POLYPUBLIE

Conception d'un logiciel de description d'éléments pour la discrétisation d'équations aux dérivées partielles

François Guibault

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

Date	1990
Type	Mémoire ou thèse / Dissertation or Thesis Mémoire de maîtrise/Master's thesis
URL de PolyPublie PolyPublie URL	https://publications.polymtl.ca/59272/
Conditions d'utilisation Terms of use	© Tous droits réservés / All rights reserved
Référence Citation	Guibault, F. (1990). Conception d'un logiciel de description d'éléments pour la discrétisation d'équations aux dérivées partielles [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/59272/

Détails de la thèse
Thesis details

Directeurs de recherche Advisors	
Programme Program	Non spécifié

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal
https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

CONCEPTION D'UN LOGICIEL DE DESCRIPTION D'ÉLÉMENTS POUR LA DISCRÉTISATION D'ÉQUATIONS AUX DÉRIVÉES PARTIELLES.

par

François GUIBAULT
DÉPARTEMENT DE MATHÉMATIQUES APPLIQUÉES
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU GRADE DE MAÎTRE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
Juillet 1990

©François Guibault 1990

al Library
ıada

Bibliothèque nationale
du Canada

ian Theses Service

Service des thèses canadiennes

Canada
4

author has granted an irrevocable non-
usive licence allowing the National Library
anada to reproduce, loan, distribute or sell
ies of his/her thesis by any means and in
form or format, making this thesis available
ıterested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

author retains ownership of the copyright
ıis/her thesis. Neither the thesis nor
stantial extracts from it may be printed or
erwise reproduced without his/her per-
sion.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-58933-7

ıada

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE

Ce mémoire intitulé:

CONCEPTION D'UN LOGICIEL DE DESCRIPTION D'ÉLÉMENTS POUR LA DISCRÉTISATION D'ÉQUATIONS AUX DÉRIVÉES PARTIELLES.

présenté par: François Guibault

en vue de l'obtention du grade de: MAÎTRE ÈS SCIENCES APPLIQUÉES (M.Sc.A.)

a été dûment accepté par le jury d'examen constitué de:

M. Louis Granger ..... M.Sc., président

M. Ricardo Camarero ..... Ph.D.

M. André Garon ..... Ph. D.

M. Dominique Pelletier ..... Ph. D.

Sommaire

Ce mémoire se veut une présentation des travaux entrepris lors de l'élaboration et de la mise en oeuvre d'un logiciel de description d'éléments pour fins de calculs numériques. Cette présentation sera divisée en trois sections principales, soit la mise en contexte du problème, la description des structures d'information utilisées et la présentation des algorithmes fondamentaux du logiciel.

Nous constaterons d'abord le rôle central que joue le concept d'élément au sein de la démarche de résolution d'un problème par la méthode des éléments finis. Ce rôle central servira de tremplin à une démarche de généralisation et de spécification du concept d'élément. Ceci permettra par la suite une dissociation pratiquement complète des étapes de description des éléments de calcul et de résolution proprement dite.

La séparation entre la description d'élément et la résolution passe essentiellement par l'identification des différents types d'information qui entrent en jeux dans un processus classique de résolution par éléments finis. C'est la caractérisation de ces différents types de données qui constitue la contribution principale du travail présenté ici; l'identification des données et l'analyse des méthodes d'acquisition,

de stockage, de hiérarchisation et de maintient de la cohérence interne des données constituent donc la portion la plus importante de ce travail.

Chaque type de données traité par le logiciel se retrouve implicitement dans tout programme de résolution numérique par éléments finis. En général, cependant, les programmes classiques de résolution utilisent une codification des éléments très liée au mode de fonctionnement interne du programme. Ce travail vise donc en premier lieu à montrer qu'il est possible de séparer nettement les étapes de description d'élément et de résolution numérique; puis donne les grandes lignes de la réalisation d'un logiciel dédié à la description des éléments qui puisse interfacer aisément avec un résoluteur conçu à cette fin.

Le bénéfice net de cette démarche est de proposer une méthodologie simple pour la construction de résoluteurs numériques, qui accepteront des descriptions génériques d'éléments, et qui n'auront donc plus à être modifiés pour être utilisés avec un nouveau type d'élément.

Abstract

The concept of element plays a central role within many numerical methods currently in use for solving physical problems trough simulation and especially in the finite element method (FEM). The work presented here originated from the difficulties encountered by researchers who needed to make their simulation tools more general and independent of the different properties associated with the elements. These difficulties have forced a restructuration of the numerical applications and lead to the development of a new piece of software whose only purpose is the description of elements used in numerical simulations.

Indeed, the description of the elements can be made totaly independent of the solving process through a carefull identification of all informations classicaly used in FEM, and by means of providing these informations to the solving process in a way that is independent of the actual element used. This work concentrates on characterizing the different data types involved in the description process, aswell as organizing this data hierarchicaly and providing means of gathering, storing and maintaining internal consistency of the data.

The net benefit of this approach is to give a simple methodology for building numerical solvers independent of the elements used for discretization and thus preventing a rewriting of the software in order to use a new kind of element.

Remerciements

Nous tenons à remercier très sincèrement notre directeur, M. Ricardo Camarero, et notre co-directeur, M. André Garon pour le support, les idées et les conseils qu'ils ont su nous prodiguer tout au long de l'élaboration de ce travail.

Notre reconnaissance va également à M. Robert Magnan pour son excellent esprit d'équipe et ses commentaires toujours pertinents; ce travail s'inscrivant dans un projet d'équipe, n'aurait pu être mené à bien sans son aide précieuse. Nous remercions également les professeurs et les étudiants du groupe MIAO pour leur support et particulièrement M. Dominique Pelletier et M. André Fortin qui ont su nous éclairer dans bien des domaines tant au travers des cours suivis avec eux que par de nombreuses discussions enrichissantes.

Enfin nous tenons à remercier de tout coeur Mme Isabelle Naud pour sa patiente compréhension et son aide inestimable dans les moments difficiles.

Table des Matières

Sommaire iv

Abstract vi

Remerciements viii

Liste des figures xiii

Introduction 1

1 La méthode des éléments finis 3

1.1 Formulation du problème 4

1.1.1 Équations 4

1.1.2 Domaine 5

1.2 Discrétisation 7

1.2.1 Discrétisation des équations 7

1.2.2 Discrétisation du domaine 9

1.3 Résolution 11

1.4 Présentation des résultats 11

1.5 Les éléments dans la méthode des éléments finis 13

2 Description générique d'un élément **14**

2.1 L'information nécessaire au résoluteur 15

2.2 Le résoluteur: Une machine à sommer qui résoud Ax = B 16

2.3 Les variables 20

2.4 Regroupement de l'information sur un élément 22

2.4.1 La géométrie 24

2.4.2 Les inconnues 27

2.4.3 Les règles d'intégration 31

2.4.4 Les interpolants 32

Méthode de Lagrange 33

Méthode d'Hermite 35

Développement en série de Taylor 37

Bulles 39

2.5 La hiérarchie de l'information 42

2.5.1 Les formes géométriques 43

2.5.2 Les groupes de degrés de liberté 46

2.5.3 Les éléments 47

2.6 Accessibilité de l'information 50

3 Quelques aspects de l'implantation du programme 'DE' 53

3.1 Quelques algorithmes fondamentaux à 'DE' 54

3.1.1 Le calcul automatique des fonctions de base 55

Interpolants de Lagrange 55

Interpolants d'Hermite 63

Interpolants en développement en série de Taylor 64

Fonctions bulles 66

3.1.2 Évaluation des fonctions de base aux points de Gauss . . . 69

3.1.3 Génération de code 70

3.2 La manipulation symbolique 71

3.2.1 Les contraintes géométriques 73

3.2.2 La définition de la forme faible 73

3.2.3 La modification des fonctions de base 74

Bibliographie **77**

Figures

1.1 Exemple de domaine de calcul 6

1.2 Exemple d'élément de calcul 8

1.3 Exemple de maillage 10

1.4 Exemple de solution: Lignes de courant 12

2.1 Modèle global du système PIRATE 17

2.2 Hiérarchie des classes d'information dans 'DE' 23

2.3 Données géométriques pour un élément simple 27

2.4 Construction de groupes de DDL à l'aide de composantes scalaires 29

2.5 Construction de groupes de DDL à l'aide de composantes vectorielles 30

2.6 Différentes configurations valides de DDL sur un élément rectangulaire 31

2.7 Quelques règles d'intégration valides sur un élément triangulaire . 32

2.8 Configuration de DDL utilisée pour l'interpolation de Lagrange . . 33

2.9 Configuration invalide de DDL sur un élément triangulaire 35

2.10 Configuration de DDL utilisée pour l'interpolation d'Hermite . . . 36

2.11 Configuration de DDL utilisée pour l'interpolation en série de Taylor 38

2.12 Configuration de DDL utilisée pour l'interpolation avec une bulle . 40

2.13 Trois groupes de DDL utilisant trois bulles linéairement indépendantes 42

2.14 Espaces de paramétrisation des quatre formes géométiques de base 44

3.1 Eléments utilisés pour la construction d'interpolants de Lagrange: exemples simples 58

3.2 Eléments utilisés pour la construction d'interpolants de Lagrange: exemple plus complexe 61

3.3 Construction de bulles ne s'annulant pas uniquement sur les frontières de l'élément 68

Introduction

L'idée de construire un système intégré de résolution numérique de problèmes de génie suscite un intérêt grandissant au sein de groupes de recherche oeuvrant dans différents domaines des sciences et du génie.

Le présent travail s'inscrit dans un projet de développement d'un outil de résolution intrégré, axé principalement sur les problèmes de mécanique des milieux continus, et plus particulièrement sur ceux de mécanique des fluides; le système PIRATE (Programme Intégré de Résolution, d'Analyse et de Traitement par Eléments finis) a donc été conçu pour répondre en premier lieu aux besoins rencontrés dans ces champs d'application. Le système a par ailleurs été réalisé de façon à être utilisable dans les domaines d'application les plus vastes et les plus diversifiés possible.

Comme l'indique le nom du logiciel, la méthode de résolution proposée est celle des éléments finis. Cette méthode, qui a fait ses preuves tant dans le domaine de la mécanique des fluides que dans de nombreuses autres sphères d'activités du génie, semblait particulièrement bien adaptée aux besoins d'un logiciel intégré comme PIRATE.

D'une part, la méthode des éléments finis repose sur de solides bases mathématiques et par ailleurs, elle offre une grande capacité d'adaptation lorsqu'il s'agit de résoudre un problème spécifique. Ces deux qualités sont pratiquement indispensables pour l'élaboration d'un système de résolution de problèmes ne reposant pas sur une connaissance préalable du type de problème à résoudre.

Les travaux présentés dans ce mémoire ont pour but de rendre le système de résolution indépendant du type d'élément utilisé pour le calcul. Cette indépendance doit être réalisée en deux étapes. D'abord en fournissant à l'intérieur du système tous les outils permettant de décrire l'élément à utiliser lors de la résolution. Et puis, en s'assurant que le reste du système n'utilise aucune information sur les éléments n'ayant pas été recueillie à l'aide des outils fournis à cette fin.

L'exposé débutera par une présentation de la méthode des éléments finis insistant principalement sur les notions pertinentes aux éléments utilisés lors du calcul. Suivra une discussion sur la méthode de représentation des données permettant d'opérer une séparation complète entre la description de l'élément et le résoluteur. Finalement, le logiciel sera présenté, et les aspects plus techniques de sa mise en oeuvre seront touchés.

Chapitre 1

La méthode des éléments finis

Les recherches présentées dans ce mémoire étant relativement spécialisées au domaine des éléments finis, on débutera cet exposé par une mise en contexte du domaine de recherche et une description de la méthodologie de résolution par éléments finis. A cette fin, un problème simple sera formulé puis résolu par la méthode des éléments finis: chaque étape de la solution permettant de faire ressortir les aspects essentiels de la méthode.

Le problème choisi est celui de l'écoulement d'un fluide parfait (irrotationnel) autour d'un cylindre, problème que l'on retrouve fréquement dans la littérature, et entre autre dans Reddy[1].

Le traitement du problème comprend trois étapes principales soit la formulation du problème, la discrétisation et la résolution. A ces étapes de résolution, sera greffée une étape de présentation des résultats, qui ne constitue pas en soi une étape particulière à la méthode des éléments finis, mais qui permet la présentation de résultats obtenus pour le problème.

1.1 Formulation du problème

1.1.1 Équations

Le point de départ de pratiquement n'importe quel problème numérique est évidemment une équation différentielle que l'on veut résoudre sur un domaine de calcul quelconque. Le problème de l'écoulement d'un fluide parfait n'est pas différent, et dans ce cas-ci l'équation à résoudre s'écrit:

$$\nabla^2 \Psi = 0 \quad sur \ \Omega \tag{1.1}$$

En utilisant la formulation du problème dite "en fonction courant", nous pouvons exprimer les composantes du champ de vitesse du fluide:

$$u_1 = \frac{\partial \Psi}{\partial y} \tag{1.2}$$

$$u_2 = -\frac{\partial \Psi}{\partial x} \tag{1.3}$$

Cette remarque sera importante lors de l'imposition des conditions limites.

La première étape dans la démarche de résolution par éléments finis consiste à réécrire l'équation différentielle sous sa forme faible, en projetant la solution Ψ sur la base d'un espace de fonctions compatibles avec les fonctions attendues pour la solution (méthode des résidus pondérés):

$$\int_\Omega \nabla \Psi \, \nabla v \, dx = \int_\Gamma \frac{\partial \Psi}{\partial n} \, v \, dS \quad \forall v \in H^1 \tag{1.4}$$

Cette écriture permet, à l'étape de discrétisation, de construire un système algébrique pour trouver les différents coefficients de pondération.

1.1.2 Domaine

Le second aspect fondamental de la formulation du problème est bien sûr la définition du domaine de calcul. On distingue généralement lors de cette étape:

la définition du domaine géométrique, la définition du zonage et l'imposition des conditions aux frontières.

La définition géométrique du domaine, sous une forme mathématiquement précise et utilisable par un ordinateur peut s'avérer une tâche complexe nécessitant des outils spécialisés. Dans le cas qui nous occupe, cependant, cette étape est expédiée rapidement puisqu'il s'agit d'un domaine simple en deux dimensions, comme le montre la figure 1.1.

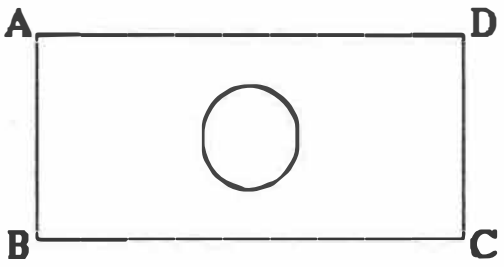


Figure 1.1: Exemple de domaine de calcul

D'autre part, il est souvent intéressant lors de la résolution d'un problème de créer des zones à l'intérieur du domaine; ceci peut permettre par exemple de résoudre des équations distinctes ou d'utiliser des discrétisations différentes sur un même domaine. Encore ici, le problème choisi ne présente pas de telles difficultés, et peut être résolu avec une seule zone couvrant l'ensemble du domaine.

La troisième étape de la formulation du problème au niveau du domaine, l'imposition des conditions aux frontières, permet de donner les conditions mathématiques de résolution du problème. Pour l'exemple choisi, les conditions sont spécifiées plus naturellement à certains endroits sur la vitesse que sur la fonction courant; c'est pourquoi les équations reliant la vitesse à la fonction courant seront

utiles ici.

De façon synthétique, nous avons les conditions suivantes de résolution:

$$
\begin{aligned}
&sur \quad AB : \quad \frac{\partial \Psi}{\partial y} = -V \\
&sur \quad BC : \quad \frac{\partial \Psi}{\partial x} = 0 \\
&sur \quad CD : \quad \frac{\partial \Psi}{\partial y} = V \\
&sur \quad DA : \quad \frac{\partial \Psi}{\partial x} = 0 \\
&sur \quad le \quad cylindre : \quad \frac{\partial \Psi}{\partial n} = 0 \\
&et \quad en \quad A : \quad \Psi = 0
\end{aligned}
$$

1.2 Discrétisation

Une fois le problème bien posé dans le continu, il s'agit de passer à une description discrète pour la résolution numérique. Tout comme la formulation du problème, la discrétisation s'opère à deux niveaux: sur les équations à résoudre et sur le domaine de calcul.

1.2.1 Discrétisation des équations

Discrétiser les équations consiste à remplacer l'espace infini des fonctions solutions et des fonctions de projection par des ensembles finis de fonctions compatibles. Avec la méthode de Galerkin, que nous utiliserons ici, les fonctions d'approximation pour la solution et pour les fonctions de projection (fonctions tests) sont les mêmes; nous parlerons désormais de fonctions de base dans un cas comme dans l'autre.

Le choix des fonctions de base est directement lié à celui de l'élément de discrétisation. En effet, la méthode classique pour choisir les fonctions de base consiste à utiliser un ensemble de fonctions d'interpolation de la géométrie de l'élément, puis de choisir l'ordre de l'interpolation en fonction de la précision de l'approximation désirée.

Pour le problème à résoudre, un choix acceptable d'élément pourrait par exemple être le triangle à trois noeuds:

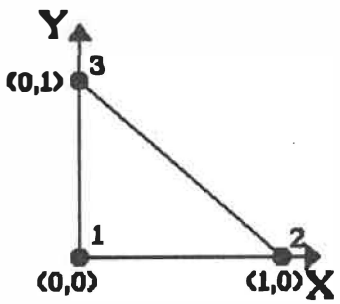


Figure 1.2: Exemple d'élément de calcul

pour lequel les fonctions d'interpolation linéaire sont:

$$N_1 = x$$

$$N_2 = y$$

$$N_3 = 1 - x - y$$

L'approximation par éléments finis consiste alors à écrire pour la solution:

$$\Psi = \sum_i \phi_i N_i \tag{1.5}$$

en considérant qu'une inconnue est associée à chaque noeud de l'élément. Les fonctions tests v sont par ailleurs choisies elles aussi parmi l'ensemble des fonctions de base et nous avons donc:

$$v = N_j, \quad j = 1, 2, 3 \tag{1.6}$$

En substituant Ψ et v dans l'équation continue de la section 1.1.1 (équation 1.4), nous obtenons un système linéaire d'équations algébriques:

$$K_{ij}\phi_j = F_j \tag{1.7}$$

avec

$$K_{ij} = \int_{\Omega_e} \nabla N_i \quad \nabla N_j \, dx \tag{1.8}$$

et

$$F_j = \int_{\Gamma_e} \frac{\partial \Psi}{\partial n} N_j \, dS \tag{1.9}$$

Cette équation sert à la construction des matrices élémentaires, elles-mêmes assemblées en une matrice globale à l'étape de résolution.

1.2.2 Discrétisation du domaine

Parallèlement à l'étape d'approximation des équations à résoudre, il est nécessaire de procéder à la discrétisation du domaine de calcul. Cette démarche effectue d'une part une subdivision du domaine et d'autre part fournit une représentation nodale des conditions limites associées au problème.

Ces deux processus sont généralement réalisés à l'aide de logiciels spécialisés utilisant une gamme étendue d'algorithmes de discrétisation pouvant satisfaire aux exigences particulières du problème à résoudre. Une certaine quantité d'information est cependant toujours pertinente quelle que soit la méthode de discrétisation; en particulier la morphologie de l'élément. Dans tous les logiciels rencontrés jusqu'ici, ce type d'information est toujours codifié de façon rigide au niveau du mailleur, qui ne peut travailler qu'avec un ensemble restreint d'éléments.

Dans la majorité des cas, les résultats de l'étape de discrétisation sont présentés sous forme d'une liste de noeuds et d'une liste d'éléments. Aux noeuds sont associées différentes propriétés, entre autre les coordonnées géométriques et la connectivité aux degrés de liberté du problème, en plus d'un code pour chaque degré de liberté indiquant s'il s'agit d'une inconnue ou d'une condition limite. Evidemment toutes ces informations sont transmises sous forme de codes qui doivent être interprétés par le résoluteur.

La discrétisation du domaine de calcul pour notre exemple pourrait donner par exemple le maillage de la figure 1.3:

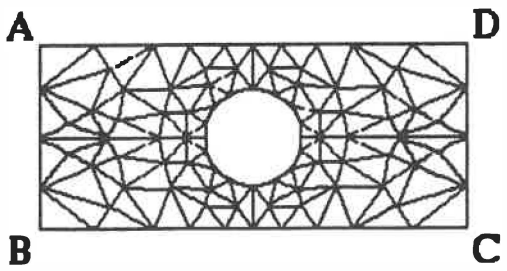


Figure 1.3: Exemple de maillage

1.3 Résolution

Cette troisième étape du processus de traitement consiste à rassembler l'information provenant des deux étapes précédentes pour construire un système algébrique qui sera résolu numériquement. Dans la majorité des cas, le résoluteur est construit en fonction d'une équation ou d'un ensemble unique d'équations, et ne travaille qu'avec un ensemble restreint de types d'éléments.

Le travail du résoluteur consiste donc à construire les matrices élémentaires et les membres de droite de chaque élément, à les assembler dans un système global, puis à résoudre le système. Pour une équation donnée, c'est généralement l'algorithme de résolution du système qui caractérise le plus un résoluteur.

Dans le cas du problème présenté ici, le résoluteur fournira une valeur de la fonction courant en chaque noeud du maillage et l'algorithme de résolution utilisé est celui du gradient conjugué.

1.4 Présentation des résultats

Bien que cette étape ne fasse pas à proprement parler partie de la méthode des éléments finis, il est indéniable que sans elle les résultats obtenus ne sont pas facile à analyser. Par exemple, pour le problème choisi, une fois la fonction courant connue en chaque point du maillage le problème est résolu. Des manipulations supplémentaires sont cependant nécessaires pour rendre les résultats utilisables par un humain. En particulier, il est sans doute très intéressant de calculer le champ de

vitesse à l'intérieur du domaine et d'en produire une représentation graphique. Ce type de manipulation constitue l'étape importante de la présentation des résultats.

Pour que le logiciel de post-traitement soit flexible et puissant, il faut qu'il ait accès à une grande quantité d'information provenant des étapes précédentes. En particulier, le post-processeur doit avoir accès à la discrétisation du domaine et aux propriétés des éléments utilisés pour la discrétisation. Ces informations sont nécessaires par exemple pour connaître la morphologie exacte du domaine de calcul ou les règles d'association des valeurs des inconnues aux noeuds du maillage.

Dans le cas de l'écoulement autour du cylindre, une représentation du champ de vitesse du fluide est illustrée à la figure 1.4.

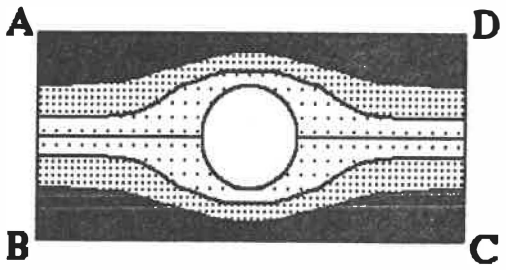


Figure 1.4: Exemple de solution: Lignes de courant

1.5 Les éléments dans la méthode des éléments finis

Ce bref aperçu de la méthode des éléments finis montre jusqu'à quel point la notion d'élément joue un rôle central dans la méthode. Chaque étape de la méthode dépend en effet d'une ou de plusieurs des propriétés des éléments utilisés pour résoudre. D'autre part, les logiciels disponibles à l'heure actuelle pour la résolution par éléments finis ne permettent que difficilement l'introduction de nouveaux éléments ou la modification des propriétés d'éléments existants en fonction de besoins nouveaux.

Les chercheurs désirant utiliser un nouvel élément sont alors confrontés à la lourde tâche soit de modifier une grande quantité de code soit carrément d'écrire un nouveau logiciel.

Dans ce mémoire, nous proposons une description générale des entités qui permet la production de résoluteurs indépendants des éléments, et ainsi rend possible l'introduction de nouveaux éléments sans modification au code existant.

Chapitre 2

Description générique d'un élément

Dans ce chapitre, nous nous attachons à décrire les différents types de données nécessaires à la description complète d'un élément; nous présentons une façon de rendre la description de l'élément de calcul indépendante du processus de calcul. Une discussion des avantages et des contraintes associés au choix de cette approche est ensuite présentée.

Le chapitre suivant, de son côté, sera consacré aux aspects plus techniques de la mise en oeuvre d'un logiciel utilisant ces données et aux questions algorithmiques.

2.1 L'information nécessaire au résoluteur

Par description générique des éléments de calcul pour le résoluteur, on entend une description sans lien explicite au niveau des structures de données et sans convention "implicite" de codification de l'information. Cette idée s'impose lors de l'analyse de l'information utilisée par un résoluteur type, analyse qui permet de formuler quelques remarques importantes:

- seul un petit nombre de modules utilisent de l'information provenant de la description des éléments,
- Pour certains éléments, toute l'information nécessaire peut être tabulée une fois pour toute et reste indépendante du problème ou de l'équation à résoudre,
- Pour certains éléments, en plus de l'information tabulée, on doit fournir une fonction d'évaluation de certains interpolants. Ces fonctions peuvent cependant être générées automatiquement.

Ces constatations nous amènent à envisager la possibilité d'écrire un résoluteur par éléments finis en terme d'un élément tout à fait général, dont les propriétés ne sont connues qu'au moment de la résolution.

Cette idée est à la base de la conception du logiciel 'DE' (Descripteur d'Elément). Voyons donc quel type d'information est nécessaire au résoluteur, et comment on peut amasser cette information.

2.2 Le résoluteur: Une machine à sommer qui résoud $\mathbf{Ax = B}$

Dans le modèle conceptuel d'un programme d'éléments finis que nous avons adopté (voir figure 2.1), le résoluteur est constitué d'un ensemble de quatre modules dont la tâche comprend le calcul des matrices élémentaires et des membres de droite, leur assemblage en un système d'équations global et la résolution de ce système.

Cette division des tâches laisse au résoluteur le choix tant de l'algorithme de résolution que de la représentation interne des données du système d'équations. En somme, le résoluteur n'est pas limité soit aux méthodes directes ou itératives de résolution du système d'équations, pas plus qu'il n'est astreint à une représentation assemblée (ou non) de la matrice.

Le résoluteur requiert de l'information de deux sources principales: du module de discrétisation (du modèle géométrique et des conditions limites), et du

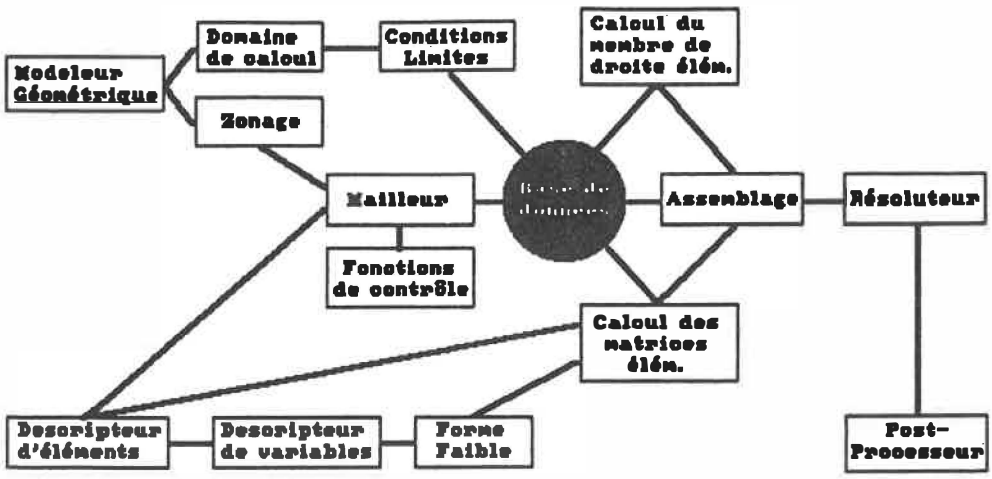


Figure 2.1: Modèle global du système PIRATE

descripteur d'élément. Du second module provient l'information permettant au résoluteur de calculer les termes de la matrice élémentaire, tandis que le premier fournit l'information qui permet d'assembler les matrices élémentaires dans la matrice globale. Rappelons que les termes de la matrice élémentaires sont en général de la forme:

$$\int_e \frac{\partial^m V}{\partial X^m} \quad \frac{\partial^n U}{\partial X^n} dX \tag{2.1}$$

que l'on évalue par intégrales numériques:

$$\sum_i W_i \frac{\partial^m V}{\partial X^m}(X_i) \quad \frac{\partial^n U}{\partial X^n}(X_i) J_e \tag{2.2}$$

avec:

- $\frac{\partial^m V}{\partial X^m}(X_i)$ la valeur de la dérivée d'ordre 'm' de la fonction test au $i^{ème}$ point d'intégration
- $\frac{\partial^n U}{\partial X^n}(X_i)$ la valeur de la dérivée d'ordre 'n' de la fonction cherchée pour les degrés de liberté au $i^{ème}$ point d'intégration

- W_i la valeur du $i^{ème}$ poids d'intégration
- J_e la valeur du déterminant de la matrice de transformation de l'élément de référence à l'élément réel.

En remplaçant U par son approximation en terme des fonctions de base élémentaires, on peut écrire:

$$U = N_j \phi^j \tag{2.3}$$

et en posant pour les fonctions test:

$$V = N_k, \quad N_k \in V_h, \quad V_h = \{N_1, N_2, N_3, ..., N_n\} \tag{2.4}$$

nous pouvons réécrire le terme général précédent (équation 2.2):

$$\sum_i W_i \frac{\partial^n (N^K)}{\partial^n X}(X_i) \frac{\partial^n (N_j \phi^j)}{\partial^n X}(X_i) J_e \quad \forall \quad N^k \in V_h \tag{2.5}$$

Lorsque les fonctions de base ne dépendent que des coordonnées locales de l'élément de référence et non des coordonnées de l'élément réel ou d'autres paramètres globaux du problème (la solution, par exemple), ce terme général possède une propriété fort importante. En effet, sous cette condition, seul le déterminant de la transformation, J_e, ne peut être calculé une fois pour toutes pour l'élément de référence. Les autres facteurs, soit les dérivées locales des fonctions test et des fonctions de base évaluées aux points de Gauss et les poids de Gauss, peuvent tous être évalués sur l'élément de référence et tabulés une fois pour toutes. Dans ce cas, le résoluteur n'a même pas à connaître l'expression des fonctions de base, des fonctions test ou de leurs dérivées pour construire la matrice A.

Il existe toutefois certaines classes d'interpolant qui utilisent, lors de leur évaluation, de l'information provenant de l'élément réel. Ces interpolants feront l'objet

d'un traitement légèrement différent, comme nous le verrons au chapitre suivant. Pour l'instant, nous concentrons notre attention sur la classe très vaste des interpolants qui sont évalués directement au niveau de l'élément de référence.

Ceci nous permet d'affirmer qu'il est inutile, dans la plupart des cas, d'évaluer les fonctions de base et leurs dérivées plus d'une fois. Ces valeurs constituent des données propres à un élément, et sont indépendantes tant de l'équation à résoudre que de la méthode de résolution ou du domaine géométrique sur lequel est posé le problème.

Un autre point de grande importance est que la valeur des fonctions de base et de leurs dérivées aux points de Gauss constituent généralement, avec le déterminant de la transformation entre l'élément de référence et l'élément réel, les seules informations nécessaires au résoluteur pour le calcul des matrices élémentaires.

Il est donc envisageable, dans ce contexte, de fournir au résoluteur exactement l'information dont il a besoin sans pour autant recourir à une codification des éléments et de leurs propriétés. Cette façon de faire devrait permettre aux usagers du système de résolution de spécifier exactement le type d'élément qu'ils veulent utiliser lors de la résolution et même de construire des nouveaux éléments.

Fournir une information du type de celle que nous venons de décrire constitue le premier objectif du logiciel 'DE'. Pour le réaliser, cependant, il faut élaborer une structure de données qui permette de représenter l'ensemble des éléments connus à l'heure actuelle, et que l'on puisse enrichir par la suite.

2.3 Les variables

Sur le plan opérationnel, cet objectif se traduit par la production d'une table des fonctions de base et de leurs dérivées évaluées à chaque point de Gauss. L'une des premières étapes sera d'identifier les quantités qui doivent être approximées. Cette étape est en fait réalisée par une petite application indépendante, 'DV', dont nous ne présenterons ici que les grandes lignes.

Tout comme 'DE' tente de rendre le reste du système de résolution indépendant du type d'élément utilisé, 'DV' isole le système de résolution de la description des variables proprement dite. Cette séparation s'opère d'autant plus facilement que les tâches de chacun sont bien définies. Tel est le cas pour 'DV', qui sert d'interface entre les usagers et les autres modules du système de résolution pour tout ce qui a trait aux quantités utilisées lors de la résolution. On entend par quantité utilisée toute inconnue, quantité dérivée ou propriété apparaîssant dans les équations à l'une ou l'autre étape du traitement numérique.

On appelle inconnues les quantités calculées directement lors de la résolution du système d'équations. Les quantités dérivées, de leur côté, sont obtenues à l'aide d'équations supplémentaires utilisant les valeurs des inconnues; les propriétés, enfin, sont des quantités connues a priori sur l'ensemble du domaine. Dans un problème typique de mécanique des fluides, certaines des inconnues seraient par exemple les vitesses, une quantité dérivée serait la vorticité, alors que la viscosité serait une propriété.

Pour chaque quantité définie, que nous appellerons désormais les variables du système, 'DV' conserve une série d'informations accessibles aux autres modules du

système. Par exemple, les variables peuvent être soit des scalaires, soit des vecteurs ou même des tenseurs d'ordre plus élevé, mais cette information sur une variable ne sera fournie que sur requête d'un des modules, et en général, les variables seront toutes traitées de la même façon quel que soit leur ordre.

Pour 'DE', les seules données utiles provenant de 'DV' sont les listes des noms des inconnues et des propriétés utilisées dans le problème. Le processus de description des éléments peut donc demeurer indépendant aussi bien de la représentation interne des variables que des méthodes de définition de ces variables (un changement dans 'DV' ne peut pratiquement pas affecter 'DE').

Il est donc essentiel que l'étape de description des variables ait été au moins partiellement complètée lorsque la description d'un élément est entreprise. Ce sont en fait les seules données utilisées par 'DE' qui ne soient pas directement sous sa responsabilité.

L'application 'DE' travaille par ailleurs autant avec les propriétés qu'avec les inconnues d'un problème. Dans les deux cas l'approche est sensiblement la même, et le but reste toujours de fournir les outils nécessaires pour approximer soit les fonctions cherchées, soit les propriétés connues en certains points du domaine. C'est pour cette raison que le reste de la discussion portera principalement sur le traitement des inconnues qui constitue bien sûr le champ d'application le plus important de 'DE'.

2.4 Regroupement de l'information sur un élément

Dans les deux sections précédentes, nous avons identifié, pour 'DE', d'une part l'information de sortie, et d'autre part la portion de l'information d'entrée provenant de modules externes. Nous décrirons maintenant l'information d'entrée provenant soit de l'usager, soit de banques de données contrôlées directement par l'application 'DE'.

L'ensemble des données nécessaires à la description complète d'un élément peut être divisé conceptuellement en quelques classes, qui sont elles-mêmes hiérarchisées. Par ailleurs, l'implantation de structures de données et des fonctions permettant de les utiliser, est souvent plus simple lorsque les structures employées correspondent étroitement aux objets qui doivent être manipulés. L'application 'DE' présente donc à l'usager une façade divisée en classes hiérarchisées, que nous présenterons immédiatement, mais utilise une structure de données interne très proche des objets à manipuler, qui sera décrite par la suite.

L'information nécessaire à la définition d'un élément se regroupe en quatre classes fondamentales:

- la géométrie,
- les inconnues (et les propriétés),
- les règles d'intégration,
- les interpolants.

Ces classes forment une hiérarchie, illustrée à la figure 2.2.

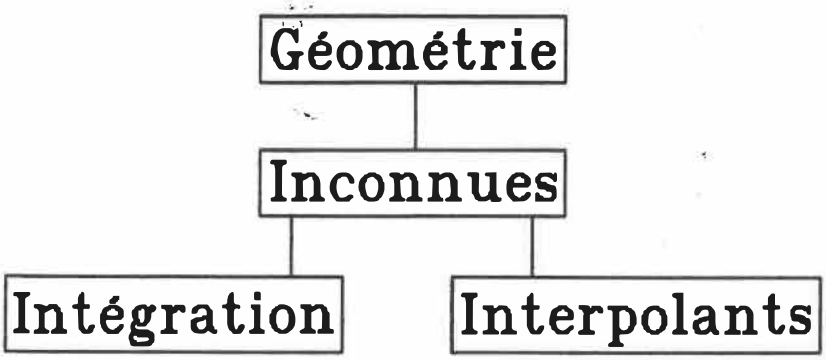


Figure 2.2: Hiérarchie des classes d'information dans 'DE'

Dans ce schéma, chaque classe comprend un sous-ensemble de données indispensable à la description d'un élément. Théoriquement, aucune spécification de données dans une classe ne pourrait être faite sans que les données nécessaires des classes inférieures n'aient été fournies. L'application 'DE' a cependant été construite de façon à constamment assurer la cohérence de ses données, et à fournir des valeurs de défaut pour toutes les données nécessaires à une description valide.

Nous présentons maintenant le type d'information pour chaque classe de données et la façon dont s'établissent les dépendances entre les classes.

2.4.1 La géométrie

Cette classe de données inclut tous les aspects morphologiques de l'élément; on y retrouve ainsi:

- la dimension,
- la forme,
- les critères de validité de la forme,
- l'interpolant géométrique,
- le nombre et la position des noeuds de calcul.

'DE' permet de construire des éléments à une, deux et trois dimensions. De la dimension de l'élément dépendent pratiquement toutes les autres données gérées par l'application. Ce paramètre doit donc être posé dès le début de la construction de l'élément, les structures de données étant réinitialisées chaque fois qu'il est changé.

La forme de l'élément (un triangle ou un quadrilatère en 2D, un tétrahèdre ou un hexahèdre en 3D), constitue un autre paramètre dont dépend la majorité des données relatives à l'élément. Le choix de la forme se fait à partir d'une liste de formes disponibles faisant partie d'une base de données gérée par 'DE'.

Les critères de validité de la forme, quant à eux, constituent une liste de contraintes déterminées par l'usager que doivent satisfaire chacun des éléments générés

lors de l'étape de discrétisation. Ces contraintes pourront être utilisées par le système lors d'une analyse du maillage et pourront fournir des critères de décision pour l'utilisation optionnelle d'étapes de remaillage sélectif ou de raffinement local du maillage. Ces données n'influencent aucun autre paramètre de l'élément, elles peuvent donc être spécifiées en tout temps et n'affectent en rien le déroulement de l'application.

De même, l'interpolant géométrique ne sera utilisé que lors des étapes de résolution (lors du calcul de la transformation de l'élément réel à l'élément de référence, par exemple) et de discrétisation; celui-ci peut donc être spécifié à tout moment durant la construction de l'élément. Les interpolants disponibles pour la géométrie sont tous de type "lagrangiens", et sont construits automatiquement pour une forme donnée en fonction du degré des polynômes spécifié par l'usager.

Du nombre et de la position des noeuds de calcul, enfin, dépendent un grand nombre des informations que l'on retrouve dans les autres classes de données de l'élément. Ces quantités déterminent en effet les positions possibles des noeuds de l'élément lorsque celui-ci est utilisé dans un processus d'interpolation. L'un des principaux objectifs de la méthode d'entrée des données est de conserver la symétrie lors de la disposition des noeuds sur l'élément.

Pour ce faire, 'DE' utilise une méthode d'entrée des données qui assure automatiquement le respect de certaines propriétés de symétrie. D'une part, 'DE' ne permet de positionner des noeuds que sur l'ensemble complet des entités géométriques d'un certain type: lorsque l'on veut, par exemple, placer un certain nombre de noeuds sur une arête, ou un noeud sur un sommet, automatiquement le programme se charge de placer des noeuds sur les autres arêtes ou sommets de l'élé-

ment. D'autre part, 'DE' utilise certaines informations contenues dans la base de données des formes géométriques et ne permet la spécification, sur les faces ou dans le volume de l'élément, que de configurations de noeuds respectant la symétrie de la forme. Enfin, chaque forme géométrique possède une fonction de validation des paramètres de positionnement des noeuds de calcul sur l'élément, qui, en dernier recours, doit pouvoir détecter les configurations de noeuds non symétriques sur l'élément.

Voici un exemple concret du type d'information que l'étape géométrie de l'application 'DE' recueille sur un élément:

- DIMENSION: 2D
- FORME: TRIANGLE
- CRITERES SUR LA FORME: Les angles aux trois sommets doivent tous êtres compris entre 50 et 70 degrés:

$$A1 = angle\ au\ premier\ sommet,$$
$$A2 = angle\ au\ second\ sommet,$$
$$A3 = angle\ au\ troisième\ sommet,$$
$$50 < A1 < 70$$
$$50 < A2 < 70$$
$$50 < A3 < 70$$

- DEGRE DE L'INTERPOLANT GEOMETRIQUE: 1 (interpolation linéaire)

- NOMBRE ET POSITION DES NOEUDS DE CALCUL:

	nombre	*paramètre*
AUX SOMMETS	1	
SUR LES ARETES	1	0.5
SUR LA FACE	1	(1/3,1/3)

Cet élément est illustré graphiquement à la figure 2.3.

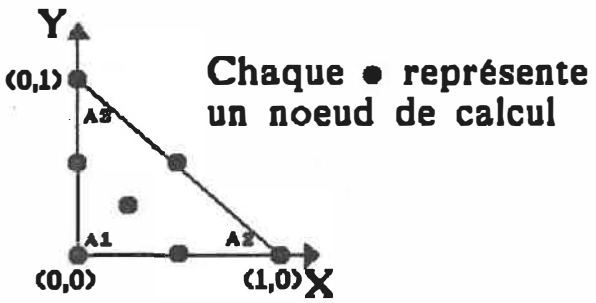


Figure 2.3: Données géométriques pour un élément simple

2.4.2 Les inconnues

La morphologie de l'élément bien établie, 'DE' est en mesure de recevoir l'information concernant les degrés de liberté associés à l'élément. Lors de cette étape de spécification des données, 'DE' construit des tables d'association des différents degrés de liberté (des inconnues) aux noeuds de calcul identifiés lors de l'étape précédente. Dans le cadre de l'application 'DE', ces tables d'association sont désignées par le terme "groupes de degrés de liberté".

Un groupe de degrés de liberté (DDL) contiendra donc, pour chaque noeud de

calcul, l'identificateur du DDL lui étant associé. Il va de soi que chaque groupe ne possède pas nécessairement des degrés de liberté à chaque noeuds, et qu'un même élément peut utiliser un nombre variable de groupes.

C'est à l'usager qu'incombe la responsabilité de spécifier la configuration de chaque groupe, et c'est ici qu'entre en jeu la liste des inconnues du problème obtenue de 'DV'. Les groupes sont en effet construits à partir des inconnues spécifiées à l'aide de 'DV'. Cette approche permet non seulement d'associer un DDL à un noeud de calcul, mais également d'identifier explicitement la variable, et donc possiblement l'équation, à laquelle est associé ce DDL.

A l'intérieur d'un groupe, un noeud ne peut supporter qu'une seule inconnue. Deux moyens sont cependant disponibles pour associer plus d'un DDL à un même noeud de calcul: on peut soit créer plusieurs groupes pour le même élément, soit utiliser des inconnues regroupant elles-mêmes plusieurs DDL sous forme d'un vecteur. Les deux approches peuvent même être utilisées conjointement sur un élément.

On pourrait par exemple utiliser l'élément décrit à la section précédente et, en employant les variables présentées dans la liste ci-dessous, construire des groupes de DDL selon l'une ou l'autre configuration présentées aux figures 2.4 et 2.5, soit à l'aide de composantes scalaires ou vectorielles.

- V le vecteur vitesse (3 composantes)
- u, v, w les trois composantes scalaires de vitesse
- P, Px, Py, Pz quatre composantes scalaires de pression obtenues par un développement de Taylor autour d'un point
- Plin le vecteur construit à partir des quatre composantes de pression

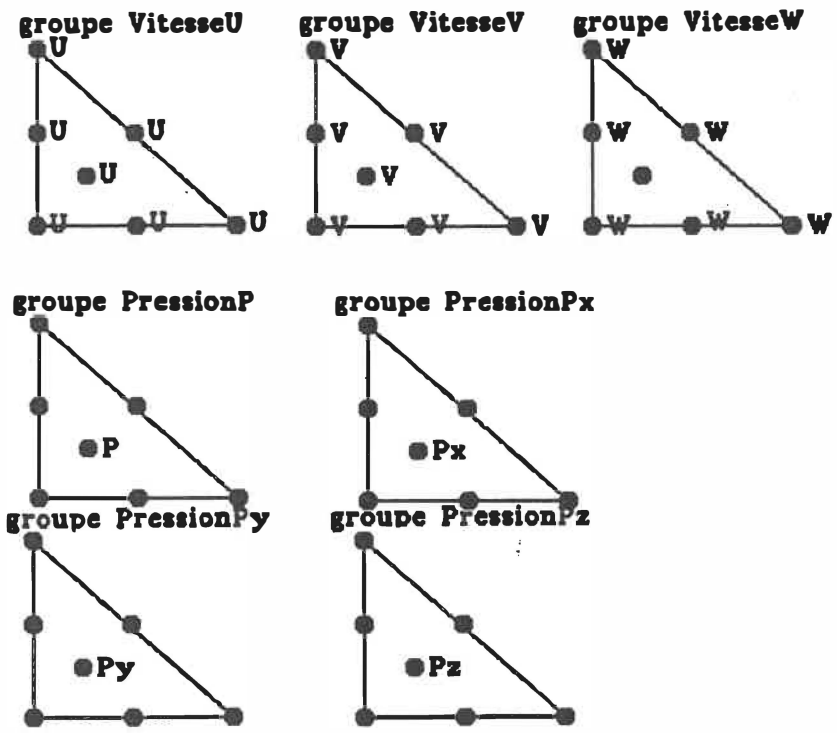


Figure 2.4: Construction de groupes de DDL à l'aide de composantes scalaires

Bien que la seconde méthode semble plus compacte et fonctionnelle, la première méthode peut également présenter plusieurs avantages pour certains problèmes. Un exemple qui vient immédiatement à l'esprit est le cas d'un problème 2D pour

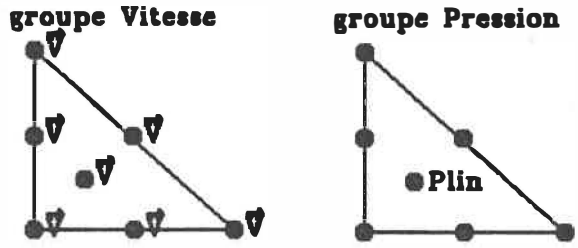


Figure 2.5: Construction de groupes de DDL à l'aide de composantes vectorielles

lequel la troisième composante de vitesse et la quatrième composante de pression sont inutiles. Dans ce cas, l'utilisation optimale de vecteurs impliquerait la modification de certaines définitions de variables, ce qui n'est pas toujours souhaitable au sein d'un projet. Par ailleurs, comme nous le verrons lorsque nous parlerons des interpolants, ces deux méthodes ne sont pas strictement équivalentes au niveau des schémas d'approximation utilisables avec chacune.

Il faut noter que 'DE' n'impose pratiquement aucune contrainte de symétrie lors de l'association des degrés de liberté aux noeuds de calcul. En fait, la seule contrainte à ce niveau se situe dans l'association d'un degré de liberté à un noeud de calcul placé sur un sommet de la forme géométrique: dans ce cas, 'DE' associe automatiquement ce DDL à tous les autres noeuds de calcul placés sur des sommets. Les éléments de la figure 2.6 seraient par ailleurs tout à fait réalisables.

Il se peut cependant que 'DE' soit incapable de construire automatiquement des interpolants sur certaines configurations de DDL.

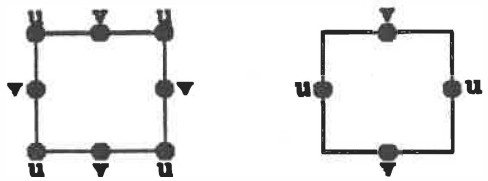


Figure 2.6: Différentes configurations valides de DDL sur un élément rectangulaire

2.4.3 Les règles d'intégration

En gardant toujours à l'esprit l'objectif de cette description, nous pouvons maintenant examiner la façon de spécifier les règles d'intégration qui seront utilisées lors de l'étape finale d'évaluation des interpolants aux différents points d'intégration.

Le choix des règles d'intégration, généralement gaussiennes, se fait par la spécification du nombre de point d'intégration et par le choix de la disposition de ces points sur la forme géométrique choisie.

Evidemment, il n'est pas toujours possible, avec un nombre arbitraire de points, de construire une règle d'intégration valide sur une forme donnée. Pour contourner ce problème, 'DE' utilise une base de données contenant pour chaque forme un certain nombre de règles d'intégration. Ansi, les règles d'intégration spécifiées sont valides, ce qui assure automatiquement la consistance interne des données.

Par exemple, dans le cas d'un triangle, 'DE' propose un grand nombre de règles, dont quelques unes sont illustrées à la figure 2.7.

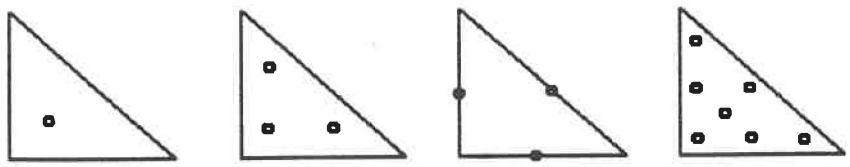


Figure 2.7: Quelques règles d'intégration valides sur un élément triangulaire

En ce qui concerne les groupes de DDL, il est important de noter que chaque groupe possède sa propre règle d'intégration. Ceci permet par exemple, sur un élément, d'utiliser des règles d'intégration réduites pour certains interpolants et d'intégrer les autres exactement.

2.4.4 Les interpolants

Le calcul automatique des interpolants représente l'essentiel de la puissance et de la versatilité de l'application 'DE'. C'est lors de cette étape que l'élément créé par l'usager est rendu générique du point de vue du résoluteur. En fait, cette étape de la description est relativement simple pour l'usager qui n'a qu'un choix restreint d'options; tout le travail est fait de façon interne et pratiquement automatiquement; l'usager étant simplement informé du résultat des calculs.

La génération automatique des interpolants peut se résumer comme suit: pour chaque groupe de DDL, 'DE' construit un ensemble de fonctions de base polynômiales à partir des spécifications de l'usager. Ces fonctions de base sont ensuite évaluées aux points d'intégration du groupe, ou bien être écrites sous forme de sous-programmes 'C' qui seront utilisés par le résoluteur.

Dans sa version courante, 'DE' prévoit quatre méthodes différentes de généra-

tion des interpolants. L'aspect mathématique de chacune des méthodes est présenté ici, alors que les considérations informatiques relatives à chaque réalisation sont exposées au chapitre suivant. Les méthodes suivantes sont donc disponibles pour le calcul des interpolants:

- interpolants de type Lagrange
- interpolants de type Hermite
- interpolants en développement en série de Taylor
- interpolants de type bulle

De plus, il est toujours possible de modifier ou remplacer certaines fonctions de base pour certains groupes de DDL.

Méthode de Lagrange

Avec l'interpolation de Lagrange 'DE' n'utilise, pour un ensemble de fonctions de base donné, qu'un seul degré de liberté à chaque noeud de calcul. Par ailleurs, 'DE' construit automatiquement autant d'ensembles de fonctions de base que nécessaire pour utiliser tous les degrés de liberté contenus dans le groupe.

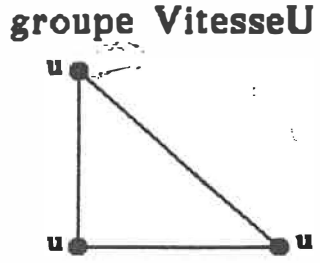


Figure 2.8: Configuration de DDL utilisée pour l'interpolation de Lagrange

Par exemple, pour le groupe suivant illustré à la figure 2.8, 'DE' ne produira qu'un seul ensemble de fonctions de base, utilisant l'unique degré de liberté disponible à chaque noeud. Nous aurons donc les fonctions de base standard:

$$N_1 = 1 - r - s$$

$$N_2 = r$$

$$N_3 = s$$

Par contre, si la variable scalaire 'u' est remplacée par la variable vectorielle à trois composantes 'V' à chaque noeud de calcul, 'DE' produira trois ensembles distincts de fonctions de base contenant chacun les trois fonctions présentées ci-haut. Chaque composante peut donc être traitée individuellement puisqu'elle possède des fonctions de base qui lui sont propres. Ansi, il est possible de choisir cette méthode de calcul des interpolants pour le groupe complet, puis de ne modifier les résultats obtenus que pour l'une ou l'autre des composantes.

'DE' lors de l'utilisation de cette méthode de calcul des interpolants, procède à certaines vérifications quant à la symétrie de l'association des degrés de liberté aux noeuds de calcul de l'élément. Il ne serait par exemple pas possible de construire des interpolants de Lagrange sur le groupe de DDL de la figure 2.9, qui ne respecte pas la symétrie de la forme géométrique.

En fait, cette restriction n'est pas vraiment propre à la méthode de génération des fonctions de base, on peut en effet construire des interpolants de Lagrange sur l'élément ci-dessus. L'exigence de symétrie se justifie plutôt par l'utilisation subséquente des éléments. Comme la description des éléments est faite de façon

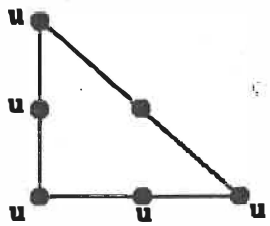


Figure 2.9: Configuration invalide de DDL sur un élément triangulaire

totalement indépendante du processus de discrétisation de la géométrie et des conditions limites, on peut supposer que certains éléments créés à l'aide de 'DE' seront utilisés dans des maillages non-structurés. Dans un tel contexte, il est donc impensable qu'un élément puisse avoir une direction privilégiée, qui ne pourrait être prise en compte lors de la discrétisation.

Méthode d'Hermite

Le calcul d'interpolants selon la méthode d'Hermite constitue en quelque sorte une généralisation de la méthode précédente. Souvent, cette seconde méthode est utilisée pour faire une approximation sur un élément d'une fonction et de ses dérivées aux mêmes points. Pour la construction de ce type d'interpolants, il est nécessaire de construire un groupe de DDL présentant plusieurs DDL associés à chaque noeud de calcul. Contrairement à la méthode de Lagrange, la méthode d'Hermite produit un seul ensemble de fonctions de base qui utilise simultanément tous les DDL. Par ailleurs, et comme c'est le cas pour la méthode précédente, les DDL doivent respecter une certaine symétrie sur l'élément; ils doivent de plus obéir à des règles précises quant à leur association aux noeuds de calcul, sans quoi 'DE' sera dans l'impossibilité de construire les interpolants demandés.

Les règles d'association des DDL aux noeuds de calcul sont fort simples et ne dépendent que de la dimension de l'élément. Il s'agit en fait de construire un vecteur de DDL comptant suffisamment de composantes pour contenir la valeur de la fonction à approximer et les valeurs des dérivées de la fonction suivant chaque paramètre géométrique jusqu'à l'ordre de dérivée désiré. Par exemple, en deux dimensions, l'approximation d'une fonctions 'u' et de ses dérivées d'ordre un selon 'x' et 'y', nécessite un vecteur à trois composantes, contenant:

$$U_{o1} = (u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}) \tag{2.6}$$

Un tel vecteur doit être associé à chaque noeud de calcul dans le groupe de DDL sur lequel on veut construire des interpolants de type Hermite:

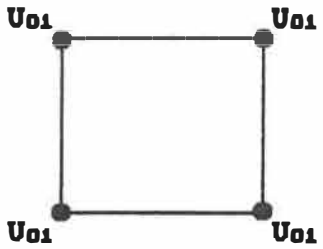


Figure 2.10: Configuration de DDL utilisée pour l'interpolation d'Hermite

Pour l'élément de la figure 2.10 en particulier, on obtient douze fonctions de base qui approximent u, $\frac{\partial u}{\partial x}$ et $\frac{\partial u}{\partial y}$ sur l'élément et qui sont exactes à chaque noeud.

De façon similaire, en trois dimensions, l'usager doit construire des vecteurs de DDL contenant la fonction à interpoler et ses dérivées selon x, y et z jusqu'à l'ordre désiré.

Développement en série de Taylor

La troisième méthode de construction d'interpolants sur un groupe de DDL proposée par 'DE' utilise un développement en série de Taylor. Cette méthode fonctionne en considérant de façon individuelle chacun des noeuds de calcul du groupe de DDL. 'DE' construit donc un ensemble de fonctions de base pour chaque noeud de calcul dans le groupe.

Les vecteurs de DDL présentés à la section précédente, sont également utilisés pour indiquer à 'DE' le nombre de termes dans la série. Le nombre de DDL contenus dans le vecteur doit donc suivre les règles énoncées à la section précédente et permettre de déterminer l'ordre maximum des dérivées utilisées dans le développement.

Le développement de Taylor construit par 'DE' se fait toujours autour du noeud de calcul pour lequel l'ensemble de fonctions de base est construit. Pour ce type d'interpolant, on utilise soit les coordonnées locales sur l'élément de référence, soit les coordonnées globales de l'élément réel. Dans le premier cas, les polynômes peuvent être construits durant l'exécution de 'DE' et les interpolants sont évalués immédiatement. Dans le second cas, les calculs doivent être faits au moment de la résolution, lorsque la géométrie, donc les coordonnées réelles des éléments, est connue. 'DE' doit produire alors une fonction externe que le résoluteur pourra appeler.

Par exemple, pour le vecteur de degrés de liberté $P_{o1} = (P, \frac{\partial P}{\partial x}, \frac{\partial P}{\partial y})$, et le groupe de degrés de liberté "Pression" sur l'élément de la figure 2.11, l'approximation de P par un développement limité en série de Taylor, avec (x_0, y_0) les coordonnées

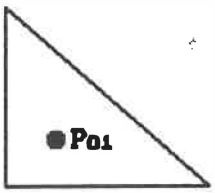


Figure 2.11: Configuration de DDL utilisée pour l'interpolation en série de Taylor

globales ou locales du noeud de calcul, s'écrit:

$$P(x,y) = P(x_0,y_0) + (x-x_0)\frac{\partial P}{\partial x}(x_0,y_0) + (y-y_0)\frac{\partial P}{\partial y}(x_0,y_0) \qquad (2.7)$$

que l'on peut réécrire:

$$P(x,y) = \sum_i N_i \phi_i \qquad (2.8)$$

avec

$$\phi = \{P, \frac{\partial P}{\partial x}, \frac{\partial P}{\partial y}\} \qquad (2.9)$$

et

$$N_1 = 1$$

$$N_2 = (x - x_0)$$

$$N_3 = (y - y_0)$$

Cette forme est donc identique à celle utilisée pour représenter les autres types d'interpolant. On note immédiatement que ce type d'interpolant est discontinu d'un élément à l'autre et que son utilisation devrait être justifiée par des propriétés particulières de la fonction à approximer.

Bulles

La dernière méthode de construction d'interpolants proposée par 'DE' exploite le concept de la fonction d'interpolation de type "bulle". On nomme bulle une fonction qui s'annule sur le bord de la forme géométrique. Ces fonctions ont la propriété intéressante de ne jamais intervenir lors de l'analyse de continuité des interpolants à l'interface entre deux types d'éléments. Elles servent à enrichir localement la qualité de l'approximation d'une fonction, le tout sans complication au niveau de la reconnection des groupes d'éléments de types différents.

'DE' associe chaque fonction bulle à un DDL spécifique. Il ne s'agit donc pas de la production d'un ensemble de fonctions de base comme c'était le cas pour les autres méthodes, mais bien de la construction d'une fonction unique associée à un DDL. A cette fin, 'DE' utilise une méthode de spécification très souple qui permet de bâtir un vaste éventail de fonctions. On peut en effet annuler une bulle non seulement sur les frontières géométriques de l'élément, mais également en chacun des autres noeuds de calcul associés à la face (en 2D) ou au volume (en 3D) de l'élément. On peut également assigner individuellement des exposants à chaque monôme servant à construire la fonction de base.

Un exemple de fonction bulle fréquement utilisée dans les calculs intervient dans la définition d'un élément triangulaire à quatre noeuds:

A cause du noeud de calcul supplémentaire au centroïde, un interpolant "standard" de type Lagrange ne peut pas être construit sur cet élément. La solution proposée par 'DE' consiste à utiliser un interpolant de Lagrange standard pour les trois premiers noeuds, puis une fonction bulle pour le quatrième noeud. Les trois

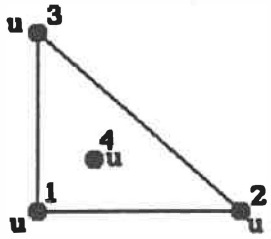


Figure 2.12: Configuration de DDL utilisée pour l'interpolation avec une bulle

premières fonctions sont ensuite modifiées automatiquement de façon à s'annuler au centroïde.

Dans le cas qui nous occupe, la fonction bulle doit s'annuler sur trois lignes de l'élément, les frontières géométriques, c'est-à-dire lorsque:

$$r = 0$$

$$s = 0$$

$$1 - r - s = 0$$

Le produit de ces trois monômes, élevés à n'importe quelle puissance supérieure ou égale à un, donne une fonction ayant les propriétés recherchées. En choisissant le degré 1 pour chaque monôme, on peut écrire la première bulle B_1:

$$B_1 = C\, r\, s\, (1 - r - s) \tag{2.10}$$

Cette fonction doit ensuite être normalisée pour valoir 1 au centroïde, c'est-à-dire en r=1/3, s=1/3.

$$B_1 = 27\, r\, s\, (1 - r - s) \tag{2.11}$$

On modifie les trois premières fonctions de base pour qu'elles s'annulent au centroïde:

$$N_1 = 1 - r - s - 9(rs - r^2 s - rs^2)$$

$$N_2 = r - 9(rs - r^2 s - rs^2)$$

$$N_3 = s - 9(rs - r^2 s - rs^2)$$

Ce qui donne, avec B_1, un ensemble complet de fonctions de base sur l'élément.

Cette méthodologie est en fait utilisée automatiquement par 'DE' pour compléter les ensembles de fonctions d'interpolation qui ne peuvent être créés directement à l'aide des méthodes classiques présentées plus haut. Par ailleurs, il peut être intéressant d'associer explicitement une bulle à un DDL, ce que permet 'DE' par son option de création de fonctions bulle.

Pour enrichir une approximation tout en conservant des propriétés de continuité des interpolants aux interfaces des éléments, on peut recourir à la méthode précédente, le développement en série de Taylor, ou bien utiliser des bulles.

La figure 2.13 présente un autre exemple d'utilisation des bulles. Dans cet exemple, trois groupes de DDL contenant chacun un seul DDL, utilisent des bulles linéairement indépendantes comme fonctions d'interpolation. Ces fonctions forment une base pour l'approximation de la fonction P et sont construites en élevant au carré tour à tour chacun des monômes de la bulle standard.

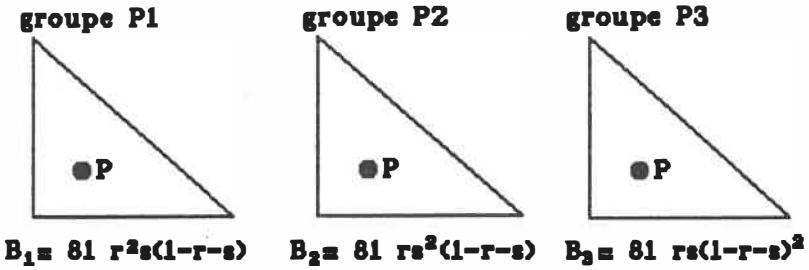


Figure 2.13: Trois groupes de DDL utilisant trois bulles linéairement indépendantes

2.5 La hiérarchie de l'information

L'information utilisée par 'DE' pour la définition d'un élément peut être décrite différemment si l'on adopte l'optique des objets qui doivent être manipulés de façon interne par l'application. En effet, à la section précédente, l'optique était celle du dialogue avec l'usager, bien différente de celle de la représentation interne des données utilisées.

Normalement, l'utilisation d'une application raisonnablement bien conçue ne nécessite pas une connaissance de l'organisation interne des données; par contre, pour modifier ou ajouter des fonctions supplémentaires, comprendre la structure des données devient indispensable.

Les données de l'application 'DE' sont structurées sous la forme d'"objets" plus ou moins complexes, qui sont manipulés et modifiés en cours d'exécution, principalement sous l'action de l'usager. L'approche "objet" pour la réalisation d'une application présente plusieurs avantages, en particulier la modularité du code et la simplicité de son entretien.

Dans 'DE' on distingue quelques classes principales d'objet, notamment:

- les formes géométriques,
- les groupes de DDL,
- les éléments.

dont nous présentons les principales caractéristiques. Cette description des objets ne se veut pas un exposé technique de l'implantation informatique réalisée dans 'DE', mais bien une initiation aux concepts les plus importants utilisés par l'application.

2.5.1 Les formes géométriques

La première classe d'objets manipulés par 'DE' est celle des formes géométriques, qui se divise immédiatement en trois sous-classes selon la dimension de la forme.

Toutes les informations pertinentes à une forme géométrique 2D ou 3D proviennent d'une banque de données sous le contrôle exclusif de 'DE'. Par ailleurs, comme en 1D la seule forme géométrique possible est la ligne droite, l'information nécessaire à 'DE' pour cette forme est incluse directement dans le code de l'application.

Le premier type d'information conservé par 'DE' sur une forme géométrique constitue une description hiérarchique de sa frontière dans un espace défini indépendamment pour chaque forme. Cette représentation de la frontière est basée sur

un ensemble de sommets, dans un espace 2D ou 3D. Ces sommets sont ensuite reliés deux à deux pour former des arêtes, elles-mêmes regroupées en faces; en 3D enfin, les faces sont groupées et forment un volume.

La paramétrisation utilisée dans 'DE' tente d'être compatible avec les conventions couramment utilisées dans la littérature, sans essayer de forcer une paramétrisation unique sur toutes les formes géométriques disponibles. Dans cet esprit, les paramétrisations illustrées à la figure 2.14 ont été utilisées pour les quatre formes géométriques de base.

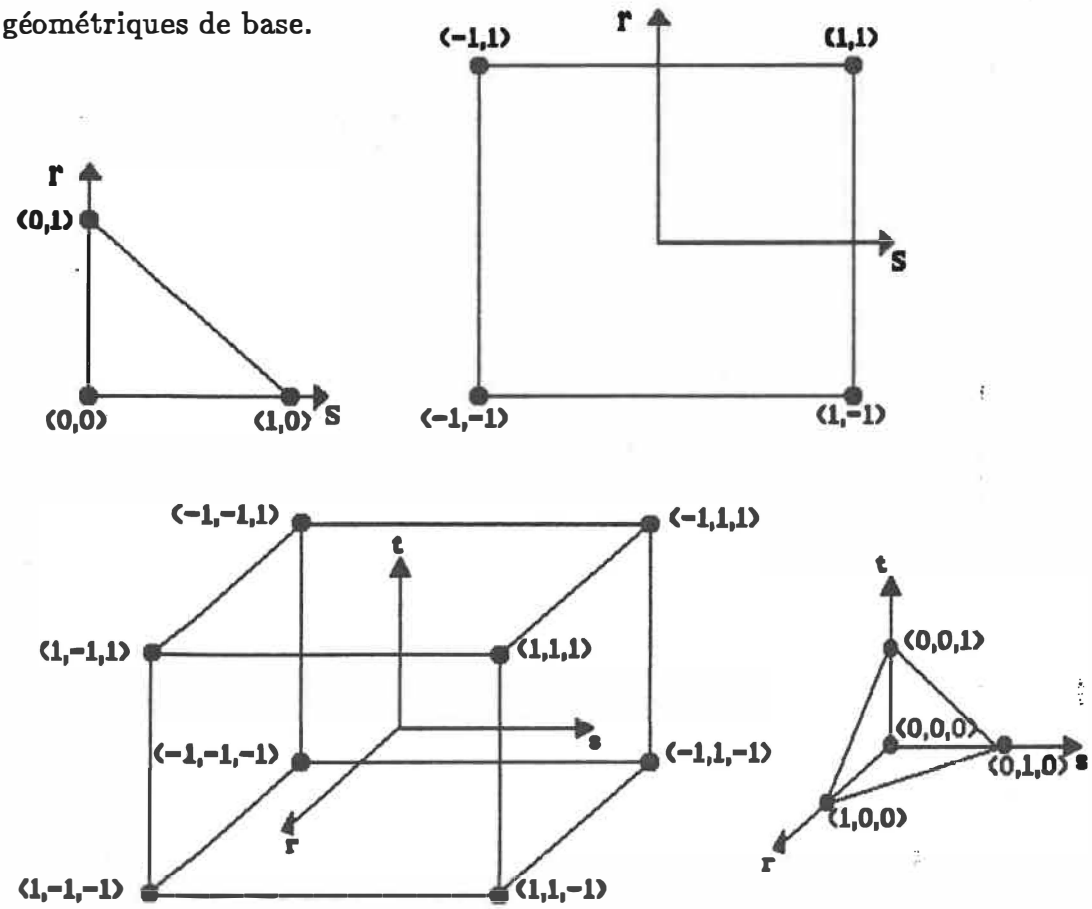


Figure 2.14: Espaces de paramétrisation des quatre formes géométiques de base

Chaque forme géométrique possède par ailleurs un numéro d'indentification

interne unique. Ce numéro est utilisé dans la définition de formes 3D à partir des formes 2D connues; chaque face d'une forme 3D peut donc être considérée comme une forme 2D et hérite automatiquement des propriétés associées à la première.

Egalement, chaque face d'une forme 3D possède automatiquement une paramétrisation locale 2D identique à celle de la forme 2D à laquelle elle correspond. De la même façon, chaque arête d'une forme géométrique 2D, tout comme une ligne en 1D, est paramétrisée de 0 à 1 indépendamment de la paramétrisation du volume ou de la face à laquelle elle appartient.

Ces considérations sont importantes pour placer des noeuds de calcul ou des points d'intégration sur l'une de ces entités. La position de ces noeuds doit être exprimée en fonction de paramètres propres à l'entité. L'approche adoptée par 'DE' consiste tout simplement à utiliser la paramétrisation définie pour la forme correspondant à l'entité sur laquelle est placé le noeud ou le point. En somme, un noeud placé au centroïde d'une face triangulaire d'un tétrahèdre est spécifié par le même ensemble de paramètres (r=1/3,s=1/3) qu'un noeud au centroïde d'un triangle en 2D. De plus, le paramètre r=0.5 indique toujours le centre d'une arête, quelle que soit la forme géométrique à laquelle appartient l'arête.

Les autres informations pertinentes à une forme géométrique utilisent directement les notions de paramétrisation dont il vient d'être question. A chaque forme dans la base de données de 'DE' correspond en effet un ensemble de configurations de noeuds de calcul et de points d'intégration compatibles avec la symétrie naturelle de la forme. Chaque configuration contient le nombre de points qui la constitue et, pour chaque point, deux ou trois paramètres (selon la dimension de la forme) donnant la position des points dans l'espace de paramétrisation de la

forme. Les configurations de points d'intégration contiennent également les poids des règles d'intégration.

Une forme géométrique, pour 'DE', constitue donc un mode de représentation totalement général de la morphologie d'un élément; ce concept permet de rendre le reste de l'application indépendant de l'espace de paramétrisation dans lequel est définie la forme et donc l'élément de référence. La forme géométrique fournit également une méthode structurée de positionnement des noeuds de calcul sur les différentes entités géométriques (sommet, arête, face, volume).

2.5.2 Les groupes de degrés de liberté

Le second objet de la structure de données interne de 'DE' est le groupe de degrés de liberté. Cet objet, bien qu'interne à 'DE', doit être connu jusqu'à un certain point de l'usager puisque c'est à ce dernier qu'incombe les responsabilités de création, de définition et de destruction des groupes.

Le concept de DDL est fondamental à la méthode des éléments finis. Par contre l'idée de regrouper plusieurs DDL pour constituer une entité supérieure ayant des propriétés distinctives est tout-à-fait nouvelle. En fait, lorsque le besoin de plusieurs ensembles de fonctions de base se fait sentir dans un même problème, les chercheurs procédent plutôt à une "superposition d'éléments de calcul", qui sont tous traités séparément à l'intérieur du code de résolution. Cette approche ne peut évidemment pas être adoptée dans le contexte de 'DE', où l'indépendance du résoluteur face à l'élément utilisé est le premier objectif. Il faut donc introduire la notion de "groupe", de manière à classifier les DDL et à leur associer certaines

propriétés, le tout à l'intérieur d'un seul et même élément.

De façon interne, un groupe de DDL est une entité relativement simple, constituée d'une liste de relations entre les variables définies pour le problème et les noeuds de calcul de l'élément. Pour chaque noeud de calcul, un groupe de DDL peut contenir soit le nom d'une variable, qui lui est alors "associée", soit une indication comme quoi aucune variable ne lui est "associée". Rappelons qu'une variable désigne soit un scalaire, donc un seul DDL, soit un vecteur ou un tenseur, ce qui peut permettre d'associer plusieurs DDL au même noeud de calcul dans un même groupe.

A cette liste d'association s'ajoute, pour chaque groupe, une règle d'intégration et un ensemble de fonctions de base; il y aura toujours dans un groupe autant de fonctions de base que de DDL.

Les groupes de DDL constituent donc une méthode de regroupement interne de l'information propre à un élément que le résoluteur pourra consulter. Le résoluteur doit donc s'attendre à rencontrer un nombre variable de groupes par élément, selon le type d'interpolant choisi par l'usager et la méthode utilisée pour organiser les DDL sur l'élément.

2.5.3 Les éléments

Le processus de description proposé par 'DE' doit ultimement produire une représentation complète d'un élément de calcul. Il va donc de soi que cet élément constitue l'objet fondamental sur lequel repose toute la structure de données de

l'application.

Au sein de 'DE', il ne peut exister qu'un seul élément à la fois, sur lequel travaille l'usager. Cet élément est toujours complet et valide, c'est-à-dire que chaque paramètre de définition de l'élément est instancié au départ à une valeur de défaut et, lorsque l'usager modifie un paramètre de l'élément, 'DE' se charge de mettre à jour tous les paramètres affectés par la modification. Par exemple, l'élément de défaut proposé par 'DE' est un triangle 2D; cependant, si l'usager décide de passer en 3D, la forme géométrique deviendra automatiquement le tétrahèdre.

Ce mécanisme de maintient interne de la cohérence présente trois avantages principaux: d'abord il élimine l'utilisation d'une étape de validation spécifique avant l'exportation des résultats de 'DE' vers les autres modules, ensuite, il facilite le dialogue avec l'usager en évitant les configurations incorrectes, et enfin, il propose une démarche structurée pour la description d'un élément complet.

Une étape de validation des données est souvent très difficile à écrire, et couvre rarement tous les cas possibles. Un exemple extrême de phase de validation serait un compilateur, qui en soi représente une application rarement simple à réaliser. Il est donc de loin préférable d'éviter l'utilisation d'un tel mécanisme à moins qu'il ne soit essentiel.

Faciliter le dialogue avec l'usager constitue l'une des pierres angulaires de la mise en oeuvre du système de résolution, toutes les façons d'atteindre cet objectif devaient donc être mises à profit. En s'assurant constamment de la cohérence de la description de l'élément, 'DE' détecte les erreurs dès leur origine et évite des pertes de temps considérables à l'usager. Cette approche permet dans la mesure

du possible d'essayer de réaliser les attentes de l'usager, en faisant "de son mieux". Par ailleurs, les messages d'erreur fournis par l'application peuvent être plus clair et plus proche du contexte dans lequel se trouve l'usager au moment où se produit l'erreur.

Dans le même ordre d'idée, le fait de proposer une démarche structurée pour la description d'un élément permet à l'usager de se familiariser plus rapidement avec l'utilisation du logiciel. En appliquant une méthodologie de description très proche de celle prévue et utilisée par le programme, l'usager apprend plus rapidement la portée de chaque action disponible, juge mieux l'influence d'une commande sur la description courante et établit plus rapidement la démarche qui lui permettra d'obtenir le résultat qu'il vise.

La démarche proposée, passant par la géométrie, l'association des variables, le choix des règles d'intégration et des interpolants, permet d'amasser l'information nécessaire pour la construction d'un élément en respectant les contraintes de dépendance entre les informations. Cette démarche évite donc à l'usager certaines surprises désagréables, où certaines de ses données seraient réinitialisées.

Il existe par ailleurs une correspondance assez étroite entre les données amassées lors du processus de description et les étapes mêmes de ce processus. L'étape de géométrie permet de choisir une forme géométrique dans la banque de données, puis de lui associer des noeuds de calcul, des contraintes, etc. L'association des variables permet de créer les groupes de DDL et de spécifier la composition de chacun. Enfin les étapes de choix des règles d'intégration et des interpolants permettent de complèter la spécification de chaque groupe de DDL, et donc celle de l'élément complet.

2.6 Accessibilité de l'information

Une fois la description de l'élément complétée, il devient important pour 'DE' de rendre cette information disponible aux autres modules du système. Dans le schéma global adopté pour le système, c'est en effet la responsabilité du module ayant une information sous son contrôle de rendre cette information disponible au reste du système.

Plusieurs méthodes de mise en commun de l'information peuvent être envisagées, et le choix d'une méthode particulière dépend de facteurs aussi variés que les temps de communication, la quantité d'information à transmettre, les conditions d'exécution des applications devant communiquer et la simplicité de programmation de la méthode.

Avant d'élaborer plus longuement sur les différentes méthodes de mise en commun de l'information, sur leurs avantages ou leurs inconvénients respectifs, il semble important d'identifier plus précisément les modules potentiellement intéressés à recevoir de l'information sous le contrôle de 'DE'. La figure 2.1, qui présentait la structure globale d'un programme d'éléments finis, établit clairement les liens qui unissent 'DE', non seulement au résoluteur, mais également aux modules de discrétisation et d'imposition des conditions aux limites. Il existe cependant, parmi ces modules, des différences fondamentales au niveau de la méthode utilisée pour leur intégration dans l'ensemble du système.

Ces différences sont exposées en détail dans le mémoire de M. Robert Magnan [2]; sans reprendre cette présentation dans son ensemble, il faut cependant relever quelques points importants: d'une part, tant le module de résolution que celui de

discrétisation sont appelés à être remplacés à volonté par les usagers du système; d'autre part, le module d'imposition des conditions aux limites fait partie intégrante du système de résolution et ne sera jamais sujet à être remplacé par les usagers.

Cette distinction importante entre les modules dits "externes" et "internes" entraîne l'utilisation de techniques de communication fort différentes d'un module à l'autre. 'DE' étant un module interne à PIRATE, au même titre que le module d'imposition des conditions aux limites, il peut partager de l'information avec ce dernier selon un schéma client/serveur, à l'intérieur d'un processus UNIX unique; les deux modules font en effet partie du même logiciel au sens informatique, et le passage d'information peut se faire au moyen d'appels à des sous-programmes de requêtes présents dans le logiciel même.

Un mécanisme plus complexe doit cependant être élaboré pour permettre à un résoluteur ou à un mailleur d'accéder aux données recueillies par 'DE'. Tout en conservant un modèle conceptuel d'interrogation de type client/serveur, un grand nombre de méthodes de mise en oeuvre peut être envisagé; en effet, il ne s'agit plus de communication à l'intérieur d'un processus (UNIX) unique, mais bien de communication inter-processus. Le système d'exploitation UNIX met à la disposition des programmeurs un jeu très complet d'outils permettant de réaliser ce type de système de communication.

Etant donné la quantité relativement faible d'information à transmettre entre 'DE' et les modules externes, le mode de communication le plus adapté aux besoins de 'DE' semble être la queue de communication. Cette méthode offre une grande flexibilité tant au niveau des types d'information qui peuvent être transmis qu'au

niveau du protocole utilisé lors de la transmission; par ailleurs, les queues de communication se prêtent facilement à la communication inter-processus au travers d'un réseau, ce qui constitue un grand avantage si l'on veut faire fonctionner 'DE' et le résoluteur sur des ordinateurs différents reliés par un réseau. L'utilisation des queues implique cependant un effort de programmation supplémentaire lors de la mise en oeuvre du résoluteur, ou du mailleur, le programme doit en effet être en mesure de formuler ses requêtes, de les coder pour la transmission par la queue, puis de décoder les réponses. Cette difficulté est cependant en grande partie éliminée par l'écriture d'un sous-programme d'interrogation standard pour tous les mailleurs ou tous les résoluteurs, fourni avec 'DE' et qui se charge de tous les aspects de communication par la queue.

Chapitre 3

Quelques aspects de l'implantation du programme 'DE'

Ce troisième chapitre est consacré à une description technique de certains aspects de la mise en oeuvre du programme 'DE'. L'étape de prototypage permet souvent de découvrir de nouvelles facettes d'un problème qui n'avaient pas été envisagées lors de sa description théorique mais qui influencent fortement le mode de fonctionnement du produit fini. Nous présentons ici les solutions apportées à certains problèmes qui se sont posés lors du prototypage.

La première section du chapitre est consacrée à quelques uns des algorithmes originaux développés pour les besoins de 'DE'. La seconde section, de son côté, présente les grandes lignes d'un outil qui doit être développé parallèlement à 'DE': le manipulateur symbolique. Cette section montrera de quelle façon cet outil peut être utilisé tant avec 'DE' qu'avec un certain nombre d'autres modules de PIRATE.

3.1 Quelques algorithmes fondamentaux à 'DE'

Cette section ne se veut pas un exposé exhaustif des algorithmes implantés dans 'DE', mais plutôt une sélection de quelques uns des algorithmes fondamentaux du logiciel. Il sera d'abord question du calcul automatique des fonctions de base, puis de l'évaluation de ces fonctions de base aux points de Gauss, et enfin de la génération automatique de code pour les fonctions de base ne pouvant être évaluées directement sur l'élément de référence.

3.1.1 Le calcul automatique des fonctions de base

Tel que mentionné au chapitre précédent, le programme 'DE' est en mesure de calculer automatiquement certaines classes d'interpolant sur les éléments décrits, lorsque ceux-ci répondent à quelques règles simples. Pour ce faire, il a fallu développer des algorithmes, dans la plupart des cas originaux, permettant de construire, à partir d'une description de l'élément, les ensembles de polynômes cherchés.

Quelques exemples simples, tirés de la pratique, serviront à illustrer les principaux mécanismes de construction des interpolants. Comme ces mécanismes varient légèrement d'une forme géométrique à l'autre, mais que la structure globale de l'algorithme reste toujours identique quelle que soit cette forme, les exemples choisis seront basés sur le triangle 2D. Les exemples porteront successivement sur chaque famille d'interpolants que permet de construire 'DE'.

Interpolants de Lagrange

La première classe d'interpolant que permet de construire 'DE' a été décrite au chapitre précédent comme la famille des interpolants de Lagrange généralisés. Cette famille n'utilise qu'un seul DDL par noeud de calcul lors de la construction d'un ensemble de fonctions de base et répète le processus de construction tant qu'il reste des DDL inutilisés dans le groupe.

Voici un algorithme schématique de la méthode de construction des interpolants de Lagrange sur un groupe de DDL, chaque étape sera explicitée par la suite:

- TANT QU'IL RESTE DES DDL INUTILISES
  - IDENTIFIER LES NOEUDS DE CALCUL OU IL RESTE UN DDL INUTILISE
  - MARQUER UN DDL PAR NOEUD COMME UTILISE
  - VERIFIER SI LE NOMBRE DE DDL A UTILISER FORME UNE BASE POLYNOMIALE COMPLETE POUR LA FERME GEOMETRIQUE
  - SI LA BASE EST COMPLETE
    * CONSTRUIRE L'ENSEMBLE DES FONCTIONS DE BASE A L'AIDE DE LA MATRICE DE VANDERMONDE
  - SI LA BASE EST INCOMPLETE
    * CONSTRUIRE UN ENSEMBLE PARTIEL DE FONCTIONS DE BASE EN CONSIDERANT UNIQUEMENT LES DDL PLACES AUX SOMMETS ET SUR LES ARETES DE L'ELEMENT (ELEMENT DE SERENDIP)
    * AJOUTER DES BULLES COMME INTERPOLANTS POUR LES DDL PLACES SUR LES FACES ET DANS LE VOLUME DE L'ELEMENT
    * MODIFIER CHAQUE FONCTION DE L'ENSEMBLE CONSTRUIT PRECEDEMENT POUR QU'ELLE S'ANNULE AUX NOEUDS SUR LES FACES ET DANS LE VOLUME

Pour chaque forme géométrique 2D ou 3D, il existe une suite infinie de bases polynômiales à deux ou trois paramètres qui permettent de construire, sur la forme, des polynômes d'approximation parcourant des espaces polynômiaux complets. Par exemple, pour un triangle, les trois premières bases peuvent s'écrire comme:

$$\{1, r, s\} \tag{3.1}$$

$$\{1, r, s, r^2, rs, s^2\} \tag{3.2}$$

$$\{1, r, s, r^2, rs, s^2, r^3, r^2s, rs^2, s^3\} \tag{3.3}$$

et permettent de construire respectivement des ensembles de polynômes d'approximation linéaire, quadratique et cubique complets. Les bases sont différentes d'une forme à l'autre, mais il existe toujours une façon systématique de les construire: ces bases sont les bases polynômiales complètes de la forme géométrique.

Lorsque 'DE' détecte que le nombre de DDL sur lesquels il construit les interpolants permet d'utiliser une base polynômiale complète pour la forme, l'algorithme de construction devient beaucoup plus simple. Dans ce cas particulier, en effet, il suffit de construire la matrice de Vandermonde de l'élément et de l'inverser pour obtenir l'ensemble des fonctions de base cherché.

Par contre, lorsque la base polynômiale n'est pas complète, 'DE' doit procéder en trois étapes: construire un ensemble partiel de fonctions de base, puis compléter cet ensemble à l'aide de fonctions bulles, et enfin modifier les premières fonctions de base pour tenir compte des bulles. L'ensemble partiel est construit en considérant uniquement les DDL placés sur les sommets et les arêtes de la forme géométrique, c'est-à-dire en utilisant les résultats obtenus pour les éléments de type "sérendip". Ce premier ensemble de fonctions de base est complété par 'DE'

à l'aide de fonctions bulles pour chacun des DDL placés sur les faces ou dans le volume de la forme. Le premier ensemble de fonctions de base est enfin modifié de façon à ce que chaque fonction de base s'annule en chaque noeud de calcul sauf un.

L'une des difficultés principales de cette approche consiste à obtenir une méthode générale pour le calcul des fonctions de base sur les éléments de type "sérendip". Pour certaines formes géométriques, il existe des expressions générales qui vont donner dans tous les cas la base polynômiale réduite à utiliser; par contre, pour d'autres formes, et c'est le cas du triangle, il ne semble pas exister d'expression de ce type. Pour contourner ce problème, 'DE' utilise une approche relativement souple, qui permet à un usager ayant des besoins particuliers auxquels 'DE' ne sait pas répondre, de founir l'information nécessaire au programme pour que le calcul puisse s'effectuer.

Les quelques exemples suivants illustrent différents aspects de l'algorithme de construction des fonctions de base, puis montrent une circonstance dans laquelle l'usager doit fournir de l'information supplémentaire au programme.

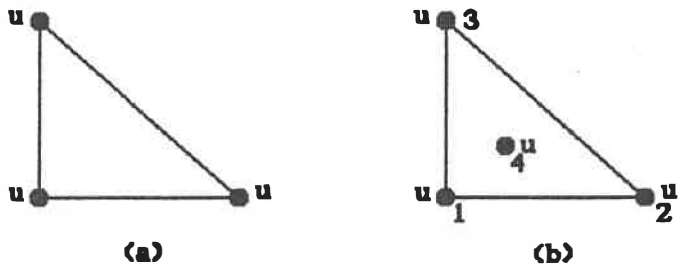


Figure 3.1: 'Eléments utilisés pour la construction d'interpolants de Lagrange: exemples simples

EXEMPLE 3.1 La figure 3.1(a) illustre l'élément triangulaire le plus simple sur lequel 'DE' peut construire un ensemble d'interpolants de Lagrange; il est clair d'après l'algorithme présenté plus haut que cette configuration de DDL permet d'utiliser la première base polynômiale complète pour la forme, soit:

$$\{1, r, s\}$$

puis de construire la matrice de Vandermonde et son inverse:

$$V = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad V^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

Chaque colonne de la matrice peut s'interpréter comme les coefficients des monômes d'une fonction de base, soit:

$$\begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} = \begin{bmatrix} 1 & r & s \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1-r-s \\ r \\ s \end{bmatrix}$$

Cette méthode de calcul s'applique donc directement à toutes les configurations de DDL permettant d'utiliser une base polynômiale complète. Le problème se complique légèrement lorsque la base à utiliser n'est plus complète, comme dans le cas illustré à la figure 3.1(b). Dans un tel cas, l'algorithme prévoit trois étapes. D'abord, il faut construire un ensemble partiel de fonctions qui ne tient compte que des DDL placés aux sommets ou sur les arêtes de la forme. Dans cet exemple, cette construction est strictement équivalente au calcul précédent et donne les trois fonctions de base N_1, N_2, N_3.

La seconde étape consiste à compléter cet ensemble en utilisant des fonctions bulles pour chaque DDL sur une face ou dans le volume de la forme. Ici, il n'y a qu'un DDL sur la face, et la bulle la plus simple pour compléter l'ensemble sera:

$$B_1 = 27 \quad r \quad s \quad (1 - r - s)$$

Finalement, il faut modifier les trois premières fonctions de base pour qu'elles s'annulent au noeud de calcul où est définie la nouvelle fonction de base. Le moyen le plus simple consiste à retrancher de chacune des fonctions la fraction de B_1 qui les annulera au noeud de calcul.

B_1 vaut évidemment 1 au quatrième noeud de calcul, alors que les trois autres fonctions de base valent toutes 1/3; il suffit donc de soustraire 1/3 B_1 de N_1, N_2 et N_3 pour former un ensemble complet de fonctions de base ayant les bonnes propriétés pour les fins d'un calcul par éléments finis:

$$N_1 = 1 - r - s - 9\,(rs - r^2 s - rs^2)$$

$$N_2 = r - 9\,(rs - r^2 s - rs^2)$$

$$N_3 = s - 9\,(rs - r^2 s - rs^2)$$

$$B_1 = 27\, r\, s\, (1 - r - s)$$

Le cas présenté ici se traite de façon relativement simple, car il est facile de construire l'ensemble partiel des fonctions de base pour l'élément sans DDL sur les faces ou dans le volume. Il suffit en effet d'utiliser une base polynômiale complète pour la forme. Ce n'est cependant pas toujours le cas, comme le montre l'exemple suivant.

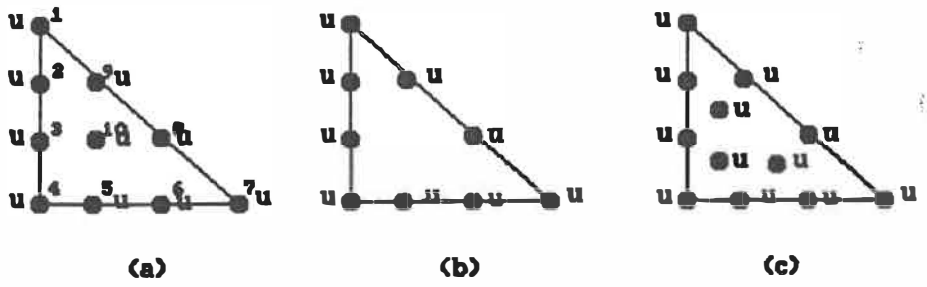


Figure 3.2: Eléments utilisés pour la construction d'interpolants de Lagrange: exemple plus complexe

EXEMPLE 3.2 La troisième base polynômiale complète pour le triangle contient 10 monômes et la configuration standard de noeuds de calcul utilisant cette base est présentée à la figure 3.2(a). Le point important à noter dans cette configuration est l'apparition d'un noeud de calcul, et donc d'un DDL, sur la face du triangle. Le problème que pose ce DDL est le suivant: pour construire un ensemble partiel de fonctions de base utilisant les neuf premiers noeuds de l'élément 3.2(a) (figure 3.2(b)), 'DE' ne peut plus employer une base complète.

En fait, constuire un ensemble de fonctions de base sur l'élément 3.2(b) implique d'éliminer un DDL dans la base polynômiale elle-même. Le problème réside dans le fait qu'il ne semble pas exister de méthode systématique pour procéder à cette élimination de DDL. La littérature documente abondamment des éléments comme celui de la figure 3.2(b), mais personne ne semble mentionner de méthode systématique pour construire une base polynômiale réduite permettant d'obtenir des fonctions de base sur un tel élément.

L'approche utilisée par 'DE' dans ce cas consiste à éliminer les DDL en trop à l'aide de contraintes internes à la base. 'DE' doit donc accumuler des ensembles de

contraintes qui lui permettront par la suite d'éliminer des DDL si nécessaire. Il est cependant possible que 'DE' ne connaisse pas les contraintes nécessaires à l'élimination des DDL dans certains cas très complexes. Le programme demandera donc à l'usager de lui fournir les contraintes appropriées. Voyons dans le cas présenté ci-haut le type de contraintes nécessaires à 'DE' et leur mode d'utilisation concret.

Dans le cas de l'élément de la figure 3.2(b), la base complète comporte dix monômes, alors que l'élément ne compte que neuf degrés de liberté. Il faut donc, pour construire une "pseudo" matrice de Vandermonde, une dixième ligne qui n'est pas fournie par la position des noeuds de calcul: c'est cette dernière ligne de la matrice qui constitue la contrainte interne à la base polynômiale.

Dans le cas qui nous occupe, la dernière ligne de la matrice, qui devrait contenir les coefficients obtenus à l'aide de la position du noeud central de l'élément, est remplacé par une ligne complètement différente, en l'occurence:

$$\begin{bmatrix} & & & & & & & & & \\ & & & & & & & & & \\ & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0.5 & 0.5 & -1 \end{bmatrix}$$

qui en fait équivaut à imposer la contrainte suivante à la base polynômiale:

$$-r^3 + \frac{r^2 s}{2} + \frac{r s^2}{2} - s^3 = 0 \qquad (3.4)$$

L'utilisation de cette contrainte pour le cas particulier du triangle à neuf noeuds donnera très précisément les fonctions de base généralement trouvées pour cet élément dans la littérature. Comme c'était le cas dans l'exemple précédent, il suffit

d'inverser la matrice calculée pour obtenir les neufs fonctions de base. Il est évident que cet algorithme est excellent dans la mesure où les contraintes nécessaires sont connues. A ce point-ci aucun processus ne semble cependant permettre d'obtenir automatiquement ces contraintes, qui, jusqu'à un certain point, semblent choisies arbitrairement. C'est dans le cas bien précis, et relativement improbable, où 'DE' ne connaîtrait pas l'ensemble de contraintes à utiliser pour éliminer les DDL en trop que celui-ci serait forcé d'interroger l'usager afin d'obtenir l'information manquante pour terminer son travail. Cette situation ne devrait toutefois se présenter que très rarement, puisqu'elle implique l'utilisation par l'usager d'éléments sur lesquels sont définis de très grands nombres de DDL (pour le triangle, par exemple, le problème ne se présente que pour les éléments possédant plus de dix DDL dans le même groupe).

Une fois l'ensemble partiel de fonctions de base trouvé, le travail de 'DE' se poursuit normalement. Dans le cas de la figure 3.2(c), 'DE' complète l'ensemble des fonctions de base par trois fonctions bulles, définies chacune en un noeud différent. Par la suite, 'DE' modifie les neufs fonctions de base calculées préalablement pour que chacune s'annule aux trois noeuds de calcul de la face.

Interpolants d'Hermite

L'algorithme de construction des interpolants d'Hermite étant pratiquement identique à celui utilisé pour les interpolants de Lagrange, il paraît inutile de reprendre l'ensemble de la discussion précédente dans ce nouveau contexte.

Qu'il suffise de mentionner que les interpolants, du point de vue de 'DE', sont

construits strictement de la même façon, c'est à dire en évaluant une matrice, qui n'est cependant plus celle de Vandermonde, puis en inversant cette matrice. La différence la plus importante se situe en fait au niveau de l'évaluation des termes de la base polynômiale au moment de la construction de la matrice. Il est en effet nécessaire, avant d'évaluer les termes pour les DDL associés aux dérivées, de dériver chaque terme de la base polynômiale.

Les considérations présentées plus haut sur les bases incomplètes s'appliquent également ici: lorsque le nombre de DDL à approximer ne permet pas d'utiliser une base polynômiale complète, 'DE' élimine des DDL dans la base en utilisant des contraintes.

Interpolants en développement en série de Taylor

La troisième classe d'interpolants supportée par 'DE', les interpolants en développement en série de Taylor, est produite selon un algorithme beaucoup plus simple que celui utilisé pour les deux classes précédentes. Cette famille d'interpolants utilise en effet un seul noeud de calcul à la fois, auquel sont associés un certain nombre de DDL. La construction des interpolants pour ce type de configuration repose en fait essentiellement sur une convention simple et sur quelques informations founies par l'usager au moment du calcul.

L'approximation d'une fonction P autour d'un noeud de calcul (x_0, y_0) étant de la forme:

$$P = P(x_0, y_0) + (x - x_0)\frac{\partial P}{\partial x} + (y - y_0)\frac{\partial P}{\partial y} + \ldots$$

'DE' a besoin de trois informations provenant de l'usager:

- le nombre de paramètres par rapport auquel 'DE' doit dériver,
- l'ordre maximal des dérivées à utiliser dans l'approximation,
- le référentiel dans lequel sont exprimées les coordonnées du noeud de calcul (local à l'élément ou global).

'DE' peut produire des fonctions de base utilisant jusqu'à trois paramètres (r, s et t ou x, y et z) dans les dérivées. Evidemment, il n'est pas possible d'utiliser moins de paramètres que la dimension de l'élément, par contre, en 1D et 2D, il est pensable d'utiliser plus de paramètres que la dimension de l'élément si le besoin s'en fait sentir. L'usager peut également spécifier l'ordre de la dérivée maximale qui sera utilisée dans le développement en série. Cet ordre est le même pour tous les paramètres et permet de choisir la qualité de l'approximation et le nombre de DDL qui seront impliqués dans cette approximation.

Finalement, l'usager doit choisir le référentiel par rapport auquel seront exprimées les coordonnées du noeud de calcul (x_0, y_0, z_0). Dans bien des cas, il n'y a pas de différence, au niveau de l'approximation, entre l'utilisation des coordonnées locales et globales. Dans cette situation, il est alors préférable d'utiliser les coordonnées locales, qui permettent de construire immédiatement et une fois pour toute une table des valeurs des fonctions de base évaluées aux points d'intégration. Par contre, certains types d'approximation nécessitent absolument l'emploi des coordonnées globales; c'est le cas par exemple d'un élément 1D ou 2D utilisé avec plus de paramètres que la dimension de l'élément.

Lors de la génération des fonctions de base, 'DE' utilise une convention très simple qui permet immédiatement de savoir à quel DDL est associée une fonction de base en particulier. La première fonction de base est toujours 1, puis 'DE' construit les fonctions de base en utilisant successivement tous les paramètres spécifiés pour un ordre de dérivée donné. Dans ce contexte, les fonctions de base générées pour une expansion à trois paramètres jusqu'à l'ordre 2 des dérivées seraient dans l'ordre:

$$1$$

$$(x - x_0)$$

$$(y - y_0)$$

$$(z - z_0)$$

$$(x - x_0)^2/2$$

$$(y - y_0)^2/2$$

$$(z - z_0)^2/2$$

La variable associée au noeud de calcul pour lequel cet ensemble de fonctions de base est généré devrait donc contenir au moins sept DDL.

Fonctions bulles

Les bulles sont le dernier type d'interpolant qu'il est possible de construire automatiquement à l'aide de 'DE'. Ces fonctions sont associées de façon individuelle aux DDL et 'DE' permet de les construire selon des spécifications relativement précises de l'usager.

Par définition, une bulle s'annule sur les frontières de l'élément auquel elle est associée. Il est par ailleurs souvent intéressant de construire des bulles qui non seulement s'annulent aux frontières, mais qui s'annulent également à certains noeuds internes de l'élément. 'DE' permet de construire de telles fonctions d'interpolation selon une mécanique relativement simple. A nouveau, la technique sera démontrée pour un triangle en 2 dimensions, mais celle-ci peut être transposée directement à n'importe quelle autre forme 2D ou 3D.

La technique habituelle de construction des bulles consiste à multiplier des termes s'annulant chacun sur une frontière. Par exemple, pour le triangle standard, les trois termes suivants sont utilisés:

$$r$$

$$s$$

$$1 - r - s$$

En multipliant ensemble ces termes, on obtient la bulle bien connue:

$$B = rs - r^2 s - rs^2$$

Cette technique a été étendue au sein de 'DE' pour permettre d'annuler une fonction ailleurs qu'aux frontières de l'élément. Par exemple, pour le triangle de la figure 3.3, il est avantageux de pouvoir annuler la fonction d'interpolation à deux noeuds internes de l'élément.

Si l'on veut construire une bulle pour le noeud de calcul numéro 1 de la figure 3.3(a), il faut pouvoir annuler la fonction aux deux autres noeuds. Pour ce faire, un moyen simple consiste à multiplier la bulle normale par un terme supplémentaire

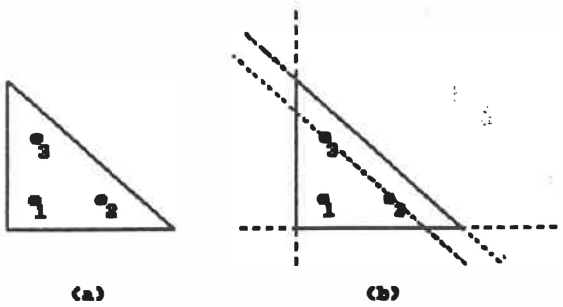


Figure 3.3: Construction de bulles ne s'annulant pas uniquement sur les frontières de l'élément

qui s'annulera aux noeuds 2 et 3. Le terme le plus simple possèdant cette propriété est bien entendu:

$$0.8 - r - s$$

qui une fois multiplié aux autres termes de la bulle normale nous donne la fonction:

$$B = rs(1 - r - s)(0.8 - r - s)$$

Cette approche consiste en fait à annuler la fonction sur une nouvelle ligne de l'élément, qui n'est plus une frontière. Cette approche s'étend d'ailleurs directement en 3D par l'utilisation de plans plutôt que de lignes.

'DE' permet donc de construire un ensemble de termes aussi complexes que nécessaire, qui permettra à la fonction cherchée de s'annuler exactement aux endroits voulus.

L'usager possède par ailleurs un autre niveau de contrôle pour la création des fonctions. Chaque terme constituant la bulle peut, avant d'être multiplié aux autres, être élevé à une puissance positive choisie par l'usager. Cette caractéristique supplémentaire permet de créer pour un même noeud de calcul différentes fonctions linéairement indépendantes, qui pourront toutes être utilisées simultanément au

moment de la résolution pour enrichir l'approximation.

3.1.2 Évaluation des fonctions de base aux points de Gauss

Il a souvent été mentionné dans le présent mémoire que certaines fonctions de base pouvaient être évaluées une fois pour toute au moment de la description de l'élément, ce qui permettait de transmettre au programme de résolution non pas une fonction mais bien une table de valeurs. Cette section-ci montrera de quelle façon les fonctions de base sont représentées au sein de 'DE' et comment ces fonctions peuvent être évaluées afin de produire une table de valeur; la section suivante, de son côté, expliquera la procédure employée par 'DE' lorsqu'il n'est pas possible de construire une telle table de valeur.

De façon interne, 'DE' conserve toutes les fonctions de base sous forme de polynômes à un, deux ou trois paramètres. En une dimension, 'DE' conserve tout simplement le nombre et la liste des coefficients des puissances du paramètre en ordre croissant:

$$a_0 + a_1 r + a_2 r^2 + ...$$

D'une façon similaire, en deux dimensions, un polynôme est stocké comme une matrice carrée NxN des coefficients des puissances de chaque paramètre de 0 à N-1, le polynôme:

$$\begin{array}{ccccc} a_{00} & + & a_{01}r & + & a_{02}r^2 & + \\ a_{10}s & + & a_{11}rs & + & a_{12}r^2 s & + \\ a_{20}s^2 & + & a_{21}rs^2 & + & a_{22}r^2 s^2 & \end{array}$$

sera donc stocké dans la matrice A_{ij}, et de même en trois dimensions, le tenseur A_{ijk} contiendra un polynôme à trois paramètres r,s et t. Cette représentation se

prête bien tant aux manipulations d'addition, de soustraction, de multiplication, qu'à la dérivation par rapport à une variable ou à l'évaluation en un point donné. C'est donc une tâche relativement simple que de construire des tables de valeurs pour chaque fonction de base et ses dérivées par rapport à chaque paramètre en chaque point d'intégration.

3.1.3 Génération de code

Dans certains cas particuliers, les fonctions de base ne peuvent pas être évaluées au moment de l'étape de description de l'élément, et 'DE' se voit alors forcé de générer non pas une table de valeurs, mais bien une fonction 'C' qui sera exécutée au moment de la résolution.

Une telle situation se présente entre autre lorsque l'usager choisit d'utiliser des fonctions de base produites par un développement en série de Taylor autour d'un noeud de calcul de l'élément, et que ce développement n'utilise pas les coordonnées locales du noeud sur l'élément, mais bien les coordonnées géométriques globales. Dans ce cas, les fonctions de base dépendent des coordonnées réelles (dans le maillage) des noeuds de calcul et elles doivent donc être évaluées de façon individuelle pour chaque élément en fonction de sa position.

Pour répondre à un tel besoin, 'DE' est en mesure de générer du code, c'est-à-dire d'écrire un sous-programme, qui une fois appelé par le résoluteur calculera la valeur d'une fonction de base ou de l'une de ses dérivées en un point donné de l'élément. L'appel aux fonctions générées par 'DE' a été standardisé, et permet au résoluteur de rester tout aussi indépendant de l'élément que si la table des

valeurs avaient été générée. En fait, le résoluteur peut procéder par requêtes à 'DE' comme s'il s'agissait d'une fonction de base standard, en lui fournissant les données nécessaires au calcul, ou bien il peut inclure directement la fonction écrite par 'DE' et l'appeler après avoir procéder à l'édition des liens.

3.2 La manipulation symbolique

Suite au regain de popularité qu'a connu le domaine de l'intelligence artificielle et de l'utilisation d'ordinateurs en logique, certaines vieilles idées refont surface. C'est le cas par exemple de l'idée d'utiliser l'ordinateur dans des tâches de manipulations algébriques et symboliques. Bien que cette idée soit caressée par les chercheurs depuis de nombreuses années, il est relativement récent que les ordinateurs dont ils disposent puissent leur fournir la puissance de calcul et la mémoire nécessaires à ce type d'opérations.

L'idée de base de la manipulation symbolique est relativement simple, il s'agit en fait d'utiliser l'ordinateur non pas comme une calculatrice, qui ne traite que des nombres et les opérations algébriques de base, mais plutôt comme un outil de manipulation de symboles mathématiques, utilisant un ensemble beaucoup plus vaste d'opérations. Les premiers concepts nouveaux inclus dans un manipulateur symbolique seront des distinctions entre nombres entiers, rationnels et irrationnels, de même que le concept de variable (d'inconnue) et les différentes définitions que prend une même opération sur chacune de ces entités.

Utilisé en mode interactif, un manipulateur symbolique devrait par exemple

être en mesure de simplifier une expression algébrique complexe, d'isoler une variable ou de substituer une valeur pour une variable dans une expression. Au-delà de ces opérations rudimentaires, cet outil, pour être vraiment utile, doit pouvoir manipuler des entités plus complexes telles des matrices, ou des équations implicites et pouvoir intégrer et différentier au niveau symbolique.

Au sein de PIRATE, un manipulateur symbolique peut rendre des services à tous les niveaux. En fait, cet outil peut constituer une couche "intelligente" de traitement de l'information qui filtre en quelque sorte les données provenant de l'usager avant de les transmettre à l'un des modules. Au niveau le plus simple, le manipulateur peut permettre à l'usager de stocker des valeurs fréquemment utilisées dans des variables, de manipuler ces variables et de les utiliser en guise de réponse à certaines questions de l'application. Ce type d'utilisation ne constitue cependant qu'une infime portion de ce qui peut être réalisé avec un tel outil; une intégration beaucoup plus étroite du manipulateur à certains modules du système permet d'offrir à l'usager des services d'entrée de données extrêmement naturels. Trois domaines se distinguent particulièrement à ce niveau:

- la définition des contraintes géométriques
- la définition de la forme faible
- la modification des fonctions de base

Voici, dans chacun de ces domaines, comment une utilisation adéquate d'un manipulateur symbolique peut améliorer de façon considérable les capacités du module en charge de l'interaction.

3.2.1 Les contraintes géométriques

La description géométrique d'un élément peut comprendre un ensemble de contraintes sur la forme géométrique elle-même. Ces contraintes prennent la forme de relations mathématiques entre différentes entités définissant la forme, telles la position des sommets, les angles entre les arêtes, les hauteurs, etc...

Pour permettre de définir ces contraintes de façon aussi naturelle que possible, 'DE' doit fournir à l'usager le moyen d'établir des relations très générales entre les entités de base constituant la forme. Avec l'aide d'un manipulateur symbolique, 'DE' peut automatiquement définir des identificateurs pour la position de chacun des sommets et pour chacun des angles de la forme. Ces identificateurs peuvent par la suite être utilisés par l'usager pour construire des relations qui seront conservées puis vérifiées par le système de discrétisation.

Le manipulateur symbolique permet d'associer un ensemble de relations à un élément, qui seront tenues pour vraies par le système et qui seront vérifiées à l'étape de discrétisation. En cas de violations des relations par certains éléments réels, le système devra prévoir des mécanismes qui permettront de corriger ou tout au moins de quantifier les erreurs commises lors de l'étape de discrétisation.

3.2.2 La définition de la forme faible

Bien que l'état actuel de PIRATE ne permette pas à l'usager de définir une nouvelle forme faible pour fins de résolution, il est tout à fait envisageable qu'à moyen terme ce service soit offert par le système. De toute évidence la mise en

oeuvre de ce module de description sera basée sur le manipulateur symbolique, qui permettra encore une fois de construire des ensembles de relations entre les variables définies pour le problème.

Cet aspect important de la description d'un problème justifie en fait à lui seul l'élaboration d'un manipulateur symbolique complet. Ce manipulateur devra pouvoir travailler non seulement sur des variables, mais permettre la définition d'opérateurs et fournir l'ensemble des services de manipulation algébrique, d'intégration et de dérivation que requiert un mathématicien dans l'exercice de son travail.

3.2.3 La modification des fonctions de base

Un dernier exemple d'application qui tirerait un grand bénéfice de l'utilisation d'un manipulateur symbolique au sein de PIRATE consisterait à permettre à l'usager de modifier les fonctions de base calculées automatiquement par 'DE' pour un élément donné.

'DE' pourrait en effet fournir un utilitaire d'édition symbolique des fonctions de base qui présenterait à l'usager les fonctions calculées automatiquement d'après les propriétés de l'élément et qui permettrait l'édition de certaines des fonctions avant l'étape de résolution.

Conclusion

Nous avons présenté une méthode générale de description des éléments utilisés pour des fins de résolution par éléments finis. Cette description a par ailleurs fait l'objet d'une première mise en oeuvre informatique au sein de l'application 'DE', qui s'inscrit dans un projet plus large de résolution intégrée de problèmes numériques.

Cette description couvre non seulement les aspects géométriques des éléments mais également les questions concernant l'association des degrés de liberté à l'élément de même que les notions d'intégration numérique et de fonctions d'interpolation propre aux éléments de calcul. L'application 'DE' permet d'ailleurs de construire une très grande variété de fonctions d'interpolation à l'aide d'algorithmes originaux présentés dans ce mémoire.

L'ensemble de l'information amassée lors d'un processus complet de description semble suffisamment générale pour permettre la représentation de tous les éléments présentement connus et utilisés couramment en élément fini. En ce sens, cette description devrait permettre la production d'une application stable sous peu.

M. André Garon a par ailleurs démontré qu'il est possible de constuire un

résoluteur acceptant une description générique des éléments de résolution. Ce type de résoluteur pourra donc interfacer directement avec 'DE'; il ne reste plus qu'à établir un protocole de communication entre les deux types d'application pour que les résoluteurs puissent disposer de toute la flexibilité que procure 'DE'.

Les champs d'application de 'DE' ne se limitent par ailleurs pas uniquement aux éléments finis: il est en effet tout-à-fait possible d'envisager utiliser 'DE' dans un contexte de volume fini, dans lequel seules les propriétés géométriques et la connectivité des inconnues seraient utilisées. La flexibilité du logiciel pourra d'ailleurs se mesurer non seulement à la facilité qu'auront les usagers à créer de nouveaux éléments dans un contexte d'éléments finis, mais également à adapter 'DE' à des problèmes numériques relevant d'autres domaines.

Bibliographie

1. REDDY, J.N., "An Introduction to the Finite Element Method", McGraw-Hill, 1984, 495 pages.

2. MAGNAN, R. "Élaboration de l'architecture d'un logiciel pour la résolution de problèmes aux dérivées partielles", Mémoire de maîtrise, Ecole Polytechnique de Montréal, 1990.

3. CROSS, M. et al., "Engineering CFD Software for the Next Decade", dans Computational Fluid Dynamics, G. de Vahl Davis et C. Fletcher (Ed.), North-Holland, 1988, page 31.

4. DHATT, G. et TOUZOT, G., "Une présentation de la méthode des éléments finis", Les presses de l'université Laval, Québec, 1981, 543 p.

5. PONCET, A., "Autour de l'écriture d'un code d'éléments finis", Thèse de doctorat présentée à l'université scientifique et médicale de Grenoble, mars 1979., 474 p.

6. SHILOV, G., "An introduction to the theory of linear spaces", Dover Publications, New York, 1974, 310 p.

7. CHLOE 3.4 - Manuel de l'usager, Groupe MIAO, mai 1989.

8. CHLOE IV - Manuel de l'usager, Groupe MIAO, à paraître.

ÉCOLE POLYTECHNIQUE DE MONTRÉAL
3 9334 00247669 3