Nessy: un Logiciel éléments finis pour développer et capitaliser des travaux de recherche

Gérard Coffignal, Philippe Lorong

Nessy: Un Logiciel éléments finis pour développer et capitaliser des travaux de recherche

G. Coffignal, P. Lorong

LMSP (Laboratoire de Mécanique des Systèmes et des Procédés), UMR 8106,
ENSAM Paris, 151, bd de l'Hôpital, F-75013 Paris
gerard.coffignal@paris.ensam.fr
philippe.lorong@paris.ensam.fr

Résumé

Nessy (*Ne*twork *S*olver *Sy*stem) est un logiciel destiné à faciliter l'intégration et la capitalisation du savoir faire du LMSP en calcul des structures. *Nessy* est ouvert sur l'avenir et vise à traiter des problèmes non-linéaires de grande taille par l'utilisation du calcul parallèle à gros grain. Outre des utilitaires généraux, *Nessy* s'appuie sur une représentation topologique des domaines (volumes connectés à des faces, faces connectées à des arêtes, etc.).

1 Introduction

Plus qu'un programme, le projet *Nessy* peut être présenté comme une plateforme logicielle destinée à fédérer et capitaliser les travaux de programmation réalisés au sein de notre laboratoire. Délibérement orienté recherche, *Nessy* ne prétend en aucun cas devenir un code généraliste. Développé et maintenu par une équipe réduite, il ne s'adresse qu'à un nombre limité d'utilisateurs.

Le développement d'un tel logiciel est cependant un projet ambitieux, surtout pour un laboratoire de taille modeste ce qui est notre cas (une vingtaine d'enseignants chercheurs et une quinzaine de doctorants). Qui plus est, c'est un travail de longue haleine, très exigeant en temps travail, et peu valorisable par des publications. Ce choix a pourtant été fait, et ce, pour plusieurs raisons. La première est une expérience préalable de plusieurs membres du laboratoire en développement de logiciel de simulation pour le calcul des structures [5][6]. Ainsi, *Nessy* repose sur cette expérience et a hérité au passage d'un certain nombre d'utilitaires de base. La seconde est que les logiciels "commerciaux", s'ils sont très performants, ne proposent généralement qu'une ouverture très limitée aux développeurs externes. De plus, l'investissement en

temps nécessaire à l'immersion dans de tels logiciels n'est pas toujours compatible avec un travail de thèse, et encore moins de mémoire de DEA, sachant qu'il s'agit pourtant de notre ressource principale. Enfin, en cas de mise à jour importante, et le plus souvent inopinée, c'est tout cet investissement qui peut être perdu. Aucune pérénité, ou incrémentation de connaissance, dans un domaine de compétence ne sont plus alors garantis.

Deux thèmes de recherche du laboratoire corroborent notre démarche :

- La simulation de l'usinage à l'échelle macroscopique (prédiction du comportement dynamique du sytème pièce/outil/machine ainsi que de l'état de la surface usinée) où l'association d'une description de la surface extérieure de la pièce (qui évolue en cours d'usinage) avec un modèle éléments finis rend très délicat l'utitisation d'un logiciel "commercial".
- La simulation de l'usinage à l'échelle mésoscopique (simulation de la formation du copeau et des mécanismes de cisaillage de la matière) qui s'effectue dans un cadre thermomécanique en transformations finies et qui nécessite, de part la complexité du sujet, une capitalisation du travail de plusieurs personnes sur plusieurs années.

Un guide important dans la conception de *Nessy* a été la volonté d'utiliser le parallélisme à gros grain (encore appelé parallélisme par envoi de messages) pour répondre à l'objectif de traiter des problèmes non-linéaires de grande taille. Ceci nous a conduit à structurer de façon "modulaire" les données et la représentation du domaine matériel. Pour les données nous avons mis en place un agencement récursif par blocs dans un tableau de travail. Pour ce qui est du domaine, nous utilisons une représentation topologique. Le domaine est décomposé en volumes connectés à des faces, les faces étant elles-mêmes connectées à des arêtes connectées à des points. Cette décomposition peut être récursive même si ce n'est pas le cas dans la version actuelle de *Nessy*.

Dans la partie 2 nous commençons par présenter l'organisation générale du logiciel. Cet aspect n'est pas anodin car nombre de fonctionnalités essentielles de *Nessy* sont en fait "externalisées". Nous donnons ensuite, dans la partie 3, les principaux choix conceptuels de *Nessy* en précisant l'état d'avancement. Nous y précisons en particulier l'ouverture de *Nessy* au parallélisme, et y exposons quelques particularités visant à rendre le code le plus accessible possible aux développeurs. Nous finissons, dans la partie 4, par présenter quelques uns des premiers résultats ainsi que les développements en cours.

2 Organisation du logiciel : *Nessy* et *GesDyn*

Nous avons introduit une séparation en deux couches logicielles : *GesDyn* et *Nessy*. *GesDyn* est un noyau qui rassemble les fonctionnalités plus ou moins évoluées pouvant être utilisées par des applications n'ayant aucun rapport avec les éléments finis. C'est

Nessy regroupe les fonctionnalités et l'ordonnancement des traitements propre à la simulation de phénomènes physiques en s'appuyant sur *GesDyn*.

Nessy, dont les spécificités sont exposées dans la partie suivante, se positionne en fait en tant qu'*application utilisatrice* de *GesDyn*.

GesDyn comporte des utilitaires de *bas niveau* tels que :

- des outils matriciels : addition, multiplication, factorisation, calcul de valeurs propres, ... ,
- des armes anti-vermine : impressions, dessins et pauses conditionnels tous déclenchés par le changement d'état d'indicateurs en interne (dans le programme compilé) ou en externe (fichier d'entrée de données), suivi d'arbres d'appel, ... ,
- des outils de gestion de listes et de piles.

Il est à noter que certains utilitaires matriciels peuvent faire appel à des sous-programmes optimisés (optimisation de l'utilisation du cache notamment) dépendant du type de calculateur employé.

GesDyn propose également des fonctionnalités de *haut niveau* parmi lesquelles :

- La gestion dynamique d'un tableau de travail organisé sous-forme de blocs d'informations récursifs. Chaque bloc d'information est *une zone mémoire contiguë* (portion d'un tableau de travail) qui contient les informations sur sa propre structure (présence de sous-blocs, etc.). Deux types de blocs d'informations ont été créés : les blocs-liste et les blocs-feuille. Un bloc-liste contient une liste de blocs-liste ou/et de blocs-feuille, un bloc-feuille contient un ensemble de paquets de valeurs organisés sous forme de matrice bidimensionnelle. Toutes les données d'un paquet sont d'un seul type (entier, réel, booléen, chaîne de caractères). Un exemple de bloc d'information de type bloc-liste est donné sur la figure 1. Son organisation dans le tableau de travail est quant à elle représentée sur la figure 2.
- Un mécanisme de lectures d'entrées (pour l'application utilisatrice) par mots-clés. La description des entrées est faite dans des fichiers externes lus et interprétés par *GesDyn*. Le module de lecture des entrées permet également d'introduire des paramètres utilisés par *Nessy* dans la description du modèle éléments finis. Ceci ouvre des perspectives sur l'optimisation des structures et les calculs de sensibilité.
- Une description externe (dans un unique fichier écrit par les développeurs lus et interprétés par *GesDyn*) de la structure des données de l'application utilisatrice. Celle-ci est très liée à l'organisation récursive des blocs. Elle est en relation avec la description de la plupart des entrées dont les valeurs lues sont automatiquement rangées, sans aucune intervention de l'application utilisatrice, dans la base de données de cette dernière.

- Une description topologique de frontières. Cette fonctionnalité est actuellement utilisée pour la simulation de l'enlèvement de matière à l'échelle macroscopique où de nombreuses intersections entre l'enveloppe du volume balayé par l'outil (à chaque pas de temps) et la pièce usinée sont à réaliser.

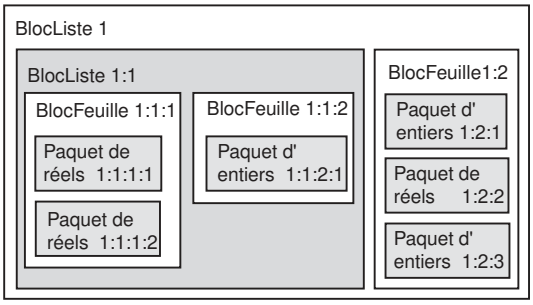

Figure 1 – *Exemple de bloc d'information*

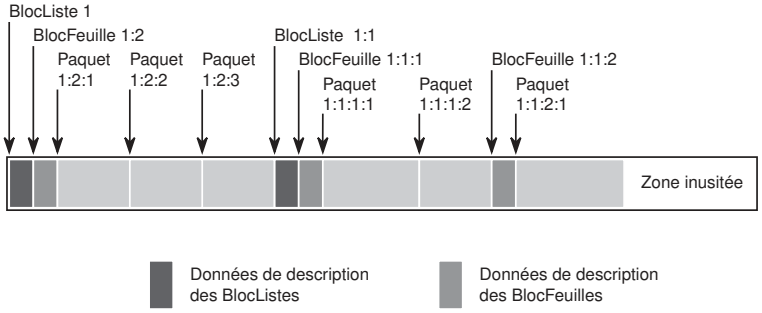

Figure 2 – *Organisation dans le tableau de travail d'un bloc d'informations*

3 Spécificités de *Nessy*

3.1 Organisation des traitements

Les traitements dans *Nessy* sont organisés par services. Un service répond à une liste de demandes qu'il traite et/ou qu'il transmet à un service subalterne. La liste des demandes peut être vue comme le micro-programme que doit réaliser le service.

On trouve par exemple le service des algorithmes qui se situe au niveau le plus haut et celui des points matériels, pour les traitements liés au comportement de la matière, qui se situe au niveau le plus bas. Entre ces deux niveaux, on trouve successivement le service des sous-domaines (un sous-domaine est une partie de la structure dont le

modèle de comportement et les traitements qui en découlent sont identiques en tout point), les services liés au *genre* des éléments issus du maillage des sous-domaines (éléments volumiques, surfaciques ou linéiques), les services liés à la *catégorie* des éléments dans chaque *genre* (par exemple hexaèdre, pentaèdres ou tétraèdres pour les éléments volumiques, quadrangles ou triangles pour les éléments surfaciques, ...), et enfin les services liés à chaque *type* précis d'élément dans lesquels sont, par exemple, construites les matrices élémentaires.

L'organisation par blocs recursifs est fortement mise à contribution afin que chaque service dispose des données qui le concernent (par exemple, un bloc pour chaque *genre* d'élément, contenant lui-même les blocs relatifs aux *catégories* d'éléments, et ainsi de suite). Ceci s'effectue au travers d'aiguillages permettant de naviguer dans les blocs-liste. On aboutit ainsi par exemple, dans les services propres à chaque type d'éléments ou dans le service point matériel, aux données qui les concernent. Ces dernières sont contenues dans des blocs-feuille.

La description externe de la structure des données de *Nessy* et l'organisation des services sont donc interdépendantes.

Il existe par contre une très grande indépendance dans *Nessy* entre le service des algorithmes (actuellement sont programmés ceux de statique, analyse modale, schéma explicite en transformations finies), le service point matériel (sont actuellement traités le comportement élastique en HPP et le comportement de type Johnson Cook en transformations finies) et les services éléments (un par type d'élément programmé: actuellement éléments hexaèdriques, poutre, triangle).

3.2 Représentation topologique

L'utilisation d'une représentation topologique dans *Nessy* se retrouve à deux niveaux :

- elui du domaine pour faciliter sa décomposition en sous-domaines grâce à une indentification naturelle des surfaces les délimitant. Ces surfaces peuvent elles-mêmes être décomposées en sous-surfaces qui sont délimitées par leurs arêtes. Cette représentation récursive des domaines et de leurs frontières peut être particulièrement utile pour les traitements en parallèles, pour la définition de paramètres dimensionnels ou comportementaux lors de la mise en place d'algorithmes d'optimisation. Enfin une identification claire des frontières et des faces d'éléments les constituant rend plus aisée la définition d'algorithmes de contact.
- elui de l'élément où la connaissance de son voisinage facilite également l'écriture d'algorithmes d'adaptation de maillage et/ou de calcul d'erreur a prosteriori.

3.3 Organisation des degrés de liberté

Une originalité de *Nessy* est de répartir les degrés de liberté en cohérence avec la représentation topologique des éléments. Chaque entité topologique possède un

ensemble de degrés de liberté qui lui est propre. On trouve ainsi des degrés de liberté internes à un élément, à une face, à une arête ou à un point.

Par exemple, pour un élément quadrangulaire à "9 noeuds" la répartition des degrés de liberté se fait sur les 4 coins, les 4 faces et dans l'élément (degrés de libertés associés au point central).

Ces degrés de liberté peuvent être définis de façon très générale pour chaque type d'entité via un regroupement en familles. Leur nombre et leurs natures (translations, rotations, températures, potentiels électriques, ...) sont définis par des listes famille par famille.

3.4 Ouverture vers le parallélisme

Le parallélisme n'est pas encore mis en place. Cependant par conception *Nessy*et *GesDyn*sont prédisposés à être parallélisés car ils intègrent une structure de données adaptée au parallélisme. L'organisation par blocs rend très aisés et très efficace une mise en veille sur disque de zones entières de données (généralement un unique bloc) ou l'échange de messages entre occurences de *Nessy* (échange de blocs). L'ouverture vers le parallélisme se décline ainsi de deux façons :

- *Introduction d'un parallélisme "stuctural"*: Les traitements et les données, associés à chaque sous-domaine ou macro-élément sont pris en compte par une o urence de *Nessy* (au moins une par processeur).
- *Introduction d'un parallélisme "matriciel"*: Représention des matrices sous forme d'hypermatrices (matrices de matrices) pour l'écriture de solveurs hypermatriels parallèles.

3.5 Le projet *Nessy* et ses développeurs

Même s'il est très difficile de rendre accessible à des développeurs novices un code en constante évolution, un certain nombre de facettes de *Nessy* y contribue. On peut en particulier citer :

- le choix d'un langage de programmation très accessible : fortran,
- une description externe des entrées et de la structure de données (information oncentrée sur quelques fichiers lisibles et commentés),
- une indépendance de la plupart des services (il est inutile d'avoir une connaissance globale du code pour intervenir de façon locale),
- une documentation présentée sous forme synthétique et structurée dans des pages html accessibles en permanence sur le site du LMSP [1]. Une grande partie de cette documentation est intégrée dans les fichiers sources fortran; un outil permet, de façon automatique, par scrutation de ces fichiers source, de l'extraire et de la convertir au format adéquat.

4 Premiers résultats – Exemple d'implémentation

4.1 Exemples de calculs réalisés avec *Nessy*

Nessy permet actuellement de traiter des problèmes de statique, d'analyse modale (Figure 3) et de dynamique explicite en transformations finies (Figure 4).

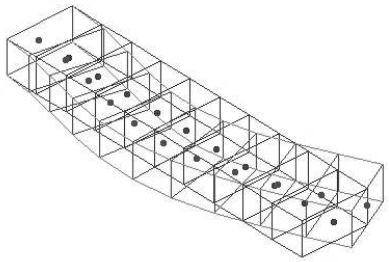

Figure 3 – $5^{ème}$ *mode de vibration d'une poutre à section rectangulaire*

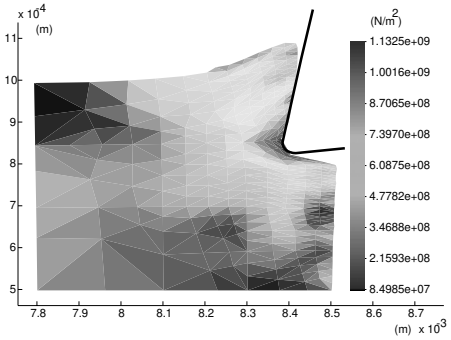

Figure 4 – *Indentation d'un pièce par un outil – Contraintes équivalentes de Mises*

4.2 Développements en cours

Actuellement, les principaux développements portent sur la mise en place d'algorithmes de contact, l'intégration complète des travaux antérieurs et récents relatifs à la simulation de la coupe à l'échelle macroscopique [2], ainsi qu'une approche sans maillage pour la simulation de l'enlèvement de matière et du découpage à l'échelle mésoscopique.

Pour ce dernier développement, une représentation topologique des cellules de Voronoi en 3D, nécessaires à l'approche, est mise en place. Une des particularités est

qu'une cellule de Voronoi est connectée à une liste de faces dont le nombre change d'une cellule à l'autre. De plus une représentation explicite de la frontière doit être introduite pour traiter convenablement les cas de frontières non convexes [10].

5 Conclusion

Ce projet de développement de logiciel a été initié il y a déjà plusieurs années. Aujourd'hui, les premières simulations ont été effectuées et un nombre important des fonctionnalités attendues est opérationnel. Ceci permet désormais à des étudiants en thèse d'effectuer leurs travaux de programmation au sein de *Nessy*.

Nous espérons que la perte de généralité et d'homogénéité dans les développements sera compensée par la capitalisation et ce particulièrement pour des sujets de recherche déjà existants au laboratoire et appelés à se développer.

Références

[1] http://www.paris.ensam.fr/lmsp ().

[2] Assouline, S., Beauchesne, E., Coffignal, G., Lorong, P., Marty, Audrey. Simulation numérique de l'usinage à l'échelle macroscopique : modèles dynamiques de la pièce. *Mécanique et Industrie*, **3**, 389–402 (2002).

[3] Bathe, Klaus-Jürgen. *Finite Element Procedures*. Prentice-Hall, first edn. (1996).

[4] Breitkopf, P., Touzot, G. Architecture des logiciels et languages de modélisation. *Revue Européenne des Eléments Finis*, **1**(3), 333–368 (1992).

[5] Coffignal, G. Optimisation et fiabilité des calculs éléments finis en élastoplasticité. (1987). Thèse d'état, Université de Pierre et Marie Curie (Paris VI).

[6] Cognard, J.Y., Dureisseix, D., Ladevèze, P., Lorong, P. Expérimentation d'une approche parallèle en calcul des structures. *Revue européenne des éléments finis*, **2**(5), 197–220 (1996).

[7] Crisfield, M.A. *Non-Linear Finite Element Analysis of Solids and Structures - Advanced Topics*, vol. 2. Wiley, first edn. (1998).

[8] Kamel, H.A., McCabe, M.W. Applications of GIFT III to structural engineering problems. *Computers and Structures*, **7**, 399–415 (1976).

[9] Lorong, P., Ali, F., Coffignal, G. Research oriented software development platform for structural mechanics : a solution for distributed computing. In *Proceedings of the fifth International Conference on Computational Structures Technology*, pp. 93–100. B.H.V. Topping, Leuven, Belgium (September 2000).

[10] Yvonnet, J., Ryckelynck, D., Lorong, P., Chinesta, F. Interpolation naturelle sur les domaines non convexes par l'utilisation du diagramme de Voronoï contraint – méthode des éléments C-naturels. *Revue européenne des éléments finis* (2003). accepted.