Chapitre 1

Introduction

1-1 Problématique :

Avec les développements récents dans le domaine de la modélisation solide en trois dimensions et ses applications étendues en CAO/FAO ainsi que dans le domaine des éléments finis, il devient nécessaire de mettre au point un environnement automatique et convivial de CAO/FAO (conception et fabrication assistée par ordinateur) qui intègre les différentes étapes de la conception mécanique des pièces et de leur analyse structurale par Éléments Finis. Ce besoin bien qu'évident n'est cependant pas facile à satisfaire car il demande l'intégration de techniques et le développement de moyens de modélisation d'analyse et sophistiqués. Dans la conception et l'analyse de modèles en trois dimensions (3D), la génération d'un maillage pour Éléments Finis (E.F.) fiable et consistant sur des solides de formes complexes représente encore un goulot d'étranglement dans l'établissement du lien nécessaire entre les calculs de validation et le modèle géométrique.

Pour être efficace le mailleur recherché doit :

I. Produire un maillage géométriquement et topologiquement consistant avec le modèle géométrique CAO (fidélité) ;
II. Être entièrement automatique et facilement contrôlable (minimiser l'intervention de l'usager) ;
III. Produire un maillage avec des éléments finis bien conditionnés et fiables.

Il existe actuellement trois grandes classes de mailleurs automatiques :

I. Ceux utilisant la technique de triangulation de Delaunay (en 2D et 3D) utilisés dans des maillages non structurés ;
II. Ceux utilisant la transformation isoparamétrique, populaires chez les ingénieurs;
III. Et ceux basés sur la décomposition récursive spatiale des domaines, par la méthode quadtree en 2D ou octree modifiée en 3D.

Ces derniers utilisent des bases de données basées sur une arborescence quaternaire ou octale et permettent un raffinement sélectif facile, suite à une analyse d'erreur et un addressage spatial de chaque élément ainsi qu'une structuration de l'analyse de façon hiérarchique. Ils offrent un avenir intéressant dans le domaine de l'analyse automatique.

Le maillage adaptatif doit être généré à partir du modèle géométrique (CSG ou B-rep.) [26-29]. C'est lui qui sera considéré ici et qui est en relation avec la représentation par quadtree en 2D ou octree en 3D.

1-2 CAO/FAO :

Afin d'effectuer des études mécaniques, l'ingénieur a désormais accès à l'outil informatique par le biais de la Conception Assistée par Ordinateur (CAO). Celle-ci lui fournit les environnements nécessaires pour modéliser ses objets et des méthodes ont été développées afin de simuler numériquement le comportement de systèmes soumis

à différents types de contraintes et de chargements.

Lorsqu'on veut résoudre un problème de mécanique par l'intermédiaire de la CAO, la démarche générale est la suivante : on commence par modéliser l'objet à étudier par l'intermédiaire d'un logiciel, généralement interactif, de saisie de sa géométrie. Ceci se fait en précisant des points, en utilisant des primitives graphiques comme : cercles, sphères, cônes, etc., ou en définissant des courbes ou surfaces paramétriques ou d'interpolation (ces logiciels sont parfois appelés modeleurs solides). Une fois cette étape franchie, il s'agit d'imposer les contraintes voulues à l'objet, et d'étudier ses réactions. Pour cela, différentes méthodes existent. L'une des plus puissantes est la méthode des Éléments finis. Elle s'applique aussi bien dans les études de résistance des matériaux qu'en aérodynamique. Cette méthode nécessite le découpage du domaine en sous-domaines appelés éléments finis afin de constituer un maillage. Une telle approche oblige à reconsidérer la modélisation géométrique de l'objet sous l'angle des éléments finis. Cependant les tendances actuelles considèrent le modèle géométrique comme une base crédible d'analyse sans recourir explicitement au maillage.

Il existe très peu d'interfaces entièrement automatiques permettant de passer du modèle géométrique au modèle d'éléments finis. L'utilisateur est souvent obligé de repenser son modèle en fonction des éléments finis, en perdant éventuellement de vue le modèle géométrique.

Lors de la conception d'un produit, des calculs interviennent à plusieurs stades [10] (figure 1-1), dont les principaux sont :

- Une fois que le principe de fonctionnement est établi, pour le prédimensionnement des divers éléments constituant ce produit ; et
- Une fois que l'avant-projet est bien avancé, pour la vérification de ses éléments constitutifs.

Bien entendu, de nombreux autres calculs peuvent intervenir dans le projet, l'avant-projet ou même le schéma fonctionnel tels que des études de prix, fabrication, ...

Schématiquement le concepteur est donc confronté à deux types de calcul :

I. Les calculs de conception ayant pour but de déterminer certains paramètres numériques (dimensions, caractéristiques des matériaux, etc.) minimaux que devront posséder certains éléments essentiels du produit.

II. Les calculs de vérification ayant pour rôle de s'assurer que le fonctionnement des divers éléments du produit se fera conformément au cahier des charges.

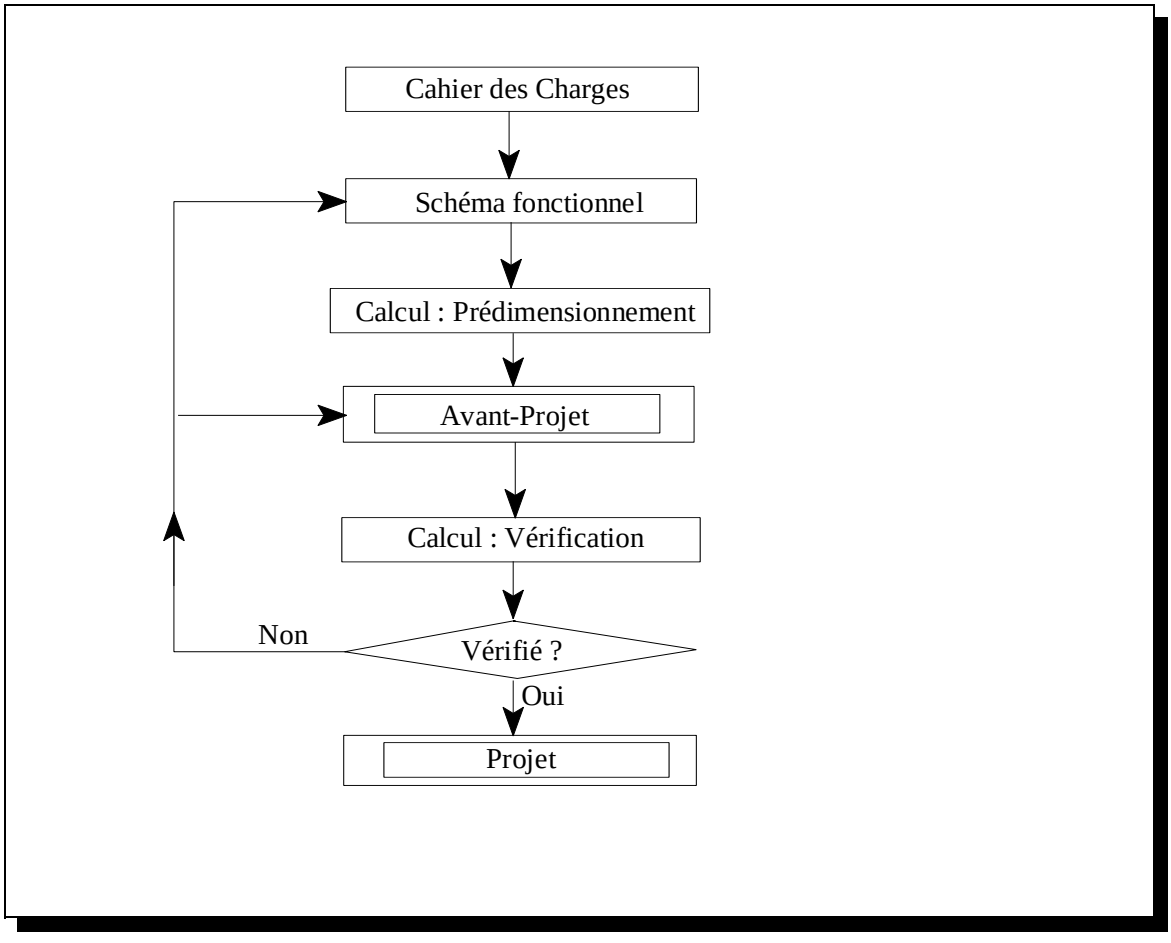

Figure 1-1 : Un exemple d'intervention des calculs dans la conception d'un produit.

L'informatique apporte, à l'heure actuelle, une aide considérable aux calculs de vérification. Les méthodes de calcul les plus puissantes et très utilisées en CAO sont la méthode des **éléments finis** et la méthode des **éléments finis de frontière**.

1-3 Présentation de la méthode des éléments finis :

Le but de la méthode des éléments finis est de résoudre de façon approchée les problèmes aux limites de la mécanique exprimés soit sous forme d'équations aux dérivées partielles soit sous forme d'équations intégrales, et sous diverses conditions de chargement.

Parallèlement aux progrès de l'informatique, les ingénieurs mécaniciens ont développé, au cours des années 60, des méthodes d'approximation pour le calcul des contraintes et des déformations basées sur la discrétisation des structures continues. La méthode des éléments finis qui sera abordée ici, est l'une de ces méthodes les plus utilisées.

De récentes études mathématiques ont permis une approche plus générale et une extension de la méthode à un grand nombre de problèmes physiques de nature différente de la mécanique des structures [1, 65-68].

1-3-1 Description générale :

La méthode des éléments finis (E.F.) est une technique particulière d'approximation des fonctions laquelle utilise une approximation nodale par sous-domaines [1]. Les inconnus, notées {U}, sont des valeurs de ces fonctions en certains points appelés noeuds de chaque sous-domaine. La forme variationnelle définie sur le milieu continu est alors représentée par une forme discrétisée, qui fait intervenir les inconnues nodales {U}. Par exemple, pour un problème d'élasticité linéaire :

- Le problème avec conditions aux limites (EDP) est défini par :

$$L\,u = f \qquad \text{sur le domaine } (v)$$
$$C\,u = b \qquad \text{sur la frontière } (\partial v)$$

où "L" et "C" sont des opérateurs différentiels et "f" et "b" sont les fonctions connues.

- La forme variationnelle correspondante s'obtient via une méthode des résidus pondérés :

$$W_{cont.} = \int_{v} (Lu - f)u^{*}\,dv = 0 \qquad \forall \underset{\sim}{u}^{*} \qquad \text{(1.1)}$$

Milieu continu

où {u*} est le vecteur de fonctions tests (ou virtuelles).

qu'une intégration par partie transforme en forme faible qui est le point de départ de l'approximation par E.F..

Pour fin d'approximation, on doit discrétiser la géométrie et les variables.

- Discrétisation de la géométrie :

  La géométrie de chaque élément est défini par :

$$X = [N]\{x_n^e\}$$

où

X = (x, y, z) : coordonnées d'un point quelconque sur un élément.

$\{x_n^e\}$: vecteur des coordonnées nodales d'un élément.

[N] : matrice des fonctions d'interpolation.

- Approximation des variables

$$u = [N]\{u_n^e\} \qquad (1.2)$$

où

{u} : valeur de la variable au point X(x, y, z).

$\{u_n\}$: vecteur des variables aux noeuds de l'élément.

- Discrétisation par Éléments Finis :

On remplace la forme variationnelle globale (1.1) par une somme de formes variationnelles sur chaque élément sur lequel on aura introduit les approximations par éléments finis ci-dessus :

$$W_{cont.} = \sum_{e=1}^{nb-éléments} W^e = \sum_{e=1}^{nb-éléments} ([k^e]\{u_n^e\} - \{f^e\})\{u_n^*\} \qquad (1.3)$$

$[k^e]$: matrice élémentaire (dite de rigidité).

$\{f_n^e\}$: vecteur élémentaire des sollicitations.

- Forme discrétisée finale :

$$W_{dis.} = <U^*>\{R\} = 0 \qquad \forall\{U^*\} \tag{1.4}$$

$d'où$

$$\{R\} = [K]\{U\} - \{F\} = \{0\} \tag{1.5}$$ *Milieu discrétisé*

1-3-2 Démarche par éléments finis :

Les différentes étapes de la méthode des éléments finis sont les suivantes :

- Représentation géométrique du domaine V.

- Discrétisation du domaine de volume V par un ensemble de sous-domaines de volume V^e (maillage) :

$$V = \sum V^e \quad ; \quad W = \sum W^e \tag{1.6}$$

- Représentation de la géométrie de chaque élément V^e :

$$\{x(\xi)\} = [N(\xi)]\{x_n\} \tag{1.7}$$

{x} position d'un point quelconque sur l'élément V^e

$\{x_n\}$ coordonnées des noeuds définissant V^e

ξ coordonnées paramétriques ξ, η, ζ

[N] fonctions d'interpolation en variables paramétriques

- Représentation (isoparamétrique) de la fonction solution {u} sur chaque élément

$$\{u(\xi)\} = [N(\xi)]\,\{u_n\} \;;\; \{u^*(\xi)\} = [N(\xi)]\,\{u_n^*\} \qquad (1.8)$$

{u} fonction solution

$\{u_n\}$ variables nodales caractérisant la fonction solution

{u*} fonctions tests

$\{u^*_n\}$ variables nodales virtuelles

- Représentation de la forme variationnelle (discrétisation) :
  Calcul élémentaire : sur chaque élément la quantité W, notée W^e, s'exprime en fonction de $\{u_n\}$ et $\{u^*_n\}$ (en utilisant (1.5) et (1.6)) :

$$W^e = <u_n^*> ([k^e]\{u_n^e\} - \{f_n^e\}) \qquad (1.9)$$

$[k^e]$: matrice élémentaire (dite de rigidité)

$\{f_n^e\}$: vecteur élémentaire des sollicitations

- Assemblage : construction de [K] et [F]

$$W = \sum_e W^e = \sum_e <u_n^*> ([k^e]\{u_n\} - \{f_n^e\}) \qquad (1.10)$$

$$W = <U^*> ([K]\{U\} - \{F\}) = 0 \qquad \forall\{U^*\} \qquad (1.11)$$

soit le système final à résoudre :

$$[K]\{U\} = \{F\} \qquad (1.11a)$$

[K] matrice globale obtenue par assemblage des matrices élémentaires

{F} vecteur global des sollicitations obtenu par assemblage des vecteurs de sollicitations élémentaires.

Dans les cas de sollicitations dynamiques :

$$\{F\} = \{F(t)\} - [M]\{\ddot{U}\} - [C]\{\dot{U}\} \quad (1.12)$$

[U] ensemble des variables nodales
{U*} ensemble des variables nodales virtuelles
[M] matrice de masse
[C] matrice d'amortissement
{U̇} vecteur de vitesse
{Ü} vecteur d'accélération
{F} = [K]{U}
{F(t)} vecteur sollicitation dynamique

- Résolution :

  En tenant compte des conditions aux limites, trouver {U} tel que :

$$\{R\} = [K]\{U\} - \{F\} = \{0\} \quad (1.13)$$

  Pour un problème linéaire :

$$\{U\} = [K]^{-1}\{F\} \quad (1.14)$$

- Évaluation des grandeurs dérivées relatives à chaque élément :
  I. extraire $\{u_n\}$ de {U}
  II. calculer les gradients de {u} en un point de l'élément (déformations)
  III. calculer les quantités représentatives du problème (contraintes, ...)

La démarche ci-dessus peut généraliser à tout problème avec conditions initiales et/ou aux limites dans presque tous les domaines en sciences et génie.

1-4 Problème de la génération de maillage automatique :

Nous venons de voir que la méthode des Éléments Finis est un outil puissant d'analyse applicable à de nombreux domaines d'études. Une des étapes essentielles à la réussite d'une analyse par éléments finis est l'étape de la génération de maillage. Lorsque cette étape est effectuée manuellement, elle est très gourmande en temps et des erreurs et imperfections préjudiciables à la qualité des résultats peuvent facilement s'introduire, notamment à cause de la grande quantité de données à générer et à manipuler. Mark A. Yerry [2] évalue que cette génération de données (géométrie des éléments, connectivité des noeuds, etc.) justifie à elle seule 80% du coût total d'une analyse par éléments finis utilisant des méthodes interactives de génération de maillage. Afin de réduire l'impact de ces problèmes, et ainsi de banaliser l'utilisation des éléments finis, de nombreuses équipes tentent de développer des méthodes visant à automatiser, du moins en partie, le processus de génération de maillage et même de l'éliminer complètement [2-5, 11-40].

Avant de présenter un bref panorama de ces développements, nous allons essayer d'énoncer les caractéristiques principales d’un bon mailleur. (On désignera désormais par *mailleur* n'importe quelle méthode de maillage visant à automatiser ce processus, et ce, quel que soit le degré d'automatisation effectivement atteint.)

1-4-1 Propriétés essentielles d'un mailleur :

Un bon mailleur pour éléments finis doit tenir compte de ces considérations de base :

I. La qualité de la solution obtenue lors de l'analyse par éléments finis dépend de la finesse du maillage produit.

Cela signifie que plus le maillage est fin (i.e. la concentration en éléments de maillage est élevée), plus précise sera la solution obtenue, ceci étant particulièrement vrai dans les zones où les gradients de solution sont importants.

II. Les coûts d'analyse augmentent généralement beaucoup plus vite que la finesse de maillage.

En d'autres termes, les coûts d'analyse (cette notion de *coût* recouvre des paramètres tels que le temps de calcul sur ordinateur, la place mémoire nécessaire, les temps de pré- et post- traitement, etc.) peuvent rapidement devenir prohibitifs si l'on tente d'appliquer uniformément un maillage très fin [3].

Une conséquence immédiate de ces considérations est que pour obtenir des solutions de bonne qualité, il faut pouvoir préciser les niveaux de maillage de façon non uniforme. On doit pouvoir imposer un maillage fin dans les zones susceptibles d'être le siège de forts gradients de la solution, et un maillage plus grossier dans les zones où les solutions n'évoluent que peu. Les éléments de maillage générés, en principe, des entités géométriques simples (triangles et quadrilatères en 2D et tétraèdre, pyramide, prisme et hexaèdre en 3D), doivent donc avoir les tailles requises par les paramètres de maillage dans les zones où ils apparaissent.

III. Le maillage généré doit être le plus fidèle possible à la géométrie du problème. En particulier, les différents contours extérieurs et intérieurs qui apparaissent

une fois le maillage généré doivent être aussi proches que possible des frontières originales.

Certains mailleurs vont générer des maillages tels que les sommets de certains éléments vont se trouver le long d'un côté d'un élément voisin. Ces mailleurs vont devoir mettre en oeuvre des méthodes destinées à résoudre les problèmes de continuité que de telles configurations entraîneraient.

IV. Il convient également de contrôler la forme des éléments finis générés. En effet, les calculs effectués au moment de l'analyse par éléments finis sont faits sur des éléments de référence (triangle ou carré) qui sont ensuite déformés pour coller aux différents éléments de maillage analogues. Des éléments trop déformés par rapport à ces éléments de référence amèneraient des problèmes de types numériques (singularité, imprécision, ...).

On peut définir des paramètres permettant de contrôler l'aspect des éléments; de tels paramètres sont appelés *coefficients de forme*. Pour un élément triangulaire, on peut, par exemple, considérer comme coefficient de forme le rapport du rayon du cercle inscrit sur le rayon du cercle circonscrit passant par les sommets de l'élément. Il faut ensuite déterminer expérimentalement les seuils permettant de séparer les éléments "corrects" du point de vue des éléments finis et les autres.

V. Enfin, si le but premier du mailleur doit être de soulager l'analyste dans sa tâche de génération d'un maillage correct, il faudrait également avoir comme objectif à plus long terme de le faire travailler (i.e. essentiellement échanger des données) avec d'autres programmes (modélisation du ou des objets, saisie des paramètres physiques tels que les matériaux utilisés, définition des sollicitations appliquées, analyse et calculs, etc.) ; il convient donc de soigner l'aspect structure de données.

1-4-2 Éléments de comparaison entre méthodes de maillage :

Tentons de préciser des critères permettant d'établir une classification des méthodes visant à automatiser le processus de génération de maillage.

Un mailleur étant sensé fournir en sortie un ensemble de noeuds et un ensemble d'éléments, le tout constituant la topologie du maillage, il est proposé [3] de retenir comme principal critère l'ordre d'application dans le temps de ces deux ensembles.

Parmi les autres éléments de comparaison, on peut distinguer le schéma selon lequel les éléments sont générés. Par exemple la *triangularisation de Delaunay* recueille pour certains types de méthodes (en particulier celles qui implantent les noeuds avant les éléments) les suffrages de nombreuses équipes (dont [5]) car elle semble produire les meilleurs éléments dans la perspective d'une analyse par éléments finis. Cette méthode tente de maximiser la somme des plus petits angles des triangles générés. Un tel schéma tend à limiter l'apparition d'éléments "plats", ce qui correspond bien à un des objectifs que nous avons défini dans le paragraphe précédent (i.e. éviter une trop grande déformation par rapport aux éléments de référence utilisés dans les calculs).

On peut distinguer les méthodes imposant un *maillage uniforme* de celles qui ne l'imposent pas. Dans les premières, un élément quelconque donné ne partagera avec ses voisins que des côtés entiers (Voir figure 1-2 a). Au contraire, si la méthode n'impose pas le maillage uniforme, on pourra observer

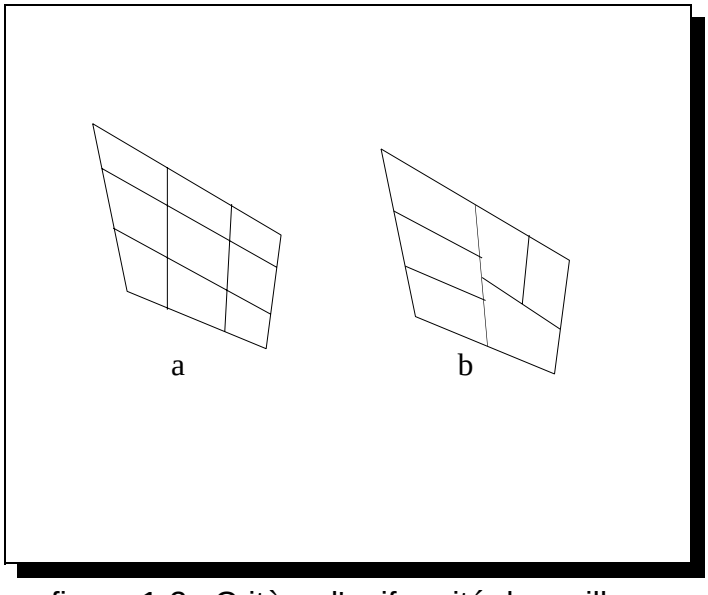

figure 1-2 : Critère d'uniformité de maillage

plusieurs éléments se partageant le même côté d'un élément donné (figure 1-2 b).

Évidemment, si ce critère n'est pas imposé, les transitions entre zones grossièrement et finement maillées seront simplifiées. En contrepartie, cette liberté impose la mise en place de traitements supplémentaires destinés à rendre l'analyse possible. Ceci est dû aux problèmes de discontinuité que de telles configurations apportent.

Enfin, on peut comparer les méthodes suivant le degré d'automatisation atteint.

1-5 Programmation orientée objet :

1-5-1 Les abstractions modulaires :

Dans la littérature, on trouve plusieurs définitions d'un module [6-7] :

I. Une séquence contiguë d'énoncés qui peuvent être référés par un nom.
II. Les familles (package) de sous-programmes et tâches (comme ADA).
III. Une sous routine qui fait partie d'une activité.
IV. Une séquence contiguë d'énoncés d'un programme, limitée par des délimiteurs, ayant un identificateur d'agrégat et pouvant être utilisé explicitement où il est visible, ailleurs dans le programme.
V. Un ensemble d'énoncés exécutables d'un programme qui rencontre les trois critères suivants :
    i. C'est une sous routine fermée.
    ii. Il peut être appelé par n'importe quel autre module d'un programme.
    iii. Il peut être compilé séparément.

Certaines caractéristiques sont souhaitables pour les modules :

- La possibilité de cacher à l'utilisateur des informations inutiles qui pourraient amener de la confusion dans l'utilisation correcte des modules. De plus, si l'on

doit modifier la structure du module on a qu'à le faire en respectant les spécifications. Il ne sera même pas nécessaire d'informer les utilisateurs. Les changements sont transparents pour eux.

- La possibilité de compiler séparément les modules afin de les tester exhaustivement et de constituer une bibliothèque de bons modules.
- La possibilité d'exécuter les modules en même temps. Ce n'est pas évident. Il faut respecter la disponibilité des intrants. De plus, ça se complique beaucoup si certaines données sont partagées par plusieurs modules.

Si on prend l'exemple du langage ADA, on retrouve deux notions de modules : le package et la tâche. Le package permet d'encapsuler des types, des déclarations et des procédures sous un nom avec des possibilités de cacher certaines parties. Chaque package peut être sur un fichier différent et être compilé séparément.

```
        (fichier A)
package module is                    -- Partie spécification
.........................            -- Partie publique
private                              -- Partie privée
end module;

        (fichier B)
package body module is               -- Partie implantation
.....................................
end module;
```

Cette partie spécification peut être incorporée à d'autres packages sans le corps du package. On ne peut pas appeler un package par son nom ; ce qui lui enlève le titre de *module*. Les tâches TASK peuvent être appelées par leur nom et sont utilisées dans un contexte de multitraitement en parallèle ou non.

Les objets ("class" de C++) permettent l'abstraction des données, l'encapsulation des données et des procédures s'y rattachant, le *polymorphisme* c'est-à-dire la possibilité de répondre différemment à un message en fonction de l'objet et l'héritage des attributs. La déclaration *class* permet de créer une structure contenant des déclarations, des données et des procédures. Pour contrôler l'accès aux informations, on dispose des mots clés *private*, *public* et *protected* [8]. Des variables déclarées *private* sont visibles à l'intérieur de la classe seulement mais pas dans les sous-classes. Par contre, des variables déclarées *protected* sont visibles dans les sous-classes. Les variables *public* sont visibles à l'extérieur de la classe.

Une procédure est un module élémentaire. La procédure représente un modèle de calcul qui peut s'appliquer sur plusieurs données différentes. Elle est aussi locale à un programme avec son propre environnement. La communication avec d'autres procédures ou programmes se fait par l'intermédiaire de passage de paramètres.

1-5-2 Langages structurés en bloc :

Le paradigme de structure de blocs est caractérisé par la notion de blocs imbriqués, de procédures et de récursivité [6-7].

Un bloc est une section de code limitée par deux délimiteurs ayant des propriétés de localité. De l'information qui doit être utilisée exclusivement dans un bloc et non nécessaire à l'extérieur peut, par un bloc, être isolée du reste du programme. Une procédure devient un bloc nommé qui peut être appelée par une autre procédure en lui transmettant des paramètres ou arguments.

Voici les avantages d'avoir des blocs :

I. Des changements ultérieurs peuvent être apportés sans interférer avec les autres blocs.

II. La preuve de programme est facilitée en permettant de mettre des pré- et post-

conditions aux blocs.

III. Plusieurs programmeurs peuvent développer des blocs différents sans avoir de conflit de variables locales.

IV. La présence de blocs favorise la modularité et permet ainsi d'isoler un sous-programme.

Comme exemples de langages structurés en bloc, on peut citer C, C++, ADA, Pascal, etc. .

1-5-3 Les langages objets :

Un objet est un regroupement de données et de procédures (encapsulation) s'y rapportant et qui communique ou non par message (polymorphisme).

La programmation "orientée objet" aborde un problème en le ramenant à des objets qui interagissent entre eux.

Une approche "descendante" décompose le problème en un ensemble d'étapes de plus en plus raffinées du problème. Les étapes les plus importantes constituent des modules.

Il y a plusieurs niveaux dans la classification des langages objets.

A. Les langages basés sur les objets supportant l'encapsulation, l'abstraction des données et la dissimulation des informations, par exemple les packages d'ADA.

B. Les langages basés sur les classes qui utilisent les objets mais avec la possibilité de les faire apparaître et disparaître dynamiquement, par exemple, les clusters de CLU et les packages génériques d'ADA.

C. Les langages orientés objets possèdent des classes allouées dynamiquement et l'héritage d'attributs d'une classe supérieure, par exemple Pascal objet et C++.

On distingue les procédures des langages structurés en bloc des procédures contenues dans les objets. Ces dernières ne sont pas accessibles directement mais à travers l'interface de l'objet. Pour les distinguer, on parle dans ce cas de *méthodes* au lieu de procédures.

L'héritage est la possibilité pour un objet d'une classe inférieure d'hériter des attributs et des méthodes (procédures) d'une classe ou de classes supérieures.

1-5-4 Choix d'un environnement de programmation :

Nous souhaitions disposer d'un langage suffisamment évolué pour pouvoir manipuler des entités à plusieurs champs dont certains variables. En outre, il fallait pouvoir générer et manipuler dynamiquement nos structures de données car la taille du problème n'est pas connue à priori. Le langage choisi devait nous permettre d'effectuer les opérations mathématiques de base, de disposer d'un minimum de capacité d'entrée/sortie (ou un minimum de lectures et d'écritures de fichiers) et être potentiellement portable et rapide (il sera donc inévitablement compilable). Le travail doit aussi être compatible avec d'autres logiciels disponibles de calcul par Élément Finis.

Ces considérations nous ont amené à arrêter notre choix sur le langage C.

Les entités et leurs divers champs vont pouvoir être déclarés sous le type composé de C appelé *structure*. Les champs variables seront eux des *unions* [9]. Tous les membres d'une union partagent le même espace de stockage. Deux membres ne peuvent donc exister simultanément. Le compilateur assure que l'espace alloué pour l'union sera suffisant pour ranger le plus gros membre. C'est au programmeur de s'assurer que la valeur qu'il cherche à récupérer est cohérente avec la dernière qui a été rangée. Ainsi dans l'exemple d'une union composée d'un membre réel et d'un membre entier, si la dernière valeur rangée était de type entier, il ne faut pas chercher

à récupérer un réel. Le compilateur ne détecte pas ce genre de configuration. Il est de toute façon dans la philosophie de C de laisser une grande liberté à l'utilisateur dans la manipulation des structures. Mais s'il n'y a pas non plus, en général, de "run-time error" émise lors d'une telle tentative, il est évident que le programme va finir par ne plus se comporter comme prévu.

Une des forces principales de C est son aptitude à manipuler des pointeurs. La richesse de C dans ce domaine, ainsi que ses capacités d'allocation (et de libération) dynamique d'espace mémoire vont nous permettre de gérer très naturellement les listes simplements chaînées, doublement chaînées, les tableaux et les arbres qui vont successivement apparaître, évoluer puis éventuellement disparaître en cours de traitement.

Comme on vient de le voir plus tôt, le langage C permet des licences qui feraient frémir le plus primaire des compilateurs. Cela autorise certaines manipulations très puissantes, mais exige d'être prudent. En outre, dans le but de générer le code le plus efficace possible, le langage C possède certaines tournures syntaxiques qui, en plus d'être d'une utilisation délicate, rendent le code source moins lisible et donc plus difficile à gérer. Les passages où on utilise ces formes devront être particulièrement bien documentées. Le langage C est performant et efficace lorsque nous avons à travailler près de l'ordinateur, comme par exemple, lors du développement de systèmes d'exploitation, de compilateurs ou d'une animation sur ordinateur. Par contre, l'écriture compacte des énoncés le rend peu lisible et la mise au point est difficile à cause de la grande liberté des types.

Pour bien utiliser les avantages du langage C et éviter ses inconvénients et aussi utiliser les avantages de la programmation modulaire basée sur les objets, on a essayé dans ce travail de simuler la programmation modulaire (objet) au sens d'ADA en environnement C.

1-6 Objectifs et contenu de la thèse :

L'objectif de ce travail est la mise au point de certains outils permettant :

i. d'affranchir l'utilisateur de certaines considérations propres aux éléments finis lors de la conception de son modèle géométrique,
ii. de réaliser un mailleur entièrement automatique en deux et trois dimensions de solides de forme quelconque et
iii. de l'appliquer dans le cadre d'une analyse automatique adaptative par Éléments finis.

Pour atteindre cet objectif global, on doit réaliser les sous objectifs suivants :

I. Étude des différentes méthodes de génération de maillage pour trouver les avantages et les inconvénients de chaque méthode et choisir la méthode plus apte à répondre à notre objectif global.
II. Choisir et intégrer les modèles géométriques pour la représentation des objets.
III. Construire les bases de données appropriées pour la représentation des objets et du maillage généré.
IV. Effectuer un maillage hiérarchique par Éléments finis correspondant, de façon automatique et créer des fichiers des noeuds, des éléments et des leurs connectivités, et leur assigner des attributs mécaniques et relationnels, et assurer la cohérence et la fiabilité du maillage.
V. Explorer la possibilité d'implanter l'algorithme de maillage sur des machines parallèles pour bénéficier des avantages du temps d'exécution.
VI. Démontrer que l'approche de maillage par quadtree/octree convient au problème de maillage adaptatif et d'estimation d'erreurs en E.F.

Pour réaliser ces objectifs, nous procéderons comme suit :

Le chapitre 2 explore différentes méthodes de génération de maillage E.F.. Après une revue bibliographique de la plupart des travaux dans le domaine de la génération de maillage, on met l'accent sur la méthode de quadtree/octree modifiée.

Dans le chapitre 3, on présente diverses techniques de modélisation géométrique en 2D et 3D. On considère la modélisation par les courbes et les surfaces de Bézier et B-spline. Afin d'arriver à une représentation unifiée, on utilise le modèle des courbes B-splines composites fermés en 2D et Surfaces B-spline en 3D. À la fin de ce chapitre, on étudie quelques problèmes de calcul géométrique comme la projection et l'intersection dans le cadre de la génération de maillage par la méthode quadtree/octree modifiée.

Le chapitre 4 est consacré à la méthode de quadtree modifiée pour la génération de maillage en 2D. Ensuite au chapitre 5, on étudie la méthode d'octree modifiée pour la génération de maillage en 3D. Dans les chapitres 4 et 5, on présente les différentes étapes de génération de maillage en détail et les algorithmes associés à chaque étape. On considère aussi la transformation des données générées par le programme de génération de maillage vers un programme E.F..

Dans le chapitre 6, on considère la possibilité de génération de maillage en utilisant les machines parallèles. Dans ce chapitre, différents algorithmes parallèles pour chaque étape de la génération de maillage sont présentés.

Dans le chapitre 7, nous procédons à l'application de techniques développées précédemment à des problèmes physiques réels. Nous faisons l'analyse des contraintes par E.F. dans les exemples en 2D et 3D.

Enfin le chapitre 8 est consacré à une rétrospective du travail et à la présentation des conclusions et recommandations pour des travaux futurs.

La difficulté de la méthode de quadtree/octree modifiée est le traitement des quadrants/octants coupés. Dans cette thèse, la méthode de quadtree/octree modifiée

pour la génération de maillage pour les éléments finis en 2D et 3D a été améliorée en prêtant une attention particulière aux quadrants et aux octants frontières.

Les aspects originaux dans ce travail sont donc :

- Afin d'arriver à une représentation géométrique unifiée, nous utilisons une approche basée sur des modèles B-spline rationnels non uniformes (NURBS) (le modèle des courbes B-splines composites fermées en 2D et de surfaces B-spline en 3D).
- En 2D, les quadrants frontières ont été générés entre les côtés libres des quadrants intérieurs et les courbes frontières, en éliminant les quadrants partiels et extérieurs. De cette manière, on élimine tous les cas particuliers de quadrants coupés déjà mentionnés.
- En 3D, on a étendu l'approche ci-dessus et on génère les octants frontières entre les faces libres des octants intérieurs et les surfaces frontières, en éliminant les octants partiels et extérieurs. De cette manière, on élimine tous les cas particuliers d'octants coupés (4096 cas différents en 3D).
- On a utilisé les algorithmes simples, basés sur le principe de projection d'un point sur une courbe ou sur une surface B-spline paramétrique.
- On a éliminé tous les traitements particuliers de génération de maillages à partir des quadrants et des octants coupés (comme calcul d'intersections), ce qui a allégé beaucoup la programmation et partant le temps de traitement.
- On a présenté les algorithmes parallèles pour les différentes étapes de génération de maillages.