Calcul de structures et parallélisme : un bilan et quelques développements récents

David Dureisseix, Laurent Champaney

Calcul de structures et parallélisme : un bilan et quelques développements récents

David DUREISSEIX et Laurent CHAMPANEY

Laboratoire de Mécanique et Technologie, E.N.S. de Cachan / C.N.R.S. / Université Paris 6
61, avenue du Président Wilson - 94235 CACHAN CEDEX

Résumé

Dans le domaine de la simulation et du calcul de structures, la demande croissante en ressource de calcul conduit à utiliser les machines les plus puissantes en service actuellement, à savoir les calculateurs à architecture parallèle. Après avoir présenté les caractéristiques des analyses par éléments finis, et celles de ces calculateurs parallèles, des méthodes de parallélisation des problèmes sont abordées. Les méthodes de décomposition de domaine aujourd'hui utilisées dans ce cadre sont ensuite plus détaillées, et leurs performances sont présentées pour l'une d'entre elles sur des exemples industriels et de grande taille.

Ceci est le *preprint* d'un article publié sous sa forme finale comme : David Dureisseix, Laurent Champaney, Calcul de structures et parallélisme : un bilan et quelques développements récents, *Mécanique et Industries* 1:43-60, Elsevier, 2000. doi : 10.1016/s1296-2139(00)00100-7.
http://www.sciencedirect.com/science/article/pii/S1296213900001007

Mots clés : Calcul de structures, parallélisme, décomposition de domaine, éléments finis.

1. Introduction

Dans le cadre du développement des mécanismes et des structures, la part de la simulation devient aujourd'hui de plus en plus grande. Elle permet la réduction des temps de développement, la limitation du nombre de prototypes, et celle du nombre d'essais intermédiaires avant les tests finaux de validation.

Cet état de fait est rendu possible par l'émergence des moyens de simulation (aussi bien dans l'évolution des machines de calcul que des techniques de calculs proprement dites). En particulier, le développement et la commercialisation de nombreux codes de calcul par la méthode des éléments finis (Nastran, Abaqus, Samcef...) rendent la simulation de systèmes mécaniques accessible à tous. Les systèmes de modélisation et de CAO (Catia, Ideas...) permettent assez aisément d'obtenir un modèle éléments finis, même pour des géométries complexes. De plus, les systèmes de post-traitement graphiques permettent de visualiser rapidement les quantités directement utiles à l'ingénieur, par exemple sous forme de cartes d'isovaleurs (de contraintes, de déformations…) avec une qualité et une interactivité vis à vis de l'utilisateur impressionnantes.

Néanmoins il convient de rester lucide et d'être conscients de la qualité (et des limites) des modélisations et des calculs que nous menons. Même si elle se banalise, la méthode des éléments finis reste un outil complexe et savoir bien la maîtriser n'est pas chose aisée.

La nécessité d'obtenir une indication sur la qualité des calculs réalisés se pose de façon de plus en plus pressante. Les outils développés jusqu'à lors commencent à cesser d'être des outils du domaine de la recherche pour devenir d'usage courant dans des codes commerciaux. Mais même avec des maillages

optimisés (coût de calcul minimum pour une précision demandée), les problèmes à résoudre continuent d'être de taille de plus en plus importante, en particulier lors des analyses tridimensionnelles. La demande de plus grandes capacités de traitement est toujours présente et les centres de calcul ou de ressources informatiques s'équipent de machines de plus en plus puissantes, quitte à se regrouper pour pouvoir accéder à de plus grandes ressources par la mise en commun des moyens économiques. Citons en France, les serveurs nationaux : l'IDRIS Orsay (http://www.idris.fr/) et le CINES Montpellier (http://www.cines.fr/), le CRIHAN Mont-Saint-Aignan (http://www.crihan.fr/)

Les calculateurs les plus puissants en service actuellement sont tous du type parallèle. Après une longue phase de stabilisation (systèmes exotiques, manque de portabilité...) qui s'est traduite par la fermeture ou le rachat de grands constructeurs (TMC et CRAY ont peut-être été les plus marquants), et bien que de nouvelles réalisations continuent de voir le jour (clusters de PC, projet Beowulf du CESDIS-Maryland...), les choses se stabilisent du point de vue de l'utilisateur, et les systèmes d'exploitation deviennent plus aisés d'utilisation (émergence de standards et portabilité).

Se posent alors des questions de développement et de maintenance pour les fournisseurs de codes de calcul : doivent-ils essayer d'adapter le logiciel existant (souvent initialement développé pour des architectures très différentes) ou doivent-ils investir dans la réécriture complète du code ?

Actuellement, les durées de développement de nouveaux codes deviennent beaucoup plus grandes que les longévités des machines de calcul (avant qu'elles ne deviennent obsolètes). Ces dernières ont maintenant tendance à être assimilées à du « consommable, » et la mise à jour et la portabilité des codes devient une issue cruciale (d'où l'utilisation de langages plus adaptés à l'évolution des codes, comme les langages orientés objet, [Breitkopf 92]).

Après avoir présenté quelques caractéristiques des problèmes de mécanique des structures abordés par les éléments finis dans la deuxième partie, on s'attachera à celles des architectures parallèles et de leur utilisation dans la troisième partie. La quatrième partie concerne quant à elle l'utilisation d'un type de méthode particulier, celui de la décomposition de domaine, pour la résolution implicite des grands systèmes linéaires issus des équations d'équilibre de la structure. Quatre méthodes actuellement en usage sont ainsi discutées quant à leur domaine d'emploi et leur performance. La cinquième partie, enfin, illustre les résultats que l'on peut obtenir avec ce type de méthodes sur des exemples réalistes.

2. Calcul de structures par la méthode des éléments finis

2.1 But

Le dimensionnement d'un système mécanique (pièce, assemblage…) nécessite la réalisation de plusieurs modélisations :

– modélisation de la géométrie,

– modélisation des conditions aux limites et des charges,

– modélisation du comportement du ou des matériaux,

de manière à obtenir une formulation mécanique du problème traité. Cela conduit au choix d'un modèle particulier parmi ceux que nous propose la mécanique (tridimensionnel, plaque, poutre…, élastique, plastique, thermique…). Généralement, les modèles ainsi construits relèvent de la mécanique des milieux continus et se traduisent par des équations aux dérivées partielles, avec des conditions aux limites entre les inconnues (champ de déplacement U, champ de contrainte σ…) et les données (conditions initiales, charges, liaisons, coefficients matériaux…).

La solution exacte (U_{ex} , σ_{ex}) d'un tel problème est rarement possible à déterminer analytiquement et la méthode la plus couramment utilisée pour trouver une solution approchée (U_h , σ_h) est la méthode des éléments finis [Zienkiewicz 91, Gupta 96]. Elle consiste à remplacer le modèle continu par un modèle discret.

2.2 Problématique du calcul de structures complexes

Non-linéarités

Il existe généralement trois types de non-linéarités dans les problèmes standards de mécanique que nous traitons :

– les non-linéarités de type « matériau » qui sont associées au caractère non-linéaire de la loi de comportement des matériaux en présence (plasticité, endommagement...) ;

– les non-linéarités de type « géométrique » qui interviennent lorsque les déplacements et/ou les déformations de la structure deviennent tels qu'il n'est plus possible de traiter le problème dans sa configuration non déformée (l'hypothèse de « petites perturbations » n'est plus valable) ;

– les non-linéarités de type « contact » qui apparaissent lorsqu'on veut modéliser des conditions de non pénétration et/ou de frottement entre deux solides. La solution dépend de l'état de la zone de contact (décollement, contact, glissement, adhérence…).

La résolution de ces problèmes entraîne donc l'utilisation d'algorithmes itératifs plus ou moins adaptés pour la résolution de systèmes non-linéaires (BFGS, line search...). De plus, lorsque le comportement de la structure varie avec l'intensité du chargement et avec son histoire (problèmes d'évolution), il est nécessaire d'introduire une discrétisation temporelle. Des méthodes incrémentales, basées sur le fait que la solution varie peu entre deux instants, permettent de calculer la solution au cours du temps. Dans tous les cas de calculs implicites, on sera aussi amené à résoudre des problèmes linéaires couplés sur toute la structure ; leur description fait partie du paragraphe suivant.

Taille des modèles

Dans le cas de modélisations tridimensionnelles, la complexité de la géométrie et la recherche d'une solution de bonne qualité conduisent à utiliser une discrétisation fine et donc à introduire un grand nombre d'éléments. Outre les non-linéarités précédemment citées, la deuxième difficulté est ainsi l'obligation de résoudre des systèmes linéaires creux qui deviennent très rapidement gigantesques (plusieurs centaines de milliers d'équations). En statique, ils traduisent l'équilibre discrétisé de la structure ; dans toute la suite, on les notera sous la forme :

$$[K]\{q\} = \{f\} \tag{1}$$

où {q} est le vecteur des déplacements nodaux recherché, {f}, le vecteur des forces nodales appliquées (ou forces généralisées) et [K] la matrice de rigidité du système. Le système linéaire a la taille du nombre d'inconnues nodales (degrés de liberté ou ddl). Il est creux et souvent mal conditionné, bien que souvent symétrique, défini et positif. La résolution directe de ce système se fait classiquement par factorisation de la matrice, avec une méthode de type Cholesky ou Crout. Ces méthodes s'avèrent cependant très coûteuses lorsque la taille du problème croît. A titre d'exemple, on montre sur la figure 1, l'évolution du nombre de degrés de liberté quand le nombre d'éléments augmente, pour un maillage 2D en quadrangles à 8 nœuds et pour un maillage 3D en cubes à 20 nœuds.

Pire encore, cette même figure reporte la taille mémoire (en Mo) nécessaire pour stocker la matrice [K] factorisée, ainsi que le nombre d'opérations nécessaires pour réaliser cette factorisation (complexité de l'algorithme, en millions d'opérations en virgule flottante), dans des axes logarithmiques. Le code de calcul employé est CASTEM 2000, développé au CEA de Saclay [Verpeaux 88] ; il utilise un stockage semi-morse, une renumérotation de Cuthill-McKee inverse, et une factorisation de Crout.

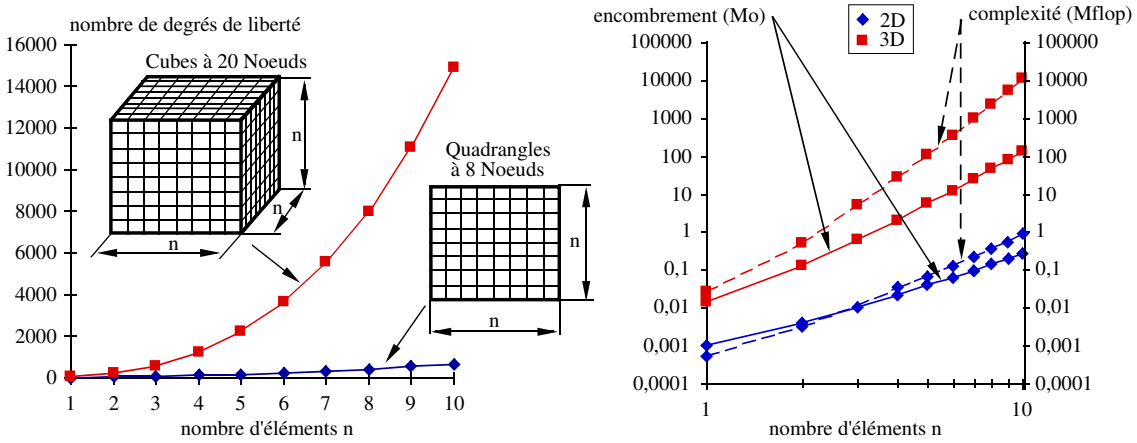


Figure 1 : comparaison du nombre d'inconnues, de la taille mémoire et de la complexité des problèmes 2D et 3D.

Problèmes numériques

Les problèmes posés sont alors multiples. La mise en place du modèle géométrique fin et du modèle de calcul contenant autant d'inconnues amène à manipuler de très grosses quantités d'informations.

De plus, la matrice de rigidité, une fois assemblée, occupe un espace mémoire très important, souvent plus grand que celui disponible sur les calculateurs courants. Son stockage et sa factorisation deviennent délicats. Des écritures sur disques sont donc nécessaires et ralentissent fortement le processus. Il est alors nécessaire, pour de telles modélisations, de générer des outils facilitant l'entrée et la manipulation des données et conduisant à des encombrements mémoire réduits.

Pour les problèmes non-linéaires, d'autres difficultés apparaissent. La volonté de prendre en compte les effets de structure (zones à fort gradient de contrainte, redistribution de contraintes…), aussi bien que celle de modéliser finement le comportement des matériaux (plasticité, viscoplasticité, matériaux composites…) en développant des lois de comportement sophistiquées prenant en compte de nombreux couplages mécaniques, de nombreuses variables internes, ou même le développement de techniques de calcul adaptées aux problèmes très fortement non-linéaires, conduisent à des coûts de calcul prohibitifs, voire hors de portée, pour les architectures des ordinateurs séquentiels actuels.

De plus, lorsque le chargement est complexe (chargement cyclique, dynamique rapide…), la discrétisation temporelle se doit d'être fine. Un grand nombre d'instants est donc à considérer. Le nombre d'opérations à réaliser est donc très important et le volume de résultats à stocker est considérable.

Les coûts de calcul sont généralement issus des deux types distincts de problèmes qui se présentent lors de l'analyse de structures : les problèmes non-linéaires, mais généralement locaux en variables d'espace qui traduisent l'évolution du comportement du matériau, et les grands systèmes globaux linéaires, qui traduisent classiquement l'équilibre de la structure.

3. Calcul parallèle

3.1 Calculateurs parallèles actuels

Les machines monoprocesseur ont vu la puissance de base de leur processeur augmenter notablement ; cependant, des limites physiques sont apparues et pour permettre le passage à une classe de machines supérieure, les architectures vectorielles puis parallèles ont émergé.

Aujourd'hui, ces dernières sont celles qui permettent d'atteindre les plus grandes puissances de calcul, figure 2 et [Dongarra 99], en particulier avec le développement de processeurs et de réseaux

d'interconnexion permettant d'obtenir un parallélisme « extensible » utilisant pleinement les possibilités d'une machine (accroissement du nombre de processeurs utilisés avec la taille du problème à traiter, avec pas ou peu de dégradation de performances).

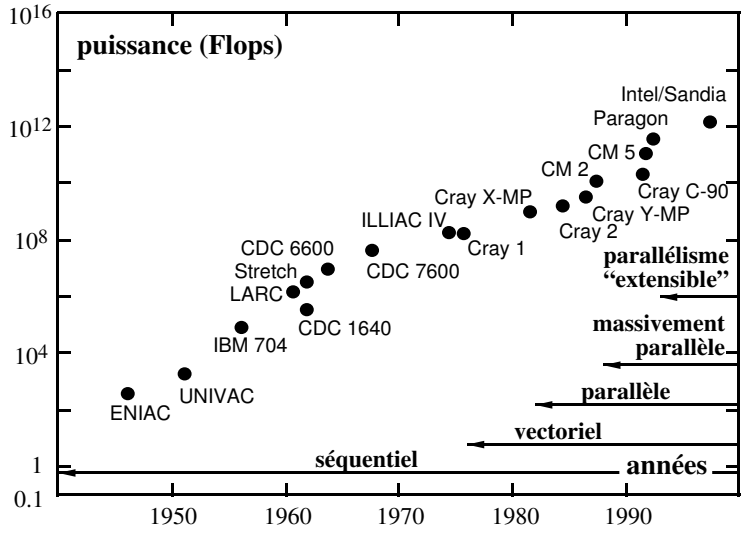


Figure 2 : évolution de la puissance de calcul des ordinateurs (d'après [Noor 97]).

Principes

Typiquement, une machine parallèle se présente comme une collection de « nœuds » de calcul sur lequel se trouve entre autres le processeur de calcul, figure 3. Les différents processeurs peuvent communiquer entre eux par l'intermédiaire du réseau d'interconnexion. La mémoire est ensuite soit distribuée (chaque processeur n'a accès qu'à sa propre mémoire), soit partagée par tous les processeurs (plus pratiquement, il s'agit d'une mémoire physiquement distribuée mais globalement adressable par tous les processeurs). Sur les machines récentes, tous les processeurs peuvent fonctionner indépendamment, c'est à dire peuvent a priori effectuer des tâches distinctes les uns des autres : c'est le mode MIMD (Multiple Instruction flow, Multiple Data flow) ; les machines à mémoire distribuée sont actuellement les machines qui peuvent être pourvues de la mémoire totale la plus grande, ce qui permet d'offrir à l'utilisateur une plus grande capacité de traitement, et une plus grande puissance de calcul (somme des puissances des processeurs utilisés).

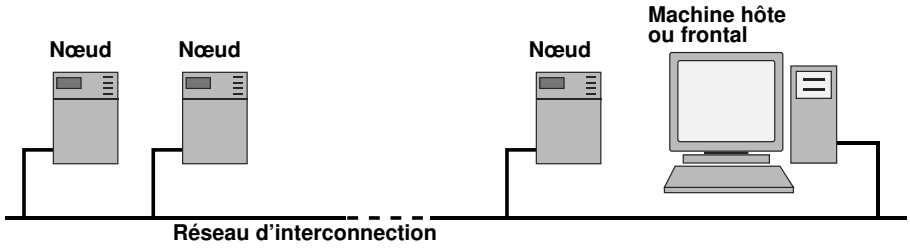


Figure 3 : principe de construction d'une machine parallèle.

Pour permettre les échanges d'information entre les processeurs, en particulier pour les machines à mémoire distribuée, il est nécessaire de pouvoir disposer d'outils permettant le transfert par le réseau d'interconnexion. De tels utilitaires existent aujourd'hui de façon standard dans les librairies d'échange de messages (ou « message passing »), telles que PVM (Parallel Virtual Machine system) ou MPI (Message Passing Interface), [Geist 94, Gropp 94].

Utilisation pour le calcul de structures

Comme on l'imagine facilement, les calculs locaux indépendants, sur des données locales, qui proviennent de l'intégration de la loi de comportement, peuvent être résolus aisément en parallèle ; par exemple en distribuant les éléments parmi les processeurs disponibles [Barragy 88, Whirley 89, Débordes 89]. Quant aux grands systèmes globaux linéaires, dans le cas de sollicitations quasi-statiques ou lentes, ils conduisent à une résolution implicite globale sur toute la structure, qui ne présente donc pas le caractère « parallèle » des équations de comportement du matériau. Notons que dans le cas de sollicitations dynamiques transitoires rapides (suivi de la propagation d'une onde dans une structure, par exemple), des techniques de résolution explicites permettent, outre une écriture plus simple, de ne traiter l'équilibre que de façon quasi-locale et se prêtent donc bien à une résolution en parallèle, [Oakley 95, Fahmy 95]. Avec ces avantages, ces méthodes sont actuellement aussi largement utilisées pour des problèmes de dynamique transitoire lente (crash par exemple) ; le problème de pas de temps critique très petit, en particulier lors de phénomènes de contact intermittents ou avec l'utilisation d'éléments finis de taille très petite, conduit cependant aussi à des résolutions très coûteuses. Des études pour le traitement en analyse implicite sont en cours. Enfin, ces techniques explicites ont aussi inspiré des extensions numériques pour les cas statiques, [Hugues 83] par exemple.

3.2 Parallélisme informatique

Pour tirer parti de ces nouvelles architectures, la parallélisation automatique (que l'on pourrait qualifier de parallélisme « informatique ») de codes de calculs existant sur machines séquentielles a été développée. Les différentes techniques utilisées vont de l'analyse automatique de dépendance de sections de code, du déroulage de boucle en vue de leur vectorisation et parallélisation [Lamour 91, Roux 91] à la génération automatique de code parallèle à partir d'un code séquentiel [André 95].

Ces techniques permettent de réutiliser au maximum un code séquentiel existant pour produire un code capable de s'exécuter sur machines modérément parallèle ; elles sont aujourd'hui les techniques les plus transparentes pour le développeur. L'inconvénient majeur est la relative faible performance parallèle produite, qui limite l'utilisation sur un nombre faible de processeurs (de l'ordre de 4). De plus, ces techniques ne sont pratiquement disponibles que sur les architectures à mémoire partagée. Elles font encore l'objet de projets européens sur HPF (High Performance Fortran) comme PPPE (Meiko UK) ou PANDORE (IRISA Rennes).

D'un point de vue moins transparent au programmeur, il existe aussi des techniques consistant à insérer des directives de parallélisation dans le source des codes pour indiquer au compilateur les opérations réalisables en parallèle. Citons notamment OpenMP (http://www.openmp.org/) qui devient actuellement un standard parmi ce type d'outils, toujours pour une mémoire partagée ou globalement adressable. Ces techniques sont bien adaptées à la résolution de l'intégration des lois de comportement, et au traitement des calculs explicites. Cependant, elles ont peu d'efficacité en ce qui concerne la résolution de grands systèmes linéaires creux ; on leur préfèrera donc dans ce cas les techniques présentées dans les paragraphes suivants.

3.3 Parallélisme numérique

Outre les techniques précédentes, se développent aussi des méthodes de résolution en parallèle de grands systèmes linéaires creux (typiques de ceux auxquels conduit une discrétisation éléments finis), que ce soit de manière directe ou itérative, de façon quasi indépendante du type de problème qui leur a donné lieu, [Utku 86, Roditis 90, Buoni 93, Pan 93]. Il s'agirait alors de parallélisme « numérique. »

Ces techniques sont encore très proches du type d'architecture de la machine cible, elles ne possèdent donc que peu de portabilité. Elles sont classiquement utilisée en « boîte noire » fournie par le constructeur de la machine, et appelées comme librairies système par le développeur de code.

Tirant mieux parti des applications traitées, pour la résolution de systèmes linéaires, elles possèdent une plus forte « granularité » (le volume de calcul à réaliser par processeur reste élevé vis-à-

vis de la quantité d'informations à échanger avec les autres processeurs au travers du réseau de communication) que les techniques précédentes. Elles permettent donc d'atteindre une meilleure performance lorsque le nombre de processeurs utilisés croît [Petiton 93].

Certaines librairies existent aussi contenant des utilitaires de résolution de grands systèmes linéaires creux sur ce mode, comme P-SPARSLIB [Saad 95].

3.4 Parallélisme mécanique - décomposition de domaine

Ici, nous nous plaçons dans une troisième catégorie : celle d'un parallélisme « mécanique, » dans laquelle se classent les méthodes de type décomposition de domaine.

Typiquement, un domaine physique est partitionné en plusieurs sous-domaines, et une technique est utilisée pour retrouver la solution par la résolution d'une succession de sous-problèmes indépendants associés aux sous-domaines. Chaque processeur gère un ou plusieurs sous-domaines de la partition, et les solutions partielles sont combinées pour produire une solution approchée du problème de départ. Ces techniques jouent sur le fait que chaque processeur peut traiter une grande part du travail indépendamment, et que le travail nécessaire pour construire la solution globale à partir de solutions locales n'est pas trop grand en comparaison de celui nécessaire pour obtenir les solutions locales.

Pour rompre la globalité en variables d'espace des équations d'équilibre d'une structure, ces méthodes procèdent à une décomposition en sous-domaines pour ne plus faire intervenir qu'une succession de résolution de problèmes quasi-locaux sous-domaine par sous-domaine. Le problème résultant est alors, bien entendu, celui du raccord entre sous-domaines voisins, [Escaig 94, Farhat 91, Yagawa 93, De Roeck 92].

Plus encore que les techniques précédentes, ces approches conduisent à un parallélisme à « grain large. » Cependant, la nature généralement synchrone de ces algorithmes pousse à se préoccuper de l'équilibrage des charges entre processeurs pour obtenir des efficacités élevées [Farhat 93]. À ce sujet, les décomposeurs automatiques de maillages ont beaucoup progressé à la fois dans la rapidité de traitement, et dans l'équilibrage statique des volumes de calculs. Actuellement, plusieurs librairies proposent un tel outil de décomposition automatique, comme METIS (U. of Minnesota, [Karypis 99]), CHACO (Sandia Nat. Lab., [Hendrickson 95]), TOP/DOMDEC (U. du Colorado, [Farhat 95]), JOSTLE (U. of Greenwich, [Walshaw 97]) ou SCOTCH (U. Bordeaux I, [Pellegrini 96]).

La description de plusieurs versions modernes de ces approches de décomposition de domaine sans recouvrement fait l'objet du chapitre suivant.

4. Méthodes de décomposition de domaine

Les méthodes de décomposition de domaine furent tout d'abord initiées sans chercher explicitement à introduire le parallélisme dans le traitement d'un problème implicite, c'est à dire qui conduit à exprimer un couplage entre toutes les inconnues du problème.

Avec l'avènement des calculateurs parallèles, de nombreuses recherches en mécanique se sont développées afin d'utiliser ces nouvelles architectures, d'abord en mécanique des fluides, puis en calcul de structures. En particulier, l'analyse par éléments finis a conduit au développement de techniques et d'algorithmes nouveaux, [White 88, Mackerle 96]. Dans ce cadre, ces méthodes connaissent un nouvel essor.

Elles mettent classiquement en jeu une décomposition en sous-domaines recouvrants (méthode de Schwarz) ou non recouvrants (méthode de Schur). Cette dernière classe de méthode est la plus utilisée en pratique pour la résolution du problème linéaire issu d'une analyse par éléments finis. La suite de cette partie lui est donc consacrée. En particulier, les approches correspondantes se classent parmi les approches primales (privilégiant les quantités cinématiques : le déplacement), duales (privilégiant les quantités statiques : les efforts), ou mixtes (traitant les deux types de quantités de façon

équivalente).

Dans toute la suite, on s'intéressera à la résolution du système linéaire (1).

4.1 Méthode directe de Schur primal

Aussi appelée méthode de sous-structuration, la méthode du complément de Schur a semble-t-il été introduite dans [Przemieniecki 63]. Grâce au partitionnement du domaine qu'elle entraîne, son intérêt est aussi apparu dans la création d'algorithmes adaptés aux calculateurs parallèles. En particulier, la figure 4 illustre une décomposition qui met en évidence différents sous-domaines géométriquement indépendants, ainsi qu'une interface globale qui les relie tous entre eux.

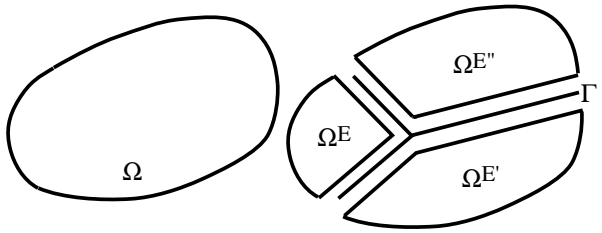


Figure 4 : décomposition en sous-domaines, et interface globale.

Condensation sur l'interface

L'approche algébrique consiste alors en une renumérotation formelle des nœuds en vue de présenter un découpage en blocs de la matrice de rigidité [K] : si on numérote en premier les degrés de liberté dont les nœuds sont internes au premier sous-domaine (au sens strict : pas de nœuds sur la frontière qui le sépare des autres sous-domaines), puis ceux du deuxième sous-domaine, ainsi de suite... jusqu'à numéroter en dernier ceux de l'ensemble des frontières entre sous-domaines Γ. Par exemple, pour une découpe en deux sous-domaines Ω_E et $\Omega_{E'}$, on est formellement amené à écrire le système sous la forme :

$$\begin{bmatrix} K_{EE} & 0 & K_{E\Gamma} \\ 0 & K_{E'E'} & K_{E'\Gamma} \\ K_{\Gamma E} & K_{\Gamma E'} & K_{\Gamma\Gamma} \end{bmatrix} \begin{Bmatrix} q_E \\ q_{E'} \\ q_\Gamma \end{Bmatrix} = \begin{Bmatrix} f_E \\ f_{E'} \\ f_\Gamma \end{Bmatrix}$$

La condensation statique sur l'interface Γ du problème discrétisé n'est alors rien d'autre qu'une substitution des inconnues q_E et $q_{E'}$ (élimination de Gauss par bloc), de façon à se ramener aux seules inconnues q_Γ :

$$[S]\{q_\Gamma\} = \{b\}$$

avec

$$[S] = [K_{\Gamma\Gamma}] - \sum_E [K_{\Gamma E}]\,[K_{EE}]^{-1}\,[K_{E\Gamma}] \quad \text{et} \quad \{b\} = \{f_\Gamma\} - \sum_E [K_{\Gamma E}]\,[K_{EE}]^{-1}\,\{f_E\}$$

[S] est appelée matrice du complément de Schur [Bjørstad 86, Agoshkov 88], et {b} est le résultat de la condensation des efforts généralisés {f} sur Γ.

Comme $[K_{\Gamma\Gamma}]$ peut être considéré comme la contribution à une partie de la rigidité, des éléments de différents sous-domaines, il peut s'écrire

$$[K_{\Gamma\Gamma}] = \sum_E [K_{\Gamma\Gamma}{}^E]$$

et de même, [S] est l'assemblage des $[S^E]$, matrices de rigidités (condensées) des Ω_E considérées comme des « super-éléments, » qui se trouvent être des matrices pleines. Il en est de même pour $\{f_\Gamma\}$ et donc pour {b}, ainsi :

$$[S] = \sum_E [S^E] \quad \text{et} \quad \{b\} = \sum_E \{b^E\}$$

où $\sum_E$ désigne ici abusivement l'assemblage des matrices locales aux sous-domaines $[S^E]$.

Parallélisation

Considérons maintenant le problème condensé. L'assemblage explicite des $[S^E]$ peut conduire, lorsque le nombre de degrés de liberté est grand sur l'interface Γ (par exemple, lors d'une décomposition en un nombre élevé de sous-domaines), à une matrice [S] creuse au sens des super-éléments mais possédant quand même une connectivité élevée et donc une grande largeur de bande.

Une première approche consiste à opérer une résolution directe du problème condensé à partir de cette construction explicite de la matrice de Schur [S]. Le calcul des rigidités « élémentaires » $[S^E]$ des super-éléments Ω_E est réalisé par une méthode de type frontale [Irons 70] appliquée aux rigidités des sous-domaines Ω_E :

$$[K^E] = \begin{bmatrix} K_{EE} & K_{E\Gamma} \\ K_{\Gamma E} & K_{\Gamma\Gamma}{}^E \end{bmatrix}$$

en conservant comme dernier front, l'ensemble des « nœuds maîtres » de Γ. La méthode frontale est en fait une forme récursive de condensation des nœuds internes sur les nœuds maîtres.

Le fait de pouvoir réaliser cette opération sur plusieurs sous-domaines Ω_E (simultanément) a conduit à qualifier cette technique de méthode multi-frontale [Duff 86, Escaig 94, Farhat 89].

Bien entendu, la phase d'assemblage des $[S^E]$ et $\{b^E\}$ est séquentielle, ainsi que la résolution du système condensé $[S]\{q_\Gamma\} = \{b\}$ (de petite taille s'il y a peu de nœuds sur l'interface Γ, mais de largeur de bande élevée). On peut néanmoins utiliser pour ce système un solveur direct en parallèle (parallélisme « numérique »), s'il est de taille suffisante pour amortir les coûts de parallélisation de cette phase, voir [Farhat 88, Heath 91, Gupta 95] par exemple.

Pour chaque sous-domaine (en parallèle)			Pour le processeur maître (en séquentiel)
création des rigidités élémentaires de $[K^E]$			
condensation locale pour $[S^E]$ et $\{b^E\}$			
envoi par message de $[S^E]$ et $\{b^E\}$	→	→	réception et assemblage des [S] et {b}
			résolution de $[S]\{q_\Gamma\} = \{b\}$
réception de $\{q_\Gamma{}^E\}$	←	←	envoi par message des $\{q_\Gamma{}^E\}$
restitution aux nœuds internes de $\{q^E\}$			

Table 1 : algorithme multi-frontal parallèle.

L'algorithme de résolution est décrit dans la table 1, en supposant une machine parallèle MIMD à mémoire distribuée, avec un mécanisme d'échange de message comme système de communication entre processeurs. La phase de post-traitement, peu coûteuse, consiste à revenir aux degrés de liberté internes aux sous-domaines par :

$$\{q^E\} = [K_{EE}]^{-1} (\{f_E\} - [K_{E\Gamma}] \{q_\Gamma\})$$

Quand le nombre de sous-domaines est important, le nombre de nœuds de Γ peut devenir assez élevé et la taille du problème condensé croît en conséquence. Par exemple, pour le problème test d'élasticité bidimensionnelle de la figure 5, la table 2 reporte en fonction du nombre de sous-domaines employés, l'encombrement des matrices de rigidité locales $[K^E]$ ainsi que celle de la matrice condensée [S] (H représente le côté d'un sous-domaine et h celui d'un élément fini). Pour pallier cet inconvénient, des étapes supplémentaires dans la condensation peuvent ainsi être envisagées : condensations successives sur des problèmes de taille de plus en plus réduite [Escaig 94]. Cette méthode apparaît performante, au moins pour des nombres de processeurs modérés (de l'ordre de 10). Pour conserver un même algorithme de numérotation des degrés de liberté dans la matrice $[K^E]$, $[K_{EE}]$ peut être numérotée à part, à condition de prévoir un stockage particulier des condensées élémentaires $[S^E]$ et des blocs qui lui sont connectés suivant des considérations de remplissage optimal [Escaig 92].

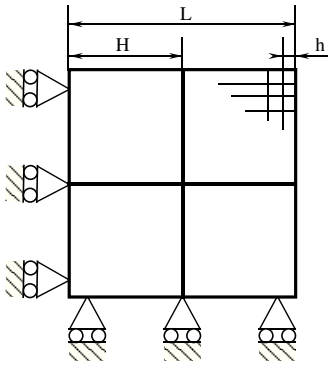


Figure 5 : problème test pour la condensation.

nombre de sous-domaines	64	16	4	1
encombrement de $[K^E]$ (nombre de ddl)	0.3 Mo (882)	2 Mo (3362)	18 Mo (13122)	148 Mo (51842)
encombrement de [S] (nombre de ddl)	13 Mo (4466)	6 Mo (1938)	1.4 Mo (650)	-

Table 2 : caractéristiques des condensations.

Issues du domaine de l'analyse numérique, les solveurs linéaires itératifs en parallèle qui sont construits sur ces approches participent souvent des techniques de gradient conjugué, beaucoup étudiées comme algorithmes itératifs utilisables sur calculateurs multiprocesseurs, [Saad 85, Johnsson 89, Mansfield 90] par exemple. C'est le cas des deux méthodes suivantes.

4.2 Méthode itérative de Schur primal

Dans le cas de figure où le nombre de sous-domaines est plus important (de l'ordre de la centaine ou plus) il n'est plus intéressant de résoudre explicitement le système condensé directement. Les méthodes itératives deviennent alors compétitives, en particulier car elles ne nécessitent pas d'expliciter la matrice du complément de Schur [S] ni de former sa factorisée. De plus, le fait que le conditionnement

de [S] soit meilleur que celui de la matrice de rigidité [K], voir [Le Tallec 94], plaide en faveur du développement d'algorithmes itératifs de résolution du problème condensé, par exemple de type gradient conjugué : le conditionnement de [S] est asymptotiquement $o(\frac{1}{H^2}(1+\frac{H}{h}))$, alors que celui de [K] est $o(\frac{1}{h^2})$.

Dans la méthode du gradient conjugué, plusieurs phases sont à réaliser : le produit de la matrice condensée [S] par un vecteur $\{q_\Gamma\}$ pour estimer le résidu $\{r\} = \{b\} - [S]\{q_\Gamma\}$ au cours des itérations, le calcul de la norme de ce résidu $\{r\}^T\{r\}$ pour obtenir un critère d'arrêt des itérations, et le calcul du pas de descente optimal, qui nécessite un produit scalaire du type $\{q_\Gamma\}^T[S]\{q_\Gamma\}$. Le fait que la résolution se fasse en parallèle implique que toutes les quantités précédentes soient accessibles uniquement sous forme distribuée, c'est à dire sous forme de contributions relatives aux différents sous-domaines comme $\{q_E\}$, $\{f_E\}$...

Le produit de [S] par un vecteur $\{q_\Gamma\}$ n'intervient que sous forme de contributions locales $[S^E]\{q_\Gamma{}^E\}$ et son évaluation ne nécessite pas la connaissance explicite de [S] : elle est effectuée en deux temps :

– résolution du problème de Dirichlet avec $\{q_\Gamma{}^E\}$ comme valeurs imposées sur le bord Γ, ce qui demande d'avoir factorisé $[K_{EE}]$ lors de la phase d'initialisation ;

– calcul des efforts bord généralisés $[S^E]\{q_\Gamma{}^E\} = [K_{\Gamma E}]\{q^E\} + [K_{\Gamma\Gamma}{}^E]\{q_\Gamma{}^E\}$

L'interprétation de $[S^E]$ est alors la suivante : à un champ de déplacement bord imposé comme seul chargement, $[S^E]$ fait correspondre le champ d'effort bord associé par le comportement du sous-domaine Ω_E. À un champ de déplacement bord, [S] fait donc correspondre le déséquilibre bord associé.

Les deux produits scalaires précédents, $\{r\}^T\{r\}$ et $\{q_\Gamma\}^T[S]\{q_\Gamma\}$, s'obtiennent avec des techniques classiques de calcul de produit scalaire réparti (en utilisant des techniques de communication du type descente logarithmique ou diffusion générale, voir [Kumar 94] par exemple). Des routines spécialisées pour réaliser ces opérations sont présentes dans les librairies d'échange de messages déjà mentionnées telles que PVM ou MPI.

L'algorithme présente donc, au cours d'une itération, deux synchronisations (tous les processeurs attendent le plus lent d'entre eux, ou le plus chargé, pour les produits scalaires distribués) pour des opérations globales, et un échange de messages entre processeurs « voisins. » Ces derniers sont déterminés par la nécessité pour les déplacements sur l'interface Γ, qui sont les inconnues principales, d'avoir une continuité (du moins pour le problème discrétisé). Le voisinage entre sous-domaines sera alors dicté par le fait que l'intersection entre frontières de deux sous-domaines est non vide ; deux sous-domaines connectés par un coin (ou une arête en 3D) sont donc considérés comme voisins et s'ils sont gérés par des processeurs différents, il faudra échanger des messages entre ceux-ci au cours des itérations pour communiquer les quantités bord locales. Les échanges de messages seront d'autant plus intensifs que la connectivité entre sous-domaines voisins est grande.

Bien entendu, comme la majorité des applications de la méthode du gradient conjugué, il est nécessaire de procéder à un préconditionnement pour accélérer la convergence. En analyse numérique, beaucoup de développements récents concernent la construction de préconditionneurs efficaces et qui puissent être réalisés en parallèle. Citons par exemple [Le Tallec 94, Mandel 93].

Le gradient conjugué sur le problème condensé peut d'ailleurs s'interpréter comme un gradient conjugué préconditionné sur le problème de départ : le préconditionneur est alors une résolution « exacte » par sous-domaine.

4.3 Méthode itérative de Schur dual - méthode FETI

Cette méthode, initiée dans [Farhat 91], utilise comme inconnue principale les inter-efforts à l'interface entre sous-domaines, et non plus les déplacements à l'interface ; il s'agit donc d'une méthode « duale. » L'équilibre du sous-domaine Ω^E s'écrit ainsi :

$$[K^E]\{q^E\} = \{f^E\} - [B^E]^T\{\lambda\} \qquad (2)$$

où les $\{\lambda\}$ sont des multiplicateurs de Lagrange, interprétés comme des inter-efforts généralisés sur l'interface Γ, et $[B^E]$ une matrice booléenne signée, qui repère juste les degrés de liberté du sous-domaine qui sont situés sur Γ. Les matrices $[B^E]$ ne sont d'ailleurs pas construites explicitement, seuls les degrés de liberté sur le bord d'un sous-domaine sont repérés. Le problème de raccord en déplacement des sous-domaines s'écrit alors $[B]\{q\} = \{0\}$ où $[B]$ désigne l'assemblage des $[B^E]$. Notons que si le sous-domaine Ω^E est flottant (n'a pas de frontière à déplacement imposé), pour que le problème (2) ait une solution, il faut que le second membre soit à résultante et moment nul. À partir de ce moment, la solution est définie à un mouvement de solide rigide près sous la forme

$$\{q^E\} = [K^E]^+ (\{f^E\} - [B^E]^T\{\lambda\}) + [R^E]\{\alpha^E\}$$

où $[K^E]^+$ est un inverse généralisé de $[K^E]$, $[R^E]$ l'ensemble des modes rigides du sous-domaine, et $\{\alpha^E\}$ les coefficients de la combinaison linéaire des modes rigides présents (vecteur de taille maximum 6 en 3D).

Condenser le problème de raccord sur les $\{\lambda\}$ s'écrit alors :

$$[F_I]\{\lambda\} = \{d\} - [B][R]\{\alpha\} \qquad (3)$$

où $[F_I] = \sum_E [B^E][K^E]^+[B^E]^T$, $\{d\} = \sum_E [B^E][K^E]^+\{f^E\}$, $[R]$ et $\{\alpha\}$ étant les assemblés des $[R^E]$ et $\{\alpha^E\}$, les $\{\lambda\}$ devant être obligatoirement dans l'ensemble des efforts à résultante et moment nul sur tous les sous-domaines. Ce problème est alors résolu par une méthode de gradient conjugué projeté dans cet ensemble et on appelle $[P]$ un opérateur de projection correspondant. Le plus simple d'entre eux est le suivant

$$[P] = [I_d] - ([B][R])\left(([B][R])^T([B][R])\right)^{-1}([B][R])^T$$

où $[I_d]$ est la matrice identité. Le problème (3) se réécrit alors (car on peut montrer aisément que $[P]^T[B][R]\{\alpha\} = \{0\}$) :

$$([P]^T[F_I][P])\{\lambda\} = [P]^T\{d\}$$

Au cours des itérations du gradient conjugué, cette fois-ci, l'application de $([P]^T[F_I][P])$ à un vecteur courant consiste à résoudre un problème de Neumann sur chaque sous-domaine (efforts imposés sur le bord lors du calcul de l'application de $[K^E]^+$ au vecteur d'effort équilibré sur chaque sous-domaine Ω^E) et une étape supplémentaire consiste à appliquer deux projections par itération. Chacune est un problème global sur l'ensemble des sous-domaines, donc plus difficilement parallélisable, mais ne porte que sur les mouvements de solide rigide de ceux-ci (au plus 6 inconnues par sous-domaine en 3D) ; l'opérateur correspondant est effectivement $([B][R])^T([B][R])$.

La présence de ce problème global permet de qualifier cette méthode de multi-échelles. L'intérêt est qu'elle devient ainsi extensible. En particulier, avec un préconditionneur de Dirichlet additionnel, le conditionnement de la matrice d'itération du problème à résoudre devient $o(1 + \log^2\frac{H}{h})$. Le nombre

d'itérations nécessaires pour converger ne dépend alors peu de H/h et pas de H, [Farhat 94] : l'algorithme est ainsi qualifié de quasi-optimal (faible dépendance en H/h) et numériquement extensible (pas de dépendance en H).

4.4 Méthode itérative mixte

Dans cette dernière approche, ni le déplacement, ni les inter-efforts aux interfaces ne sont a priori continus au cours des itérations (sauf à convergence). Les approches en déplacement (Schur primal) utilisent des résolutions locales à déplacement imposé (conditions aux limites de Dirichlet) ; les approches en effort (duales) utilisent des résolutions locales à effort imposé (conditions aux limites de Neumann) ; la méthode mixte ici présentée utilise des résolutions locales à conditions aux limites mixtes de type Robin, ou Fourier. Elle est issue des approches à grand incrément de temps, développées au LMT-Cachan, [Ladevèze 96], et sa particularité est de traiter une interface $\Gamma^{EE'}$ entre les sous-structures Ω^E et $\Omega^{E'}$ comme une entité mécanique à part entière, possédant son propre comportement (celui décrit ici sera du type interface parfaite).

Une sous-structure Ω^E doit donc posséder un déplacement égal à $\{q^{EE'}\}$ sur le bord en relation avec l'interface $\Gamma^{EE'}$ et doit aussi équilibrer un effort $\{f^{EE'}\}$, pour tous les voisins :

$$[K^E]\{q^E\} = \sum_{E'} \{f^{EE'}\} \quad \text{et} \quad \{q^E\}_{\subseteq \Gamma^{EE'}} = \{q^{EE'}\} \tag{4}$$

Une interface $\Gamma^{EE'}$ parfaite doit assurer la continuité des déplacements et des efforts :

$$\{q^{EE'}\} = \{q^{E'E}\} \quad \text{et} \quad \{f^{EE'}\} + \{f^{E'E}\} = 0$$

Une solution satisfaisant successivement à chacun de ces deux groupes d'équations est successivement construite par l'algorithme lors des deux étapes que constitue une itération :

– l'étape locale consiste, connaissant une solution approchée $(\{q^{EE'}\},\{f^{EE'}\})$ satisfaisant au comportement des sous-structures, à construire une solution approchée $(\{\hat{q}^{EE'}\},\{\hat{f}^{EE'}\})$, satisfaisant au comportement des interfaces. Pour ce faire, une direction de recherche est nécessaire ; elle est choisie sous la forme :

$$(\{\hat{f}^{EE'}\} - \{f^{EE'}\}) + [k^{EE'}](\{\hat{q}^{EE'}\} - \{q^{EE'}\}) = \{0\}$$

– l'étape linéaire consiste, connaissant une solution approchée $(\{\hat{q}^{EE'}\},\{\hat{f}^{EE'}\})$, à construire $(\{q^{EE'}\},\{f^{EE'}\})$ satisfaisant au comportement des sous-structures. Une direction de recherche conjuguée de la précédente est utilisée :

$$(\{f^{EE'}\} - \{\hat{f}^{EE'}\}) - [k^{EE'}](\{q^{EE'}\} - \{\hat{q}^{EE'}\}) = \{0\}$$

associée aux conditions (4), elle conduit au problème de Robin local par sous-structure :

$$([K^E] + \sum_{E'} [k^{EE'}])\{q^E\} = \sum_{E'} \{\hat{f}^{EE'}\} + [k^{EE'}]\{\hat{q}^{EE'}\}$$

La convergence de l'algorithme est assurée dès que les $[k^{EE'}]$ sont définies strictement positives. Ce sont les paramètres de la méthode. De plus, la présence de ces « rigidités bord » prévient la présence de sous-structure flottante. Leur construction se fait de façon analogue à celle de matrices de convection en thermique, avec un coefficient « matériau » (ici direction de recherche) k qui peut être mis sous la forme $k = \frac{E}{L} I_d$, E étant le module d'Young et L une dimension caractéristique de la structure. Il influe

sur le taux de convergence, mais pas sur la solution à convergence (solution du problème de départ).

L'intérêt de cette approche est la possibilité d'utiliser très simplement des comportements non-linéaires d'interface (comme le contact unilatéral) de façon modulaire ; de tels exemples seront présentés dans la partie suivante. Dans le cas du comportement élastique linéaire, l'algorithme qui en résulte se confond avec celui proposés dans [Lions 90], présenté à partir du cas limite de la méthode de Schwarz avec recouvrement, [Glowinski 90] et [Ladevèze 85], à partir de lagrangiens augmentés. Il s'agit alors d'une méthode d'Uzawa de résolution de problèmes de point-selle (version dénommée ALG3 dans [Fortin 82]).

Des développements en dynamique sont aussi possibles [Boucard 99], de même qu'avec des comportements non-linéaires d'interface [Blanzé 95], l'utilisation de maillages non conformes [Champaney 99], des extensions multi-échelles [Dureisseix 98] pour assurer l'extensibilité, ainsi que des approches d'homogénéisation [Ladevèze 99].

4.5 Bilan

De façon générale, lorsque la taille des problèmes augmente, les méthodes itératives deviennent compétitives avec les méthodes directes. Dans des cas limites, les méthodes directes (factorisation de Cholesky ou de Crout) requièrent trop de mémoire pour pouvoir être raisonnablement effectuées.

Les méthodes de décomposition de domaine utilisent avantageusement une résolution directe (pour des raisons de robustesse) des problèmes locaux seulement (indépendants d'un sous-domaine à l'autre). Avec une approche multi-échelles (pour l'extensibilité), ces méthodes peuvent être vues comme un moyen de repousser les limitations des solveurs directs en combinant les avantages des solveurs itératifs (réduction de la ressource machine nécessaire, parallélisme à gros grain) et celles des solveurs directs (robustesse, efficacité sur les problèmes locaux de taille plus réduite). Il est à noter qu'à même niveau d'optimisation, un code même séquentiel utilisant une méthode de décomposition de domaine devient plus efficace qu'un code séquentiel direct classique lorsque la taille du problème croît.

Des résultats récents conduisent à penser que l'optimum du point de vue efficacité consiste à avoir un nombre de sous-domaines élevé, d'affecter par conséquent plusieurs sous-domaines par processeur, ce qui assoupli les contraintes d'équilibrage statique des charges, par une distribution dynamique possible des sous-domaines sur les processeurs.

5. Exemples d'application

Pour illustrer les performances des méthodes de décomposition de domaine, la technique de sous-structuration mixte présentée précédemment est utilisée. Cette approche est particulièrement utile aussi pour traiter des problèmes d'assemblages de structures élastiques, et permet aussi facilement de traiter les liaisons par contact, avec ou sans frottement, jeux ou joints élastomères..., qui rendent le problème non-linéaire. Les développements sont effectués au sein du code de calcul CASTEM 2000 développé au CEA de Saclay, [Verpeaux 88].

5.1 Exemple d'assemblage : liaison complète pignon-arbre creux

Cet exemple a été réalisé dans le cadre d'un bureau d'étude en préparation à l'agrégation de Mécanique à l'ENS de Cachan. Le thème de ce bureau d'étude est le dimensionnement d'un réducteur à arbre creux. La présentation faite aux étudiants avait pour but de montrer les apports du calcul dans le dimensionnement de la liaison encastrement entre le pignon et l'arbre creux (figure 6) par rapport aux méthodes analytiques classiques en bureau d'étude. L'étude porte sur le nombre de boulons nécessaires à la transmission du couple, leur précharge et la répartition de pression en charge sur la zone de contact entre le pignon et l'épaulement de l'arbre.

La figure 6 présente la géométrie de la liaison et son éclaté (la denture du pignon n'est pas représentée). Le modèle est complètement paramétré. L'arbre est supposé encastré en bout et un effort

— paramétré par le couple à transmettre et par la géométrie de la denture — est appliqué sur la partie supérieure du pignon. La figure 7 présente la décomposition en 54 sous-structures et 109 interfaces pour un modèle à neuf boulons de fixation. Ce modèle à neuf boulons comporte 141 000 degrés de liberté.

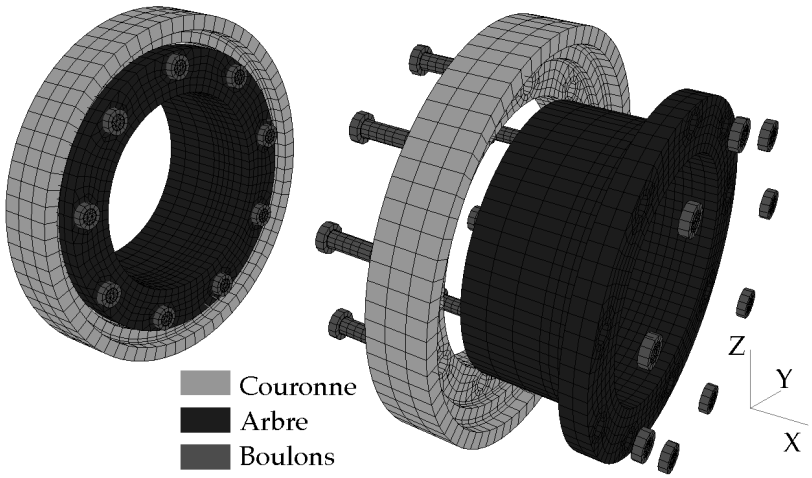


Figure 6 : pignon vissé sur un arbre creux.

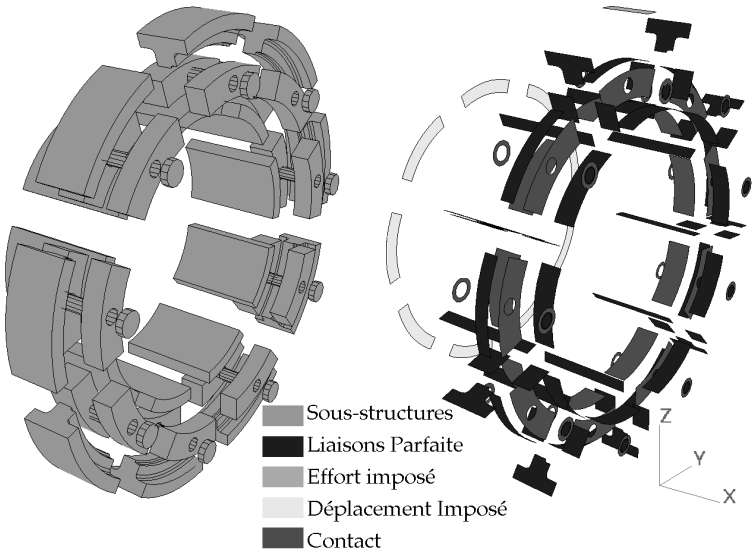


Figure 7 : décomposition en sous-structures et interfaces.

La figure 8 présente, pour un couple appliqué de 1 000 Nm, la répartition des contraintes axiales dans une portion de l'assemblage et dans les boulons associés. Elle présente ainsi, la répartition de pression sur la zone de contact qui est très éloignée de l'hypothèse de répartition constante effectuée pour le calcul analytique. On remarque alors que le calcul analytique classique est fortement surdimensionnant.

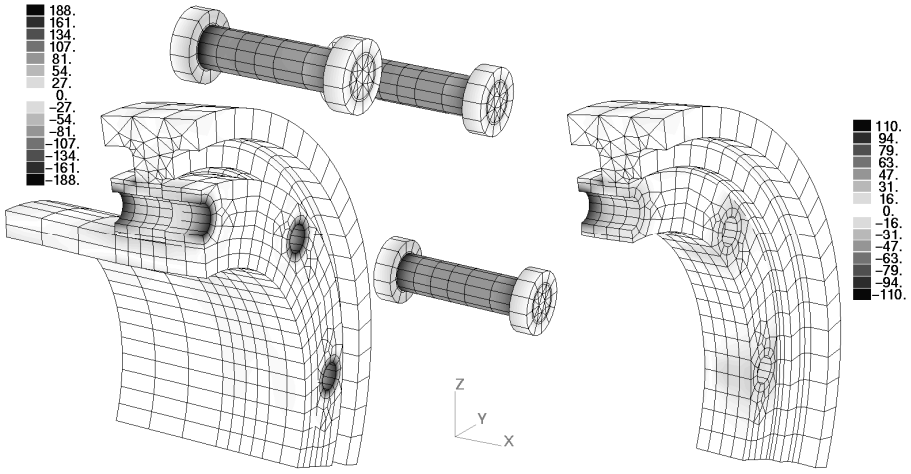


Figure 8 : contraintes axiale sur une portion de l'assemblage.

5.2 Un exemple à grand nombre de degrés de liberté : l'éprouvette biaxiale

La figure 9 représente les trois quarts d'une éprouvette biaxiale utilisée au LMT-Cachan pour des tests de fatigue sous chargement complexe. La zone utile est la partie centrale, et les peignes sont le dispositif d'accrochage à la machine d'essai. Un calcul de structure en vue d'optimiser la géométrie, [Cognard 96], est réalisé sur un seizième de l'éprouvette ; la même figure 9 présente une discrétisation intermédiaire (a) et les éléments utilisés sont des prismes à 15 nœuds et des cubes à 20 nœuds. Afin de montrer les performances d'une méthode de sous-structuration, la taille de ce même problème a été accrue par raffinement du maillage pour obtenir 7 problèmes de tailles croissantes (table 3). Un découpage manuel en 31 puis 62 sous-structures presque équilibrées (en nombre d'éléments) a été réalisé (figure 10).

maillage	1	2	3	a	b	c	d
nombre de ddl (de sous-structures)	175 740 (31)	259 077 (31)	340 734 (31)	340 734 (62)	483 702 (62)	627 486 (62)	1 199 553 (62)
coût CPU total sur IBM SP1	2 910 s	5 100 s	8 700 s				
coût CPU total sur SGI Origin 2000	930 s	1 635 s	2 681 s	1 945 s	3 368 s	5 476 s	17 217 s

Table 3 : caractéristiques des problèmes traités.

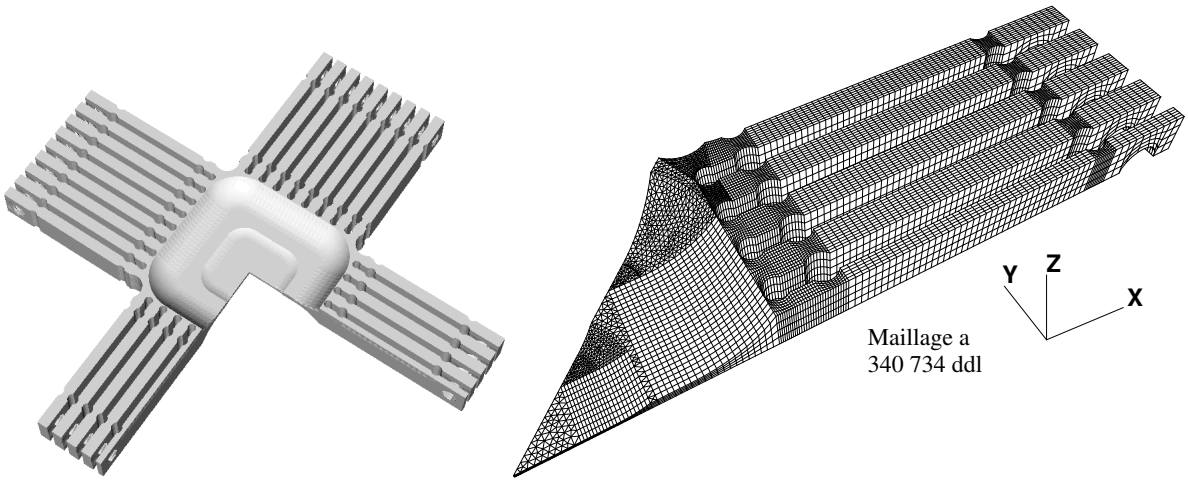


Figure 9 : éprouvette biaxiale et problème discrétisé en éléments finis.

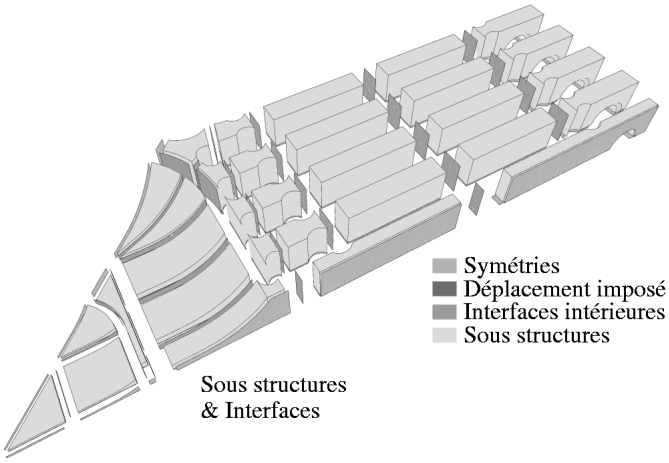


Figure 10 : décomposition en 31 sous-structures et interfaces.

La figure 11 reporte la capacité mémoire nécessaire à stocker l'ensemble des matrices de rigidité du problème pour les différents cas ; le calcul séquentiel correspondant n'a pu être mené que sur la première discrétisation, à cause des ressources trop grandes nécessaires à son traitement. Pour ce cas, traité sur un nœud d'une machine IBM SP1 du CEA de Saclay, le temps de calcul CPU est de 4 300 s. Pour cet exemple, traité par une méthode de décomposition de domaine, une bonne solution est obtenue après 100 itérations. En distribuant les sous-structures et les interfaces sur les 16 processeurs de l'IBM SP1, et en utilisant le système d'échange de messages PVM déjà mentionné, les trois premiers tests ont pu être traités. La table 3 reporte les coûts de calcul CPU correspondants (cumulés sur tous les processeurs). Lorsque la taille du problème croît encore, il est nécessaire d'utiliser des nombres de sous-structures et de processeurs plus importants. Les 7 cas déjà présentés ont ainsi été traités sur une machine SGI Origin 2000 à 32 processeurs et 8 Go de mémoire centrale partagée, du pôle parallélisme Île de France Sud pour les matériaux et les structures (http://www.lmt.ens-cachan.fr/Pole_IDFS/Welcome.html). La table 3 contient les coûts correspondants. La figure 12 reporte aussi ces derniers résultats.

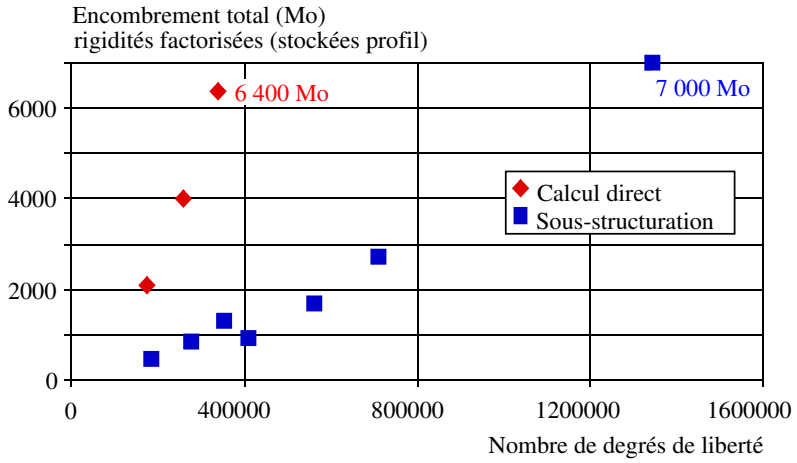


Figure 11: capacité mémoire nécessaire au traitement des différents problèmes.

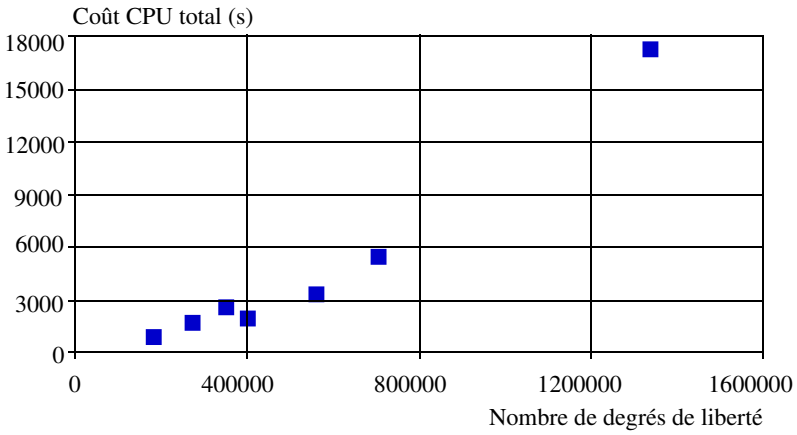


Figure 12 : coût CPU cumulé sur les processeurs pour résoudre les différents problèmes.

4.5 Exemple industriel : le différentiel automobile

L'exemple proposé est celui d'un différentiel automobile, pour lequel on modélise uniquement le boîtier, la couronne, le chapeau et les vis de fixation (figure 13). On considère des actions sur une dent de la couronne et les actions des roulements à billes sur le boîtier sont modélisées par l'intermédiaire de liaisons de type « contact unilatéral sans frottement. » L'axe des satellites est supposé encastré. Les actions des satellites et des planétaires sur le boîtier sont modélisées par des efforts dont l'amplitude et la direction sont paramétrées par le couple appliqué et par la géométrie des dentures (figure 14). Les précontraintes des vis sont imposées par l'intermédiaire d'interfaces adaptées, situées entre le corps et la tête de vis. Les sollicitations mécaniques associées à la rotation d'ensemble sont aussi prises en compte. Des interfaces de type contact unilatéral avec frottement sont utilisées pour modéliser les liaisons entre les différents composants (boîtier, couronne, chapeau et vis).

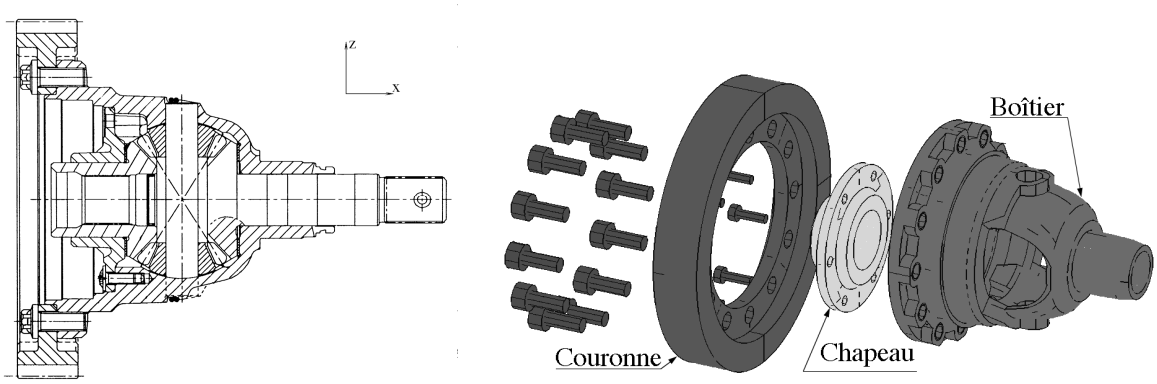


Figure 13 : différentiel et son modèle.

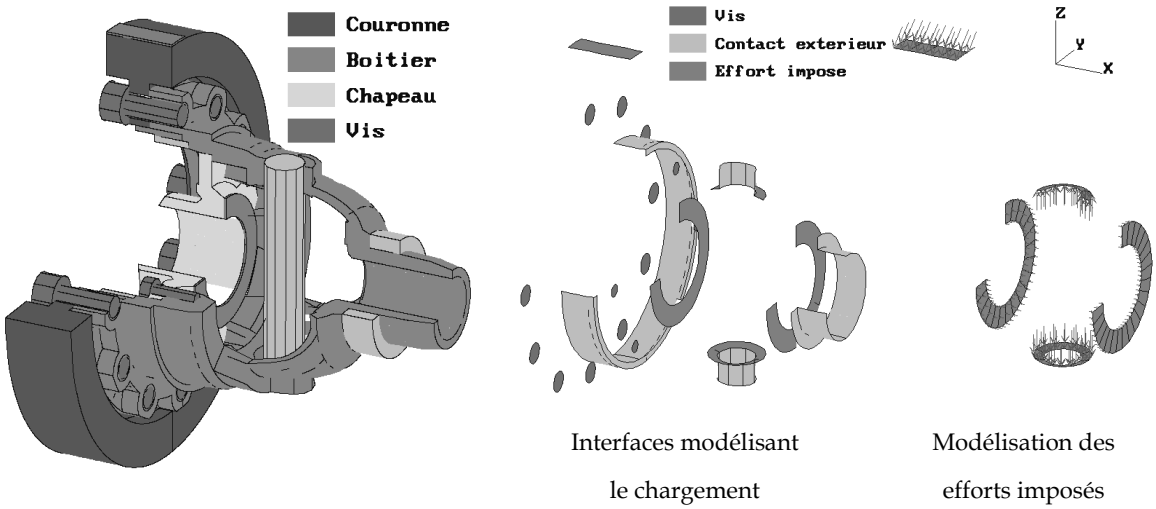
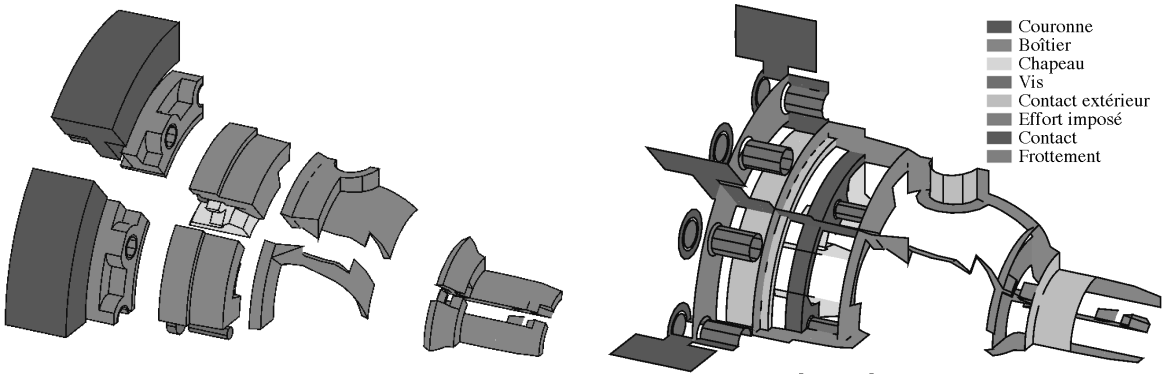


Figure 14 : modèle utilisé et chargement (3/4 du modèle).

Six modélisations différentes ont été réalisées (3 maillages différents et 2 décompositions par maillage). Le premier découpage en 39 sous-structures est la décomposition minimale de l'assemblage : le boîtier, le chapeau, la couronne, 18 corps de vis et 18 têtes. Les symétries du modèle et un décomposeur automatique ont été utilisés pour effectuer le second découpage en 84 sous-structures et 223 interfaces (figure 15). La table 4, présente les caractéristiques des différents maillages adoptés et l'encombrement de l'ensemble des matrices de rigidité pour les différents découpages. Ces résultats montrent que l'augmentation du nombre de sous-structures permet de réduire considérablement l'encombrement du problème.



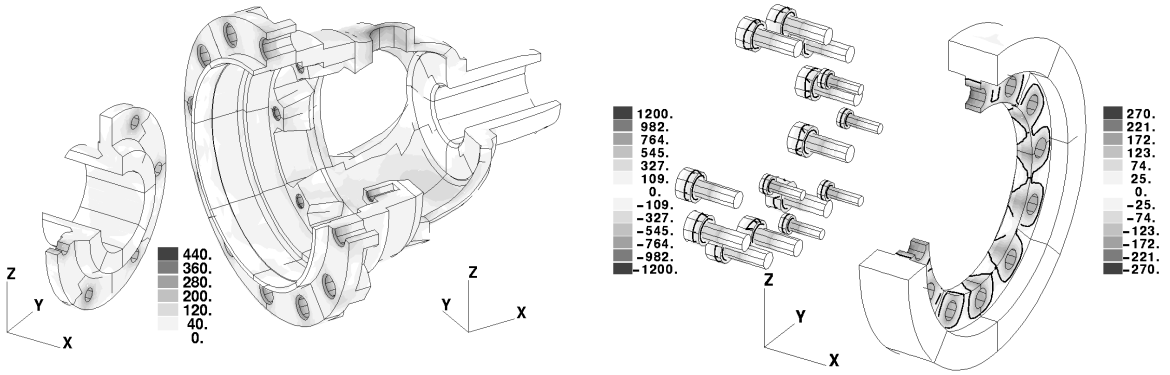
Sous-structures Interfaces

Figure 15 : découpage en 84 sous-structures (1/4 du modèle).

maillage	1	2	3
nombre de ddl	35 136	129 924	236 976
encombrement pour 39 sous-structures	128 Mo	1 391 Mo	4 574 Mo
encombrement pour 84 sous-structures	31 Mo	310 Mo	790 Mo

Table 4 : caractéristiques des problèmes traités.

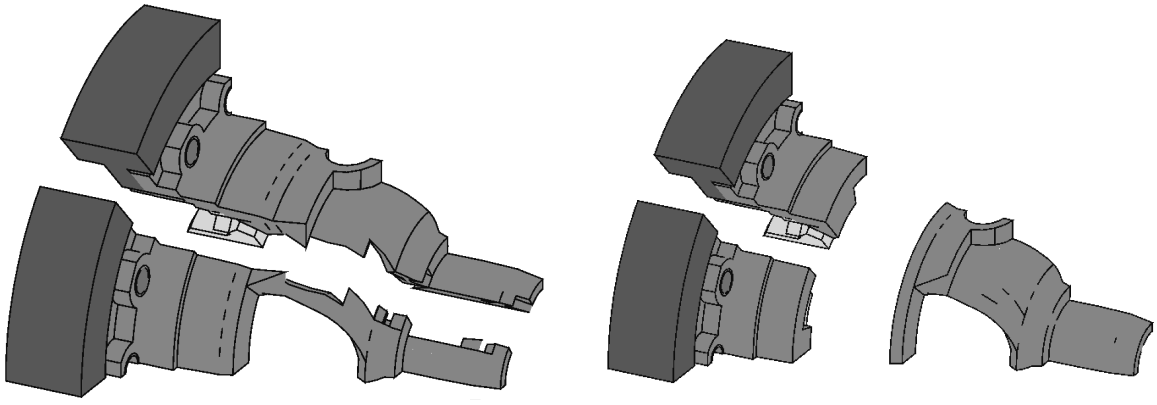
Pour un couple appliqué de 2 000 Nm et une vitesse de rotation de 2 000 tr/min, la convergence est atteinte après 200 itérations. La figure 16 donne les contraintes équivalentes de Mises à l'itération 200 tracées sur le boîtier et sur le chapeau. On donne aussi les contraintes axiales (σ_{xx}) sur les vis et sur la couronne. Cette figure donne la répartition de pression de contact entre la couronne et le boîtier (liaison avec frottement).



Contraintes équivalentes de Mises Contrainte axiale σ_{xx}

Figure 16: résultats à l'itération 200.

La figure 17 montre, sur un quart du différentiel et pour la deuxième décomposition, de quelle façon les sous-structures ont été groupées pour un fonctionnement avec 8 et 12 processeurs.



8 processeurs 12 processeurs

Figure 17 : regroupement des sous-structures pour 8 et 12 processeurs (1/4 du modèle).

Plusieurs tests ont été effectués sur un CRAY T3D de l'IDRIS (Orsay) et sur un IBM SP2 du CEA (Saclay). Ce dernier dispose de 16 processeurs avec chacun 128 Mo de RAM. Le temps de calcul CPU cumulé en fonction du nombre de processeurs pour le modèle à 35 136 degrés de liberté (figure 18) montre que le coût des communications est relativement faible (pour un processeur, il n'y a pas de communication). La figure 18 présente aussi le temps CPU cumulé en fonction de la taille de la modélisation pour l'utilisation de 12 processeurs. Il est important de noter que le coût total augmente lentement avec la taille du problème.

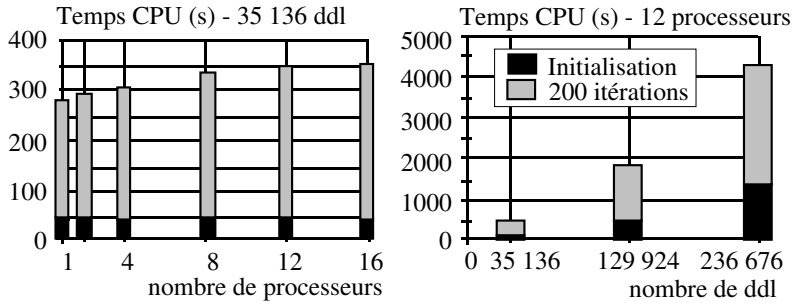


Figure 18 : temps CPU cumulé sur l'IBM SP2.

5. Conclusions et Perspectives

Nous avons présenté les problèmes numériques intervenant dans les calculs éléments finis de structures complexes. Ces problèmes s'expriment en termes de difficulté de gestion des données, de taille et de coût de calcul.

Plusieurs techniques de parallélisation existent, soit pour adapter un code de calcul existant à une architecture parallèle, soit pour développer de nouveaux algorithmes de résolution. La grande généralité et la minimisation des coûts de développement du code des approches de parallélisation automatique est contrebalancée par les performances moyennes obtenues. À contrario, la spécificité des approches de sous-structuration pour le type de problème traité permet d'atteindre des performances élevées.

Les techniques utilisables en pratique feront sans doute intervenir ces différentes compétences ; par exemple : un compilateur adapté pour utiliser de façon cachée un éventuel parallélisme interne à l'unité de traitement de chaque nœud de calcul, une parallélisation automatique des phases de calcul locales (intégration de la relation de comportement...), une parallélisation explicite par l'utilisation de décomposition de domaine pour la résolution des grands systèmes linéaires, ou même un couplage de

codes de calcul spécialisés pour traiter des problèmes à phénomènes couplés. Avec toutes ces techniques, l'utilisateur des codes de calcul utilisera donc le parallélisme de façon plus ou moins transparente.

Références

[Agoshkov 88] **V. I. Agoshkov** — Poincaré-Steklov's operators and domain decomposition methods in finite dimensional spaces, dans *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, R. Glowinski, G. H. Golub, G. A. Meurant, J. Périaux éd., SIAM, Philadelphia, 73-112 (1986).

[André 95] **F. André, M. Le Fur, F. Mahéo, J.-L. Pazat** — The Pandore Data Parallel Compiler and its Portable Runtime, *International Conference and Exhibition on High-Performance Computing and Networking*, Springer Verlag, 176-183, Milan, Italie (1995).

[Barragy 88] **E. Barragy, G. F. Carey** — A parallel element-by-element solution scheme, *International Journal of Numerical Methods in Engineering*, vol. 26, 2367-2382 (1988).

[Bjørstad 86] **P. E. Bjørstad, O. B. Widlund** — Iterative methods for the solution of elliptic problems on regions partitioned into substructures, *SIAM Journal on Numerical Analysis*, vol. 23 (6), 1097-1120 (1986).

[Blanzé 95] **C. Blanzé, L. Champaney, J.-Y. Cognard, P. Ladevèze** — A modular approach to 3D structure assembly computations: application to contact problem, *Engineering Computations*, 10, 79-93 (1995).

[Boucard 99] **P.-A. Boucard, H. Lemoussu, P. Ladevèze** — A modular approach to 3D impact computations with frictional contact, *Computers & Structures*, à paraître.

[Breitkopf 92] **P. Breitkopf, G. Touzot** — Architecture des logiciels et langages de modélisation, *Revue Européenne des Eléments Finis*, vol. 1 (3), 333-368 (1992).

[Buoni 93] **J. J. Buoni, P. A. Farrell, A. Ruttan** — Algorithms for LU decomposition on a shared memory multiprocessor, *Parallel Computing*, vol. 19, 925-937 (1993).

[Champaney 96] **L. Champaney** — *Une nouvelle approche modulaire pour l'analyse d'assemblages de structures tridimensionnelles*, Thèse de doctorat, ENS de Cachan (1996).

[Champaney 97] **L. Champaney, J.-Y. Cognard, D. Dureisseix, P. Ladevèze** — Large scale applications on parallel computers of a mixed domain decomposition method, *Computational Mechanics*, vol. 19 (4), 253-262 (1997).

[Champaney 99] **L. Champaney, J.-Y. Cognard, P. Ladevèze** — Modular analysis of assemblages of three-dimensional structures with unilateral contact conditions, *Computers & Structures*, vol. 73, 249-266 (1999).

[Chan 90] **T. F. Chan, R. Glowinski, J. Périaux, O. B. Widlund** éd. — *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia (1990).

[Cognard 96] **J.-Y. Cognard, V. Feuardent, J.-M. Virely** — Optimisation d'une structure pour essais mécaniques biaxiaux, *Proc. of the First International Conference on Integrated Design and Manufacturing in Mechanical Engineering*, 295-304 (1996).

[Débordes 89] **O. Débordes, J. C. Michel** — Parallélisation des problèmes non-linéaires, dans *Calcul des Structures et Intelligence Artificielle*, vol. 4, Pluralis, 223-232 (1989).

[De Roeck 92] **Y.-H. De Roeck, P. Le Tallec, M. Vidrascu** — A domain decomposed solver for nonlinear elasticity, *Computer Methods in Applied Mechanics and Engineering*, n° 99, 187-207 (1992).

[Dongarra 99] **J. J. Dongarra, H. W. Meuer, E. Strohmeier** — TOP 500 supercomputer sites, *Supercomputer '99 Conference*, Mannheim (1999). Disponible sur http://www.top500.org/ et http://www.netlib.org/benchmark/top500.html

[Duff 86] **I. S. Duff** — Parallel implementation of multifrontal schemes, *Parallel Computing*, vol. 3, 192-204 (1986).

[Dureisseix 97] **D. Dureisseix** — *Une approche multi-échelles pour des calculs de structures sur ordinateurs à architecture parallèle*, Thèse de doctorat, ENS de Cachan (1997).

[Dureisseix 98] **D. Dureisseix, P. Ladevèze** — A 2-level and mixed domain decomposition approach for structural analysis, Domain Decomposition Methods 10, *Contemporary Mathematics*, vol. 218, AMS, 238-245 (1998).

[Escaig 92] **Y. Escaig —** *Décomposition de domaine multiniveaux et traitements distribués pour la résolution de problèmes de grande taille*, Thèse de doctorat, Université de Technologie de Compiègne (1992).

[Escaig 94] **Y. Escaig, M. Vayssade, G. Touzot —** Une méthode de décomposition de domaine multifrontale multiniveaux, *Revue Européenne des Éléments Finis*, vol. 3, 311-337 (1994).

[Fahmy 95] **M. W. Fahmy, A. H. Namini —** A survey of parallel nonlinear dynamic analysis methodologies, *Computers & Structures*, vol. 53 (4), 1033-1043 (1995).

[Farhat 88] **Ch. Farhat, E. Wilson —** A parallel active column equation solver, *Computers & Structures*, vol. 28, 289-304 (1988).

[Farhat 89] **Ch. Farhat, L. Crivelli —** A general approach to non linear finite element computations on shared-memory multiprocessors, *Computer Methods in Applied Mechanics and Engineering*, vol. 72, 153-171 (1989).

[Farhat 91] **Ch. Farhat, F.-X. Roux —** A method of finite element tearing and interconnecting and its parallel solution algorithm, *International Journal of Numerical Methods in Engineering*, vol. 32, 1205-1227 (1991).

[Farhat 93] **Ch. Farhat, M. Lesoinne —** Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics, *International Journal of Numerical Methods in Engineering*, vol. 36, 745-764 (1993).

[Farhat 94] **Ch. Farhat, F.-X. Roux —** Implicit parallel processing in structural mechanics, dans *Computational Mechanics Advances*, J. Tinsley Oden éd., vol. 2 (1) (1994).

[Farhat 95] **Ch. Farhat, S. Lanteri, H. D. Simon —** TOP/DOMDEC: A Software Tool for Mesh Partitioning and Parallel Processing, *Journal of Computing Systems in Engineering*, vol. 6 (1), 13-26 (1995).

[Fortin 82] **M. Fortin, R. Glowinski —** Méthodes de Lagrangien augmenté, dans *Méthodes Mathématiques de l'informatique*, P.-L. Lions éd., vol. 9 (1982).

[Geist 94] **A. Geist, A. Berguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam —** *PVM: A user's guide and tutorial for networked parallel computing*, Oak Ridge National Laboratory, MIT Press, Tennessee (1994). Pour accéder à des informations sur PVM, consulter http://www.epm.ornl.gov/pvm/pvm_home.html

[Glowinski 90] **R. Glowinski, P. Le Tallec —** Augmented Lagrangian Interpretation of the Nonoverlapping Schwarz Alternating Method, dans [Chan 90], 224-231 (1990).

[Gupta 95] **A. Gupta, G. Karypis, V. Kumar —** Highly Scalable Parallel Algorithms for Sparse Matrix Factorization, *IEEE Transactions on Parallel and Distributed Systems*, vol. 8 (5), (1995).

[Gropp 94] **W. D. Gropp, E. Lusk, A. Skjellum —** Using MPI: portable parallel programming with the message-passing interface, *Scientific and Engineering Computation*, MIT Press, Cambridge (1994). Pour accéder à des informations sur MPI, consulter http://www-unix.mcs.anl.gov/mpi/

[Gupta 96] **K. K. Gupta, J. L. Meek —** A brief history of the beginning of the finite element method, *International Journal of Numerical Methods in Engineering*, vol. 49, 3761-3774 (1996).

[Heath 91] **M. T. Heath, E. Ng, B. W. Peyton —** Parallel algorithms for sparse linear systems, *SIAM Review*, vol. 33 (3), 420-460 (1991).

[Hendrickson 95] **Bruce Hendrickson, Robert Leland —** An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations, *SIAM J. Sci. Stat. Comput.*, vol. 16 (2), 452-469 (1995). Pour plus d'informations, consulter http://www.cs.sandia.gov/CRF/chac.html

[Hugues 83] **T. J. R. Hugues, I. Levit, J. Winget —** An element-by element solution algorithm for problems of structural and solid mechanics, *Computer Methods in Applied Mechanics and Engineering*, vol. 36, 241-254 (1983).

[Irons 70] **B. M. Irons —** A frontal solution program for finite element analysis, *International Journal of Numerical Methods in Engineering*, vol. 2, 5-32 (1970).

[Johnsson 89] **S. L. Johnsson, K. K. Mathur —** Experience with the conjugate gradient method for stress analysis on a data parallel supercomputer, *International Journal of Numerical Methods in Engineering*, vol. 27, 523-546 (1989).

[Karypis 99] **George Karypis, Vipin Kumar —** A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs, *SIAM Journal on Scientific Computing* (to appear). Pour plus d'informations, consulter http://www-users.cs.umn.edu/~karypis/metis/

[Kumar 94] **V. Kumar, A. Grama, A. Gupta, G. Karypis —** *Introduction to parallel computing: design and analysis of algorithms*, Addison-Wesley (1994).

[Ladevèze 85] **J. Ladevèze —** Algorithmes adaptés aux calculs vectoriel et parallèle pour des méthodes de décomposition de domaines, dans les *Actes du 3e colloque Tendances Actuelles en Calcul de Structures*, J. P. Grellier, G. M. Campel éd., Pluralis, 893-907 (1985).

[Ladevèze 96] **P. Ladevèze —** *Mécanique non-linéaire des structures - Nouvelle approche et méthodes de calcul non incrémentales*, Hermès, Paris (1996).

[Ladevèze 99] **P. Ladevèze, D. Dureisseix —** Une nouvelle stratégie de calcul micro/macro en mécanique des structures, à paraître dans les *Comptes Rendus à l'Académie des Sciences*.

[Lamour 91] **F. Lamour —** Sur la restructuration des boucles Fortran en vue de leur parallélisation, *La recherche Aérospatiale*, n° 5, 1-8 (1991).

[Le Tallec 94] **P. Le Tallec —** Domain decomposition methods in computational mechanics, dans *Computational Mechanics Advances*, vol. 1 (2), North-Holland (1994).

[Lions 90] **P.-L. Lions —** On the Schwarz alternating method III: a variant for nonoverlapping subdomains, dans [Chan 90], 202-223 (1990).

[Mackerle 96] **J. Mackerle —** Implementing finite element methods or supercomputers, workstations and PCs, *Engineering Computations*, vol. 13 (1), 33-85 (1996).

[Mandel 93] **J. Mandel —** Balancing domain decomposition, *Communications in Applied Numerical Methods*, vol. 9, 233-241 (1993).

[Mansfield 90] **L. Mansfield —** On the conjugate gradient solution of the {S}chur complement system obtained from domain decomposition, *SIAM Journal on Numerical Analysis*, vol. 27, 1612-1620 (1990).

[Noor 97] **A. K. Noor —** New computing systems and future high performance computing environment and their impact on structural analysis and design, *Computers & Structures*, vol. 64 (1-4), 1-30 (1997).

[Oakley 95] **D. R. Oakley, N. F. Knight jr. —** Adaptive dynamic relaxation algorithm for non-linear hyperelastic structures. Part III. Parallel implementation, *Computer Methods in Applied Mechanics and Engineering*, vol. 126, 111-129 (1995).

[Pan 93] **V. Pan, J. Reif —** Fast and efficient parallel solution of sparse linear systems, *SIAM J. Sci. Comput.*, vol. 22 (6), 1227-1250 (1993).

[Pellegrini 96] **F. Pellegrini, J. Roman —** SCOTCH: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs, *Proc. of HPCN'96*, Brussels, Belgium., 493-498, Springer (1996).

[Petiton 93] **S. Petiton, Y. Saad, K. Wu, W. Ferng —** Basic sparse matrix computations on the CM-5, *International Journal of Modern Physics C*, 4-1 (1993).

[Przemieniecki 63] **J. S. Przemieniecki —** Matrix structural analysis of substructures, *Am. Inst. Aero. Astro. J.*, vol. 1, 138-147 (1963).

[Roditis 90] **Y. S. Roditis, P. D. Kiousis —** Parallel multisplitting, block Jacobi type solutions of linear systems of equations, *International Journal for Numerical Methods in Engineering*, vol. 29, 619-632 (1990).

[Roux 91] **F.-X. Roux —** *Programmation des super-calculateurs scientifiques vectoriels et parallèles*, Note Technique 1991-11, ONERA (1991).

[Saad 85] **Y. Saad, M. H. Schultz —** *Parallel implementations of preconditioned conjugate gradient methods*, Research report 425, Dept Computer Science, Yale University (1985).

[Saad 95] **Y. Saad, A. V. Malevsky —** P-SPARSLIB: a portable library of distributed memory sparse iterative solvers, *Proc. of Parallel Computing Technologies*, V. E. Malyshkin éd., St Petersbourg, 1995. Pour plus d'informations, consulter
http://www.cs.umn.edu/Research/arpa/p_sparslib/psp-abs.html

[Utku 86] **S. Utku, M. Salama, R. J. Melosh —** Concurrent Cholesky factorization of positive definite banded Hermitian matrices, *International Journal for Numerical Methods in Engineering*, vol. 23, 2137-2152 (1986).

[Verpeaux 88] **P. Verpeaux, T. Charras, A. Millard —** CASTEM 2000 une approche moderne du calcul des structures, dans *Calcul des structures et intelligence artificielle*, J. M. Fouet, P. Ladevèze et R. Ohayon éd., Pluralis, 261-271 (1988). Pour plus d'informations, consulter http://www.castem.org:8001/

[Walshaw 97] **C. Walshaw, M. Cross, M. Everett —** Parallel Dynamic Graph Partitioning for Adaptive Unstructured Meshes, *J. Par. Dist. Comput.*, 47 (2), 102-108 (1997). Pour plus d'informations, consulter http://www.gre.ac.uk/jostle/

[Whirley 89] **R. G. Whirley, J. O. Hallquist, G. L. Goudreau —** An assessment of numerical algorithms for plane stress and shell elastoplasticity on supercomputers, *Engineering Computations*, vol. 6 (2), 116-126 (1989).

[White 88] **D. W. White, J. F. Abel —** Bibliography on finite elements and supercomputing, *Communications in Applied Numerical Methods*, vol. 4, 279-294 (1988).

[Yagawa 93] **G. Yagawa, A. Yoshioka, S. Yoshimura, N. Soneda —** A parallel finite element method with a supercomputer network, *Computers & Structures*, vol. 47, 407-418 (1993).

[Zienkiewicz 91] **O. C. Zienkiewicz, R. L. Taylor —** *The finite element method*, Mc Graw-Hill (1991).