Conseils de mise en œuvre de calculs non-linéaires

Résumé

L'objectif de cet document est de donner des conseils à un utilisateur souhaitant réaliser des calculs non-linéaires avec *Code_Aster*. L'utilisateur de calculs linéaires y trouvera aussi des informations et des conseils utiles.

Les aspects suivant seront abordés :

- maillage et modélisation,
- chargements et conditions aux limites,
- matériaux et lois de comportement,
- résolution non-linéaire,
- gestion mémoire/temps et optimisation,
- contrôle de l'erreur et qualité.

Ce document ne traite pas des questions spécifiques à la dynamique.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

1 Maillage et modélisation

Le choix des éléments finis influe directement sur la solution. Par choix, on entend le choix du degré de l'approximation éléments-finis ainsi que le choix d'une formulation. Le degré de l'approximation étant généralement relié au degré des mailles, les paragraphes suivant traitent du choix du degré du maillage et le choix d'une formulation éléments-finis, avec un point particulier pour les problèmes d'incompressibilité.

1.1 Généralités sur le choix du degré du maillage et le choix des éléments finis

De manière générale, l'utilisation d'un maillage linéaire est conseillé en thermique et l'utilisation d'un maillage quadratique est conseillé en mécanique. La commande CREA_MAILLAGE (LINE_QUAD et QUAD_LINE) permet de passer d'un maillage linéaire à un maillage quadratique et vice-versa, mais ne permet pas de générer des éléments quadratiques à bords courbes. Il est d'ailleurs préférable de générer directement les éléments quadratiques avec l'outil de maillage (le module de maillage de Salome-Meca par exemple). Pour la mécanique de la rupture, l'utilisation d'éléments de Barsoum en fond de fissure améliore la qualité du résultat (commande MODI_MAILLAGE / NOEUD_QUART).

Le choix des éléments finis est réalisé dans la commande AFFE_MODELE. En thermique, il est conseillé d'utiliser des éléments lumpés (_DIAG). En mécanique, les éléments finis classiquement utilisés sont les éléments iso-paramétriques (3D, D_PLAN, C_PLAN, AXIS). Cependant, ces éléments sont mal adaptés aux problèmes quasi-incompressibles. Le choix d'une formulation répondant à ce problème fait l'objet du paragraphe suivant.

Pour plus de détails, voir [U2.01.10] « Notice d'utilisation sur le choix des éléments finis »

1.2 Choix d'une formulation pour traiter les problèmes quasi-incompressibles

Un problème est quasi-incompressible si le coefficient de Poisson est proche de 0,5 ($\nu > 0{,}45$) ou si le taux de déformations plastiques est élevé. Cela se traduit par une oscillation de la trace des contraintes. Il existe 6 formulations pour traiter ce problème dans *Code_Aster* dont le choix dépend du type de mailles, du modèle de déformation et du ratio qualité/coût calcul.

Les formulations possibles sont :

- en petites déformations : _SI, _INCO_UP, _INCO_UPG, INCO_UPO
- en grandes déformations : _INCO_UP, _INCO_UPG.

Toutes les formulations sont compatibles avec les modélisations 3D, D_PLAN et AXIS mais en contraintes planes (C_PLAN) seule la formulation sous-intégrée (_SI) est possible.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Le tableau ci-dessous présente la compatibilité entre formulation quasi-incompressible et type de mailles (limitées au principales mailles 3D).

	_SI	_INCO_UP	_INCO_UPG	_INCO_UPO
HEXA20	X	X	X	
PENTA15		X	X	
PYRAM13				
TETRA10	X	X	X	
HEXA8	X			X
PENTA6				X
PYRAM5				X
TETRA4		X		X

Dans le cas général, il est donc nécessaire de mixer différentes formulations pour couvrir tous les types de mailles (par exemple, comme aucune formulation ne gère les mailles PYRAM13, il faut aussi affecter une modélisation iso-paramétrique).

Dans le cas où on reste en petites déformations, il est conseillé d'utiliser un mélange de formulations (le maillage pouvant comporter plusieurs types de mailles) :

	qualité	coût
MODELISATION=('3D', '3D_INCO_UPG', '3D_INCO_UPO')	+++	+++
MODELISATION=('3D', '3D_INCO_UP' , '3D_INCO_UPO')	++	++
MODELISATION=('3D', '3D_SI' , '3D_INCO_UPO')	+	+

Pour plus de détails, voir [U2.01.10] « Notice d'utilisation sur le choix des éléments finis »

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

2 Chargements, conditions aux limites, conditions initiales

2.1 Cœur du problème

Les chargements et conditions aux limites sont importants pour une bonne modélisation. Leur influence sur les résultats est la plupart du temps directe, si bien que l'on peut vérifier leur application par un calcul préliminaire simple, par exemple à l'aide de MECA_STATIQUE, ou en visualisant les conditions aux limites *via* le mot-clé CONCEPT de IMPR_RESU au format MED.

En ce qui concerne les conditions aux limites, il s'agit souvent d'en introduire le minimum pour bloquer les déplacements de solide rigide, donc pour éviter d'avoir des corps flottants dans la structure, ce qui provoquerait un pivot nul ou une matrice singulière lors de la résolution.

Certaines conditions aux limites sont non-linéaires, par exemple le contact unilatéral. Leur vérification ne peut donc être faite à l'aide d'un premier calcul linéaire (MECA_STATIQUE) : les différents objets en contact ne doivent pas avoir de mouvement de corps rigide [U2.04.04] Notice d'utilisation du contact. Si c'est le cas, on peut éviter les pivots nuls en ajoutant des éléments discrets (ressorts) de faible rigidité.

Les chargements (autre que les blocages) peuvent consister soit en conditions de déplacement imposés, soit en efforts imposés, soit de type champ initial imposé.

Dans tous les cas, il importe de bien représenter la réalité. On pourra utiliser avec profit les conditions de symétrie ou d'antisymétrie [1].

De façon générale, une condition de type déplacement imposé ne fournit pas les mêmes résultats qu'une condition de type force imposée : un déplacement imposé sur une partie de la frontière impose que le déplacement (imposé) soit constant en espace sur cette partie, donc apporte de la rigidité, voire des singularités. Prenons l'exemple du poinçonnement d'un massif : le déplacement imposé sur le bord du poinçon induit des singularités de contraintes de même nature que celles qui sont rencontrées au fond d'une fissure.

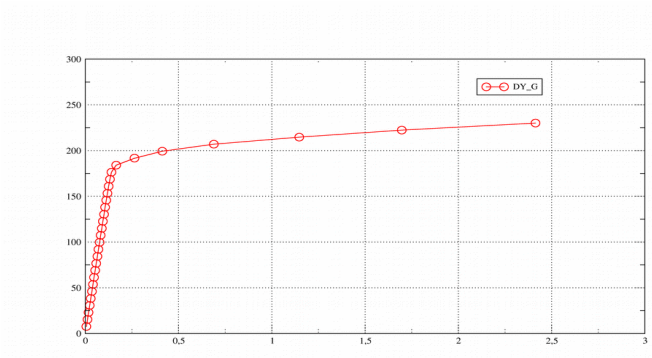


Figure 2.1-1: Exemple de courbe force-déplacement

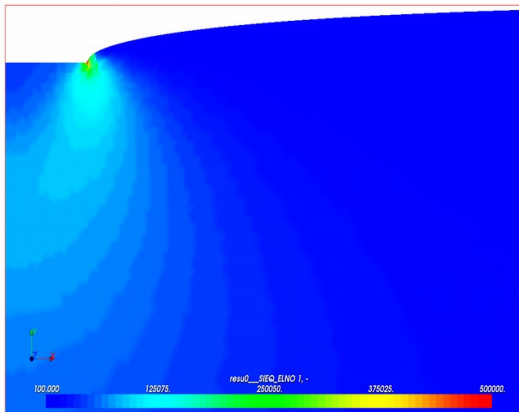


Figure 2.1-2: poinçonnement à déplacement imposé

De plus, pour les calculs non linéaires, la conduite du calcul n'est pas la même : en cas d'adoucissement ou de charge limite, le chargement à force imposée peut être illicite (au-delà du chargement limite). Prenons par exemple le poinçonnement (voir Figure 2.1-1 et Figure 2.1-2) : la réponse en termes de force résultante sur le poinçon en fonction du déplacement montre que plus la force imposée s'approche de la charge limite, plus la convergence sera difficile, jusqu'à être impossible au-delà de la charge limite. Si la modélisation correcte nécessite d'appliquer une force imposée, le problème peut être résolu *via* le pilotage de la force imposée par rapport au déplacement

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

d'un point ou d'un ensemble de points (méthode de continuation ou de longueur d'arc) (voir [R5.03.80] Méthodes de pilotage du chargement).

Les documentations d'utilisation relatives à l'application des chargements et conditions aux limites sont [U4.44.01] AFFE_CHAR_MECA et [U4.44.03] AFFE_CHAR_CINE .

2.2 Précautions pour l'application des chargements

Il est utile pour l'application des chargements de parcourir les différents mots-clés de [U4.44.01] AFFE_CHAR_MECA. Attention au vocabulaire, en particulier, les chargements de type FORCE_FACE, PRES_REP, FORCE_CONTOUR, FORCE_ARETE, FORCE_INTERNE, FORCE_COQUE, FORCE_TUYAU correspondent à des forces réparties (linéique, surfacique, volumique) et **sont exprimées en unité de contraintes** (par exemple des Pa en unité S.I.). Seules les FORCE_NODALE sont exprimées en unité de forces (N), et les forces réparties de poutres (FORCE_POUTRE) en unités de force divisée par une longueur.

En ce qui concerne les pressions (PRES_REP, FORCE_COQUE/PRES), la pression appliquée est positive suivant le sens contraire de la normale à l'élément. Il est fortement conseillé de réorienter ces normales *via* l'opérateur MODI_MAILLAGE, mots-clés ORIE_PEAU_*, ORIE_NORM_COQUE.

Pour les milieux continus 2D, 3D, l'utilisation de forces ponctuelles (FORCE_NODALE) est à proscrire, car elle entraîne toujours des singularités.

L'application de charges fonction du temps peut être effectuée de deux façons :

- soit en définissant des charges constantes, puis en appliquant un coefficient multiplicateur fonction du temps à ces charges lors de la résolution (mot-clé FONC_MULT sous EXCIT dans STAT_NON_LINE),
- soit en définissant directement des charges fonction du temps via AFFE_CHAR_MECA_F.

Dans le cas où les chargements sont de type déplacements imposés, ceux-ci peuvent être introduits soit par AFFE_CHAR_MECA(_F), soit par AFFE_CHAR_CINE(_F). L'utilisation de cette dernière commande permet un gain de temps calcul qui peut être appréciable en non-linéaire (car elle n'ajoute pas de multiplicateurs de Lagrange, il y a donc moins d'équations à résoudre).

Les variables de commandes ne font pas partie des commandes AFFE_CHAR_MECA(_F). Elles sont pourtant, dans beaucoup de modélisations, assimilables à des chargements : par exemple la dilatation thermique peut à elle seule engendrer des états de contraintes et de déformations conduisant à des non-linéarités de comportement. Les variables de commandes sont appliquées via AFFE_MATERIAU.

Il faut toutefois signaler ici une précaution de modélisation concernant ces variables de commande : en effet, dès que le calcul est incrémental, STAT_NON_LINE utilise à chaque pas de temps l'incrément de variable de commande pour calculer les déformations correspondantes (cf. par exemple [R5.03.02] Intégration des relations de comportement élasto-plastique de Von Mises). En général, il est préférable qu'à l'instant initial, la structure soit non contrainte, non déformée. Si ce n'est pas le cas, il faut ajouter un instant préliminaire où la structure est au repos, voir par exemple le test FORMA30 [V7.20.101] FORMA30 - Cylindre creux thermoélastique. En cas de détection de contraintes dues aux variables de commandes à l'instant initial on obtient l'alarme :

```
--------------------------------------------------------------------------------
<A> <MECANONLINE2_97> :
-> Les variables de commandes initiales induisent des contraintes incompatibles :
l'état initial (avant le premier instant de calcul) est tel que les variables de commande
(température, hydratation, séchage...) conduisent à des contraintes non équilibrées.

-> Risque & Conseil : dans le cas d'une résolution incrémentale, on ne considère que la
variation des variables de commande entre l'instant précédent et l'instant actuel. On ne prend
donc pas en compte d'éventuelles contraintes incompatibles dues à ces variables de commande
initiales.
Pour tenir compte de ces contraintes vous pouvez :
– partir d'un instant fictif antérieur où toutes les variables de commande sont nulles ou
égales aux valeurs de référence
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
–   choisir des valeurs de référence adaptées
Pour plus d'information, voir la documentation de STAT_NON_LINE (U4.51.03) mot-clé EXCIT, et le
test FORMA30 (V7.20.101).
--------------------------------------------------------------------------------
```

2.3 Les différents types de conditions aux limites

Les conditions aux limites les plus simples servent principalement à introduire des conditions de symétrie, d'une part, et à empêcher les mouvements de solide rigide, d'autre part. Quand elles sont de type degré de liberté imposé, on peut utiliser soit AFFE_CHAR_MECA, soit (pour optimiser le temps calcul) AFFE_CHAR_CINE [U2.01.02] Notice d'utilisation des conditions aux limites traitées par élimination.

Un conseil méthodologique : il est toujours préférable d'appliquer les conditions aux limites sur des groupes de mailles plutôt que sur des groupes de nœuds ou des listes de nœuds ou de mailles. En effet, les groupes de mailles correspondent à des zones géométriques, et sont conservés lors d'un raffinement du maillage (manuel ou à l'aide de Homard), alors que les groupes de nœuds sont modifiés. C'est pourquoi dans AFFE_CHAR_MECA les mot-clés DDL_IMPO, FACE_IMPO, LIAISON_UNIF, ainsi que tous les mots clés de AFFE_CHAR_CINE comportent le mot clé GROUP_MA, à privilégier donc.

Il est parfois nécessaire d'imposer des relations linéaires entre degrés de liberté. Les mots-clés LIAISON_* de AFFE_CHAR_MECA permettent d'introduire ce type de relation. Les relations élémentaires peuvent être définies à l'aide de LIAISON_DDL, mais cela devient fastidieux si les équations à écrire sont nombreuses. Des fonctionnalités de plus haut niveau sont disponibles, par exemple :

- LIAISON_SOLIDE pour rigidifier une partie de la structure, (en petites déformations et petits déplacements uniquement) [R3.03.02] Conditions de liaison de corps solide ;
- LIAISON_UNIF pour assurer qu'une partie de la frontière gardera le même déplacement (inconnu a priori) ; [U4.44.01] AFFE_CHAR_MECA
- LIAISON_MAIL, LIAISON_GROUP pour relier deux bords,
- et d'autres mots clés spécifiques de AFFE_CHAR_MECA.

De plus il est possible de raccorder (au sens énergétique) des modélisations de natures différentes, via LIAISON_ELEM et plusieurs options (en petites déformations et petits déplacements uniquement)

- '2D_POU','3D_POU', poutres ou discrets avec éléments 2D ou 3D [R3.03.03] Raccords 2D-poutre et 3D-poutre;
- 'COQ_TUYAU','3D_TUYAU', éléments tuyau avec avec plaques et coques, 3D ;
- 'COQ_POU', poutres ou discrets avec plaques et coques [R3.03.06] Liaison coque-poutre.

Remarque : dans certains cas, les conditions aux limites ou les chargements de type déplacement imposé ne conviennent pas car ils apportent trop de rigidité locale. Il peut être intéressant de remplacer ces conditions par une liaison entre la partie de la frontière concernée et un élément discret, sur lequel seront appliquées les conditions de déplacement imposés ou un torseur d'efforts imposés. Cela signifie que les déplacements de la frontière seront égaux en moyenne au déplacement de l'élément discret, sans introduire de contraintes parasites. Par contre, ce procédé introduit des degrés de liberté supplémentaires (multiplicateurs de Lagrange), la résolution des systèmes linéaires peut donc être plus coûteuse.

Le contact unilatéral, avec ou sans frottement, est un type de conditions aux limites particulier. Il est fortement non linéaire, et son traitement dans STAT_NON_LINE / DYNA_NON_LINE nécessite des itérations supplémentaires pour obtenir la convergence. Il est fortement conseillé de consulter le document [U2.04.04 Notice d'utilisation du contact] pour modéliser correctement ces phénomènes.

2.4 Les conditions initiales

Dans AFFE_CHAR_MECA, il est possible de définir des chargements de déformations moyennes ou de contraintes moyennes, globalement uniformes (mots clés PRE_EPSI, PRE_SIGM).

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Il ne faut pas confondre ces chargements et les déformations et contraintes initiales utilisées en non linéaire, car ces quantités n'interviennent pas directement dans l'expression de la loi de comportement, mais seulement au second membre.

Les champs de contraintes, déplacements et variables internes initiaux sont à fournir directement dans STAT_NON_LINE (mot clé ETAT_INIT). Pour construire ces champs, il peut être utile de consulter le document [U2.01.09] Définition analytique d'un champ de contraintes et d'un champ de variables internes initiaux.

3 Matériaux et comportements

3.1 Choix de la loi de comportement

Le choix de la loi de comportement est bien sûr fonction du matériau que l'on modélise, mais également des phénomènes à traiter : par exemple, un même acier sera élasto-plastique à basse température, et visco-plastique à haute température. Les valeurs des paramètres de ces lois (dans DEFI_MATERIAU) sont souvent identifiées dans une gamme de déformation, de vitesse, de température bien spécifiques.

- Pour les comportements élastoplastiques, voir [U2.04.03] Choix du comportement élasto-(visco)-plastique
- Pour les lois avec endommagement (cas du béton par exemple), voir [U2.05.06] Réalisation de calculs d'endommagement en quasi- statique
- Pour la métallurgie, voir [U2.03.04] Notice d'utilisation pour des calculs thermométallomécaniques sur des aciers
- Pour les milieux poreux en THM, voir [U2.04.05] Notice d'utilisation du modèle THM et [R7.01.11] Modèles de comportement THHM
- Pour l'utilisation des éléments CZM, voir [U2.05.07] Notice d'utilisation des modèles de zones cohésives

3.2 Mots-clés de COMPORTEMENT

3.2.1 DEFORMATION

Ce mot-clé permet de définir les hypothèses utilisées pour le calcul des déformations : par défaut, on considère de petits déplacements et petites déformations. Le type de déformation utilisé peut avoir une grande influence sur le calcul dès qu'une composante des déformations dépasse quelques % (typiquement 5%).

- **PETIT** : petites déformations, petits déplacements. Linéarité de l'opérateur déformation.

- **GROT_GDEP** : permet de traiter les grandes rotations et les grands déplacements, mais en restant en petites déformations. Ceci est particulièrement utile pour les structures élancées (modélisées en poutres, coques, ou 3D) et l'étude du flambement (voir par exemple le test [V6.02.134] SSNL134 - Ruine élasto-plastique du portique de Lee).

  En ce qui concerne les lois hyper-élastiques de type ELAS_VMIS , elles ne sont pas adaptées aux grandes déformations (perte d' existence de la solution, voir le §2,1 de [R5.03.20] Relation de comportement élastique non linéaire en grands déplacements). Il faut utiliser soit un modèle de grandes déformations avec VMIS_ISOT , soit le comportement ELAS_HYPER .

- **GDEF_LOG** : modèle de grandes déformations, utilisant une mesure de déformation logarithmique, et qui permet d'utiliser des lois de comportement élastoplastiques à écrouissage isotrope ou cinématique (voir la liste des comportements dans [U4.51.11] Comportements non linéaires). La relation contraintes-déformation étant hypo-élastique, cette formulation est limitée aux faibles déformations élastiques (mais grandes déformations plastiques).

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- **SIMO_MIEHE** : modèle de grandes déformations des lois de comportements s'appuyant sur un critère de Von Mises à écrouissage isotrope, et tous les comportements à écrouissage isotrope associés à un matériau subissant des changements de phases métallurgiques. La relation contraintes-déformations élastique est hyper-élastique, ce qui permet de traiter les grandes déformations élastiques (pour peu que cela ait un sens pour le matériau utilisé).

- d'autres formulations existent, voir [U4.51.11] .

3.2.2 ALGO_INTE , ITER_INTE_MAXI , RESI_INTE_RELA , ITER_INTE_PAS

Permet de préciser le type de schéma d'intégration pour résoudre l'équation ou le système d'équations non-linéaires formé par les équations constitutives des modèles de comportement à variables internes. Une méthode de résolution par défaut est prévue pour chaque comportement. Toutefois, il est possible de modifier la méthode de résolution par défaut pour un certain nombre de comportements (voir [U4.51.11]).

Dans le cas d'une résolution itérative (c'est à dire si ALGO_INTE n'est pas ANALYTIQUE), les mots-clés ITER_INTE_MAXI et RESI_INTE_RELA définissent le nombre maximum d'itérations et le résidu relatif pour intégrer le comportement. Si cette intégration échoue, il est possible de subdiviser le pas de temps, soit localement (mot-clé ITER_INTE_PAS) soit globalement (commande DEFI_LIST_INST). La valeur par défaut de ITER_INTE_MAXI est 20. Cela peut être insuffisant pour certains comportements (MONOCRISTAL par exemple). Ne pas hésiter dans ce cas à augmenter ce paramètre (100 par exemple). Par contre il est fortement déconseillé d'augmenter RESI_INTE_RELA (10^{-6} par défaut), sous peine d'obtenir des solutions non convergées.

La subdivision locale du pas de temps est moyen d'améliorer la robustesse de l'intégration locale, par contre elle ne permet pas de fournir une matrice tangente cohérente (perte de la convergence quadratique du problème global).

3.2.3 POST_ITER='CRIT_RUPT'

Définition d'un critère de rupture en contrainte critique en post-traitement des itérations de Newton, à chaque pas de temps. Si la plus grande contrainte principale moyenne dans un élément dépasse un seuil donné σ_c, le module d'Young est divisé au pas de temps suivant par un coefficient. Ces deux coefficients sont définis sous le mot- clé CRIT_RUPT de l'opérateur DEFI_MATERIAU.

3.2.4 RESI_RADI_RELA

Mesure de l'erreur due à la discrétisation en temps, directement reliée à la rotation de la normale à la surface de charge. On calcule l'angle entre la normale au critère de plasticité au début du pas de temps et la normale au critère de plasticité calculée à la fin du pas de temps. Le pas de temps est découpé (*via* DEFI_LIST_INST) si l'erreur est supérieure à la tolérance définie par l'utilisateur.

Ce critère est opérationnel pour les comportements élastoplastiques de Von Mises à écrouissage isotrope, cinématique linéaire et mixte et pour les comportements élasto-visco-plastiques de Chaboche.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

4 Résolution non-linéaire

Les 3 grands types de non-linéarités dans *Code_Aster* sont les suivants :
- non-linéarité liée au comportement du matériau (par exemple plastique) ;
- non-linéarité liée à la géométrie (par exemple en grands déplacements) ;
- non-linéarité liée au contact-frottant.

Les conseils généraux de résolution de problèmes non-linéaires font l'objet d'un document spécifique dont on recommande fortement la lecture : [U2.04.01] « Conseils d'utilisation de STAT_NON_LINE ».

Avant tout calcul non-linéaire, il est indispensable de s'assurer que le calcul fonctionne correctement en élasticité linéaire. Les non-linéarités pourront alors être ajoutées les unes après les autres.

On rappelle brièvement les conseils majeurs sur la discrétisation temporelle et les paramètres numériques de résolution non-linéaire dans les paragraphes suivants.

4.1 Non-linéarité et discrétisation temporelle

La résolution d'un problème non-linéaire nécessite généralement d'appliquer le chargement extérieur progressivement, par incrément de charge. Ainsi le temps (ou pseudo-temps en quasi-statique) est discrétisé en pas de temps et à chaque pas de temps correspond un incrément de charge (voir §2.2). Plus le pas de temps est petit, moins le problème est non-linéaire donc plus facile à résoudre. Afin d'autoriser le sous-découpage du pas de temps en cas d'échec de convergence, il est indispensable d'utiliser la commande DEFI_LIST_INST (pour plus de détails sur la gestion de la liste d'instants, voir le paragraphe §3.1 du document [U2.04.01])

Pour vérifier la cohérence des données (système d'unités, conditions aux limites, caractéristiques élémentaires, effet des variables de commandes), il est toujours utile d'effectuer un premier calcul élastique linéaire (STAT_NON_LINE/RELATION='ELAS', ou MECA_STATIQUE), avant toute étude non linéaire, puis d'ajouter les non linéarités les unes après les autres.

4.2 Paramètres numériques pour l'algorithme de résolution non-linéaire

Par défaut, l'algorithme de résolution non-linéaire est basé sur la méthode de Newton-Raphson (STAT_NON_LINE / METHODE='NEWTON').

Il est conseillé d'utiliser une matrice tangente réactualisée à chaque itération de Newton : REAC_ITER=1 afin de faciliter la convergence de l'algorithme (voir le paragraphe §2.2 du document [U2.04.01]).

Il est fortement recommandé de ne pas augmenter le critère de convergence de la méthode de Newton (RESI_GLOB_RELA=10^{-6} par défaut). D'autres critères de convergence sont également disponibles (voir le paragraphe §3.3 du document [U2.04.01]).

Remarques :
- Pour des problèmes adoucissants, *Code_Aster* offre la possibilité d'utiliser une méthode alternative à la méthode Newton-Raphson : c'est la méthode IMPLEX qui est une méthode robuste mais approchée (voir le paragraphe §4.5 du document [U2.04.01]).
- Pour une optimisation du temps de calcul, *Code_Aster* offre la possibilité d'utiliser une méthode de Newton inexacte (à condition que le solveur linéaire choisi soit un solveur itératif) : c'est la méthode NEWTON_KRYLOV.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

5 Gestion mémoire/temps et optimisation

Avant de parler d'optimisation, la première question que l'on se pose est comment choisir les paramètres d'exécution dans Astk : mémoire totale et temps pour que le premier lancement du calcul se termine correctement. Ensuite, une fois que le calcul sera passé une première fois, il sera toujours possible d'essayer de l'optimiser en fonction des informations que l'on aura récupérées du 1er calcul.

5.1 Première exécution d'un calcul

La connaissance de la taille de la mémoire dont on dispose est nécessaire. La réponse dépend bien entendu de la machine d'exécution : serveur centralisé, machine locale.... Sur une machine locale, il est toujours possible d'autoriser la mémoire maximale disponible alors que sur le serveur centralisé, il peut être opportun de ne pas demander trop de mémoire sous peine de devoir attendre que la classe de job associée se libère.

Estimation de la mémoire nécessaire : en quasi-statique, la mémoire totale est généralement fonction de la taille du ou des systèmes linéaires à résoudre. Pour les éléments finis iso-paramétriques, il est assez facile d'estimer la taille globale de ces systèmes. Le nombre de degrés de liberté peut être estimé par la relation : (nombre de nœuds du maillage) x (dimension du problème (2 ou 3))
Cette relation ne prend pas en compte les ddls relatifs aux conditions aux limites dualisées. Mais généralement, cette part est faible. Si les éléments finis ne sont pas iso-paramétriques (formulations mixtes par exemple), cette estimation est plus délicate.

De toute manière, la taille des systèmes linéaires est affichée à chaque résolution dans le fichier .mess : nombre total d'équations ou la matrice est de taille n équations. Il est **important de connaître ce nombre** (bien qu'à lui seul, il ne soit pas toujours un indicateur totalement fiable, car la mémoire et le temps nécessaire à la résolution d'un système linéaire dépend de bien d'autres paramètres : taille de la largeur de bande, nombre de termes non nuls dans la matrice, renuméroteur... mais cela nous emmènerait trop loin). Donc si on ne sait pas estimer la taille du système linéaire, il est intéressant de **lancer une première fois l'étude en mode dégradé** : lecture du maillage, affectation des éléments-finis, du matériau et résolution (MECA_STATIQUE ou STAT_NON_LINE) avec le solveur direct MUMPS. Lancer l'étude simplifiée en mode interactif avec suivi interactif, puis arrêter le calcul une fois l'information sur la taille du système connue.

Dans l'idéal[1], il faut disposer d'environ 30 Go pour faire passer un calcul à un million de degrés de liberté (cas du cube maillé en HEXA8, dont une face est encastrée par dualisation), la mémoire minimale[2] étant 9 Go. Le tableau ci-dessous donne des ordres de grandeur de la mémoire « idéale » (ou optimale) et minimale pour résoudre un système de taille donnée (avec le solveur direct MUMPS).

Nombre de degrés de liberté	Mémoire minimale	Mémoire « idéale »
200000	1 Go	4 Go
400000	2,5 Go	9 Go
600000	4 Go	15 Go
800000	6 Go	23 Go
1000000	9 Go	31 Go

Une fois la mémoire nécessaire estimée, on peut lancer le calcul complet avec le solveur direct MUMPS. A la suite de ce 1er calcul, il est très intéressant de regarder :

1 Idéal voulant dire ici gestion de mémoire IN_CORE
2 Minimale voulant dire ici pour une gestion OUT_OF_CORE

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- Dans le fichier .mess la mémoire totale (JEVEUX + python + librairies externes) utilisée par le calcul : cette information est affichée à la fin du fichier : voir MAXIMUM DE MEMOIRE UTILISEE PAR LE PROCESSUS ;
- Dans le fichier .resu : le temps de résolution (MECA_STATIQUE / STAT_NON_LINE) par rapport au temps total du calcul.

5.2 Optimisation d'un calcul

L'optimisation d'un calcul sur les aspects temps et mémoire est une opération intéressante mais un peu délicate car il n'existe pas de recette miracle. D'ailleurs il n'est pas toujours facile de faire la distinction entre optimisation de la mémoire et optimisation du temps de calcul, donc dans les paragraphes suivants, on donne des pistes d'optimisation « au sens large » et on renvoie vers les documents adéquats.

Le parallélisme
L'utilisation du parallélisme est certainement la solution la plus facile et la plus sûre pour diminuer le temps de calcul (et parfois aussi la mémoire nécessaire). Le parallélisme consiste à exécuter des opérations sur plusieurs processeurs en même temps. La mise en place du parallélisme n'est efficace que si le temps passé dans STAT_NON_LINE est prépondérant par rapport au temps total du calcul. Pour cela, il suffit de choisir un solveur adéquat (exemple : MUMPS, PETSC), une version MPI de *Code_Aster* et de spécifier le nombre de processeurs dans Astk (menu Options/mpi_nbcpu). Sur le serveur centralisé aster4, on conseille de commencer par choisir mpi_nbcpu=2, d'observer le gain en temps puis de recommencer avec mpi_nbcpu=4, puis éventuellement mpi_nbcpu=8. Il est possible de choisir mpi_nbcpu=16 mais on risque d'attendre longtemps avant que tous ces processeurs soient disponibles, donc avant que le calcul ne démarre (tout dépend de la charge machine !).
De nombreux conseils sur le parallélisme sont donnés dans le document [U2.08.06] « Notice d'utilisation du parallélisme ». Des informations plus fines sur les temps passés dans chaque partie de la résolution sont également affichées, et peuvent servir à mieux paramétrer le calcul (temps, mémoire, nombre de processeurs...). On renvoie pour cela à la documentation [U1.03.03] « Indicateurs de performance d'un calcul (temps/mémoire) ».

Le solveur linéaire
Suivant la taille de la matrice, l'utilisation d'un solveur itératif (comme PETSC) à la place d'un solveur direct permettra un gain en temps. À titre d'information, on considère qu'un solveur itératif est plus rapide qu'un solveur direct (par défaut) à partir d'environ 200 000 équations en 3d. Mais cela dépend du type de structure[3]. La contre-partie de l'utilisation d'un solveur itératif est une robustesse non garantie. Il est à noter que l'utilisation de PETSC nécessite une version MPI de *Code_Aster* (même si on n'utilise qu'un seul processeur). A partir d'une certaine taille de problème (quelques millions de ddls), l'utilisation d'un solveur direct devient impossible et il est indispensable d'utiliser un solveur itératif.
En cas d'échec de la résolution par un solveur itératif, les leviers d'actions sont le choix d'un autre préconditionneur (préconditionneur direct simple précision par défaut LDLT_SP, préconditionneur incomplet LDLT_INC, …), le choix d'un autre algorithme itératif (GMRES par défaut, CG, CR, …).
De nombreux conseils sur le choix du solveur sont donnés dans [U2.08.03] « Notice d'utilisation des solveurs linéaires » .

La gestion de la base globale
En cas de dépassement de la mémoire limite, on peut jouer sur l'archivage des résultats. L'archivage (mot-clé facteur ARCHIVAGE de STAT_NON_LINE) permet de réduire sensiblement la taille des bases en sélectionnant les instants sauvegardés. Par défaut, tout est archivé (même les instants issus du sous-découpage du pas de temps). Dans la phase de mise en place d'une étude, cela peut s'avérer utile, mais trop couteux en mémoire. L'observation et le suivi de certaines grandeurs peut répondre de manière efficace au besoin d'archivage (voir §3.4 de [U2.04.01] « Conseils d'utilisation de STAT_NON_LINE »). Une fois l'étude mise en place, on peut restreindre l'archivage à un nombre limité de pas de temps (instants de post-traitement par exemple).
Pour réduire l'encombrement de la structure de données resultat, il est possible de choisir *a posteriori* les champs à archiver soit en indiquant les champs à conserver, soit en indiquant les

3 Un cube est pénalisant pour le solveur direct, alors qu'une structure mince ou élancée donne un avantage au solveur direct

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

champs à exclure (commande EXTR_RESU). L'extraction peut aussi se faire sur une partie du maillage ou du modèle. La commande DETRUIRE peut aussi être utilisée pour détruire un concept. Suite à ces opérations d'extraction ou de suppression, il faut spécifier RETASSAGE='OUI' dans la commande FIN afin de récupérer effectivement l'espace disque associée à la base globale.

Pour réduire la taille des fichiers résultat au format MED (.rmed) : utiliser IMPR_RESU/RESTREINT. Dans un objectif d'une poursuite, on peut dans certains cas remplacer la base par l'impression de champs MED. La « poursuite » débutera alors par les commandes DEBUT puis LIRE_RESU.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

6 Contrôle de la qualité

De façon générale, la qualité globale d'une étude dépend de plusieurs facteurs. Parmi les causes qui peuvent altérer la qualité d'une étude, on peut distinguer en particulier :
- les incertitudes sur les données,
- les choix de modélisation (passage de la physique au numérique),
- les erreurs de discrétisation, une fois les données établies et la modélisation choisies.

6.1 Incertitudes sur les données

Sur ce point les incertitudes peuvent être multiples car elles peuvent concerner chacun des aspects suivants.

- Les données matériau : une fois la loi de comportement choisie, et ses paramètres correctement identifiés sur des essais de caractérisation appropriés, il peut subsister une incertitude sur les valeurs des paramètres de cette loi, due à la variabilité des matériaux.
- La géométrie exacte de la structure à modéliser peut comporter des incertitudes : les dimensions réelles ne sont pas celles prévues à la conception, ou bien la dégradation de la forme d'un pièce conduit à une géométrie mal maîtrisée (en dehors de tout aspect de discrétisation).
- Les chargements (y compris thermiques ou dus à d'autres variables de commandes : irradiation, hydratation, séchage, etc..) et les conditions aux limites peuvent présenter de nombreuses incertitudes.
- Les conditions initiales, en particulier celles dues à fabrication (contraintes résiduelles, écrouissage initial).

Pour chacun des paramètres incertains évoqués ci-dessus, il sera parfois nécessaire d'effectuer une analyse de sensibilité :

- soit l'utilisateur effectue quelques simulations pour des valeurs extrêmes des paramètres incertains, ce qui suffit dans le cas d'une variation monotone du résultat à ces paramètres (mais ce n'est pas toujours le cas : par exemple la dépendance des contraintes aux valeurs extrêmes de température, avec des coefficients matériau qui en sont fonction, c'est pas triviale.)
  Signalons à ce propos la possibilité d'effectuer très simplement des calculs paramétriques avec *Code_Aster* (cf. [U2.08.07] Distribution de calculs paramétriques)
- soit dans le cas de paramètres hautement variables, il effectue une analyse mécano–fiabiliste, avec OpenTurns (http://www.openturns.org) couplé à *Code_Aster*, dans la plate-forme Salome-Meca [SV1.04.01] Gestionnaire d'étude chaînée Openturns/Aster).

6.2 Choix de modélisation et vérification des données

Même si les données sont fiables, la qualité des résultats dépend aussi de choix effectués par l'utilisateur relatifs à la simulation à effectuer. Les paragraphes précédent décrivent ces choix. Rappelons quelques conseils généraux :

- Au delà du maillage, il peut être utile de vérifier les données du calcul, comme par exemple, visualiser les conditions aux limites et chargements (sens et lieu d'application), les affectations des matériaux, et des caractéristiques élémentaires (orientation des poutres, épaisseurs, sections). Pour cela, utiliser `IMPR_RESU/CONCEPT`.

- D'autres vérifications peuvent être effectuées directement dans le fichier de commandes ; par exemple, vérifier la cohérence du système d'unités, les conditions aux limites, les caractéristiques élémentaires, il est toujours utile d'effectuer un premier calcul élastique, avant toute étude non linéaire.

- Pour vérifier les données matériau, il peut être utile d'effectuer avec la mise en données de *Code_Aster* un essai plus simple que l'étude, par exemple sur point matériel (`SIMU_POINT_MAT`). Retrouve-t-on la courbe de traction-compression ?

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- Le calcul de la masse (ou du volume) de la structure fait également partie des vérifications simples mais parfois utiles.

6.3 Maîtriser les erreurs dues à la discrétisation spatiale

Sur ce point, nous rappelons ici quelques généralités, mais la lecture de [U2.08.01] Utilisation des indicateurs d'erreur et stratégies d'adaptation de maillages associées est plus que conseillée.

Indépendamment de tout désir de réaliser de l'adaptation de maillage, nous conseillons de vérifier le maillage initial avec la commande MACR_INFO_MAIL [U7.03.02] Macro-commande MACR_INFO_MAIL. Cette commande permet de réaliser, à peu de frais, les vérifications suivantes :

- vérifier la concordance du maillage avec la géométrie initiale (en dimension, en volume) ;
- lister les GROUP_MA et GROUP_NO, pour une bonne modélisation des conditions aux limites ;
- diagnostiquer d'éventuels problèmes (connexité, trous, interpénétration de mailles) ;
- fournir des critères de qualité des mailles (évalués maille par maille).

Pour maîtriser plus facilement l'effet de la qualité du maillage, une stratégie automatique d'adaptation de maillage est opérationnelle : elle s'appuie sur le logiciel d'adaptation de maillage Homard, qui peut être appelé directement depuis le fichier de commandes de *Code_Aster* ou dans la plate-forme Salome-Meca, et utilise soit des indicateurs d'erreur permettant de piloter l'adaptation, soit les valeurs d'un champ, soit le saut d'un champ d'un élément à l'autre. Plusieurs motivations apparaissent pour adapter un maillage :

- Le maillage est très compliqué à réaliser : on part d'une version simple et on confie à un processus automatique la charge de l'affiner.
- On veut s'assurer de la convergence de la solution numérique : plutôt que de réaliser à la main des maillages de plus en plus fins, on laisse le logiciel chercher lui-même les endroits où il faudrait affiner le maillage pour augmenter la précision du résultat.
- Les conditions du calcul changent au cours de son déroulement : les zones qui doivent être maillées finement se déplacent. Si on maille fin partout dès le début, le maillage est trop gros. En adaptant au fur et à mesure, (raffinement - déraffinement) le maillage ne sera fin qu'aux endroits nécessaires : sa taille sera réduite et la qualité de la solution sera bonne.

Les zones à raffiner peuvent être repérées :

- Soit avec un indicateur d'erreur. Notons que les plus simples (ZZ1, de type Zhu-Zienkiewicz) sont très robustes, disponibles en 2D et 3D, et même s'ils ne fournissent pas les meilleurs majorants de l'erreur, leurs variations suffisent à piloter l'adaptation. Les estimateurs en quantité d'intérêt sont plus pertinents pour fournir une borne de l'erreur commise.
- Soit avec un champ (déformation, variable interne) pertinent. Attention, en plasticité, la contrainte de Von Mises ne l'est pas toujours. On peut utiliser :
  - soit les valeurs extrêmes de ce champ (par exemple demander de raffiner les 5% d'éléments qui ont la valeurs la plus forte),
  - soit piloter l'adaptation par le saut des valeurs de ce champ à la frontière entre deux élements finis.
- Soit par des boîtes (raffinement uniforme dans des zones géométriques).

Pour plus d'information, voir [U2.08.01] Utilisation des indicateurs d'erreur et stratégies d'adaptation de maillages associées

6.4 Erreurs de discrétisation en temps

Pour la plupart des non linéarités (comportement, contact, grandes déformations) la discrétisation temporelle influence le résultat. Des résultats de convergence existent (fort heureusement) dans tous les cas classiques (élasto-visco-plasticité, contact) , mais il reste néanmoins à s'assurer que pour un pas de temps choisi la solution est suffisamment proche de la solution continue en temps.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

La solution la plus simple est similaire au raffinement uniforme pour l'adaptation de maillage : elle consiste à raffiner uniformément le pas de temps sur tout le transitoire et relancer STAT_NON_LINE. Ceci peut être automatisé grâce à une fonctionnalité de DEFI_LIST_INST (voir [U2.04.01] et [U4.34.03]).

Un autre critère de maîtrise du pas de temps est la subdivision en fonction d'une grandeur d'intérêt : DEFI_LIST_INST / DELTA_GRANDEUR permet en effet de redécouper le pas de temps si la variation maximum d'une quantité donnée (par exemple une composante de déformation plastique) est supérieure à un seuil fourni.

De plus, pour les comportements élastoplastiques, il est possible d'estimer directement l'erreur due à la discrétisation en temps, de façon analogue à RESI_RADI_RELA (cf.3.2.4). Si ce critère n'a pas été pris en compte lors du calcul, il est possible de le calculer en post-traitement, dans CALC_CHAMP : la composante ERR_RADI de l'option DERA_ELGA contient l'estimation d'erreur à chaque instant et en chaque point d'intégration.

7 Références

[1] « Mécanique des structures », p.658, F.Voldoire ; Y.Bamberger, Presses de l'ENPC 2008.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)