Mise en œuvre de STAT_NON_LINE et de DYNA_NON_LINE

Résumé :

On décrit ici la mise en œuvre informatique de l'algorithme de résolution des problèmes quasi-statiques non linéaires et dynamique non-linéaires. Les documents décrivant en détail ces algorithmes sont supposés connus ([R5.03.01] et [R5.05.05]), on n'en rappellera que les principales étapes. On trouvera dans ce document un rappel des notations, l'organigramme simplifié de la routine op0070, permettant de discerner les principales articulations logiques de l'opérateur STAT_NON_LINE et DYNA_NON_LINE de *Code_Aster*, son arbre d'appel, une description des objets informatiques et des principales routines, et quelques pièges à éviter lors du développement dans cet opérateur.

Sommaire

1 Introduction

L'opérateur op0070.f est constitué de plusieurs parties :

1) Lecture des données ;
2) Initialisation des données ;
3) Construction et résolution d'un système linéaire
4) Mise-à-jour des champs ;
5) Archivage des résultats, post-traitements ;

Le tout étant encapsulé dans trois niveaux de boucle : pas de temps, boucles de point fixe (pour le contact) et itérations de Newton avec une gestion de type évènementielle.

Ce document propose la description de ces différentes zones de l'opérateur, en insistant particulièrement sur le coté structurant des SD (§ 6), des évènements (§ 31) et sur l'algorithme général, boucle par boucle (§ 35).

A noter que pour aider au débugage, il est possible de sauvegarder les champs de déplacements dans un fichier MED, à chaque itération de Newton. Pour cela, il faut surcharger la routine `dbgcha.f` en changeant le booléen `DBG=.FALSE .` en `DBG=.FALSE.` Les champs de déplacements seront alors sauvegardés dans un fichier au format MED, sur l'unité logique 80 (ne pas oublier d'ajouter un fichier sur cette unité en sortie dans astk)

2 Structures de données

Une SD passe par plusieurs phases :
- Création de la SD;
- Lecture des informations de l'utilisateur et stockage dans la SD ;
- Initialisations de la SD;
- Destruction de la SD ;

Les SD ne passent pas nécessairement par toutes ces phases.

2.1 Les types de SD

On introduit quatre niveaux de SD :
1.Les concepts provenant d'autres commandes ou produits par la commande elle-même ;
2.Les SD « bas-niveau » sont des variables Fortran (des tableaux de type simple), sans routine d'accès ;
3.Les SD « moyen-niveau » sont des variables Fortran (des tableaux de type simple) disposant de routines d'accès. Dans cette catégorie, on trouvera le vecteur des fonctionnalités activées FONACT et les variables-chapeaux qui sont des tableaux de chaînes de caractères contenant des objets plus complexes (matrices, vecteurs, etc.) ;
4.Les SD « haut-niveau » sont des objets JEVEUX spécifiques à l'opérateur (avec routines d'accès) ou des SD standards de Code_Aster qui resteront internes à l'opérateur ;
Les SD bas-niveau, sont, en général, un reliquat des anciennes versions d'op0070.f . Leur disparition totale et leur résorption au profit des SD de haut-niveau reste à faire.
NB : Dans la description des SD, la colonne # est un numéro d'ordre permettant de les repérer.

2.1.1 SD de type concept

Les concepts proviennent d'un autre opérateur ou sont construits par l'opérateur pour être utilisés dans d'autres commandes. Elles sont construites exclusivement sur des SD standards dans *Code_Aster* (champs, numérotations, etc.). Leur contenu est décrit dans les documentations de type D4. Voici la liste exhaustive de ces SD dans op0070.f :

#	Description	Type	Nom généralement utilisé
1	Maillage	sd_mailla	M AILLA
2	Modèle	sd_modele	MODELE
3	Résultat	sd_resultat	RESULT
4	Caractéristiques élémentaires	sd_cara_elem	CARELE
6	Chargements	sd_l_charges	LISCHA
7	Champ de matériaux et variables de commande	sd_cham_mater	MATE
8	Définition des contacts	sd_contact	DEFICO
9	Définition des liaisons unilatérales	sd_contact	DEFICU

2.1.2 SD de bas-niveau

Ces SD sont des tableaux, leur longueur est donc importante. Pour éviter les erreurs (mauvaise recopie des longueurs), ils sont pré-dimensionnés dans op0070.f, via un PARAMETER, la taille est alors (*) dans les routines appelées.

#	Description	Type	Nom
11	Paramètres de résolution	REAL(ZPMET)	PARMET
12	Critères de convergence	REAL(ZPCRI)	PARCRI

13	Informations sur la convergence du calcul	REAL(ZCONV)	CONV
14	Paramètres du critère de convergence en contrainte	REAL(ZPCON)	PARCON
15	Méthode de résolution	CHAR*16(ZNMETH)	METHOD

2.1.3 SD de niveau moyen

Les SD de niveau moyen sont intermédiaires entre les SD « bas-niveau » et les SD « haut-niveau ». En effet, ces variables sont des objets Fortran simples mais disposent de routines d'accès. Elles sont donc « bas-niveau » du point de vue du stockage, mais leur accès se fait par des routines spécifiques, tout comme les SD de « haut-niveau ».

#	Description	Type	Nom
10	Fonctionnalités activées	INTEGER(100)	FONACT
16	Variable-chapeau – Matrices élémentaires	CHAR*19 (ZMEELM)	MEELEM
17	Variable-chapeau – Matrices assemblées	CHAR*19 (ZMEASS)	MEASSE
18	Variable-chapeau – Vecteurs élémentaires	CHAR*19 (ZVEELM)	VEELEM
19	Variable-chapeau – Vecteurs assemblés	CHAR*19 (ZVEASS)	VEASSE
20	Variable-chapeau – Champs solutions	CHAR*19 (ZSOLAL)	SOLALG
21	Variable-chapeau – Champs solutions incrémentales	CHAR*19 (ZVALIN)	VALINC

2.1.4 SD de haut -niveau

Toutes ces SD sont construites sur des objets JEVEUX et leur accès se fait par l'intermédiaire de routines dédiées (encapsulation des données) quand les SD sont spécifiques . On ne revient pas sur les neuf premières, qui sont des concepts contenus dans les mot-clefs de l'opérateur ou produits par l'opérateur (§ 6).

#	Description	Type	Nom
5	Carte de comportement	sd_carte	COMPOR
22	Solveur	sd_solveur	SOLVEU
23	Matrice de préconditionnement	sd_matr_asse	MAPREC
24	Matrice de résolution assemblée	sd_matr_asse	MATASS
25	Carte des critères de convergence pour le comportement	sd_carte	CARCRI
26	Carte des variables de commande de référence	sd_carte	COMREF
27	Carte des codes-retours erreur du comportement	sd_carte	CODERE
28	Numérotation	sd_nume_ddl	NUMDDL
29	Numérotation fixe (utilisée pour le contact méthode continue)	sd_nume_ddl	NUMFIX
30	Gestion des impressions	spécifique à op0070.f	SDIMPR
31	Gestion des mesures de temps	spécifique à op0070.f	SDTIME
32	Gestion des erreurs de l'algorithme	spécifique à op0070.f	SDERRO
33	Gestion de l'archivage et de l'état initial (IN et OUT)	spécifique à op0070.f	SDIETO
34	Gestion des statistiques	spécifique à op0070.f	SDSTAT
35	Gestion de la convergence	spécifique à op0070.f	SDCONV
36	Gestion du SUIVI_DDL	spécifique à op0070.f	SDSUIV
37	Gestion des critères de qualité	spécifique à op0070.f	SDCRIQ
38	Gestion du pilotage	spécifique à op0070.f	SDPILO

39	Gestion des numérotations	spécifique à op0070.f	SDNUME
40	Gestion de la dynamique	spécifique à op0070.f	SDDYNA
41	Gestion de la discrétisation temporelle	spécifique à op0070.f	SDDISC
42	Gestion des critères de convergence	spécifique à op0070.f	SDCRIT
43	Gestion de l' OBSERVATION	spécifique à op0070.f	SDOBSE
44	Gestion du post-traitement (MODE_VIBR et CRIT_STAB)	spécifique à op0070.f	SDPOST
45	Gestion de l'énergie	spécifique à op0070.f	SDENER
46	Résolution du CONTACT	spécifique à op0070.f	RESOCO
47	Résolution de LIAISON_UNIL	spécifique à op0070.f	RESOCU

Le reste du document va décrire essentiellement le contenu, l'utilisation et l'accès aux SD spécifiques à op0070.f.

2.2 Création des SD

Le tableau suivant résume l'origine de la création des SD.

#	Description	Type	Création
1	Maillage	sd_mailla	vient du .comm
2	Modèle	sd_modele	vient du .comm
3	Résultat	sd_resultat	nmnoli.f
4	Caractéristiques élémentaires	sd_cara_elem	vient du .comm
5	Carte de comportement	sd_carte	nmdocc.f
6	Chargements	sd_l_charges	nmdoch.f
7	Matériau codé	sd_cham_mater	vient du .comm
8	Définition des contacts	sd_contact	vient du .comm
9	Définition des liaisons unilatérales	sd_contact	vient du .comm
10	Fonctionnalités activées	INTEGER(100)	op0070.f
11	Paramètres de résolution	REAL(ZPMET)	op0070.f
12	Critères de convergence	REAL(ZPCRI)	op0070.f
13	Informations sur la convergence du calcul	REAL(ZCONV)	op0070.f
14	Paramètres du critère de convergence en contrainte	REAL(ZPCON)	op0070.f
15	Méthode de résolution	CHAR*16 (ZNMETH)	op0070.f
16	Variable-chapeau – Matrices élémentaires	CHAR*19 (ZMEELM)	op0070.f
17	Variable-chapeau – Matrices assemblées	CHAR*19 (ZMEASS)	op0070.f
18	Variable-chapeau – Vecteurs élémentaires	CHAR*19 (ZVEELM)	op0070.f
19	Variable-chapeau – Vecteurs assemblés	CHAR*19 (ZVEASS)	op0070.f
20	Variable-chapeau – Champs solutions	CHAR*19 (ZSOLAL)	op0070.f

21	Variable-chapeau – Champs solutions incrémentales	CHAR*19 (ZVALIN)	op0070.f
22	Solveur	sd_solveur	nmlect.f
23	Matrice de préconditionnement	sd_matr_asse	pendant l'algo
24	Matrice de résolution assemblée	sd_matr_asse	pendant l'algo
25	Carte des critères de convergence pour le comportement	sd_carte	nmdocr.f
26	Carte des variables de commande de référence	sd_carte	nmvcre.f
27	Carte des codes-retours erreur du comportement	sd_carte	pendant l'algo
28	Numérotation	sd_nume_ddl	nmprof.f
29	Numérotation fixe (utilisée pour le contact méthode continue)	sd_nume_ddl	nmpro2.f
30	Gestion des impressions	spécifique	nminim.f
31	Gestion des mesures de temps	spécifique	nmcrti.f
32	Gestion des erreurs de l'algorithme	spécifique	nmcrga.f
33	Gestion de l'archivage et de l'état initial (IN et OUT)	spécifique	nmetcr.f ntetcr.f
34	Gestion des statistiques	spécifique	nmcrst.f
35	Gestion de la convergence	spécifique	nmcrcg.f
36	Gestion du SUIVI_DDL	spécifique	nmcrdd.f
37	Gestion des critères de qualité	spécifique	nmcrer.f
38	Gestion du pilotage	spécifique	nmdopi.f
39	Gestion des numérotations	spécifique	nmnume.f
40	Gestion de la dynamique	spécifique	ndcrdy.f
41	Gestion de la discrétisation temporelle	spécifique	nmcrli.f
42	Gestion des critères de convergence	spécifique	nmcrcv.f ntcrcv.f
43	Gestion de l' OBSERVATION	spécifique	nmcrob.f
44	Gestion du post-traitement (MODE_VIBR et CRIT_STAB)	spécifique	nmdopo.f
45	Gestion de l'énergie	spécifique	eninit.f
46	Résolution du CONTACT	spécifique	cfmxsd.f
47	Résolution de LIAISON_UNIL	spécifique	cfmxsd.f

2.3 Lecture des données utilisateurs

La lecture des données utilisateurs se fait majoritairement sous la routine nmdata.f, cette routine lit les données utilisateurs et crée éventuellement les SD nécessaires. La routine nmdome.f lit les caractéristiques données par les mots-clefs MODELE, CHAM_MATER, CARA_ELEM et EXCIT en traitant également le cas des reprises de calcul (reuse) qui utilise les informations stockées dans la SD résultat. Le tableau ci-dessous rassemble toutes les SD qui vont lire directement des informations dans le fichier de commande (et donc utilisant les routines getxxx de la communication entre le Fortran et le superviseur Python).

#	Description	Mots-clefs	Routine de lecture
2	Modèle	MODELE	nmdome.f
3	Résultat	*concept créé*	nmnoli.f
4	Caractéristiques élémentaires	CARA_ELEM	nmdome.f
5	Carte de comportement	COMPORTEMENT	nmdocc.f

6	Chargements	EXCIT	nmdome.f
7	Matériau codé	CHAM_MATER	nmdome.f
8	Définition des contacts	CONTACT	cfmxsd.f
9	Définition des liaisons unilatérales	CONTACT	cfmxsd.f
11	Paramètres de résolution	METHODE RECH_LINEAIRE	nmdomt.f
12	Critères de convergence	CONVERGENCE	nmdocn.f
14	Paramètres du critère de convergence en contrainte	CONVERGENCE	nmdocn.f
15	Méthode de résolution	METHODE RECH_LINEAIRE	nmdomt.f
22	Solveur	SOLVEUR	cresol.f
25	Carte des critères de convergence pour le comportement	COMPORTEMENT	nmdocr.f
30	Gestion des impressions	IMPRESSION	nmdoim.f
33	Gestion de l'archivage et de l'état initial (IN et OUT)	ARCHIVAGE ETAT_INIT	nmetcr.f nmcrar.f nmdoet.f
36	Gestion du SUIVI_DDL	SUIVI_DDL	nmcrdd.f
37	Gestion des critères de qualité	CRIT_QUALITE	nmcrer.f
38	Gestion du pilotage	PILOTAGE	nmdopi.f
40	Gestion de la dynamique	SCHEMA_TEMPS MASS_DIAG PROJ_MODAL MODE_STAT AMOR_MODAL	ndlect.f
41	Gestion de la discrétisation temporelle	DISCRETISATION	nmcrsu.f
43	Gestion de l' OBSERVATION	OBSERVATION	nmcrob.f
44	Gestion du post-traitement (MODE_VIBR et CRIT_STAB)	CRIT_STAB MODE_VIBR	nmdopo.f
45	Gestion de l'énergie	ENERGIE	eninit.f
46	Résolution du CONTACT	CONTACT	cfmxsd.f
47	Résolution de LIAISON_UNIL	CONTACT	cfmxsd.f

2.4 Description des SD

2.4.1 SD de bas-niveau

2.4.1.1 Paramètres de résolution – PARMET

Les paramètres de résolution sont contenus dans le vecteur PARMET.

	SD PARMET
1	Valeur de REAC_INCR
2	Valeur de REAC_ITER
3	Valeur de PAS_MINI_ELAS
4	Valeur de REAC_ITER_ ELAS
5	Valeur de ITER_ LINE_MAXI
6	Valeur de R ESI_LINE_RELA

7	Valeur de RHO_MIN
8	Valeur de RHO_MAX
9	Valeur de RHO_EXCL

2.4.1.2 Critères de convergence – PARCRI

Les critères de convergence sont contenus dans le vecteur PARCRI.

	SD PAR CRI
1	Valeur de ITER_GLOB_MAXI
2	Valeur de RESI_GLOB_RELA
3	Valeur de RESI_GLOB_MAXI
4	Valeur de ARRET
5	Valeur de ITER_GLOB_ELAS
6	Valeur de REAC_ REFE_RELA
7	Valeur de CONVERGENCE/TYPE
8	Valeur de PLATEAU_ITER
9	Valeur de PLATEAU_RELA
1 0	Valeur de RESI_COMP_RELA

2.4.1.3 Informations sur la convergence du calcul – CONV

Les informations sur la convergence sont contenus d ans le vecteur CONV.

	SD CONV
1	Nombre d'itérations de recherche linéaire
2	Coefficient de recherche linéaire
3	Valeur de RESI_GLOB_RELA
4	Valeur de RESI_GLOB_MAXI

2.4.1.4 Paramètres du critère de convergence en contrainte – PARCON

Les paramètres du critère de convergence en contrainte (RESI_REFE_RELA) sont contenus dans le vecteur PARCON.

	SD PARCON
1	Valeur de SIGM_REFE
2	Valeur de EPSI_REFE
3	Valeur de FLUX_THER_REFE
4	Valeur de FLUX_HYD1_REFE
5	Valeur de FLUX_HYD2_REFE
6	Valeur de VARI_REFE
7	Valeur de FORC_REFE(1)
8	Valeur de FORC_REFE(2)
9	Valeur de DEPL_REFE

10	Valeur de LAGR_REFE

2.4.1.5 Méthode de résolution – METHOD

Les méthodes de résolution sont contenues dans le vecteur METHODE .

	SD METHOD
1	Méthode de résolution (Newton, IMPLEX, Newton-Krylov)
2	Type de matrice en correction
5	Type de prédiction
6	Nom de la SD RESULTAT pour PREDICTION='DEPL_CALCULE'
7	Méthode de recherche linéaire

2.4.2 SD de niveau moyen

2.4.2.1 Fonctionnalités activées – FONACT

Cette SD est très utile car elle permet de savoir quelles fonctionnalités sont actives dans l'algorithme à n'importe quel moment. L'idée principale de cette SD est un DISMOI très rustique permettant de répondre à une question simple sur les fonctionnalités actives dans op0070.f. Par exemple :
•Il y a du contact : ISFONC(FONACT,'CONTACT') est vrai ;
•La recherche linéaire est activée : ISFONC(FONACT,'RECH_LINEAIRE') est vrai ;
On ne fera pas la liste des fonctionnalités interrogeables par ce mécanisme dans le présent document car ce vecteur est souvent modifié, il suffit de lire la routine isfonc.f. Même si cet objet est bas-niveau (simple tableaux d'INTEGER), on doit y accéder par l'intermédiaire de trois routines uniquement :

Opération sur le vecteur des fonctionnalités activées – FONACT	Routine
Préparation des fonctionnalités activées	nmfonc.f
Interrogation d'une fonctionnalité activée	isfonc.f
Règles d'exclusion de certaines fonctionnalités	exfonc.f

Il est ***impératif*** de modifier le vecteur FONACT uniquement dans la routine nmfonc.f (appelée dans nminit.f) et de toujours prévoir la question correspondant à cette fonctionnalité dans isfonc.f. De même, c'est ce vecteur qu'on utilisera de manière prioritaire pour tester les compatibilités entre certaines fonctionnalités (routine exfonc.f, appelée par nmfonc.f).

2.4.2.2 Variable-chapeau – Matrices élémentaires MEELEM

Cette variable-chapeau contient le nom de toutes les matrices élémentaires utilisables dans op0070.f. Il s'agit donc d'une liste de SD de type sd_matr_elem (sd_resu_elem). Le code de repérage dans les variables chapeaux MEELEM/MEASSE est le même s'il s'agit des mêmes objets.

Variable-chapeau – Matrices élémentaires MEELEM	Code de repérage
Matrices élémentaires de rigidité	MERIGI
Matrices élémentaires des conditions limites de Dirichlet dualisées	MEDIRI
Matrices élémentaires de masse	MEMASS
Matrices élémentaires d'amortissement	MEAMOR
Matrices élémentaires des chargements suiveurs	MESUIV
Matrices élémentaires des sous-structures (macro-éléments)	MESSTR
Matrices élémentaires de rigidité géométrique	MEGEOM

Matrices élémentaires de contact (méthode CONTINUE et XFEM)	MEELTC
Matrices élémentaires de frottement (méthode CONTINUE et XFEM)	MEELTF

2.4.2.3 Variable-chapeau – Matrices assemblées MEASSE

Cette variable-chapeau contient le nom de toutes les matrices assemblées utilisables dans op0070.f. Il s'agit donc d'une liste de SD de type sd_matr_asse. Le code de repérage dans les variables chapeaux MEELEM/MEASSE est le même s'il s'agit des mêmes objets.

Variable-chapeau – Matrices assemblées MEASSE	Code de repérage
Matrice assemblée de rigidité	MERIGI
Matrice assemblée de masse	MEMASS
Matrice assemblée d'amortissement	MEAMOR
Matrice assemblée des sous-structures (macro-éléments)	MESSTR

2.4.2.4 Variable-chapeau – Vecteurs élémentaires VEELEM

Cette variable-chapeau contient le nom de tout les vecteurs élémentaires utilisables dans op0070.f. Il s'agit donc d'une liste de SD de type sd_vect_elem (sd_resu_elem). Le code de repérage dans les variables chapeaux VEELEM/VEASSE est le même s'il s'agit des mêmes objets.

Variable-chapeau – Vecteurs élémentaires VEELEM	Code de repérage
Vecteur élémentaire des forces internes	CNFINT
Vecteur élémentaire des réactions d'appui pour les conditions limites de Dirichlet dualisées	CNDIRI
Vecteur élémentaire des conditions limites de Dirichlet dualisées	CNBUDI
Vecteur élémentaire des forces nodales	CNFNOD
Vecteur élémentaire des conditions limites de Dirichlet données	CNDIDO
Vecteur élémentaire des conditions limites de Dirichlet pilotées	CNDIPI
Vecteur élémentaire des conditions limites de Neumann données	CNFEDO
Vecteur élémentaire des conditions limites de Neumann pilotées	CNFEPI
Vecteur élémentaire des conditions limites de type Laplace	CNLAPL
Vecteur élémentaire des conditions limites de type onde plane	CNONDP
Vecteur élémentaire des conditions limites de Neumann suiveuses et données	CNFSDO
Vecteur élémentaire des conditions limites de type impédance (en prédiction)	CNIMPP
Vecteur élémentaire des conditions limites de Dirichlet différentielles	CNDIDI
Vecteur élémentaire des forces sur les sous-structures	CNSSTF
Vecteur élémentaire des forces de contact (méthode CONTINUE et XFEM)	CNELTC
Vecteur élémentaire des forces de frottement (méthode CONTINUE et XFEM)	CNELTF
Vecteur élémentaire des forces de référence (RESI_REFE_RELA)	CNREFE
Vecteur élémentaire des variables de commande pour l'état initial	CNVCF1
Vecteur élémentaire des variables de commande pour la convergence	CNVCF0
Vecteur élémentaire des conditions limites de type impédance (en correction)	CNIMPC

2.4.2.5 Variable-chapeau – Vecteurs assemblés VEASSE

Cette variable-chapeau contient le nom de tout les vecteurs assemblés utilisables dans op0070.f. Il s'agit donc d'une liste de SD de type sd_cham_no. Ces vecteurs sont essentiellement utilisés dans la construction des seconds membres et dans l'évaluation de la convergence. Le code de repérage dans les variables chapeaux VEELEM/VEASSE est le même s'il s'agit des mêmes objets.

Variable-chapeau – Vecteurs assemblés VE ASSE	**Code de repérage**
Vecteur assemblé des forces internes	CNFINT
Vecteur assemblé des réactions d'appui pour les conditions limites de Dirichlet dualisées	CNDIRI
Vecteur assemblé des conditions limites de Dirichlet dualisées	CNBUDI
Vecteur assemblé des forces nodales	CNFNOD
Vecteur assemblé des conditions limites de Dirichlet données	CNDIDO
Vecteur assemblé des conditions limites de Dirichlet pilotées	CNDIPI
Vecteur assemblé des conditions limites de Neumann données	CNFEDO
Vecteur assemblé des conditions limites de Neumann pilotées	CNFEPI
Vecteur assemblé des conditions limites de type Laplace	CNLAPL
Vecteur assemblé des conditions limites de type onde plane	CNONDP
Vecteur assemblé des conditions limites de Neumann suiveuses et données	CNFSDO
Vecteur assemblé des conditions limites de type impédance (en prédiction)	CNIMPP
Vecteur assemblé des conditions limites de Dirichlet différentielles	CNDIDI
Vecteur assemblé des forces sur les sous-structures	CNSSTF
Vecteur assemblé des forces de contact (méthode CONTINUE et XFEM)	CNELTC
Vecteur assemblé des forces de frottement (méthode CONTINUE et XFEM)	CNELTF
Vecteur assemblé des forces de référence (RESI_REFE_RELA)	CNREFE
Vecteur assemblé des variables de commande pour l'état initial	CNVCF1
Vecteur assemblé des variables de commande pour la convergence	CNVCF0
Vecteur assemblé des conditions limites de type impédance (en correction)	CNIMPC
Vecteur assemblé des conditions limites de Dirichlet éliminées	CNCINE
Vecteur assemblé des sous-structures	CNSSTR
Vecteur assemblé des forces de frottement (méthode DISCRETE)	CNCTDF
Vecteur assemblé des variables de commande pour le calcul	CNVCPR
Vecteur assemblé des forces dynamiques	CNDYNA
Vecteur assemblé de l'amortissement modal (en prédiction)	CNMODP
Vecteur assemblé de l'amortissement modal (en correction)	CNMODC
Vecteur assemblé des forces de contact (méthode DISCRETE)	CNCTDC
Vecteur assemblé des forces unilatérales (méthode LIAISON_UNIL)	CNUNIL
Vecteur assemblé des forces extérieures	CNFEXT
Vecteur assemblé des conditions limites de type VECT_ISS	CNVISS

2.4.2.6 Variable-chapeau – Vecteurs des solutions SOLALG

Cette variable-chapeau contient le nom des champs aux nœuds qui servent dans l'algorithme pour calculer la solution.

Variable-chapeau – Vecteurs des solutions SOLALG	**Code de**

	repérage
Solution en déplacement de l'itération de Newton courante	DDEPLA
Déplacement cumulé depuis le début du pas de temps	DEPDEL
Incrément de déplacement du pas de temps précédent	DEPOLD
Solution en déplacement de la prédiction	DEPPR1
Solution en déplacement de la prédiction (partie pilotée)	DEPPR2
Solution en vitesse de l'itération de Newton courante	DVITLA
Vitesse cumulée depuis le début du pas de temps	VITDEL
Incrément de vitesse du pas de temps précédent	VITOLD
Solution en vitesse de la prédiction	VITPR1
Solution en vitesse de la prédiction (partie pilotée)	VITPR2
Solution en accélération de l'itération de Newton courante	DACCLA
Accélération cumulée depuis le début du pas de temps	ACCDEL
Incrément d'accélération du pas de temps précédent	ACCOLD
Solution en accélération de la prédiction	ACCPR1
Solution en accélération de la prédiction (partie pilotée)	ACCPR2
Solution du système	DEPSO1
Solution du système (partie pilotée)	DEPSO2

2.4.2.7 Variable-chapeau – Solutions incrémentales VALINC

Cette variable-chapeau contient le nom des champs aux nœuds ou des champs de type ELGA qui seront les solutions du problème non-linéaire.

Variable-chapeau – Solutions incrémentales V ALINC	Code de repérage
Déplacements au début du pas de temps	DEPMOI
Contraintes au début du pas de temps	SIGMOI
Variables internes au début du pas de temps	VARMOI
Vitesses au début du pas de temps	VITMOI
Accélérations au début du pas de temps	ACCMOI
Variables de commande au début du pas de temps	COMMOI
Variables pour les éléments multifibres au début du pas de temps	STRMOI
Forces extérieures au début du pas de temps (pour le calcul des énergies)	FEXMOI
Forces d'amortissement au début du pas de temps (pour le calcul des énergies)	FAMMOI
Forces de liaison au début du pas de temps (pour le calcul des énergies)	FLIMOI
Forces nodales au début du pas de temps (pour le calcul des énergies)	FNOMOI
Déplacements à la fin du pas de temps	DEPPLU
Contraintes à la fin du pas de temps	SIGPLU
Variables internes à la fin du pas de temps	VARPLU
Vitesses à la fin du pas de temps	VITPLU
Accélérations à la fin du pas de temps	ACCPLU
Variables de commande à la fin du pas de temps	COMPLU

Variables pour les éléments multifibres à la fin du pas de temps	STRPLU
Forces extérieures à la fin du pas de temps (pour le calcul des énergies)	FEXPLU
Forces d'amortissement à la fin du pas de temps (pour le calcul des énergies)	FAMPLU
Forces de liaison à la fin du pas de temps (pour le calcul des énergies)	FLIPLU
Forces nodales à la fin du pas de temps (pour le calcul des énergies)	FNOPLU
Contraintes extrapolées (méthode IMPLEX)	SIGEXT
Déplacements à l'itération de Newton courante (gestion des grandes rotations)	DEPKM1
Vitesses à l'itération de Newton courante (gestion des grandes rotations)	VITKM1
Accélérations à l'itération de Newton courante (gestion des grandes rotations)	ACCKM1
Rotations à l'itération de Newton précédente (gestion des grandes rotations)	ROMK
Rotations à l'itération de Newton courante (gestion des grandes rotations)	ROMKM1

2.4.2.8 Accès aux variables-chapeaux

L'accès aux variables-chapeaux se fait par l'intermédiaire de cinq routines.

Opération sur la variable-chapeau	**Routine**
Création d'une variable-chapeau	nmcha0.f
Récupération de l'index où est stocké le nom de la variable dans la variable-chapeau	nmchai.f
Recopie d'une variable-chapeau	nmchcp.f
Récupération du nom de la variable dans la variable-chapeau	nmchex.f
Recopie une variable-chapeau en changeant éventuellement un nom de variable	nmchso.f
Création des CHAM_NO pour VALINC , SOLAG et VEASSE	nmcrch.f

La taille de ces SD est indiquée de la même manière que les SD bas-niveau. Néanmoins, il convient de répercuter un changement de taille sur la routine principale nmchai.f. Si on veut modifier le contenu d'une variable-chapeau (ajouter, supprimer ou modifier le contenu d'une variable-chapeau), il faut :
- Impacter éventuellement la longueur dans op0070.f, nmini0.f (ASSERT de protection) et nmchai.f ;
- Modifier dans nmchai.f ;
- Créer la variable-chapeau (voir § 8) ;
- Initialiser éventuellement le contenu de la variable-chapeau ;

Ces routines se contentent de gérer les variables-chapeaux en tant que liste de noms, le contenu proprement dit de ces variables-chapeaux ne dépend pas d'elles. Par exemple, la variable-chapeau VEASSE contient des vecteurs assemblés. Aucune des routines du tableau précédent ne s'occupent de gérer la SD CHAM_NO des objets contenus dans la variable-chapeau, juste leur nom. Pour les trois variables-chapeaux définissant des CHAM_NO, les champs sont créés dans la routine nmcrch.f, en utilisant les informations sur les fonctionnalités actives FONACT.

2.4.3 SD de haut-niveau

2.4.3.1 Gestion des impressions – SDIMPR

L es informations de l a SDIMPR permettent de gérer l'affichage, en particulier celui du tableau de convergence. La SDIMPR est construite à partir des concepts appelés « SD objets génériques de type STRUCT » (voir § 45). La STRUCT de type <AFFICHAGE> contient différentes informations :

Paramètres de la SD <AFFICHAGE>		
Nom	**Description**	**Type**
TABL_CONV_CSV	.True. Si on sort le tableau de convergence sur un fichier externe (AFFICHAGE/UNITE)	<Booléen>

INFO_RESIDU	.True. Si on ajoute des colonnes d'information sur les résidus (AFFICHAGE/INFO_RESID)	<Booléen>
INFO_TEMPS	.True. Si on ajoute des colonnes d'information sur le temps (AFFICHAGE/INFO_TEMPS)	<Booléen>
PRINT	.True. si on affiche les informations dans le fichier .mess pour le pas de temps courant	<Booléen>
UNITE	Unité de sortie du tableau de convergence sur un fichier externe (AFFICHAGE/UNITE)	<Entier>
REAC_AFFICHAGE	Fréquence de réactualsiation de l'affichage dans le fichier .mess (AFFICHAGE/PAS)	<Entier>
TABLEAU_CONV	Tableau de convergence	<TABLEAU>

On notera la présence de l'objet TABLEAU_ CONV, de type <TABLEAU> qui est le tableau de convergence affiché pendant le calcul. Le STRUCT de type <TABLEAU> permet de décrire un tableau à afficher ou à sortir sur un fichier.

SD <TABLEAU> – Paramètres		
Nom	**Description**	**Type**
SORTIE_CSV	.True. Si on sort le tableau de convergence sur un fichier externe	<Booléen>
LARGEUR_LIGNE	Largeur d'une ligne du tableau de convergence	<Entier>
UNITE_CSV	Unité logique pour sortir le tableau de convergence	<Entier>
HAUTEUR_TITRE	Nombre de lignes de titre du tableau de convergence	<Entier>
COLONNES_DISPOS	Liste des colonnes disponibles dans le tableau de convergence	<SDLIST> <TABLEAU_COLONNE>

Un <TABLEAU> est constitué de colonnes COLONNES_DISPOS, qui sont des SDLIST de type <TABLEAU_COLONNE>. La SDLIST de type <TABLEAU_ COLONNE> permet de décrire une colonne dans un tableau. Chaque colonne est décrite par le type de données qu'elle contient (chaîne, entier, réel), sa largeur, son titre (contenu et hauteur), une éventuelle marque supplémentaire (uniquement pour les colonnes de type entier et réel) et le fait qu'elle est affectée effectivement d'une valeur. La notion d'affectation est importante car une colonne peut ne p as contenir de valeur à chaque itération de Newton. Dans certain cas, cela se traduit par une chaîne vide, par le message « SANS_OBJET » ou par une erreur (la valeur aurait dû être affectée).

SD <TABLEAU_COLONNE> – Paramètres		
Nom	**Description**	**Type**
VALE_AFFE	.true. si la valeur a été affectée	<Booléen>
ENTIER	.true. Si la colonne contient un entier	<Booléen>
REEL	.true. Si la colonne contient un réel	<Booléen>
CHAINE	.true. Si la colonne contient une chaîne	<Booléen>
NON_AFFE_ERREUR	.true. Si une valeur affectée provoque une erreur	<Booléen>
NON_AFFE_VIDE	.true. Si une valeur affectée provoque l'affichage d'un blanc	<Booléen>
NON_AFFE_SANSOBJ	.true. Si une valeur affectée provoque l'affichage SANS_OBJET	<Booléen>
LARGEUR	Largeur d e la colonne	<Entier>
HAUTEUR_TITRE	Nombre de lignes de titre de la colonne	<Entier>
VALE_I	Valeur de la colonne si c'est un entier	<Entier>

VALE_R	Valeur de la colonne si c'est un réel	<Réel>
VALE_K	Valeur de la colonne si c'est une chaîne	<Chaîne>
TYPE_COLONNE	Type de la colonne (repère pour <SDLIST>)	<Chaîne>
TITRE_LIGN_1	Première ligne du titre de la colonne	<Chaîne>
TITRE_LIGN_2	Deuxième ligne du titre de la colonne	<Chaîne>
TITRE_LIGN_3	Troisième ligne du titre de la colonne	<Chaîne>
MARQUE	Marquage de la colonne	<Chaîne>

La SDIMPR est construite à partir des STRUCT, on peut donc y accéder via les routines d'accès des STRUCT (voir §46). Néanmoins, la plupart des opérations sur la SDIMPR sont encapsulées dans des routines de plus haut-niveau.

Opération sur la gestion des impressions – SDIMPR	**Routine**
Impression d'une ligne du tableau de convergence	nmimpr.f
Impression d'une ligne de séparation	nmimpx.f
Impression de la récapitulation des résidus d'équilibre en fin de pas	nmimps.f
Affectation d'une valeur dans une colonne de type <Réel>	nmimcr.f
Affectation d'une valeur dans une colonne de type <Entier>	nmimci.f
Affectation d'une valeur dans une colonne de type <Chaîne>	nmimck.f
Activation/désactivation d'une colonne	nmimca.f
Désactivation de toutes les colonnes	nmimr0.f
Activation ou désactivation de l'affichage pour le pas de temps courant	nmimpa.f

De même, la STRUCT de type <TABLEAU> peut être manipulée avec des routines plus pratiques :

Opération sur le STRUCT de type <TABLEAU>	**Routine**
Création d'une ligne vide avec les séparateurs de colonnes	obtlig.f
Affectation d'une marque dans une colonne	obtsdm.f
Création de l'entête du tableau	obttit.f
Calcul de la largeur d'une ligne	obtcla.f
Création de la ligne de séparation dans le tableau	implis.f
Impression d'une ligne du tableau	impsdl.f

Il existe quatre routines bas-niveau s'occupant du formatage de la chaine d'information à afficher suivant son type.

Opération de formatage	**Routine**
Formatage d'un réel	impfor.f
Formatage d'un entier	impfoi.f
Formatage d'une chaîne	impfok.f
Formatage d'un temps (transformation secondes en heures/min/s)	impfot.f

2.4.3.2 Gestion des mesures de temps – SDTIME

Cette SD sert à faire les différentes mesures de temps dans les phases de calcul non-linéaires.

SDTIME – Objets	
Nom	**Description**
SDTIME(1:19)//'.TDEB'	Stockage du temps initial
SDTIME(1:19)//'.TIMN'	Stockage des mesures pour l'itération de Newton
SDTIME(1:19)//'.TIMP'	Stockage des mesures pour le pas de temps
SDTIME(1:19)//'.TIMT'	Stockage des mesures pour le transitoire complet
SDTIME(1:19)//'.TMP1'	Timer temporaire 1
SDTIME(1:19)//'.TMP2'	Timer temporaire 2
SDTIME(1:19)//'.TPAS'	Timer pour le pas de temps
SDTIME(1:19)//'.TITE'	Timer pour l'itération de Newton
SDTIME(1:19)//'.TARC'	Timer pour l'archivage
SDTIME(1:19)//'.TPSD'	Timer pour le post-traitement

Quatre timers sont réservées aux mesures récurrentes (pas de temps, itération de Newton, archivage et post-traitement) et deux autres timers servent pour d'autres mesures (factorisation temps dans le contact , etc.).
La SD est conçue pour stocker les temps cumulés sur les phases (sur tout le pas de temps, sur tout le transitoire), et ce, de manière automatique.
Il est donc préférable d'utiliser la SDTIME plutôt que les timers directement (routines UTTCPU) car les statistiques sont faites automatiquement, ainsi que les affichages. Les routines principales de manipulation se la SD sont les suivantes :

Opération sur la gestion des mesures de temps – SDTIME	**Routine**
Création de la SDTIME – C'est dans cette routine que l'on donnera le nombre de choses mesurables (variable NBMESU)	nmcrti.f
Remise à zéro des statistiques et des timers	nmrini.f
Gestion des timers (initialisation, mesure de temps)	nmtime.f
Sauvegarde d'un temps donné dans les statistiques du timer désigné	nmrtim.f
Mesure du temps restant à la fin d'une itération de Newton ou d'un pas de temps	nmtima.f
Lecture des informations stockées dans SDTIME	nmtimr.f
Mesure du temps « perdu » en découpes du pas de temps lors d'un calcul	nmlost.f
Affichage des statistiques et des temps	nmstat.f

Les deux routines les plus importantes sont nmcrti.f avec laquelle on pourra ajouter une nouvelle mesure et nmtime.f qui réalise la mesure.
Exemple d'utilisation :

```
CALL NMTIME(SDTIME,'INI','ASSE_MATR')
CALL NMTIME(SDTIME,'RUN','ASSE_MATR')
       /.OPERATION ./
CALL NMTIME(SDTIME,'END','ASSE_MATR')
```

La routine nmstat.f permet de faire l'affichage des informations (utilisée également par la SDSTAT, voir § 22).

2.4.3.3 Gestion des erreurs de l'algorithme – SD ERRO

Cette SD s'occupe de gérer les erreurs de l'algorithme. Elle contient sept objets de même taille, celui du nombre d'évènements (variable ZEVEN) traitables par l'algorithme (modifiable dans nmcrga.f , cf § 33)

SDERRO – Objets

Nom	Description
SDERRO(1:19)//'.ENOM'	Nom de l'évènement
SDERRO(1:19)//'.ECOV'	Valeur du code-retour lié à l'évènement
SDERRO(1:19)//'.ECON'	Nom du code-retour lié à l'évènement
SDERRO(1:19)//'.ENIV'	Type et niveau de déclenchement de l'évènement
SDERRO(1:19)//'.EFCT'	Fonctionnalité activant un évènement de type convergence ou divergence
SDERRO(1:19)//'.EACT'	État de l'évènement (activé ou non)
SDERRO(1:19)//'.EMSG'	Code du message à afficher quand l'évènement se déclenche

Les deux autres objets permettent de gérer l'état de la boucle et de stocker le dernier évènement déclenché (servira aux affichages).

SDERRO – Objets	
Nom	**Description**
SDERRO(1:19)//'.CONV'	État de la boucle
SDERRO(1:19)//'.EEVT'	Information sur le dernier évènement déclenché

La SD ERRO s'utilise en utilisant certaines informations contenues dans la SDDISC (§26), provenant des définitions des évènements de la commande DEFI_LIST_INST.

Opération sur la gestion des erreurs de l'algorithme – SDERRO	Routine
Création de la SD ERRO	nmcrga.f
Enregistrement d'un évènement	nmcrel.f
Enregistrement d'un évènement à partir d'un code-retour	nmcret.f
Changement de l'état d'une boucle	nmeceb.f
Lecture de l'état d'une boucle	nmleeb.f
Remise à zéro des évènements	nmeraz.f
Retourne l'état d'un évènement (actif ou non) suivant son nom	nmerge.f
Émission du message d'information sur l'évènement	nmevim.f
Évaluation de l'état de convergence d'une boucle	nmevcv.f
Retourne l'état d'un évènement (actif ou non) suivant son type	nmltev.f

L'utilisation de ces routines dans le cadre de la gestion des évènements est détaillée dans le § 33 .

2.4.3.4 Gestion de l'archivage et de l'état initial (IN et OUT) – SDIETO

Cette SD sert à effectuer les opérations suivantes :
- Initialisation des champs (mots-clefs ETAT_INIT , prenant en compte un éventuel enrichissement de la SD résultat par l'utilisation de reuse) ;
- Archivage des champs : rythme d'archivage, exclusion de certains champs ;

Cette SD n'a que deux objets. Un qui donne des informations générales, et un second qui regroupe les informations nécessaires pour chaque champ. Ce dernier objet est proportionnel au nombre de champs (variable NBMAX) traitables.

SDIETO – Objets	
Nom	**Description**

SDIETO(1:19)//'.LCHA'	Informations sur les champs
SDIETO(1:19)//'.INFO'	Informations générales (nombre de champs, nombre de champs en entrée, nombre de champs en sortie)

Les informations sur le champ sont contenues dans des DATA dans la routine nmetcr.f, ce sont les suivantes :

Description d'un champ pour SDIETO		
Description	**DATA**	**A renseigner**
Nom du champ dans la SD résultat	NOMCHS	OUI
Nom du champ servant à initialiser	modifier nmetc0.f	
Statut du champ après initialisation (nul, donné individuellement par l'utilisateur, donné dans une SD RESULTAT, issu d'un calcul stationnaire, constant avec une valeur donnée),	NON	
Nom de la grandeur support du champ	NOMGD	OUI
Mot-clef pour ETAT_INIT	MOTCEI	Pas obligatoire
Localisation du champ (NOEU/ELGA/ELEM)	LOCCHA	OUI
Dit si le champ doit être à l'état initial (il serait nul ou lu par ETAT_INIT)	LETIN	OUI
Dit si le champ doit être archivé	LARCH	OUI
Mot-clef pour OBSERVATION	MOTCOB	Pas obligatoire
Nom du champ correspondant pendant l'algorithme (par exemple, dans la variable chapeau SOLALG)	modifier nmetcc.f	

L'ensemble des opérations courantes est prévue dans des routines en général préfixées par nmet*. Lorsqu'on veut ajouter un champ en gestion ARCHIVAGE/OBSERVATION/ETAT_INIT :
•Il faut que le développeur active ou pas le champ suivant les fonctionnalités activées (par exemple). Il faut impacter la routine nmetac.f ;
•Il faut que le développeur précise le champ initial (si champ en reprise). Il faut impacter la routine nmetc0.f ;
•Il faut que le développeur précise le nom du champ dans l'opérateur. Il faut impacter la routine nmetcr.f ;
•Quand un champ est archivé dans la SD RESULTAT , il faut impacter rscrsd.f.

Opération sur la gestion de l'archivage et de l'état initial (IN et OUT) – SDIETO	**Routine**
Création de la SDIETO	nmetcr.f
Ajout d'un champ dans la SDIETO	nmetci.f
Ajout du nom du champ qui correspond dans l'algorithme d'op0070.f	nmetcc.f
Nom et création du champ nul	nmetc0.f
Lecture de l'état initial	nmdoet.f
Lecture d'un champ – Cas de la SD résultat dans ETAT_INIT	nmetl1.f
Lecture d'un champ – Cas champ par champ dans ETAT_INIT	nmetl2.f
Lecture d'un champ – Vérifications diverses : •Vérifie la grandeur et la localisation •Traite le cas de la précontrainte (nmsigi.f) •Vérifie la cohérence du champ des variables internes (vrcomp.f)	nmetl3.f
Retourne le nom du champ utilisé dans op0070.f (par indice de champ)	nmetnc.f
Retourne l'indice du champ observable (par nom du champ)	nmetob.f

Transformation des champs de type variables-chapeaux de type "MOI" (instant précédent) en type "PLU" (instant suivant)	nmetpl.f
Conversion de la localisation d'un champ	nmetcv.f
Écriture du champ dans la SD résultat	nmeteo.f

2.4.3.5 Gestion des statistiques – SDSTAT

Cette SD permet de faire des statistiques (collecte et affichage). Elle contient des objets qui sont tous dimensionnés au nombre de choses à mesurer (variable NBSTAT dans nmcrst.f).

SDCONV – Objets	
Nom	**Description**
SDSTAT(1:19)//'.VLIT'	Statistiques pour le transitoire
SDSTAT(1:19)//'.VLIP'	Statistiques pour le pas de temps
SDSTAT(1:19)//'.VLIN'	Statistiques pour l'itération de Newton

Si on désire modifier une statistique, seules les trois premières routines suivantes sont à modifier. La routine nmstat.f permet tant de faire l'affichage des informations (utilisée également par la SDTIME, voir §18 .

Description	Routine
Création de la SDSTAT	nmcrst.f
Lecture/écriture dans la SDSTAT	nmrvai.f
Affichage des statistiques et des temps	nmstat.f
Réinitialisation des statistiques	nmrini.f

2.4.3.6 Gestion de la convergence – SDCONV

Cette SD gère la collecte et l'affichage des informations relatives aux résidus d'équilibre (boucle <RESI>, voir §31). Elle contient des objets qui sont tous dimensionnés au nombre de résidus à considérer (variable NRESI dans nmcrcg.f).

SDCONV – Objets	
Nom	**Description**
SDCONV(1:19)//'.TYPE'	Type des résidus (RESI_GLOB_RELA, RESI_GLOB_MAXI, etc.)
SDCONV(1:19)//'.LIEU'	Lieu où le résidu est maximum
SDCONV(1:19)//'.VALE'	Valeur du résidu
SDCONV(1:19)//'.ACTI'	.True. si le résidu est activé (sert à la convergence)
SDCONV(1:19)//'.NCOL'	Nom des colonnes dans la SDIMPR

L'activation d'un résidu pour la convergence n'est pas nécessairement constant au cours du calcul. Il arrive parfois que l'on bascule automatiquement de RESI_GLOB_RELA à RESI_GLOB_MAXI par exemple).

Pour la gestion de la convergence, on modifiera essentiellement les routines nmcrcg.f, nmimre.f et nmcore.f. La bascule d'un résidu à l'autre par exemple, se fera dans nmcore.f. La routine d'impression nmimps.f est « auto-portante » : les informations contenues dans SDCONV lui suffit . ***Elle ne doit donc pas être modifiée.***

Opération sur la gestion de la convergence – SDCONV	Routine
Création de la SDCONV	nmcrcg.f

Vérification des critères d'arrêt sur les résidus	nmcore.f
Impression des résidus récapitulatifs en fin de pas de temps	nmimps.f
Impression des informations sur les résidus dans le tableau de convergence	nmimre.f

2.4.3.7 Gestion de l'extraction de champs – SDEXTR

Le SUIVI_DDL partage avec l'OBSERVATION (voir §23) une SD commune appelé SDEXTR, routine d'extraction des valeurs des champs. On va donc commencer par décrire cette dernière SD. Les trois premiers objets sont généraux et leur longueur est proportionnelle au nombre d'occurrences du mot-clef SUIVI_DDL ou OBSERVATION .

SDEXTR – Objets	
Nom (attention aux blancs!)	**Description**
SDEXTR(1:14)//' .INFO'	Informations diverses sur les données d'extraction
SDEXTR(1:14)//' .EXTR'	Type d'extraction •Sur le champ (NOEU et ELGA) •Sur la maille (si ELGA) •Sur les composantes ou formule entre les composantes
SDEXTR(1:14)//' .ACTI'	Extraction active ou pas

On ne peut avoir plus de 99 occurrences des mots-clefs car on construit le nom de certains objets à partir de ce numéro d'occurrence OCC. Ces objets sont les suivants :

SDEXTR – Objets	
Nom (attention aux blancs!)	**Description**
SDEXTR(1:14)//OCC//' .NOEU'	Nœuds concernés par l'extraction
SDEXTR(1:14)//OCC//' .MAIL'	Mailles concernées par l'extraction
SDEXTR(1:14)//OCC//' .POIN'	Points d'intégration concernés par l'extraction
SDEXTR(1:14)//OCC//' .SSPI'	Sous-points d'intégration concernés par l'extraction
SDEXTR(1:14)//OCC//' .CMP'	Composantes concernées par l'extraction

2.4.3.8 Gestion du SUIVI_DDL – SDSUIV

Cette SD sert à gérer la fonctionnalité SUIVI_DDL . En plus des références à l'extraction des champs (SDEXTR, §23), nous avons un objet spécifique pour le SUIVI_DDL.

SDSUIV – Objets	
Nom (attention aux blancs!)	**Description**
SD SUIV (1:14)//' . TITR '	Titres des colonnes

2.4.3.9 Gestion de l'OBSERVATION – SDOBSE

Cette SD sert à gérer la fonctionnalité OBSERVATION. L'OBSERVATION partage avec le SUIVI_DDL (voir §23) une SD commune appelé SDEXTR, déjà décrite dans (§23).

Ensuite, nous avons des objets spécifiques pour l'OBSERVATION. Un qui va donner le nom de la table et un qui stocke les informations relatives à la fréquence d'observation (objet utilitaire SDSELI, voir §28), indicé par le numéro d'occurrence OCC du mot-clef OBSERVATION.

SDOBSE – Objets	
Nom (attention aux blancs!)	**Description**
SDOBSE(1:14)//' .TABL'	Nom de la table
SDOBSE(1:14)//OCC//'.LI'	Accès à la SDSELI, liste des instants à observer

2.4.3.10 *Gestion des critères de qualité – SDCRIQ*

Cette SD sert à évaluer les critères de qualité (mot-clef CRIT_QUALITE). Elle est très simple et ne comporte qu'un objet.

SDCRIQ – Objets	
Nom	**Description**
SDSUIV (1:1 9)//'.ERRT'	Valeur des erreurs THM espace et temps, coefficient THETA

2.4.3.11 *Gestion du pilotage – SDPILO*

Cette SD sert au pilotage (méthode de continuation). La plupart de ces objets sont regroupés dans une logique de type (réels avec réels, chaînes avec chaînes), plutôt que dans une logique de fonctionnalité.

SDPILO – Objets	
Nom	**Description**
SDPILO(1:19)//'.PLTK'	Contient les paramètres du pilotage (de type chaîne) ou les noms d'objets nécessaires au pilotage
SDPILO(1:19)//'.PLIR'	Contient les paramètres du pilotage (de type réel)
SDPILO(1:14)//'.PLCR'	Liste des coefficients pour le pilotage
SDPILO(1:14)//'.PLSL'	Liste des DDL de type DX, DY et DZ pour le pilotage
SDPILO(1:14)//'.PLCI'	Liste des coefficients pour le pilotage – Cas XFEM

Il n'y a pas de routines d'accès de haut-niveau, La SD est créée dans la routine nmdopi.f. Il faut attaquer directement les SD.

2.4.3.12 *Gestion des numérotations – SDNUME*

Cette SD vient en complément des deux SD NUMDDL et NUMFIX pour gérer la numérotation des équations. NUMDDL est la numérotation des équations, qui peut être variable au cours du transitoire, quand on utilise des fonctionnalités comme le contact CONTINUE ou le contact XFEM. NUMFIX est la numérotation fixe, elle est utile dans le cas des macro-éléments, pour pouvoir combiner les matrices. La SDNUME contient trois autres objets :

SDNUME – Objets	
Nom	**Description**
SDNUME(1:19)//'.NDRO'	Repérage des DDL pour les grandes rotations
SDNUME(1:19)//'.NUCO'	Repérage des DDL pour les Lagrangiens de contact et frottement
SDNUME(1:19)//'.ENDO'	Repérage des DDL pour l'endommagement aux nœuds

Ces trois objets ont la même logique : ils sont dimensionnés au nombre total de degrés de liberté de la structure, avec la même numérotation que les matrices et les CHAM_NO utilisés dans op0070.f. Une valeur particulière (en général 1), sert à repérer a priori des DDL spécifiques : ceux correspondant aux grandes rotations, ceux correspondant aux lagrangiens de contact/frottement et ceux correspondant à l'endommagement. On les utilise alors dans certains cas (par exemple, pour la mise à jour spécifiques des

champs dans le cas des grandes rotations ou pour filtrer les composantes dans l'évaluation des résidus). Il n'y a pas de routines d'accès spécifiques.

2.4.3.13 *Gestion de la dynamique – SDDYNA*

Cette SD contient toutes les informations nécessaires au calcul en dynamique.

SDDYNA – Objets	
Nom	**Description**
SDDYNA(1:15)//'.PARA_SCH'	Paramètres des schémas en temps
SDDYNA(1:15)//'.INFO_SD'	Paramètres de la dynamique
SDDYNA(1:15)//'.NOM_SD'	Nom des SD pour la dynamique (modes statiques, vecteurs, ...)
SDDYNA(1:15)//'.TYPE_FOR'	Type de formulation (déplacement, vitesse ou accélération)
SDDYNA(1:15)//'.COEF_SCH'	Coefficients à utiliser dans le calcul
SDDYNA(1:15)//'.TYPE_CHA'	Informations relatives au chargement ONDE_PLANE
SDDYNA(1:15)//'.NBRE_CHA'	Informations relatives à certains chargements spécifiques à la dynamique
SDDYNA(1:15)//'.VEEL_OLD'	Vecteurs élémentaires du pas précédent pour les schémas multi-pas
SDDYNA(1:15)//'.VEAS_OLD'	Vecteurs assemblés du pas précédent pour les schémas multi-pas
SDDYNA(1:15)//'.VECENT'	Quantités en entrainements (déplacements, vitesses et accélération)
SDDYNA(1:15)//'.VECABS'	Quantités absolues (déplacements, vitesses et accélération)

L'accès à ces informations se fait via quatre routines dédiées chacune à un type, avec une chaine posant la question (sur le modèle de la routine DISMOI) :

Opération sur la gestion de la dynamique – SDDYNA	**Routine**
Lecture information de type <booléen>	ndynlo.f
Lecture information de type <réel>	ndynre.f
Lecture information de type <entier>	ndynin.f
Lecture information de type <chaîne>	ndynkk.f

A noter que ces questions vont entraîner une erreur fatale si on n'est pas en dynamique, sauf dans le cas de la question NDYNLO(SDDYNA,'DYNAMIQUE') qui répondra .false. si on est en statique (c'est-à-dire qui sait détecter le cas où SDDYNA n'existe pas). En dehors de ces quatre routines, l'accès à la SDDYNA se fait directement dans deux routines :

Opération sur la gestion de la dynamique – SDDYNA	**Routine**
Lecture informations dans le fichier de commande	ndlect.f
Enregistrement valeur des différents coefficients	ndnpas.f

On revient sur les coefficients nécessaires pour le calcul en dynamique car ils sont très nombreux. Ces coefficients sont construits à partir de trois informations :
- Le type de schéma ;
- Les paramètres du schémas (coefficients ALPHA , BETA , KAPPA , etc.) ;
- L'incrément de temps ;

Le fait de dépendre du pas de temps implique que les coefficients sont ré-évalués à chaque pas (dans la routine ndnpas.f).

Coefficient	**Description**

COEF_MATR_RIGI	Coefficient devant la matrice de rigidité
COEF_MATR_AMOR	Coefficient devant la matrice d'amortissement
COEF_MATR_MASS	Coefficient devant la matrice de masse
COEF_DEPL_DEPL	Prédicteur en déplacement : coefficient devant le déplacement du pas précédent
COEF_DEPL_VITE	Prédicteur en déplacement : coefficient devant la vitesse du pas précédent
COEF_DEPL_ACCE	Prédicteur en déplacement : coefficient devant l'accélération du pas précédent
COEF_VITE_DEPL	Prédicteur en vitesse : coefficient devant le déplacement du pas précédent
COEF_VITE_VITE	Prédicteur en vitesse : coefficient devant la vitesse du pas précédent
COEF_VITE_ACCE	Prédicteur en vitesse : coefficient devant l'accélération du pas précédent
COEF_VITE_DEPL	Prédicteur en accélération : coefficient devant le déplacement du pas précédent
COEF_VITE_VITE	Prédicteur en accélération : coefficient devant la vitesse du pas précédent
COEF_VITE_ACCE	Prédicteur en accélération : coefficient devant l'accélération du pas précédent
COEF_DEPL	Coefficient devant l'incrément de déplacement
COEF_VITE	Coefficient devant l'incrément de vitesse
COEF_ACCE	Coefficient devant l'incrément d'accélération
COEF_MPAS_FEXT_PREC	Coefficient devant les forces extérieures du pas précédent (schéma multipas)
COEF_MPAS_FINT_PREC	Coefficient devant les forces intérieures du pas précédent (schéma multipas)
COEF_MPAS_FEXT_COUR	Coefficient devant les forces extérieures du pas courant (schéma multipas)
COEF_MPAS_EQUI_COUR	Coefficient devant les autres termes du second membre (inertie, amortissement)
COEF_FDYN_MASSE	Coefficient devant les forces de rappel dynamique (inertie)
COEF_FDYN_AMORT	Coefficient devant les forces de rappel dynamique (amortissement)
COEF_FDYN_RIGID	Coefficient devant les forces de rappel ?????? (sert à Krenk ???)
COEF_FORC_INER	Coefficient devant les forces d'inertie à mettre au dénominateur dans le résidu d'équilibre
INST_PREC	Pas de temps précédent (uniquement utile pour la poursuite avec, à la fois, le schéma HHT complet et les lois de comportement qui ont besoin de ce paramètre)

2.4.3.14 *Gestion de la discrétisation temporelle – SDDISC*

La structure de données SDDISC contient toutes les informations provenant de l'opérateur DEFI_LIST_INST (création du concept SDLIST) et des informations relatives à la gestion de la discrétisation temporelle dans op0070.f. Dans un premier temps, les informations provenant de l'opérateur DEFI_LIST_INST (voir [D4.06.17]) sont recopiées localement dans la SDDISC.

SDDISC – Objets	
Nom	**Description**
SDDISC(1:19)//'.LINF'	Informations sur la liste d'instants (voir contenu dans [D4.06.17]) Recopie de l'objet SDLIST(1:8)//'.LIST.INFOR'
SDDISC(1:19)//'.DITR'	Liste des instants – Cette liste est dynamique (en cas de découpe ou d'accélération du pas de temps)
SDDISC(1:19)//'.DINI'	Indicateur du niveau de sous-découpage pour chaque pas de temps. Initialement, le niveau de découpe vaut 1. Il ne peut y avoir plus d'un niveau de découpe entre deux pas successifs (vérification dans la routine nmdcin.f) – Cette liste est dynamique (en cas de découpe ou d'accélération du pas de temps)

SDDISC(1:19)//'.ITER'	Nombre d'itérations de Newton qu'il a fallu pour chaque pas de temps – Cette liste est dynamique (en cas de découpe ou d'accélération du pas de temps)
SDDISC(1:19)//'.EPIL'	Indicateur du choix de la solution de pilotage (action AUTRE_PILOTAGE)
SDDISC(1:19)//'.LIPO'	Liste des instants de calcul obligatoires. Cet objet sert dans le cas de l'adaptation automatique du pas de temps, il indique les instants qui seront calculés quoiqu'il arrive (même en cas d'agrandissement du pas de temps). On parle aussi d'instants « jalons ». Sa longueur est celle de la liste d'instants initiale donnée par l'utilisateur dans DEFI_LIST_INST. Il s'agit donc d'une liste ***non-dynamique***.
SDDISC(1:19)//'.REPC'	Indicateur d e gestion de la réactualisation du préconditionnement (action REAC_PRECOND)
SDDISC(1:19)//'.EEVR'	Recopie de l'objet SDLIST(1:8)// '.ECHE.EVENR' Voir contenu dans [D4.06.17]
SDDISC(1:19)//'.EEVK'	Recopie de l'objet SDLIST(1:8)// '.ECHE.EVENK' Voir contenu dans [D4.06.17]
SDDISC(1:19)//'.ESUR'	Recopie de l'objet SDLIST(1:8)// '.ECHE.SUBDR' Voir contenu dans [D4.06.17]
SDDISC(1:19)//'.AEVR'	Recopie de l'objet SDLIST(1:8)// '.ADAP.EVENR' Voir contenu dans [D4.06.17]
SDDISC(1:19)//'.AEVK'	Recopie de l'objet SDLIST(1:8)// '.ADAP.EVENK' Voir contenu dans [D4.06.17]
SDDISC(1:19)//'.ATPR'	Recopie de l'objet SDLIST(1:8)// '.ADAP.TPLUR' Voir contenu dans [D4.06.17]
SDDISC(1:19)//'.ATPK'	Recopie de l'objet SDLIST(1:8)// '.ADAP.TPLUK' Voir contenu dans [D4.06.17]
SDDISC(1:19)//'.AEXT'	Objet pour le prolongement de la découpe (traitement du cas de la COLLISION)
SDDISC(1:19)//'.IFCV'	Objet stockant l'état de convergence pour l'adaptation V(1) – Valeur du MAX(ITER_GLOB_MAXI,ITER_GLOB_ELAS) V(2) – Valeur du MIN(ITER_GLOB_MAXI,ITER_GLOB_ELAS) V(3) – Nombre maximum d'itérations possibles (y compris les itérations supplémentaires possibles par ITER_SUPPL) V(4) – Valeur de 'PAS_MINI_ELAS' V(5) – Valeur de 'RESI_GLOB_RELA' V(6) – Valeur de 'RESI_GLOB_MAXI' V(7) – Type de résidu demandé par l'utilisateur : =1 pour RESI_GLOB_RELA =2 pour RESI_GLOB_MAXI =3 pour RESI_GLOB_RELA et RESI_GLOB_MAXI V(8) – Valeur de 'INIT_NEWTON_KRYLOV' V(9) – Valeur de 'ITER_NEWTON_KRYLOV' V(1 0) – Indique qu'on autorise des itérations en plus
SDDISC(1:19)//'.IFRE'	Objet stockant la valeur des résidus à chaque itération de Newton V(3*(ITER-1)+0) – Valeur de RESI_GLOB_RELA V(3*(ITER-1)+1) – Valeur de RESI_GLOB_MAXI V(3*(ITER-1)+2) – Valeur du chargement

Pour manipuler la SDDISC, on utilise une série d'utilitaires.

Opération sur la gestion de la discrétisation temporelle – SDDISC	**Routine**

Routine utilitaire générale qui permet d'accéder au contenu des objets .LINF, .EEVR, .EEVK, .ESUR, .AEVR, .AEVK, .ATPR, .ATPK, .REPC et .EPIL. L'accès peut se faire en lecture ou en écriture .	utdidt.f
Retourne .RUE. si on sort de la liste d'instants (fin du transitoire)	didern.f
Valeur de l'instant selon numéro de l'instant	diinst.f

2.4.3.15 *Gestion de la sélection d'un instant – SDSELI*

La structure de données SDSELI permet de sélectionne un instant suivant une fréquence, une liste de valeurs, en prenant en compte une tolérance et un type de critère (absolu ou relatif).

SDSELI – Objets	
Nom	**Description**
SDSELI(1:19)//'.INFL'	Informations générales sur la sélection (paramètres de sélection)
SDSELI(1:19)//'.LIST'	Liste des instants donnés par l'utilisateur

Pour opérer sur cette SD, on utilise principalement deux routines :
- une qui lit les paramètres de l'utilisateur (nmcrpx.f) ;
- recherche si l'instant donné est sélectionné (nmcrpo.f) ;

Opération sur la gestion de la sélection d'un instant – SDSELI	**Routine**
Lecture de la liste des instants à sélectionner	nmcrpa.f
Lecture de la précision et du type de critère (relatif ou absolu)	nmcrpp.f
Lecture de toutes les informations (appel à nmcrpa.f et nmcrpp.f)	nmcrpx.f
Recherche d'un réel dans une liste (avec précision et critère)	utacli.f
Routine principale de recherche de l'instant	nmcrpo.f
Recherche de l'indice dans la SD résultat juste avant un instant donné	nmttch.f

Cette SD est utilisée pour la gestion de la discrétisation temporelle, de l'état initial, de l'archivage et de l'observation.

2.4.3.16 *Gestion de s critères de convergence – SDCRIT*

Cette SD gère des informations relatives aux critères de convergence, elle sert en particulier à stocker les résidus dans la SD RESULTAT.

SD CRIT – Objets	
Nom	**Description**
SDCRIT(1:19)//'.CRTR'	Valeurs des informations (résidus, nombre d'itérations, etc.)
SDCRIT(1:19)//'.CRDE'	Nom des informations pour le stockage dans la SD RESULTAT (résidus, nombre d'itérations, etc.)

Cette SD est aussi utilisée pour la commande THER_NON_LINE (attention, les objets ne sont pas de la même dimension !). La sauvegarde des informations pour cette SD se fait dans la routine nmcore.f qui évalue la convergence à chaque itération de Newton.

Opération sur la gestion des critères de convergence – SDCRIT	**Routine**
Création de la SD	nmcrcv.f ntcrcv.f

Sauvegarde des informations dans la SDCRIT	nmcore.f op0186.f
Sauvegarde des informations dans la SD RESULTAT	nmarc0.f ntarc0.f

2.4.3.17 *Gestion du post-traitement – SDPOST*

Cette SD gère des informations relatives à ce qu'on désigne comme « post-traitement ». Les post-traitements sur les analyses spectrales (modes vibratoires ou modes de stabilité) que l'on peut faire à chaque pas de temps. La SD est constituée de sept objets. Trois d'entre eux sont de simples vecteurs rassemblant les informations et les paramètres nécessaires, classés suivant leur type.

SDCRIT – Objets	
Nom (attention à la longueur!)	**Description**
SD POST (1:19)//'. INFI '	Paramètres de type <entier>
SD POST (1:19)//'. INFR '	Paramètres de type <réel>
SD POST (1:19)//'. INFK '	Paramètres de type <chaîne>
SD POST (1:1 4)//'. VIBR '	Nom de la SD de type SDSELI (§28) pour les modes vibratoires
SDPOST(1:14)//'.EXCL'	Nom de la SD stockant les noms des degrés de liberté exclus pour les modes de stabilité
SDPOST(1:14)//'.STAB'	Nom de la SD stockant les noms des degrés de liberté pris en compte pour les modes de stabilité
SD POST (1:1 4)//'. FLAM '	Nom de la SD de type SDSELI (§28) pour les modes de stabilité ou de flambement

Voici la liste des paramètres :

Paramètre	Type	Description
CRIT_STAB	<entier>	Vaut 1 si on fait un calcul de type modes de stabilité ou de modes de flambement
MODE_VIBR	<entier>	Vaut 1 si on fait un calcul de type modes vibratoires
OPTION_CALCUL_FLAMB	<chaîne>	Type de calcul de stabilité : statique ou dynamique
OPTION_CALCUL_VIBR	<chaîne>	Type de calcul de modes vibratoires : forcément dynamique
TYPE_MATR_VIBR	<chaîne>	Type de matrice de rigidité (élastique, sécante ou tangente) en modes vibratoires
OPTION_EXTR_VIBR	<chaîne>	Type de recherche (« plus petite » ou « bande ») pour les modes vibratoires
NB_FREQ_VIBR	<entier>	Nombre de fréquences à calculer pour les modes vibratoires
BANDE_VIBR_1	<réel>	Première borne si recherche « bande » pour les modes vibratoires
BANDE_VIBR_2	<réel>	Seconde borne si recherche « bande » pour les modes vibratoires
TYPE_MATR_FLAMB	<chaîne>	Type de matrice de rigidité (élastique, sécante ou tangente) en modes de flambement (utilise l'information donné dans le mot-clef PREDICTION de NEWTON).
NB_FREQ_FLAMB	<entier>	Nombre de fréquences à calculer pour les modes de stabilité ou de flambement
BANDE_FLAMB_1	<réel>	Première borne si recherche « bande » pour les modes de

		stabilité ou de flambement
BANDE_FLAMB_2	<réel>	Seconde borne si recherche « bande » pour les modes de stabilité ou de flambement
RIGI_GEOM_FLAMB	<chaîne>	Calcul des modes de stabilité ou de flambement avec ou sans prise en compte de la matrice de rigidité géométrique
OPTION_EXTR_FLAM	<chaîne>	Type de recherche (« plus petite » ou « bande ») pour les modes de stabilité ou de flambement
NB_DDL_EXCLUS	<entier>	Nombre de degrés de liberté exclus pour les modes de stabilité
SOLU_FREQ_VIBR	<réel>	Valeur de la fréquence pour le mode vibratoire
SOLU_FREQ_FLAM	<réel>	Valeur de la fréquence pour le mode de flambement
SOLU_NUME_VIBR	<entier>	Indice du mode vibratoire
SOLU_NUME_FLAM	<entier>	Indice du mode de flambement
SOLU_MODE_VIBR	<chaîne>	Mode vibratoire (champ de déplacement)
SOLU_MODE_FLAM	<chaîne>	Mode de flambement (champ de déplacement)
NOM_DDL_EXCLUS	<chaîne>	SDPOST(1:14)//'.EXCL'
NOM_DDL_STAB	<chaîne>	SDPOST(1:14)//'.STAB'
NB_DDL_STAB	<entier>	Nombre de degrés de liberté pris en compte pour les modes de stabilité
SOLU_FREQ_STAB	<réel>	Valeur de la fréquence pour le mode de stabilité
SOLU_NUME_STAB	<entier>	Indice du mode de stabilité
SOLU_MODE_STAB	<chaîne>	Mode de stabilité (champ de déplacement)
COEF_DIM_FLAMB	<entier>	Coefficient du sous-espace (voir analyse modale) pour les modes de stabilité ou les modes de flambement
COEF_DIM_VIBR	<entier>	Coefficient du sous-espace (voir analyse modale) pour les modes vibratoires
MODI_RIGI	<chaîne>	Modification de la rigidité
SIGN_INSTAB	<chaîne>	Signe pour détecter une instabilité
PREC_INSTAB	<réel>	Précision pour détecter une instabilité

3 Gestion de l'algorithme

Nous allons nous intéresser à la gestion de l'algorithme. Pour gérer la convergence, les différentes niveaux de boucle et les erreurs survenues lors d'un calcul, on utilise un formalisme commun, basé sur la notion d'**évènement** .

3.1 Les différentes boucles

Dans l'algorithme, il y a cinq niveaux de boucles <BOUC> :
•<RESI> : la boucle sur les différents résidus d'équilibre demandés par l'utilisateur (RESI_GLOB_RELA, RESI_GLOB_MAXI, etc...) ;
•<NEWT> : la boucle sur les itérations de Newton ;
•<FIXE> : la boucle sur les points fixes (contact) ;
•<INST> : la boucle sur les instants de calcul ;
•< CALC > : la boucle sur le calcul .
La «boucle » <CALC> n'en est pas vraiment une. Elle sert à gérer deux situations :
1.Le calcul se termine normalement (pas d'erreur) mais autrement que lorsqu'on a atteint la fin de la boucle sur les pas de temps. Pour l'instant, le seul cas est celui où le pilotage a atteint ses bornes ;
2.Un événement est déclenché à l'extérieur de la boucle sur les pas de temps, c'est-à-dire pendant le post-traitement (détection d'instabilité par modes vibratoires) ;
Les différentes boucles sont réparties entre op0070.f, nmnewt.f (statique et dynamique implicite) et ndexpl.f (dynamique explicite). Pour la boucle sur les résidus d'équilibre, il faut regarder dans la routine nmcore.f, appelée par nmconv.f. Pour la boucle sur les points fixes, on utilise les routines nmible.f/nmtble.f, contenues dans nmnewt.f.

3.2 État des boucles

L'état d'une boucle est stocké dans l'objet SDERRO (§19) , on y accède par nmleeb.f et nmeceb.f . Il existe six états :
1.CONT : la boucle continue ;
2.CONV : la boucle a convergé (évènements de type convergence et divergence) ;
3.ERRE : une erreur (évènement de type erreur) est survenue pendant la boucle, on va la traiter ;
4.EVEN : un événement (événement de type informatif) est survenu pendant la boucle ;
5.STOP : une erreur (évènement de type erreur) est survenue pendant la boucle, on s'arrête ;
6.CTCD : état particulier de la boucle de Newton pour le contact discret ;

3.3 Les évènements

On peut classer les évènements selon leur origine :
•Les évènements définis par l'utilisateur dans DEFI_LIST_INST (par exemple, DELTA_GRANDEUR) ;
•Les évènements intrinsèques à l'algorithme: les erreurs et la convergence des différentes boucles ;

3.3.1 Types des évènements

Un évènement est défini par son type qui est une chaîne de caractère en deux parties <XXXX_YYYY>. La première chaîne <XXXX> décrit le type de l'évènement :
•Les évènements donnant lieu à une erreur sont préfixés par <ERRC_*> ou <ERRI_*> ;
•Les évènements donnant lieu à une convergence sont préfixés par <CONV_*> ;
•Les évènements donnant lieu à une divergence sont préfixés par <DIVE_*> ;
•Les évènements informatifs sont préfixés par <EVEN_*> ;
La seconde chaine <YYYY> décrit le niveau de boucle de l'évènement :
•Boucle sur les résidus d'équilibre <*_RESI> ;
•Boucle sur les itérations de Newton <*_NEWT> ;
•Boucle de point fixe pour le contact <*_FIXE> ;
•Boucle sur les pas de temps <*_INST> ;
Cette information décrit dans quelle boucle les évènements peuvent potentiellement se déclencher .

3.3.1.1 Évènements de type erreur <ERR*_*>

Des erreurs sont à traiter immédiatement (c'est-à-dire dès déclenchement de l'évènement) car elles sont bloquantes pour le processus, elles sont notées <ERRI>. Les erreurs à traiter uniquement à convergence de la boucle sont notées <ERRC>.
Par exemple :
•<ERRI_NEWT> est une erreur à traiter immédiatement dans une itération de Newton comme un défaut d'intégration de la loi de comportement empêche ou une matrice non factorisable ;
•<ERRC_NEWT> est une erreur à traiter à convergence de Newton, par exemple une sortie d'un critère physique hors des bornes de son domaine de définition ;
•<ERRI_FIXE> est une erreur à traiter immédiatement dans une boucle de point fixe ;

Important : les erreurs typées <ERRI_CALC> entraine l'arrêt immédiat du calcul car aucune action ne peut les traiter. Ce sont les arrêts en manque de temps CPU et l'arrêt extérieur par utilisateur.

3.3.1.2 Évènements de type convergence <CONV_*>

Les évènements de type convergence se traitent à chaque niveau de boucle. Cette convergence peut-être liée à une fonctionnalité activée ou pas (cf §12).

3.3.1.3 Évènements de type divergence <DIVE_*>

Les évènements de type divergence se traitent à chaque niveau de boucle. Cette divergence peut-être liée à une fonctionnalité activée ou pas (cf §12).
En pratique, l'état de chaque boucle est plutôt traitée en di*vergence*. En effet, pour éviter d'avoir à vérifier qu'une fonctionnalité spécifique est active (pilotage, Deborst, contact, etc...) et donc tester la convergence d'une boucle en vérifiant cette fonctionnalité, on préfère émettre un évènement de divergence qu'un évènement de convergence : on ne peut avoir divergence que si la fonctionnalité est activée, alors qu'on pourrait avoir convergence même si la fonctionnalité n'est pas activée.

3.3.1.4 Évènements de type informatif <EVEN>

Les évènements de type informatifs sont les autres évènements (ni erreur, ni convergence). Par défaut, ils ne servent qu'à donner des indications à l'utilisateur (souvent sous la forme de messages d'alarmes, le passage de RESI_GLOB_RELA à RESI_GLOB_MAXI par exemple). Certains ne sont activables que sur demande de l'utilisateur (commande DEFI_LIST_INST).
Ces évènements pourront aussi être traité explicitement (autrement que par l'émission d'une alarme ou d'une information) via DEFI_LIST_INST.
Ces évènements ne sont pas liés à un niveau de boucle.

3.3.1.5 Le cas des code-retour

Un code-retour est un entier unique donnant le statut d'un opération. Il est possible de lier un évènement à la valeur d'un code-retour. Chaque valeur du code-retour peut générer un évènement différent, de n'importe quel type. Les codes retour -1 et 0 ont toujours le même sens :
•-1 : on n'a rien fait (pas de factorisation , pas d'intégration de loi de comportement, etc.) ;
•0 : on a fait quelque chose et tout s'est bien passé ;
Les autres valeurs de codes retour ont un sens dépendant de leur type :

Type	Description	Signification du code	
FAC	Retour de la factorisation	-1	Pas de factorisation
		0	Tout s'est bien passé
		1	La matrice est singulière
		2	La factorisation a échoué
		3	On ne sait pas dire si la matrice est singulière

PIL	Retour du pilotage	-1	Pas de pilotage
		0	Tout s'est bien passé
		1	Pas de solution de l'équation de pilotage
		2	On a atteint une borne (fin du calcul)
LDC	Retour de l'intégration de la loi de comportement	-1	Pas d'intégration du comportement
		0	Tout s'est bien passé
		1	Échec lors de l'intégration de la loi de comportement
		2	Un paramètre physique n'est pas dans son domaine de définition
		3	La contrainte SIZZ n'est pas nulle (algorithme de De Borst)
CTC	Retour du traitement du contact discret	-1	Pas de contact discret
		0	Tout s'est bien passé
		1	Le nombre maximum d'itérations de contact a été atteint
		2	La matrice du contact est singulière

3.4 Gestion des évènements

3.4.1 Modification, ajout ou suppression d'un évènement

La plupart des évènements standardisés sont prévus dans la structure de données et les utilitaires qui y sont liés. Dans la plupart des cas, il ne s'agit de modifier que la routine nmcrga.f (voir §19 en y ajoutant la caractéristique du l'évènement dans les DATA en début de routine .

Description	**DATA**
Nom de l'évènement	NEVEN
Nom du code-retour lié à l'évènement (XXX si pas code-retour)	NCRET
Valeur du code-retour lié à l'évènement (99 si pas code-retour)	VCRET
Type et niveau de déclenchement de l'évènement	TEVEN
Fonctionnalité activant un évènement de type convergence ou divergence	FEVEN
Code du message à afficher quand l'évènement se déclenche	MEVEN

Le système de vérification des messages fait que le code du message à afficher (MEVEN) n'est pas suffisant, il faut aussi impacter nmevim.f.

3.4.2 Émissions des évènements

Pour déclencher un évènement le développeur doit le prévoir dans l'algorithme en appelant la routine nmcrel.f. Par exemple:

Commande	**Effet**
CALL NMCREL(SDERRO,'ERRE_TIMN',.TRUE.)	Manque de temps CPU pendant une itération de Newton
CALL NMCREL(SDERRO,'RESI_MAXR',.TRUE.)	Passage de RESI_GLOB_RELA à RESI_GLOB_MAXI
CALL NMCREL(SDERRO,'DIVE_FIXG',. FALSE .)	Pas de divergence de la boucle de point fixe sur la géométrie

Si l'évènement est de type <ERRI_BOUC>, le statut de la boucle <BOUC> est automatiquement modifié. C'est une précaution pour éviter que le développeur oublie de traiter l'évènement (et donc risque de provoquer des résultats faux). Une erreur immédiate de niveau <ERRI> est répercutée à tous les niveaux de boucle pour les mêmes raisons.
Pour activer un évènement lié à un code-retour, il faut utiliser la routine nmcret.f au lieu de nmcrel.f. Par exemple CALL NMCRET(SDERRO,'LDC',LDCCVG) s'occupe de transformer en évènement la valeur du code retour de la loi de comportement.

3.4.3 Traitement des évènements

L'idée principale est de traiter les évènements à tous les niveaux de boucle, de changer l'état des boucles selon le résultat de ce traitement.

3.4.3.1 Évènements de type convergence et divergence

Les évènements de type convergence ou divergence sont examinés à chaque niveau de boucle, dans une routine dédiée appelée nmevcv.f . Cette routine modifie l'état de la boucle :
1. On suppose initialement que la boucle continue (état CONT, cf §31)
2. On boucle sur les évènements de type <CONV_*> et <DIVE_*> en prenant en compte des éventuelles fonctionnalités activées ;
3. On calcule l'état résultat pour ce niveau de boucle : si tous les évènements de divergence sont faux et tous les évènements de convergence sont vrais, alors ce niveau de boucle est dans l'état convergé ;
4. On vérifie les états de convergence des boucles intérieures à ce niveau. Pour que l'état final de cette boucle soit convergé, toutes les boucles intérieures doivent être convergées. Si c'est le cas, la boucle passe dans l'état convergé (état CONV, cf §31)
5. On modifie l'état de la boucle courante en appelant la routine nmeceb.f ;
6. La routine nmeceb.f transmet l'état de la boucle aux boucles supérieures (évite les erreurs non-traitées) ;

Boucle	Code	Routine	Routine appelante
Résidus	RESI	nmcvgr.f	nmconv.f
Newton	NEWT	nmcvgn.f	nmnewt.f
Point fixe	FIXE	nmcvgf.f	nmtble.f
Pas de temps	INST	nmcvgp.f	op0070.f
Calcul	CALC	nmcvgc.f	op0070.f

3.4.3.2 Évènements de type erreurs

Les évènements de type erreur sont particulièrement sensibles et doivent être traités de manière très rigoureuse pour éviter les résultats faux. En premier lieu, l'évènement de type erreur a été émis par le biais des routines standards : nmcrel.f et nmcret.f. On rappelle que dans le cas des évènements erreur, on modifie systématiquement l'état de la boucle en mode ERRE dans l'optique d'éviter de continuer ces boucles si jamais l'erreur n'a pas été traitée convenablement par l'algorithme (oubli du développeur).
De manière systématique, les évènements de type erreur sont testés dans l'algorithme via la routine nmltev.f. Si un évènement de type erreur est activé, un GOTO est immédiatement fait dans le programme.

4 Algorithme général

Le processus est standardisé au maximum , pour chaque niveau de boucle :
•Évaluation de la convergence par appel aux routines nmcvg*.f ;
•Action suite à la situation de la boucle par appel aux routines nmact*.f ;

4.1 Lectures et initialisations globales

La routine op0070.f gère l'algorithme global et se découpe en trois parties :
•Lecture et initialisations des données ;
•Boucle sur les pas de temps ;
•Post-traitements ;
•Gestion des erreurs globales ;
•Archivage ;
Pour la lecture et l'initialisation des données, on se reportera en particulier aux informations relatives à la gestion des SD (§6). L'ensemble des initialisations faites à ce niveau sera valable pour tout le calcul. La routine principale d'initialisation est nminit.f. Voici les opérations prévues dans cette routine :

Opérations d'initialisations dans nminit.f	**Routine**
Création de la SDTIME	nmcrti.f
Création de la SDSTAT	nmcrst.f
Saisie et vérification de la cohérence du chargement de contact	nmdoct.f
Création du profil de la matrice et de la SDNUME	nmnume.f
Création des variables-chapeaux	nmchap.f
Création du vecteur des fonctionnalité activées FONACT	nmfonc.f
Création de la SD pour le contact RESOCO	cfmxsd.f
Création de la SD pour les liaisons unilatérales RESOCU	cucrsd.f
Création des vecteurs dans les variables-chapeaux	nmcrch.f
Création de la structure de données pilotage	nmdopi.f
Duplication NUME_DDL pour SDNUME	nmpro2.f
Préparation de la carte COMPOR (transformation en CHAM_ELEM_S)	nmdoco.f
Création de la SDIETO	nmetcr.f
Lecture de l'état initial	nmdoet.f
Création de SDDISC et SDOBSE	diinit.f
Initialisation des variables de commande	nmcrvv.f
Pré-calcul des MATR_ELEM constants au cours du calcul	nminmc.f
Pré-calcul des VECT_ELEM et VECT _ASSE constants au cours du calcul	nminvc.f
Création de la SDCRIT	nmcrcv.f
Initialisation du calcul par sous-structuration	nmlssv.f
Création de la SD pour le chargement FORCE _SOL	nmexso.f
Calcul de l'accélération initiale	accel0.f
Création de la SDCONV	nmcrcg.f
Initialisation de la SDIMPR	nminim.f
Pré-calcul des MATR_ASSE constants au cours du calcul	nminma.f
Réalisation d'un e observation initiale	nmobsv.f

Création de la SD RESULTAT (EVOL_NOLI)	nmnoli.f
Création de la table des grandeurs (pour ERRE_THM)	cetule.f
Calcul du second membres initial pour les schémas multi-pas en dynamique	nmihht.f
Initialisations SDTIME et SDSTAT	nmrini.f

L'algorithme général de op0070.f est résumé sur l'organigramme (1).

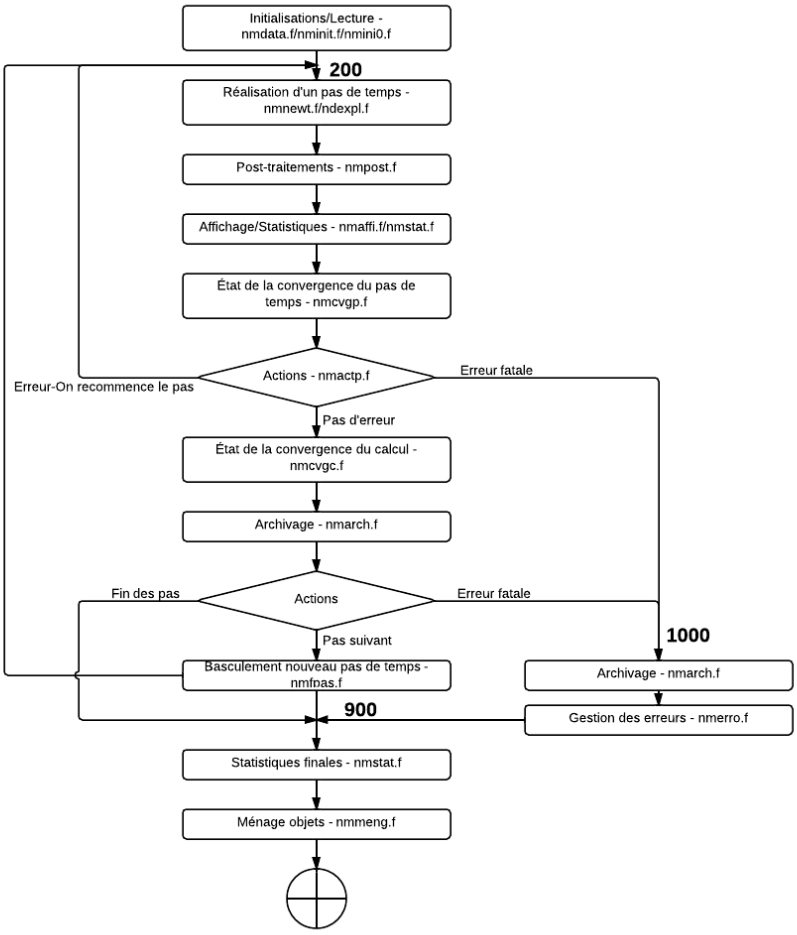

Figure 1: Organigramme général de `op0070.f`

4.2 Réalisation d'un pas de temps

Un pas de temps utilise plusieurs niveaux de boucles imbriqués :
- Des boucles de point fixe, utilisées exclusivement pour le contact (jusqu'à trois niveaux de boucle) ;
- Une boucle sur les itérations de Newton. Un processus de Newton contient une *prédiction d'Euler* et des *corrections de Newton* ;

L'algorithme général de `nmnewt.f` est résumé sur l'organigramme (2) .

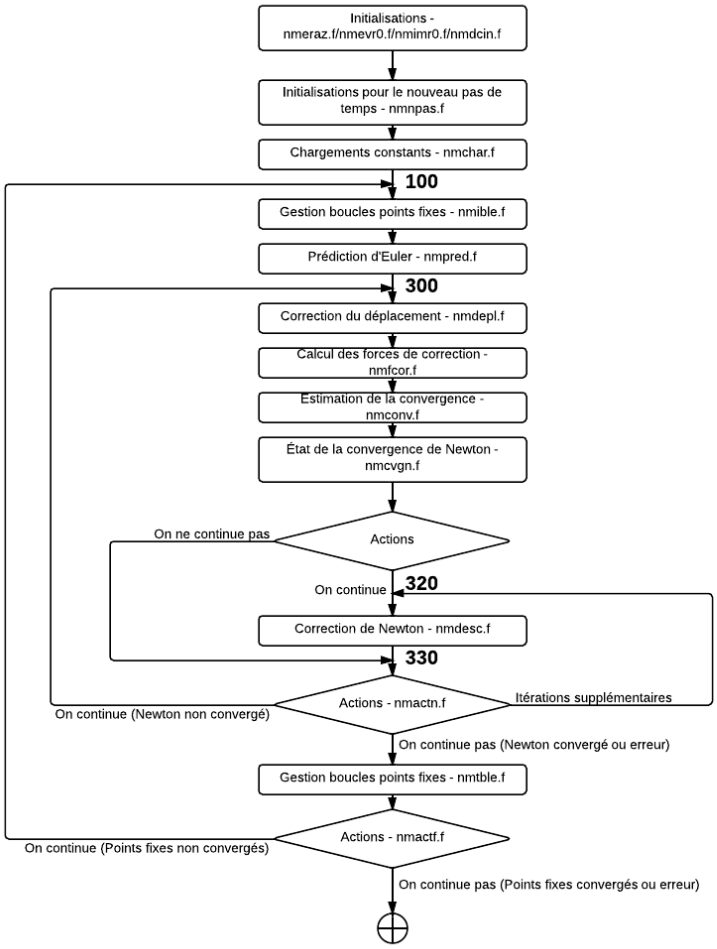

Figure 2: Organigramme général de `nmnewt.f`

4.2.1 Initialisations du pas

Chaque pas de temps est initialisé par une série de routines :
•Les routines `nmeraz.f`, `nmevr0.f` s'occupent d'initialiser la gestion des évènements ;
•La routine `nmdcin.f` vérifie que la découpe du pas de temps est correcte (contrôle du nombre de niveau de découpe d'un pas de temps à l'autre ;
La routine la plus importante est la routine `nmnpas.f` . Elle réalise les opérations suivantes :
•Les routines `nmimr0.f` et `nminin.f` initialisent l'affichage, en particulier le tableau de convergence (ce qui permet de faire varier la nature de celui-ci d'un pas de temps à l'autre) ;
•La routine `nmvcre.f` prépare les variables de commande pour le pas de temps courant ;
•On initialise à zéro l'incrément de déplacement cumulé depuis le début du pas de temps (vecteur `DEPDEL` , voir § 14). Attention l'initialisation est spéciale à cause de la prise en compte des grandes rotations ;
•On initialise toutes les données pour la dynamique, en particulier les différentes coefficients (voir § 25) dans `ndnpas.f` ;
•On initialise différents informations pour Newton-Krylov et le contact (routines `cfinit.f` , `mmapin.f` et `nmnkft.f`) ;

4.2.2 Prédiction d'Euler

La prédiction d'Euler est une estimation de l'incrément de déplacement en linéarisant le problème par rapport au temps. Il existe plusieurs méthodes et plusieurs types de matrice utilisables. Le calcul proprement dit est réalisé dans la routine nmpred.f (et ses -jolies- filles), au moyen des principes établis dans le §40.

4.2.3 Mise à jour des champs

La mise à jour des champs est réalisée dans la routine nmdepl.f. Cette mise à jour consiste à modifier les vecteurs solutions (déplacements, vitesses et accélérations), en prenant en compte éventuellement :
- La recherche linéaire sans pilotage;
- La recherche de η pour le pilotage avec ou sans recherche linéaire ;
- Le contact (formulation discrète) ou les liaisons unilatérales ;

Voici les phases :
1.Recalcul des efforts extérieurs, routine nmfext.f ;
2.Conversion des incréments de solution DEPSO1 et DEPSO2 (voir §14), issus de la résolution du système linéaire, vers les incréments en déplacement/vitesse/accélération (prise en compte des coefficients de changement de schéma), via la routine nmincr.f ;
3.Calcul recherche linéaire et pilotage, routines nmreli.f, nmpich.f et nmrepl.f ;
4.Mise à jour de la direction de descente (prise en compte des coefficients issus de la recherche linéaire et du pilotage), routine nmpild.f ;
5.Modification des déplacements par le contact et les liaisons unilatérales , par la routine nmcoun.f ;
6.Actualisation des champs de solutions dans nmmajc.f ;
La phase 39 consiste à mettre à jour les champs « plus » (DEPPLU, VITPLU, ACCPLU) et les champs cumulés (DEPDEL, VITDEL, ACCDEL), en prenant en compte les éventuelles grandes rotations (mise à jour des quaternions) et de l'endommagement aux nœuds.

4.2.4 Forces de correction

Puisque les déplacements ont été modifiés (voir § 39), il est nécessaire de ré-évaluer les forces variables : soit les forces extérieurs de type « suiveuse », soit les forces intérieur e s et les réactions de contact/frottement, soit les forces liées à la dynamique (inertie, amortissement). L'ensemble est réalisé dans la routine nmfcor.f .

4.2.5 Estimation de la convergence

L'estimation de la convergence de Newton est un moment critique, qui va conditionner la justesse du calcul. L'ensemble de l'estimation est réalisé dans la routine nmconv.f . Cette routine s'occupe du calcul des résidus (nmresi.f), de l'estimation de leur convergence (nmcore.f). C'est donc dans cette routine qu'on va traiter la boucle <RESI> sur les résidus d'équilibre. Mais on doit aussi prendre en compte le nombre d'itérations de Newton, le cas de la convergence du contact (formulations discrète ou continue), de la méthode de De Borst ou de la méthode IMPLEX. ***Il convient d'être très prudent dans la modification de cette routine.***

4.2.6 Correction de Newton

Si le processus n'a pas convergé, on procède au calcul d'une correction de Newton. On réalise donc le calcul d'un système linéaire (voir §40) dans la routine nmdesc.f (comme direction de ***desc***ente).

4.2.7 Boucle sur les points fixes

Il existe trois boucles dite « de point fixe » entre la boucle sur les pas de temps et la boucle sur les itérations de Newton. Ces boucles sont utilisés pour le contact :
1.Boucle de point fixe sur la géométrie ;
2.Boucle de point fixe sur les seuils de frottement ;
3.Boucle de point fixe sur les statuts de contact ;
Ces trois boucles sont gérées par le duo de routine nmible.f/nmtble.f, via le passage de la variable NIVEAU, qui indique dans quel type de boucle de point fixe on se trouve actuellement.

5 Construction et résolution des systèmes

Une partie importante de l'algorithme non-lin é aire consiste à résoudre des systèmes linéaires , que l'on construit en quatre temps :
- Calcul et assemblage des chargements ;
- Calcul et assemblage des seconds membres ;
- Calcul et assemblage de la matrice ;
- Résolution du système linéaire ;

Dans cette partie, nous allons décrire les différentes phases et les principales routines à considérer.

5.1 Systèmes à résoudre

Actuellement, il existe plusieurs systèmes différents qui sont résolus dans l'opérateur `op0070.f` . Ils auront toujours la forme suivante (sauf pour le post-traitement, qui utilise les opérateurs de recherche de valeurs propres) :

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

$\left[K\right]$ est la matrice de rigidité ou une combinaison linéaire de matrices et $\left[B\right]$ est la matrice des Lagrange de Dirichlet.

$\left\{\delta r\right\}$ est l'incrément de solution des valeurs nodales inconnues du système et $\left\{\delta s\right\}$ est l'incrément des paramètres de Lagrange associés à la dualisation des conditions limites. Concrètement, les « valeurs nodales inconnues » du systèmes peuvent être des déplacements, des vitesses, des accélérations, des pressions, des températures, etc. Les deux seconds membres sont également des valeurs nodales. Le détail des systèmes est donné dans les documentations de référence [R5.03.01] et [R5.05.05].

5.2 La routine `merimo.f`

Pour le calcul des matrices tangentes (options `FULL_MECA`, `FULL_MECA_ELAS`, `RIGI_MECA_TANG`, `RIGI_MECA_ELAS`, `RIGI_MECA`, `RIGI_MECA_IMPLEX`) et les efforts intérieurs (option `RAPH_MECA`), on passe systématiquement par la routine `merimo.f`. Cette routine prépare les champs d'entrées (nombreux dans ce cas), elle est appelée à la fois pour le calcul des vecteurs élémentaires pour les efforts intérieurs (voir §42) mais aussi pour le calcul des matrices élémentaires tangentes de rigidité (voir §40).

5.3 Calcul des matrices

Nous considérons ici des matrices élémentaires (`MEELEM`) et des matrices assemblées (`MEASSE`). Tout comme pour le chargement (§42) , ce calcul utilise un système en deux temps :
- Création d'une ***liste locale*** des matrices à calculer et/ou assembler ;
- Calcul et/ou assemblage des matrices ;

Pour la phase de création de la liste des matrices à calculer et/ou assembler, il y a plusieurs routines (contrairement aux chargements qui n'utilisent que la routine `nmchar.f`). Ces routines sont les suivantes :

Opérations	Routine
Construction de la liste locale pour les matrices utilisées en dynamique explicite	`ndxprm.f`
Construction de la liste locale pour les matrices utilisées en correction	`nmcoma.f`
Construction de la liste locale pour les matrices utilisées en prédiction	`nmprma.f`
Construction de la liste locale pour les matrices utilisées en post-traitement (modes vibratoires ou modes de flambement)	`nmflma.f`
Construction de la liste locale pour les matrices constantes durant tout le calcul	`nminmc.f`

Pour la phase de calcul et/ou assemblage des matrices, on retrouve des routines similaires au cas des chargements.

Opérations	Routine
Ajout d'une matrice à assembler et/ou calculer dans la liste locale Initialisation de la liste locale	nmcmat.f nmcmat.f
Aiguillage calcul et/ou assemblage des matrices	nmxmat.f
Calcul des matrices	nmcalm.f
Assemblage des matrices	nmassm.f
Calcul et assemblage des matrices de rigidité	nmrigi.f

Il y a néanmoins un exception notable : le calcul des matrices tangentes non-linéaires utilise un appel direct par nmrigi.f car cela nécessite des paramètres supplémentaires (appel à merimo.f, voir 40) dont n'ont pas besoin les autres matrices. Le type de matrice MERIGI utilise donc nmrigi.f à la place de nmcalm.f et nmassm.f .
Si on désire modifier une matrice, il faut donc :
•Ajouter son calcul et/ou son assemblage au bon endroit (ndxprm.f, nmcoma.f, nmprma.f, nmflma.f ou nminmc.f) selon le cas, voire ajouter une nouvelle routine ;
•Ajouter le calcul élémentaire et/ou l'assemblage dans nmcalm.f et nmassm.f ;
Les routines nmxmat.f et nmcmat.f de manipulation de la liste locale ***ne doivent pas être modifiées*** .

5.4 Calcul de la matrice résultante MATASS

Selon le type de matrice résultante que l'on désire obtenir (en prédiction, en correction ou pour l'accélération initiale), on utilise trois routines globales qui vont s'occuper de cette construction. Ce sont les routines nmprac.f , nmcoma.f et nmprma.f. Ces trois routines gèrent également l'affichage (option de calcul de la matrice de rigidité) et les statistiques (temps passé dans les opérations). Elles ont le même schéma de principe :
•Calcul et/ou assemblage de matrices élémentaires (MEELEM) dans des matrices assemblées (MEASSE), (voir § 40) ;
•Gestion des paramètres de réactualisation des matrices (rigidité, masse, amortissement), routine nmchrm.f ;
•Gestion du type de la matrice de rigidité (nom de l'option), routine nmchoi.f ;
•Construction de la matrice résultante par combinaison linéaire des autres matrices. En dynamique, les différentes contributions dans la matrice résultante (matrices de masse, d'amortissement et de rigidité) se combinent en utilisant un coefficient multiplicateur dépendant du schéma en temps utilisé et du pas de temps, on récupère ces coefficients via la routine d'accès NDYNRE (voir § 25). C'est la routine nmmatr.f ;
•Factorisation (ou pas) de la matrice résultante par appel à la routine preres.f ;

Opérations	Routine
Calcul de la matrice assemblée résultante pour le calcul de l'accélération initiale	nmprac.f
Calcul de la matrice assemblée résultante pour la phase de prédiction	nmprma.f
Calcul de la matrice assemblée résultante pour la phase de correction	nmcoma.f
Gestion des paramètres de réactualisation des matrices (rigidité, masse , amortissement)	nmchrm.f
Gestion du type de la matrice de rigidité (nom de l'option)	nmchoi.f
Choix de re-création de la numérotation	nmrenu.f
Calcul de la matrice résultante MATASS	nmmatr.f
Prise en compte des chargements suiveurs avec leur fonction multiplicatrice	ascoma.f
Prise en compte de la modification de la matrice résultante par le contact/frottement (méthode DISCRETE)	nmasfr.f

5.5 Calcul du second membre

Le calcul du second membre est la partie la plus différente d'un système linéaire à l'autre, il va contenir :
•Les chargements donnés (voir le § 42)

•Les forces internes provenant de l'intégration du comportement (§42) ;
•Les quantités liées aux conditions limites de Dirichlet comme les réactions d'appui (§43) ;
•Les quantités liées aux variables de commande (§43) ;
•Les différentes quantités liés au contact/frottement (pas de détails dans ce document) ;
•Les contributions d'inertie et d'amortissement pour la dynamique (§43) ;
Dans le cas de la dynamique, pour les schémas multi-pas (Newmark complet avec MODI_EQUI='OUI') on va de plus ajouter les contributions des pas de temps précédents.

5.5.1 Calcul des chargements

Il y a trois modes d'évaluation des chargements et deux phases. Les modes sont les suivants :
•Mode <ACCI> des chargements pour l'évaluation de l'accélération initiale en dynamique ;
•Mode <FIXE> des chargements pour l'évaluation des chargements fixes ne dépendant pas de la solution ;
•Mode <VARI> des chargements pour l'évaluation des chargements variables dépendant de la solution (chargements suiveurs) ;
Il ya également deux phases. Soit on est en correction, soit on est en prédiction. Ces phases ne sont utiles que pour des cas très particuliers : amortissement modal, méthode IMPLEX et impédance modale.
La routine principale qui calcule les chargements est nmchar.f . Dans cette routine, suivant les fonctionnalités activées (FONACT) et suivant le mode/phase de calcul, on va provoquer le calcul des vecteurs élémentaires (variable-chapeau VEELEM) et leur assemblage en vecteurs assemblées (variable-chapeau VEASSE). Pour cela, on utilise un certain nombre de routines utilitaires qui gèrent des listes locales à nmchar.f .

Opérations	Routine
Ajout d'un chargement à assembler et/ou calculer, avec une option éventuelle Initialisation de la liste des chargements (initialisations listes locales de nmchar.f)	nmcvec.f nmcvec.f
Aiguillage calcul et/ou assemblage des chargements	nmxvec.f
Calcul des chargements	nmcalv.f
Assemblage des chargements	nmassv.f

Certains chargements n'ont pas de phase de calcul élémentaire mais seulement la création d'un vecteur assemblé directement. Si on désire ajouter un chargement, il faut donc :
•Définir quelle phase et quel mode va l'activer ;
•Ajouter son calcul et/ou son assemblage dans nmchar.f selon la fonctionnalité activée ;
•Ajouter le calcul élémentaire et/ou l'assemblage dans nmcalv.f et nmassv.f ;
Les routines nmxvec.f, nmcvec.f ***ne doivent pas être modifiées***.

5.5.2 Calcul des quantités liées aux efforts intérieurs

Le calcul des efforts intérieurs peut être réalisé de deux manières :
•Par intégration de la loi de comportement, c'est l'option de calcul RAPH_MECA ;
•Par ré-utilisation des contraintes calculées par ailleurs, c'est l'option de calcul FORC_NODA ;
Cette distinction est importante car elle n'engendre pas les mêmes coûts (le second cas est beaucoup plus rapide). L'intégration du comportement est une opération couteuse qui implique de modifier variables internes et contraintes, et, souvent, de résoudre localement (à chaque point de Gauss), un système non-linéaire. Comme le calcul des contraintes est également nécessaire lors de l'évaluation des matrices tangentes, le comportement est aussi intégré lors du calcul des matrices tangentes (option FULL_MECA). En statique, la phase de prédiction calcule les efforts intérieurs par l'option FORC_NODA. En dynamique, on intègre systématiquement le comportement.

Opérations	Routine
Calcul des vecteurs élémentaires pour les efforts intérieurs (intégration du comportement)	nmfint.f
Assemblage des vecteurs élémentaires pour les efforts intérieurs (intégration du comportement)	nmaint.f

Le calcul de l'option FORC_NODA est fait par le mécanisme d'évaluation des chargements (§42).

5.5.3 Calcul des quantités liées aux conditions limites dualisées

La dualisation des conditions limites de Dirichlet (voir [R3.01.01]) entraîne le calcul de deux quantités de type vecteur assemblée :
•Les réactions d'appui, par le produit de la matrice des conditions limites dualisées avec les Lagrange correspondant aux degrés de liberté $[B].\{\lambda\}$, il s'agit du vecteur identifié par CNDIRI ;
•La valeur des degrés de liberté imposée par dualisation $[B].\{u\}$. A convergence, cette quantité sera égale aux conditions limites données $\{u^d\}$, le vecteur est identifié par CNBUDI ;

Opérations	Routine
Calcul des vecteurs élémentaires pour les réactions d'appui	nmdiri.f
Assemblage des vecteurs élémentaires pour les réactions d'appui	nmadir.f
Calcul et assemblage des quantités imposées par dualisation	nmbudi.f

5.5.4 Calcul des quantités liées aux variables de commande

Le calcul de ces quantités se fait par le mécanisme d'évaluation des chargements (§42), type CNVCF0, CNVCF1 et CNVCPR.

5.5.5 Calcul des quantités constantes pendant le calcul

Un certain nombre de vecteurs sont constants durant le calcul, ils utilisent alors un mécanisme similaire aux chargements (§42), à l'aide des routines nmxvec.f, nmcvec.f, nmcalv.f et nmassv.f, sauf qu'on passe par la routine nminvc.f au lieu de nmchar.f.

Opérations	Routine
Calcul et assemblage des vecteurs constants durant le calcul	nminvc.f

5.5.6 Calcul des quantités liées à la dynamique – Forces d'inertie et d'amortissement

Le calcul de ces quantités se fait par le mécanisme d'évaluation des chargements (§ 42), type CNDYNA . Ces quantités n'existent pas à l'état élémentaire puisqu'ils sont le résultat d'un produit matrice/vecteur.

Opérations	Routine
Calcul des quantités dynamiques pour le second membre	ndfdyn.f
Calcul des forces d'inertie	nminer.f
Calcul des forces d'amortissement	nmhyst.f

Selon les schémas en temps, il est nécessaire d'utiliser des coefficients multiplicateurs spécifiques devant chacune de ces quantités, on récupère ces coefficients via la routine d'accès NDYNRE (voir § 25).

5.5.7 Seconds membres résultants

Il en reste plus qu'à construire les différents seconds membres par combinaison linéaire de toutes les quantités détaillées précédemment. Les routines principales de construction de ces seconds membres sont au nombre de sept :

Opérations	Routine
Calcul du second membre pour la correction	nmassc.f
Calcul du second membre pour la prédiction – Option DEPL_CALCULE ou EXTRAPOLE	nmassd.f

Calcul du second membre pour l'accélération initiale	nmassi.f
Calcul du second membre pour la prédiction	nmassp.f
Calcul du second membre pour la prédiction – Cas statique	nsassp.f
Calcul du second membre pour la prédiction – Cas dynamique implicite	ndassp.f
Calcul du second membre pour la prédiction – Cas dynamique explicite	nmassx.f

Toutes ces routines reposent sur les mêmes principes :
•Récupération des quantités nécessaires, déjà assemblées ou calculées localement dans ces routines, via la variable-chapeau VEASSE (voir les §42 à §43) ;
•Récupération des différents coefficients multiplicateurs. En dynamique, selon les schémas en temps, il est nécessaire d'utiliser des coefficients multiplicateurs spécifiques devant chacune de ces quantités, on récupère ces coefficients via la routine d'accès NDYNRE (voir § 25) ;
•Construction d'un ou de deux seconds membres par combinaison linéaire (utilisation des routines standards vtaxpy.f). Il y a deux seconds membres dans le cas du pilotage ;
Pour améliorer la lisibilité, on utilise des routines utilitaires standardisées pour regrouper les cas les plus fréquents.

Opérations	**Routine**
Calcul des composantes du vecteur second membre pour les chargements de type Dirichlet, fixes au cours du pas de temps, pour le cas normal et le cas piloté	nmasdi.f
Calcul des composantes du vecteur second membre pour les chargements de type Neumann, fixes au cours du pas de temps, pour le cas normal et le cas piloté	nmasfi.f
Calcul des composantes du vecteur second membre pour les chargements de type Neumann, variables (suiveurs) au cours du pas de temps, pour le cas normal (pas de cas piloté)	nmasva.f
Calcul des composantes du vecteur second membre pour les chargements spécifiques de la dynamique, variables (suiveurs) au cours du pas de temps, pour le cas normal (pas de cas piloté)	ndasva.f
Calcul des composantes du vecteur second membre pour les chargements spécifiques de la dynamique, calcul de l'accélération initiale, variables (suiveurs) au cours du pas de temps, pour le cas normal (pas de cas piloté)	nmacva.f

5.6 Résolution du système

La résolution se fait via l'intermédiaire de la routine nmresd.f, qui prend en entrée la matrice résultante assemblée (voir § 41) et le ou les seconds membres idoines (deux seconds membres si on a du pilotage).

Opérations	**Routine**
Résolution du système	nmresd.f
Résolution du système sur les degrés de liberté physiques (appel à resoud.f)	nmreso.f
Résolution du système sur les degrés de liberté généralisés	nmresg.f

La routine de résolution va donc retourner un ou deux champs solutions DEPSO1 et DEPSO2, stockés dans la variable-chapeau SOLALG.

6 Les SD génériques STRUCT

L'idée est d'utiliser un système d'objets génériques appelés STRUCT pour contenir les informations dans op0070.f. Pour l'instant, et pour des raisons de performances liées à l'utilisation massives d'objets JEVEUX, le déploiement des STRUCT a été limité à l'impression (SDIMPR , voir §16).
Il y a trois objectifs :
- Rationaliser l'accès aux SD internes à op0070.f et modérer leur multiplication ;
- Disposer d'un système générique pour toutes les SD ;
- Essayer de faciliter le développement et la maintenabilité ;

Un STRUCT est construit sur les préceptes suivants :
- Routines uniques d'accès et de manipulation ;
- Protection et encapsulation des données ;
- Possibilité de créer des STRUCT génériques pour les opérations fréquentes ;

Un STRUCT est une suite d'objets JEVEUX contenant des informations données par l'utilisateur dans la commande ou créées pour la gestion de l'algorithme.
Un STRUCT est constitué d'objets JEVEUX:
- Deux objets JEVEUX contenant quelques informations nécessaires ;
- Un objet JEVEUX de type V V K24 contenant le nom de tous les paramètres ;
- Un objet JEVEUX de type V V I contenant les paramètres de type <booléen> ;
- Un objet JEVEUX de type V V I contenant les paramètres de type <entier> ;
- Un objet JEVEUX de type V V R contenant les paramètres de type <réel> ;
- Un objet JEVEUX de type V V K24 contenant les paramètres de type <chaîne> ;
- Un objet JEVEUX de type V V K24 contenant les param è tres de type <objet>

Comme on n'utilise que des noms, y compris pour les paramètres de type <objet>, on peut se permettre une certaine récursivité , c'est-à-dire qu'un paramètre de type <objet> peut contenir un STRUCT lui-même. Ce qui permet d'avoir des routines de manipulation (destruction, copie, etc), sans les difficultés (les dangers) à faire du récursif en F77.
La contrepartie est que ces routines de manipulation seront incomplètes au sens habituel l'orienté-objet : elles ne manipuleront que les objets JEVEUX de la STRUCT, le nom des paramètres, les valeurs des paramètres « scalaires » et les noms des paramètres.
Par exemple, la copie d'un STRUCT dans un autre STRUCT est une copie pauvre :
- Recopie du nom des paramètres ;
- Recopie des paramètres scalaires: booléens, entiers, réels, chaînes ;
- Recopie des ***noms*** des paramètres <objet> ;

Idem pour la destruction.

Même si ces deux types de paramètres <chaîne> et <objet> sont fondamentalement des chaines (K24), ce ne sont pas les mêmes. Les types <objet> sont des noms d'<objet>, les types <chaîne> sont des cha î nes.

Un STRUCT a le cycle de vie suivant :
- Création des objets JEVEUX génériques du STRUCT (initialisation du nom des paramètres) ;
- Lecture des informations utilisateurs ;
- Initialisation des valeurs des paramètres ;

6.1 Créer un STRUCT

Un STRUCT est créé par une série de trois routines successives : obcrea.f, routines obcrxx.f et obcrcr.f. Le développeur crée un STRUCT d'un type prédéfini par appel à la routine obcrea.f.

Principe: Chaque type de STRUCT a son propre obcrxx.f (avec xx variant de 00 à 99).

Arbre de construction :
- obcrea.f appelle obcrxx.f (avec xx suivant le type du STRUCT) ;
- obcrxx.f contient le nombre et le nom des paramètres (dans des DATA), il appelle ensuite obcrcr.f ;
- obcrcr.f crée la liste des paramètres avec leur nom (création des objets JEVEUX idoines)

Exemple : CALL OBCREA('AFFICHAGE',SDIMPR) va créer la STRUCT de type 'AFFICHAGE' et de nom SDIMPR.

6.2 Créer un nouveau STRUCT

Pour créer un nouveau type de STRUCT , il faut :
•Impacter obcrea.f pour ajouter le IF vers le nouveau obcrxx.f ;
•Créer le nouveau obcrxx.f avec la liste des paramètres et leur nom ;
Règles :
•La création d'un STRUCT doit se faire exclusivement dans la routine nmini0.f ;
•Un STRUCT peut n'exister que si la fonctionnalité est activée (pilotage, contact). Le test de la fonctionnalité ne peut se faire que sur l'absence ou la présence d'un mot-clef facteur (sauf pour la dynamique: on teste le nom de la commande) ;

6.3 Lire les données utilisateurs

Pas de généricité possible: chaque STRUCT a son propre processus.

Règles :
•Tous les appels au superviseur (getxxx) doivent être fait lors de cette phase. Les getxxx sont interdits dans le reste de l'opérateur (sauf pour détecter des fonctionnalités lors de la création du STRUCT) ;
•Pour des raisons d'encombrement mémoire, les données utilisateurs impliquant beaucoup d'informations (une liste de mailles par exemple) sont une exception à la règle de séparation Création/Lecture/Initialisation. Les objets JEVEUX nécessaires pour stocker ces informations peuvent être créés dans cette phase au lieu de la phase d'initialisation. ;

6.4 Initialiser un STRUCT

Pas de généricité possible: chaque STRUCT a son propre processus.

Règles :
•Les objets JEVEUX contenu dans les paramètres de type STRUCT doivent être nommés suivant la règle générale de Code_Aster: préfixe par le nom de la routine qui fait l'opération.

6.5 Accéder aux valeurs des paramètres

Deux séries de cinq routines pour lire ou écrire un paramètre :
•Récupérer un paramètre : obgeti.f, obgetb.f, obgetr.f, obgetk.f, obgeto.f (respectivement <entier>, <booléen>, <réel>, <chaine> et <objet>) ;
•Écrire un paramètre : obseti.f, obsetb.f, obsetr.f, obsetk.f, obseto.f (respectivement <entier>, <booléen>, <réel>, <chaine> et <objet>) ;
Il y a un contrôle de type lors de l'accès. La routine obgett.f retourne le type de la SD. (et non le type du paramètre).
Exemples :
•CALL OBGETI(SDDYNA,'NBRE_ONDE_PLANE',NCHOND) ;
•CALL OBSETR(SDPILO,'COEF_MULT_PREC',COEMUL) ;

Règles:
•Le nom et le nombre de paramètres ne peut pas être changé postérieurement à la création du STRUCT (aucune routine d'accès direct à ces paramètres) ;

6.6 STRUCT générique : la SDLIST

La SDLIST est une liste de STRUCT. C'est elle-même un STRUCT. Elle contient une liste de STRUCT et peut activer ou désactiver/un STRUCT dans cette liste.

Pour repérer un STRUCT dans la SDLIST, ce STRUCT doit contenir un paramètre de type chaine, unique, donné lors de la création. On peut accéder aux STRUCT de la SDLIST, soit par l'indice dans la liste, soit par son paramètre d'accès. On déclare ce paramètre au moment de la création de la SDLIST.

Description	Routine
Création d'une SDLIST	oblcre.f
Génération automatique d'un nom de STRUCT dans la SDLIST	oblgen.f
Dit si le STRUCT dans la SDLIST est actif ou pas (accès par indice)	oblgai.f
Désactivation de tous les STRUCT dans la SDLIST	oblraz.f
Retourne l'indice d'une STRUCT dans la SDLIST (accès par paramètre)	oblgip.f
(Dés)-activation d'un STRUCT dans la SDLIST (accès par indice)	oblsai.f
(Dés)-activation d'un STRUCT dans la SDLIST (accès par paramètre)	oblsap.f
Récupération du STRUCT dans la SDLIST (accès par indice)	oblgoi.f
Récupération du STRUCT dans la SDLIST (accès par paramètre)	oblgop.f
Mise d'un STRUCT dans la SDLIST (accès par indice)	oblsop.f

Exemples :
1.Création du STRUCT SDLIST :
•CALL OBLCRE(SLCOLO,'TABLEAU_COLONNE','TYPE_COLONNE',NBCOLO) ;
•La STRUCT SDLIST est constituée de ***nbcolo*** STRUCT de type TABLEAU_COLONNE , chaque STRUCT est identifiée par son paramètre TYPE_COLONNE ;
2.Désactivation de tous les STRUCT : CALL OBLRAZ(SLCOLO) ;
3.Activation d'un STRUCT : CALL OBLSAP(SLCOLO,'ITER_NUME',.TRUE.) ;
4.Désactivation d'un STRUCT : CALL OBLSAP(SLCOLO,'ITER_NUME',.FALSE.) ;