <CONV_*> the events of type convergence are treated with each
•level of loop . This convergence perhaps related to an activated functionality or not (cf § 2.4.2.1). Events of type divergence <DIVE_*> the events
•of type divergence are treated with each level of loop. This divergence perhaps

related to an activated functionality or not (cf § 2.4.2.1). In practice, the state of each loop is rather treated in divergence. Indeed, to avoid having to check that a specific

3.3.1.2 functionality is active (control ,

Deborst, contact, etc…) and thus to test the convergence of a loop by checking this functionality, one prefers to emit an event of divergence that16

3.3.1.3 of convergence: one can have

divergence only if the functionality is activated, whereas one could have convergence even if the functionality is not activated . Events of the informative16
<EVEN> events of the informative type are the other events (neither error, nor convergence). By default, they are only used to give indications to the user (often in the form of alarm messages, the transition of RESI_GLOB_RELA with RESI_GLOB_MAXI for example). Some are activables only on request of the user (command DEFI_LIST_INST). These events could as be treated explicitly (otherwise as by the emission of an alarm or information) via DEFI_LIST_INST

3.3.1.4 . These events are not related to

a level of loop. The case of the code-return a code-return is a single integer 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. Return codes -1 and 0 always have the same meaning: -1: nothing was done (not
factorization, not integration of constitutive law, etc); 0: one did something and very did well; The other values of
return codes have a meaning dependant on their type: Standard Description

3.3.1.5 Meaning

of code FAC Return of factorization -1 No factorization 0 Whole occurred 1 well the matrix is singular 2 factorization failed 3 One cannot say if the matrix is singular PIL Return of control -1 No control 0 Whole did without 1 well Step solution of
•the equation of control 2 One reached a limit (end of computation) LDC Return of the integration of
•the constitutive law -1 No integration of the behavior
0 Whole did without 1 well Failure during integration constitutive law

	2 a physical	**parameter is not**	
	in its field of definition	-1	3 stress SIZZ
			is not not null (algorithm
		1	of Borst) CTC Return
		2	of the processing of the contact discrete
		3	-1 discrete contact No 0 Whole did without 1
	well the maximum number	-1	iterations contact
			was reached 2

		1	the matrix of the contact is singular Management
		2	the events Modification, addition or
suppression	of an event Most standardized	-1	is envisaged in data structure
			and the
		1	utility which are dependant there. In most case,
		2	is a question of modifying only the routine nmcrga.f (see § 2.4.3.3 by adding
		3	the characteristic of the event in DATED to it at the beginning of
	routine. Description DATED Name from	-1	event NEVEN Name
		of	the code-return related to
		1	(XXX if not code-return) NCRET Value from the code-return
		2	related to the event (99 if not code

3.4 - return) VCRET Standard and level
3.4.1 of release of event TEVEN Functionality

activating an event of type convergence or divergence FEVEN Codes message to be displayed when the event starts MEVEN the verification system of the messages makes that the code of25 to be displayed (MEVEN) is not sufficient, it is also necessary to impact nmevim.f. Emissions of the events

to start an event	
the developer must envisage it in the algorithm by calling	the routine
nmcrel.f. 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.) Transition of RESI_GLOB_RELA with RESI_GLOB_MAXI	
CAL NMCREL (SDERRO, “DIVE_FIXG”. FALSE.) No the divergence	of

the fixed loop of point on the geometry If the event is of type <ERRI_BOUC>, the statute of loop <BOUC> is automatically modified.

3.4.2 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, it	is necessary to use the routine nmcret.f instead of nmcrel.f
. For example CAL NMCRET (SDERRO ,” LDC	”, LDCCVG) deals with transforming into event the value of the return code

of the constitutive law. Processing 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 result of this processing. Events of type convergence and divergence the events of type convergence or divergence are examined on each level of loop
, in a dedicated routine called nmevcv.f. This routine modifies the state of the loop : It is supposed initially that the loop continues (state CONT, cf §3.2) One buckles on the events of the type <CONV_*> and <DIVE_*> in taking into account

3.4.3 of the possible activated

features; One result calculates the state for this level of loop: if all the events of divergence are false and all the events

3.4.3.1 of convergence are true, then this level

of loop is in the converged state; 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
1. must be converged. If it is the case,38
2. passes in the converged state (state CONV , cf § 3.2) One modifies the state of the current loop by calling the routine nmeceb.f;
3. The routine nmeceb.f transmits the state of the loop to the higher loops (avoids the NON-treated errors); Buckle Code Routine appealing Routine Residues RESI nmcvgr.f nmconv.f Newton NEWT nmcvgn.f nmnewt.f Not fixes FIXES nmcvgf.f nmtble.f
4. Time step INST nmcvgp.f op0070.f Computation CALC nmcvgc.f op0070.f Events of the errors type the events of type error are particularly sensitive and must be treated in a very rigorous way to avoid the false results. Initially 38
5. the event of type error was emitted by the means of the routines standards
6.: nmcrel.f and nmcret.f. It is pointed out that in the case of the events error, one systematically modifies the state of

the loop	in	mode WANDERS	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	,

3.4.3.2 the events of type error

are tested in the algorithm via the routine nmltev.f. If an event of type error is activated, a GOTO is immediately made in the program . General algorithm the process is standardized to the maximum, for each level of loop: Evaluating of convergence by call to the routines nmcvg*.f; Action following the situation of the loop by call to the routines nmact*.f; Total readings and initializations the routine op0070.f manages the total algorithm and cuts out in three parts: Reading and initializations of the data
; Buckle on time step; Postprocessings; Management of the total errors; Archivage; For the reading and the initialization of the data, one will refer in particular to the relative information with

4 the management of the SD (

§2). All the initializations made on this level will be valid for all
•computation. The principal routine of initialization is nminit.f .
•Here operations envisaged in this routine: Operations of initializations

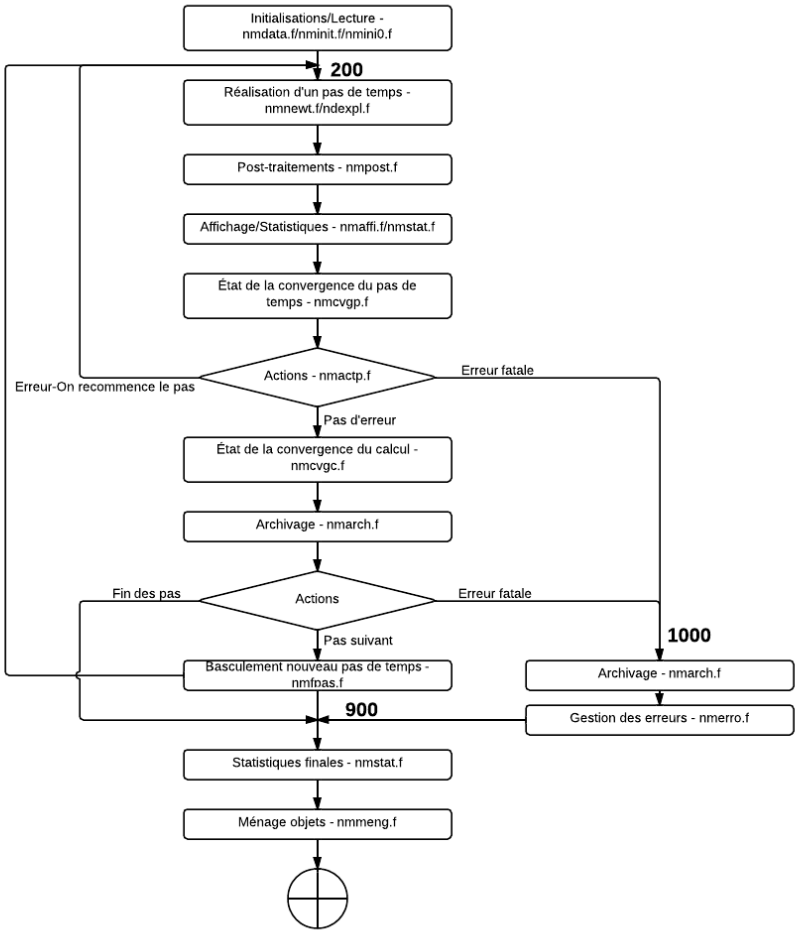
4.1 in nminit.f Routine Creation

of the SDTIME nmcrti.f Creation of the SDSTAT nmcrst.f Seized and checking of the coherence
•of the loading of contact nmdoct.f Creation
•of the profile of the matrix and
•the SDNUME nmnume.f
•Creation of the variable-hats
•nmchap.f Creation
of the vector of functionality activated FONACT nmfonc.f Creation of the SD for contact RESOCO cfmxsd.f Creation of the SD for6 unilateral connections RESOCU cucrsd.f Creation of the vectors in the variable-hats nmcrch.f Creation of the data structure control nmdopi.f Duplication NUME_DDL for SDNUME nmpro2

.f Préparation of card COMPOR (transformation	**in**
CHAM_ELEM_S) nmdoco.f	Creation
of the SDIETO nmetcr.f Reading	of
the initial state nmdoet.f Creation of SDDISC and SDOBSE diinit.f Initialization	of
the constant command variables nmcrvv.f Precalculation of	the MATR_ELEM
during computation pre nminmc.f	- computation
of the VECT_ELEM and VECT_ASSE constant during computation	nminvc.f
Creation of the SDCRIT nmcrcv.f Initialization	of computation
by substructuring nmlssv.f Creation of the SD for	loading
FORCE _SOL nmexso.f Computation of initial acceleration	accel
0.f Creation of the SDCONV nmcrcg.f Initialization	of
the constant SDIMPR nminim.f Precalculation of	the MATR_ASSE
during computation nminma.f Realization of an E initial observation	nmobsv.f
Creation of SD	RESULTAT
(EVOL_NOLI) nmnoli.f Creation	of
the array of the quantities (for _THM	WANDERS)
cetule.f Computation of the second members initial for	the multi-step
diagrams in dynamics nmihht.f Initializations SDTIME	and SDSTAT
nmrini.f the general algorithm of op0070 .f is summarized on the flow chart	(1

). Figure 1: Top-level flowchart	
of op0070.f Realization of time step	a One time step
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	
a prediction of Eulerian and corrections	of Newton
; The general algorithm of nmnewt.f is summarized	on
the flow chart (2). Figure 2: Top-level flowchart	of nmnewt.f
Initializations of the step Each time step is initialized by a series of	routines
: The routines nmeraz.f , nmevr	0.f deal

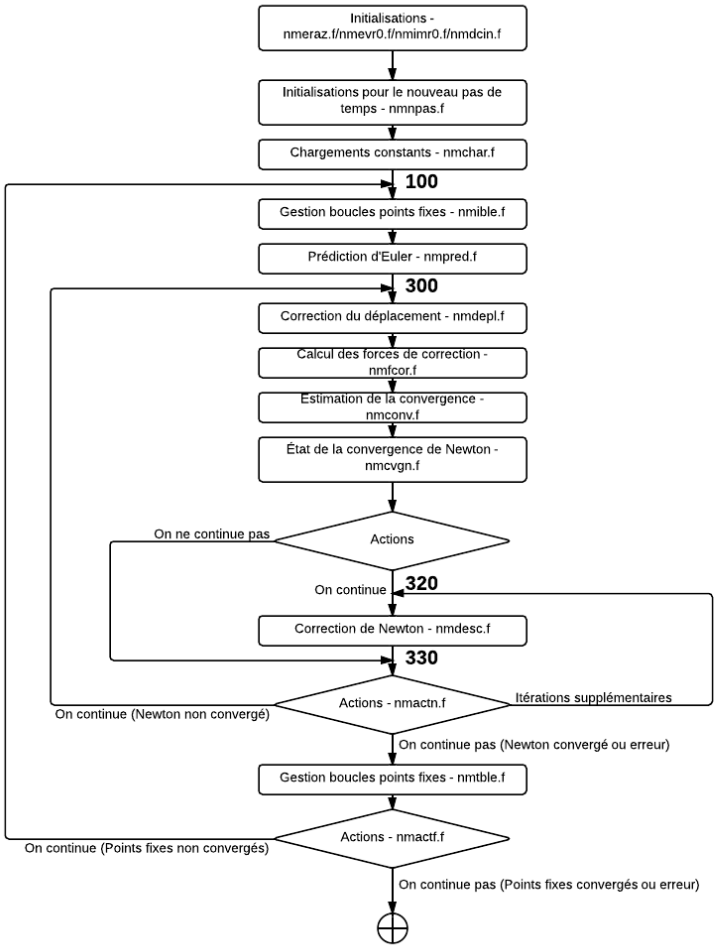
with initializing the management of the events; The routine nmdcin.f1 checks



that 1the cutting of time step `is correct`

4.2 (control amongst level

of cutting of time step to the other; The most important
•routine is the routine nmnpas.f. It carries out the following operations: The routines nmimr0.f and
•nminin.f initialize the display, in particular the table of convergence (what *makes it possible to vary the nature of this one of* time step
with the other); `The routine` nmvcre.f prepares the command variables2



for 2time step running; `One initializes`

4.2.1 to zero the displacement increment

cumulated since the beginning of time step (vector
•DEPDEL, `to see §` 2.4.2.6 `)` . Attention the initialization is special because of the taking
•into account `of large rotations` ; One initializes all the data for the dynamics, in particular different the coefficients (see § 2.4.3.13
) in ndnpas.f; One initializes different `information` for Newton-Krylov and the contact (routines
•cfinit.f, `mmapin.f and` nmnkft.f `);` Prediction of Eulerian the prediction of Eulerian is an estimate of the displacement increment by linearizing the problem compared to time. There exist
•several `methods` and several types of matrix usable. The computation itself
•is realized in the routine nmpred.f (and its - pretty girls), by means of the principles established in `the §5`. Update 19 the update of the fields is carried out in the routine nmdepl.f. This update consists in
•modifying the vectors solutions (displacements, velocities and accelerations), in taking into account possibly 31 : The linear `search`
•without control; The search for control with or without linear search `;` `The contact (` discrete `formulation`

4.2.2) or unilateral

connections; Here phases: Recalculation of the external forces, routine nmfext.f; Conversion of the increments of solution DEPSO1 and DEPSO2 (see § 2.4.2.6), resulting from the resolution of the linear system, towards the increments in displacement/velocity /acceleration (taken into account of the coefficients of change of diagram)48 via

4.2.3 the routine nmincr.f;

Computation searches linear and control, routines nmreli.f, nmpich.f and nmrepl.f; Update of the direction of descent (taken into account of the coefficients resulting from the linear search and control), routine nmpild.f;
- Modification of displacements by
- the contact and the unilateral η connections, the routine nmcoun.f; Actualization
- of the fields of solutions in nmmajc.f; Phase 6. consists in

updating
1.the fields “more” (DISPLEASED, VITPLU, ACCPLU) and the
2.cumulated fields (DEPDEL, VITDEL, ACCDEL), in taking into account 19 large rotations (updated of the quaternions) and of the damage to the nodes. Forces of correction Since displacements were modified (see § 4.2.3), it is necessary
3.to revalue the variable forces: either the forces outsides of the type "following
4.", or the forces interior E S and the reactions of contact/friction, or forces related to the dynamics (inertia, damping).
5.The group is carried out in the routine nmfcor.f. Estimate of convergence the estimate of the convergence
6.of Newton is one moment critical, which will condition
the accuracy 46 of computation. The group of the estimate is carried out in the routine nmconv.f. This routine deals with the computation of the residues (nmresi.f), of the estimate of their convergence (nmcore.f). It is thus in this routine that one will treat loop <RESI> on

4.2.4 the residues of equilibrium

. But one must also take into account the nombre of iterations 46 of Newton, the case of convergence of the contact (formulations discrete or continues), of the method of Borst or method IMPLEX. It is advisable to be very careful in the modification of this routine. Correction of Newton If the process did not converge, one carries out the computation of a correction

4.2.5 of Newton. One thus carries out

the computation of a linear system (see §5) in the routine nmdesc.f (like direction of descent). Buckle on the fixed points It exists three loops known as “of fixed point ” between the loop on time step and buckles it on the iterations of Newton. These loops are used for the contact: Buckle of fixed point on the geometry ; Buckle of fixed point on the thresholds of friction; Buckle of fixed point on the statutes of contact; These three loops are managed by the duet of routine nmible.f/nmtble.f, via the transition of the variable NIVEAU, which indicates in ***which type of loop of point one fixes is currently. Construction***

4.2.6 and resolution of

the systems an important part of the algorithm NON-flax é area consists in solving linear systems, which one builds in four times48: Computation and assembly of the loadings; Computation and assembly

4.2.7 of the second members

; Computation 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
1.. Systems to be currently solved,
2.there exist several different systems which are solved
3.in the operator op0070.f. They will always have
the following form (except for postprocessing, which uses the operators of search for eigenvalues): formulate (1) formula is the stiffness matrix or a linear combination

5 of matrixes and formula is the matrix

of Lagrange of Dirichlet. formula is the increment of solution of the unknown nodal values of the system and formula 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, velocities, accelerations, pressures, temperatures, etc the two second

5.1 members are also

nodal values. The detail of the systems is given in documentations of reference [R5.03 .01] and [R5.05.05]. The routine merimo.f For the computation of the tangent matrixes (options FULL_MECA, FULL_MECA_ELAS, RIGI_MECA_TANG, RIGI_MECA

$$\begin{bmatrix} K & B^T \\ B & 0 \end{bmatrix} \cdot \begin{Bmatrix} \delta r \\ \delta s \end{Bmatrix} = \begin{Bmatrix} L_r \\ L_s \end{Bmatrix} \quad \text{\_ELAS1}$$

$[K]$, RIGI_MECA_IMPLEX) and the internal forces (option RAPH_MECA), $[B]$ systematically by the routine merimo.f.
$\{\delta r\}$ prepares the inputted fields (many in this case), it is called $\{\delta s\}$ for the computation of the elementary vectors for the internal forces (see §5.5.2) but also for the computation of the tangent elementary matrixes of stiffness (see §5.3). Computation of the matrixes We consider here elementary matrixes (MEELEM) and assembled matrixes (MEASSE). Just like for the loading (§5.5.1), this computation uses a system in two times: Creation of a local list of the matrixes

5.2 to calculating and/or assembling

; Computation and/or assembly of the matrixes ; For the phase of creation of the list of the matrixes to calculating and/or assembling , there are several routines (contrary to the loadings which use only the routine nmchar.f). These routines are the following ones: Operations Routine Construction of the local list for the matrixes used in dynamics clarifies ndxprm.f Construction of the local list for the matrixes50 used in correction nmcoma.f Construction of the local list for the matrixes used48

5.3 in prediction nmprma.f

Construction of the local list for the matrixes used in postprocessing (oscillatory modes or buckling modes) nmflma.f50 of the local list for the constant matrixes
•during all ***computation*** nminmc.f For the phase of computation and/or assembly
•of the matrixes, one finds routines
similar to the case of the loadings. Operations Routine Addition of a matrix to assemble and/or calculate in the local list Initialization of the local list nmcmat.f nmcmat.f Shunting computation and/or assembly of the matrixes

nmxmat.f	Computation
of the matrixes nmcalm.f Assembly of the matrixes nmassm.f Computation and assembly of the stiffness matrixes	
nmrigi.f There is nevertheless a notable exception: the computation of the nonlinear	tangent
matrixes uses a direct call by nmrigi.f because that requires	additional
parameters (call to merimo.f, to see 5.2) which the other matrixes do not need. The type of matrix MERIGI thus	uses
nmrigi.f instead of nmcalm.f and nmassm.f. If one wishes to modify a matrix, it	is thus necessary

: To add its computation and/or its assembly to the good place (ndxprm.f, nmcoma.f, nmprma.f, nmflma.f or nminmc.f) according to the case,

to even add	a new
routine; To add elementary computation and/or the assembly in nmcalm.f and nmassm.f; The routines	nmxmat.f and nmcmat.f
of handling of the local list should not	be modified
. Computation of	matrix
resulting 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.f, nmcoma.f and nmprma.f. These three routines also 48 the display (computation option of the stiffness matrix) and the statistical (time spent in the operations). They have
the same general diagram: Computation and/or assembly
•of elementary matrixes (MEELEM) in assembled matrixes (MEASSE), (see § 5.3); Management of the parameters of reactualization of the matrixes (stiffness, mass
•, damping), routine nmchrm.f; Management of the type of the stiffness matrix
(name of the option), routine nmchoi.f ; Construction of the matrix resulting ***by linear combination*** from

5.4 the other matrixes. In dynamics, the various

contributions in the resulting matrix (mass matrixes, of damping and stiffness) by means of combine a multiplying coefficient depend on the time scheme used and time step, one recovers these coefficients via the routine of access NDYNRE (see § 2.4.3.13). It is the routine nmmatr.f; Factorization (or not) of the resulting matrix by call to the routine preres.f; Operations Routine Computation of the matrix assembled
•resulting for computation from initial acceleration nmprac.f Computation of the matrix assembled resulting 48
•the phase from prediction nmprma.f Computation of the matrix assembled resulting for the phase from correction nmcoma.f Management
•of the parameters of reactualization of the matrixes (stiffness, mass , damping
•) nmchrm.f Management of the type of the stiffness matrix (name of the option) nmchoi.f Choice of re-creation of classification nmrenu.f Computation of matrix resulting MATASS nmmatr.f Taken into account from the following loadings with their multiplying function ascoma.f Taken into account from the modification from the matrix resulting by the contact/friction (method DISCRETE) nmasfr.f Computation from the second member The computation 31 the second member is the
•most different part 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	bearing (§
5.5.3); Quantities related to the command variables (§5.5.4);	Various
quantities related to the contact/friction (not of details in this document); Contributions	
of inertia and damping for the dynamics (§5.5.6);	In the case
of the dynamics, for the diagrams multi	- not (complete

Newmark with MODI_EQUI=' OUI') one	moreover
will add the contributions of time step preceding. Computation of the loadings	There
are three modes of evaluating of the loadings and two phases. The modes are the following: Mode <ACCI> of the loadings	

5.5 for the evaluating of

initial acceleration in dynamics; Mode <FIXE> of the loadings for the evaluating of the fixed loadings
•not depending on the solution; Mode 50
•> of the loadings for the evaluating of the variable loadings depending50 on
•the solution (following loadings); There are also two phases. Either one is in correction51 one
•is in prediction. These phases are useful only51 very
•particular cases: modal damping, method IMPLEX and modal impedance. The principal
•routine which calculates the loadings is nmchar.f. In this routine51
the activated features (FONACT) and according to the mode/phase of computation, one will cause the computation of the elementary vectors (variable-hat VEELEM) and their

5.5.1 assembly in vectors

assembled (variable-hat VEASSE). For that, one uses a certain number of utility routines
•which manage local lists with nmchar.f. Operations Routine Addition of a loading
•to be assembled and/or calculate, with a possible option Initialization of the list of the loadings (initializations
• local lists of nmchar.f) nmcvec.f nmcvec.f Shunting computation and/or assembly of the loadings nmxvec.f Computation of the loadings
nmcalv.f Assembly of the loadings nmassv.f Certain loadings do not have a phase of elementary computation 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 computation and/or its assembly in nmchar.f according to the activated functionality; To add elementary computation and/or the assembly in nmcalv.f and nmassv.f; The routines nmxvec.f , nmcvec.f should not be modified. Computation of the quantities related to the internal forces The computation of the internal forces can be carried out of two ways: By integration

of the constitutive law	
, it is computation option RAPH_MECA; By re-use of the stresses calculated in addition, it is computation option 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 local variables and forced, and, often, to solve locally (at each Gauss point), a nonlinear system. As the computation of
•the stresses is also necessary during the evaluating
•of the tangent matrixes, the behavior is also integrated during the computation of the tangent
•matrixes (option FULL_MECA). In static, the phase of prediction calculates
the internal forces by option ***FORC_NODA. In dynamics ,***

5.5.2 the behavior systematically is integrated. Operations

Routine Computation of the elementary vectors for the internal forces (integration
•of the behavior) nmfint.f Assembly of the elementary vectors for
•the internal forces (integration of the behavior) nmaint.f The computation of option FORC_NODA is made by the mechanism of evaluating of the loadings (§5.5.1). Computation of the quantities related to the dualized limiting conditions the dualisation of the limiting conditions of Dirichlet (see [R3.01.01]) involves the

computation of two quantities of type vector assembled: The reactions of bearing, by the product of the matrix of the limiting conditions dualized with Lagrange corresponding to the degrees of freedom, it is the vector identified by CNDIRI; The value of the degrees of freedom imposed by dualisation formulates. A convergence, this quantity will be equal to the limiting conditions data formulates, the vector is identified by CNBUDI; Operations Routine Computation of

the elementary vectors	
for the reactions of bearing nmdiri.f Assembly of the elementary vectors for the reactions	of bearing
nmadir.f Computation and assembly of the quantities imposed by dualisation nmbudi.f Computation of the quantities	related to

the command variables The computation of these quantities is done by the mechanism of evaluating50

5.5.3 (§5.5.1), standard CNVCF0, CNVCF1 and CNVCPR. Computation of

the constant quantities during computation A certain number of vectors are constant during computation, they then use a mechanism

- similar to the loadings (§5.5.1), using the routines nmxvec.f, nmcvec.f, nmcalv.f and nmassv.f, except that one passes by the routine nminvc.f instead of $[B].\{\lambda\}$ nmchar.f. Operations Routine Computation and assembly
- of the constant vectors during computation nminvc.f Computation of $[B].\{u\}$ related to the dynamics – Inertia forces and of damping The computation $\{u^d\}$ quantities is done by the mechanism of evaluating

of	**the loadings**
(§ 5.5.1), standard CNDYNA. These quantities do not exist	in an elementary
state since they are result of a product stamps/vector	. Operations
Routine Computation of the dynamic quantities for the second	member

5.5.4 ndfdyn.f Computation of the inertia forces nminer.f Computation of the damping forces

nmhyst.f According to the time schemes, it is necessary to use50 multiplying coefficients

5.5.5 in front of each one of these quantities, one recovers these

coefficients via the routine of access NDYNRE (see § 2.4.3.13). Second resulting members It remains about it more than to build50 the various second members by linear combination of all the quantities detailed previously . The principal routines

of construction	**of**
these second members are seven: Operations Routine	Computation

5.5.6 of the second member for the correction nmassc.f Computation of the second member for the prediction

– Option DEPL_CALCULE or EXTRAPOLATES nmassd.f Computation of the second member for initial 50 nmassi.f Computation of the second member for the prediction nmassp.f Computation of the second member for the prediction – static Case nsassp.f Computation

of the second	**member**

for the prediction – implicit dynamic Case ndassp.f	Computation of
the second member for the prediction	-
dynamic Case clarifies nmassx.f All	these routines

rest on the same principles: Recovery of necessary quantities, already assembled or locally calculated in these routines, via variable-hat VEASSE (see the §5.5 .1 with §5.5.6) 31

5.5.7 of the various multiplying

coefficients. In dynamics, according to the time schemes, 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 §	**2.4.3.13**
); Construction of one or two second	members by
linear combination (use of the routines standards vtaxpy.f). There are two second members	
in the case of control; To improve legibility	
, one uses utility routines standardized	
to gather the most frequent cases. Operations Routine	Computation
of the components of the second member vector for the loadings of the Dirichlet type	,
fixed during time step, for the normal case and the case controlled nmasdi.f	Computation

of the components of the second member vector for the loadings
•of the Neumann type, fixed during time step, for the normal case and the case controlled nmasfi.f Computation of the components of the second member vector for50 51of the Neumann
•type, variables (follower) during time step, for the normal case (not of controlled case) nmasva.f Computation of the components of the second member vector for the specific loadings of the dynamics, variables (follower) during time step, for the case normal (not of controlled case 31 Computation
•of the components of the second member vector for the specific loadings of the dynamics, computation of initial acceleration, variables (follower) during time step
, for the normal case (not of controlled case) nmacva.f Resolution of the system the resolution is made via the intermediary of the routine

nmresd.f,	**which takes**
as starter the assembled resulting matrix (see § 5.4) and the second suitable members (two second members if there is control). Operations	Routine Resolution
of the system nmresd.f Resolution of the system on the physical degrees of freedom (call to resoud.f) nmreso.f Resolution of the system on the generalized	
degrees of freedom nmresg.f the routine of resolution will thus turn over one or two fields solutions DEPSO1 and DEPSO2, stored in variable-hat SOLALG. The SD credits	
STRUCT the idea are to use a system of called generic objects STRUCT to contain information in op0070.f. For time, and reasons of performances related	to the use
massive of JEVEUX objects, the deployment of the STRUCT was restricted with printing (SDIMPR, to see § 2.4.3.1). There are three purposes: To rationalize the access to the internal SD with op0070.f and to moderate their multiplication	

5.6 ; To have a generic

system for all the SD; To try to facilitate the development and maintainability; A STRUCT is built on 49 precepts: Single routines of access and handling; Protection and

encapsulation	of
the data; Possibility	of creating
generic STRUCT for the frequent operations; A STRUCT is a succession	of JEVEUX objects
containing of the information given by the user in	the command

or created for the management of the algorithm. A STRUCT consists of JEVEUX objects : Two JEVEUX objects containing some information

6 necessary ; A JEVEUX object

of type V V K24 containing the name of all the parameters ; A JEVEUX object of type V V I containing the parameters of the <booléen> type; A JEVEUX object of type V V I containing the parameters of the <entier> type; A JEVEUX object of type V V R containing the parameters of the <réel> type21
of type V V K24
•containing the parameters of the <chaîne type >; A JEVEUX object of type V V K24 containing
•the param è very <objet> type As only
•names are used, including for the parameters of the type <objet>,
one can allow a certain recursion, i.e.
•a parameter of the <objet> type can contain
•a STRUCT itself. What makes it possible
•to have routines of handling (destruction, copy, etc), without the difficulties
(dangers) of making the recursive one in F77. The counterpart is that these routines of handling will be incomplete with the usual meaning the direct-object: they
will handle only the JEVEUX objects of
•the STRUCT, the name of the parameters, the values of the “scalar
•” parameters and the names of the parameters . For example, the copy of a STRUCT
•in another STRUCT is a poor copy: Recopy name of the parameters
•; Recopy scalar parameters : Boolean, whole, real, character strings
•; Recopy names of the parameters <objet>; Idem for the destruction. Even
•if these two types of parameters <chaîne> and <objet> are basically
•character strings (K24), they are not the same ones. The <objet> types are names
d'<objet>, the <chaîne> types are cha î born. A STRUCT has the following life cycle: Creation of the generic JEVEUX objects of the STRUCT (initialization of the name of the parameters); Reading of the user news ; Initialization of the values of the parameters; To create a STRUCT a STRUCT is created by a series of three successive routines
: obcrea.f, routines obcrxx.f and obcrcr.f. The developer creates a STRUCT of a preset type by call to the routine obcrea.f. Principle: Each type of STRUCT has its clean obcrxx.f (with xx variable from 00 to 99). Shaft of construction: obcrea.f calls obcrxx.f (with xx
following the type of the STRUCT) ; obcrxx.f contains the number and the name of the parameters
•(in DATED), it calls then
•obcrcr.f; obcrcr.f creates the list of the parameters with their name (creation of
•the idoines ***JEVEUX objects***) Example: CAL OBCREA
(“DISPLAY”, SDIMPR)

will create the STRUCT of the type “DISPLAY” and name SDIMPR. To create a new STRUCT to create a new type of STRUCT, it is necessary: To impact obcrea.f to add the IF towards new the obcrxx.f; To create new

the obcrxx.f with the list of the parameters
•and their name; Rules: The creation of a STRUCT must be done exclusively in the routine
•nmini0.f; A STRUCT can exist only
•if the functionality is activated (control

6.1 , contact). The test

of the functionality can be done only on the absence or the presence of a keyword factor (except for the dynamics : one tests the name of the command); To see the user data Not possible

generics: each STRUCT has its own process . Rules : All the supervisor calls

(getxxx) must
•be made during this phase . The getxxx are prohibited in the rest
•of the operator (except detecting features during the creation of the STRUCT); For
•reasons of obstruction memory, the user data implying much information

(a list of meshes for example) are an exception to the rule of separation Creation /Lecture

6.2 /Initialisation. The JEVEUX objects

necessary to store this information
•can be created in this phase instead of the phase of initialization
•. ; To initialize a STRUCT Not possible generics: each
STRUCT has
•its own process . Rules : The JEVEUX objects contained in the parameters of
•the type STRUCT must be named according to the general rule of Code_Aster: prefixes by the name of the routine which makes the operation. To reach the values of the parameters Two series of five routines for reading or writing a parameter: To recover

6.3 a parameter: obgeti.f

, obgetb.f, obgetr.f, obgetk.f, obgeto.f (respectively <entier>, <booléen>,

<réel>, <chaine
•> and <objet>); To write a parameter: obseti.f, obsetb.f, obsetr.f, obsetk.f, obseto.f (respectively <entier>, <booléen>, <réel>, <chaine> and <objet>); There is a control of the type at the time of the access. The routine
•obgett.f turns over the type of the SD. (and not the type of the parameter). Examples: CAL OBGETI (SDDYNA, “NBRE_ONDE_PLANE”, NCHOND); CAL OBSETR (SDPILO, “COEF_MULT_PREC ”, COEMUL); Rules: The name and the number of parameters cannot be changed subsequently to the creation of the STRUCT (no routine of direct access to these parameters); Generic STRUCT

6.4 : the SDLIST the SDLIST

is a list of STRUCT. It is itself a STRUCT. It contains

a list
•of STRUCT and can activate or désactiver/un STRUCT in this list. To locate a STRUCT in the SDLIST, this STRUCT must contain a parameter of the type chains, single, given during

6.5 creation. One can reach

the STRUCT of the SDLIST, either by the index in the list, or by his
•parameter of access. One declares this parameter at the time of the creation of the SDLIST. Description Routine Creation of an automatic SDLIST oblcre.f
•Generation of a name of STRUCT in the SDLIST oblgen.f Says if the STRUCT in the SDLIST is active or not (access by index) oblgai.f Deactivation of
all the STRUCT in the SDLIST oblraz.f Turns over the index of a STRUCT in the SDLIST (access by parameter) oblgip.f (Dice) - activation
of
•a STRUCT in the SDLIST (access by index)
•oblsai.f (Dice) - activation of a STRUCT in

the SDLIST (access
•by parameter) oblsap.f Recovery of the STRUCT in the SDLIST (access by index) oblgoi.f Recovery of the STRUCT in the SDLIST (access by parameter

6.6) oblgop.f Put of one STRUCT in

the SDLIST (access by index) oblsop.f Examples: Creation of STRUCT SDLIST: CAL OBLCRE (SLCOLO , “TABLEAU _COLONNE”, “TYPE_COLONNE”, NBCOLO); STRUCT SDLIST

consists of nbcolo STRUCT of the type TABLEAU _COLONNE , each STRUCT is identified by its parameter TYPE_COLONNE; Deactivation of all the STRUCT : CAL OBLRAZ (SLCOLO); Activation of a STRUCT: CAL

OBLSAP (SLCOLO, “ITER_NUMÉRIQUE”, .TRUE.); Deactivation of a STRUCT: CAL OBLSAP (SLCOLO,” ITER_NUMÉRIQUE

”, .FALSE.);	

1.
•
•
2.
3.
4.