Modélisation et paramétrisation d'objets naturels de formes complexes en trois dimensions : application à la simulation stochastique de la distribution d'hétérogénéités au sein des réservoirs pétroliers

Liliane Wietzerbin

Institut
National
Polytechnique
de Lorraine

Ecole Nationale Supérieure de Géologie de Nancy
(E.N.S.G. - L.I.A.D.)
Centre de Recherche Informatique de Nancy
(C.R.I.N.)

THESE

présentée devant l'Institut National Polytechnique de Lorraine
en vue de l'obtention du titre de

Docteur de l'I.N.P.L.

Informatique

Par

Liliane WIETZERBIN

Modélisation et paramétrisation d'objets naturels de formes complexes en trois dimensions.

Application à la simulation stochastique de la distribution d'hétérogénéités au sein des réservoirs pétroliers

Soutenue le 5 mars 1994, devant le Jury composé de :

André	JOURNEL	Président et Rapporteur
Didier	ARQUEZ	Rapporteur
René	SCHOTT	Rapporteur
	DUBRULE	Examinateur
	HAAS	Examinateur
	MALLET	Examinateur

Institut
National
Polytechnique
de Lorraine

Ecole Nationale Supérieure de Géologie de Nancy
(E.N.S.G. - L.I.A.D.)
Centre de Recherche Informatique de Nancy
(C.R.I.N.)

THESE

présentée devant l'Institut National Polytechnique de Lorraine
en vue de l'obtention du titre de

Docteur de l'I.N.P.L.

Informatique

Par

Liliane WIETZERBIN

Modélisation et paramétrisation d'objets naturels de formes complexes en trois dimensions.

Application à la simulation stochastique de la distribution d'hétérogénéités au sein des réservoirs pétroliers

Soutenue le 5 mars 1994, devant le Jury composé de :

André	JOURNEL	Président et Rapporteur
Didier	ARQUEZ	Rapporteur
René	SCHOTT	Rapporteur
Olivier	DUBRULE	Examinateur
André	HAAS	Examinateur
Jean-Laurent	MALLET	Examinateur

A Philippe,
dont la compréhension, le soutien constant et l'infinie patience ont permis l'aboutissement des travaux...

Remerciements

C'est en tout premier lieu au professeur Jean Laurent Mallet, directeur du Laboratoire d'Informatique et d'Analyse des Données, que je voudrais exprimer ma reconnaissance. Il m'a non seulement accueillie avec bienveillance au sein du groupe GOCAD, mais il m'a également encouragée et soutenue tout au long des travaux. Il a surtout éclairé de ses conseils techniques les voies qui paraissaient si obscures au début de cette thèse.

Mes remerciements vont également à la société Elf Aquitaine Production, et plus particulèrement à Messieurs André Haas, Olivier Dubrule et François Alabert. Ils m'ont aidée à pénétrer dans le domaine de la géologie et à comprendre quels étaient les défis à relever.

Je tiens tout particulièrement à remercier le Professeur André Journel pour m'avoir si gentiment reçue au "Stanford Center for Reservoir Forecasting". Il m'a permis de mieux comprendre ce qu'était la géostatistique et de trouver un moyen de l'appliquer dans le cadre de mes travaux. En ce sens, je voudrais dire ma gratitude à son élève, Clayton Deutsch, pour m'avoir aidé et éclairé dans mes recherches.

Je remercie également Messieurs René Schott et Didier Arquèz de l'honneur qu'ils me font en acceptant d'être rapporteurs de cette thèse.

Je dis aussi un grand Merci à Toute l'équipe du liad pour m'avoir assistée et (bien) entourée durant près de trois années. Madame Cugurno a été une secrétaire dont l'efficacité s'est révélée précieuse dans bien des circonstances. Merci plus particulièrement à Pascal pour avoir si patiemment relu ce manuscrit.

J'adresse enfin mes remerciements aux sponsors du projet GOCAD dont la plupart ont participé activement aux recherches en cours. Parmi eux, j'aimerais plus particulièrement exprimer ma gratitude à J.C Dulac, John Gibson et David Goggin.

Résumé

Le but des travaux effectués au cours de cette thèse est de proposer une méthode générale pour simuler, en trois dimensions, la géométrie et la distribution des hétérogénéités au sein de réservoirs pétroliers naturels.

Dans un premier temps, nous proposons un nouvel outil de modélisation d'un objet naturel de forme complexe et limitant un volume fermé dans l'espace. Il s'agit de l'objet *gshape* qui se définit par une ligne conductrice, le *backbone* et par un ensemble de sections planaires le long de cette ligne, les *sections*. Leur interpolation indépendante conduit à la définition d'une *enveloppe* caractérisant la géométrie d'un objet *gshape*. Cela permet une modélisation globale de celle-ci et introduit une grande flexibilité dans la manière de paramétriser et de déformer une forme complexe.

Dans un second temps, nous avons utilisé l'objet *gshape* pour la représentation d'une hétérogénéité. Chacune d'elle est considérée comme un objet de type *gshape* dont la géométrie doit satisfaire un certain nombre de contraintes. La distribution des hétérogénéités au sein du réservoir est ensuite simulée de façon stochastique et de manière à satisfaire les données disponibles. L'algorithme de simulation proposé est basé sur le principe des méthodes de recuit et intègre le formalisme de la géostatistique non paramétrique.

L'étude de trois cas réels permet la validation des méthodes développées et met en évidence les avantages qu'elles offrent par rapport à des méthodes de simulations booléennes classiques.

Abstract

This work proposes a new general method for simulating the geometry and the distribution of 3D reservoir heterogeneities. We first present the *gshape* object which introduces a new way of caracterizing the complex shape of a 3D natural object enclosing a portion of space. A *gshape* is defined through a *backbone* and a set of planar sections along the *backbone*. Both *sections* and *backbone* are interpolated through an independant process, leading to a great deal of flexibility for parameterizing and modifying the *skin* of a *gshape* object in a global way.

We then show how we use this new formalism for the caracterization of reservoir heterogeneities distribution. A single heterogeneity is considered as a *gshape* type object which geometry must honor some specified constraints The distribution of the heterogeneities is then modeled using a stochastic simulation method based on Simulated Annealing and Non parametric Geostatistics. The process accounts for both hard and soft data. The application to three real cases allows for the validation of our method and stresses its advantages compared to traditional boolean techniques.

Table des matières

Introduction

0.1 Présentation de la thèse

Dans le domaine pétrolier, on qualifie d'hétérogénéité un corps géologique de taille relativement réduite (i.e par rapport au réservoir qui le contient), dont le materiau constitutif est de nature homogène mais différente de celui composant son environnement. Ce sont en partie–on le pressent–des considérations d'ordre pratique qui fondent la notion d'hétérogénéité. A cet égard, sera considéré comme une hétérogénéité un corps susceptible de modifier de façon significative l'écoulement des hydrocarbures. Ainsi, certains de ces corps favoriseront leur écoulement, alors que d'autres, au contraire, lui feront barrage. Dans les deux cas, il convient d'identifier ces hétérogénéités au plus tôt afin de prévoir les régimes d'écoulement et d'optimiser le choix des emplacements des nouveaux forages. La modélisation des hétérogénéités de réservoir est ainsi un des problèmes clé posés à l'industrie pétrolière et il est crucial de posséder des méthodes fiables pour caractériser la forme et la localisation des hétérogénéités au sein du réservoir.

En fait, on parlera plus de "prédiction" que de caractérisation. En effet, les données disponibles pour caractériser les corps géologiques sont en général peu nombreuses et souvent, incertaines. On emploie donc fréquemment des méthodes de type probabiliste pour les modéliser. Le travail de recherche présenté ici a pour but de caractériser les hétérogénéités de réservoir en ce qui concerne:

1. La géométrie de leur enveloppe externe. Il s'agit ici d'un problème de modélisation surfacique.

2. Leur répartition et leur disposition relative au sein du réservoir. Il s'agit de déterminer la géométrie dans l'espace de chacune des surfaces modélisées.

Notre recherche ne concernera que les réservoirs dont la forme des hétérogénéités peut être globalement connue. Cette connaissance globale est souvent à notre portée lorsque des phénomènes sédimentaires, par ailleurs bien étudiés, sont en jeu (sédimentation deltaïque, fluviatile...). Notre objectif est de caractériser le réservoir, aussi bien dans ses aspects déterministes, lorsque les données sont suffisamment nombreuses, que dans ses aspects probabilistes, lorsqu'une connaissance trop limitée ne permet pas de caractériser les hétérogénéités qu'il contient. Cette caractérisation concernant essentiellement des objets tridimensionels, nous aurons besoin d'un modeleur géométrique appliqué aux problèmes géologiques. Dans cette optique, GOCAD ([3]) a été proposé parce qu'il comporte à

la fois des capacités de modélisation en trois dimensions de surfaces géologiques ([8], [9],[23],[44],[47],[59]), et des possibilités de contraindre ces surfaces à des données que l'on peut pondérer et pouvant être de natures très différentes ([43]).

0.2 Modèles stochastiques: définition

Le terme "stochastique" s'applique à un phénomène qui peut être traduit par un ensemble de solutions équiprobables ([25]). Ainsi, une modélisation stochastique a pour but de générer le plus large éventail possible de solutions honorant les hypothèses de départ; en ceci, c'est une façon de prendre en compte le manque de données ([25],[34],[36]). Les méthodes de simulation stochastique s'opposent essentiellement aux méthodes d'interpolation qui conduisent toujours à une solution unique.

Le traitement d'une simulation stochastique se fait généralement en comparant les différentes réalisations. L'incertitude est directement liée aux différences observées entre les réalisations: forte lorsque les écarts sont importants, faibles lorsqu'ils sont minimes.

L'utilisation de méthodes stochastiques est de plus en plus répandue en Géologie, où les données sont peu nombreuses et souvent imprécises, en particulier celles relatives à la forme et à la disposition relative des hétérogénéités de réservoir.

0.3 Quelques termes géologiques

Les forages et la connaissance générale du contexte géologique permettent de déterminer les familles de corps présentes dans la zone à étudier. On classera par exemple les hétérogénéités correspondant à des corps allongés, issus du dépôt de sédiments dans une rivière, dans la famille des "chenaux", celles correspondant à des corps plus ramassés dans la famille des "lobes" ou des "lentilles".

Dans la suite de ce manuscrit, nous utiliserons les termes suivants:

1. Nous appelerons "réservoir" une portion du sous-sol géologique limitée dans l'espace et supposée contenir des hydrocarbures. Dans le cadre de notre étude, nous ne nous intéresserons qu'à des réservoirs contenant des hétérogénéités dont la géométrie est globalement connue.

2. Le terme de "faciès" sera appliqué à une unité géologique composée de matériel homogène. Lorsque cette unité est composée d'hétérogénéités, nous dirons qu'un faciès est une "famille d'hétérogénéités". Dans le modèle de réservoir présenté dans la figure 0.1, les familles d'hétérogénéités sont les faciès *chenaux*, *lobes* et *slump*.

3. Nous appelerons "matrice" ou "encaissant" le matériel n'appartenant à aucune hétérogénéité du réservoir. La matrice du réservoir est en fait un faciès particulier ne contenant pas d'hétérogénéités. Dans la Figure 0.1 le faciès *matrice* est présent dans tout le réservoir et constitue son faciès majoritaire. Il est composé des matériaux *argile, laminé sableux* et *laminé argileux*.

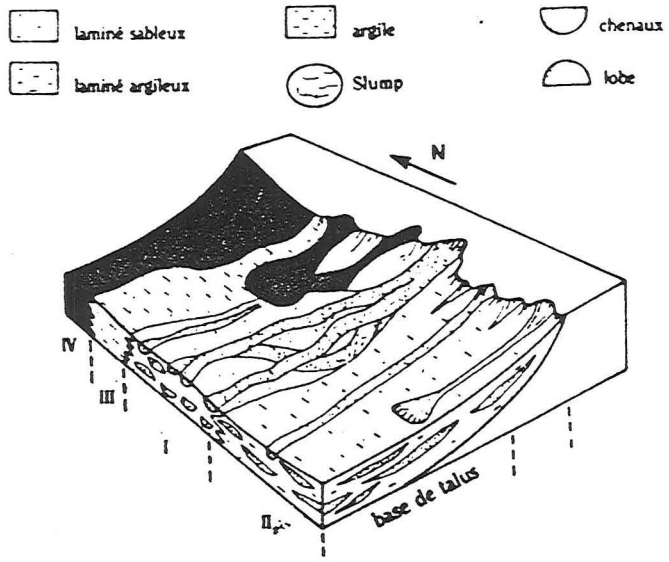


Figure 0.1 Modèle de réservoir composé des faciès *chenaux*, *lobes*, *slump*, *matrice*. Zone I et II: *chenaux*, *lobes* et *matrice*; Zone III et zone IV: *matrice* et *slump*.

4. Nous appelerons "puits", une portion du sous sol le long d'un forage. On suppose que les faciès observés le long de ce forage sont connus (voir Figure 0.2, page 4).

5. Nous dirons que deux puits sont "corrélés" pour un faciès lorsqu'ils intersectent tous deux un corps appartenant à ce faciès (voir Figure 0.3, page 5). On suppose que le faciès concerné est une "famille d'hétérogénéités".

0.4 Description des données

Les données dont on dispose pour modéliser la distribution des hétérogénéités de réservoir ont été répertoriées ci-dessous. Elles peuvent être regroupées dans les deux catégories suivantes:

1. **Les données dites "dures"**, considérées comme certaines. Dans le domaine de la géologie elles correspondent essentiellement aux données de puits. Ces puits permettent de disposer des faciès localisés dans le sous-sol à un intervalle de profondeur donné. Certains de ces faciès correspondent à des hétérogénéités, et c'est à eux que nous nous intéresserons.

2. **Les données dites "molles" ou "floues"**, considérées comme incertaines. Il s'agit souvent de données portant sur la géométrie des corps géologiques. On pourra

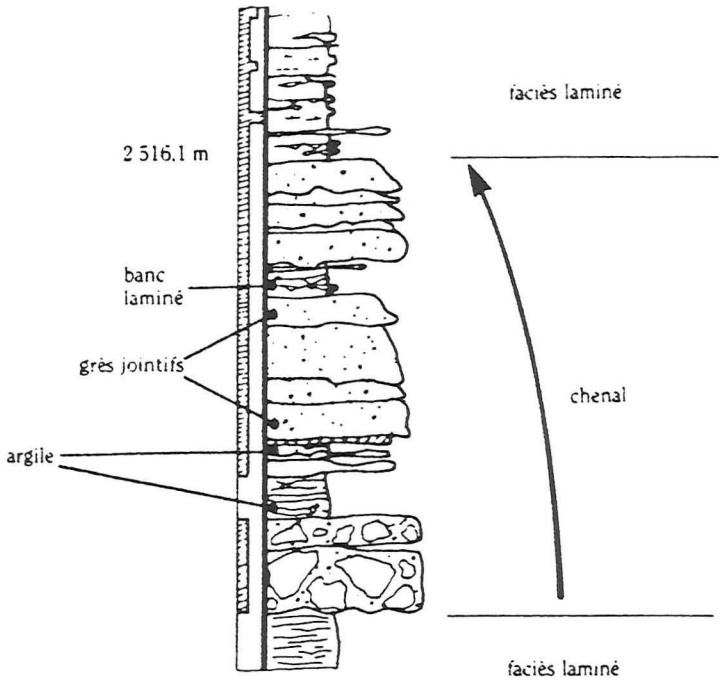


Figure 0.2 Exemple de forage; le faciès *matrice* est constitué du materiau laminé.

ainsi posséder des distributions de tailles, de directions d'élongation, de proportions de familles d'hétérogénéités dans le réservoir.

En ce qui concerne les données utiles à la description de la géométrie et de la distribution des hétérogénéités, nous avons répertorié les données dures et molles de la manière suivante:

1. Données dures

   (a) Différents faciès en présence; ces faciès résultent des observations faites aux puits.

   (b) Absence ou présence d'un faciès au niveau d'un puits;

   (c) Corrélation ou non corrélation entre puits pour un faciès particulier (voir Figure 0.4 et Figure 0.3);

2. Données molles

   (a) **Distribution de tailles** de corps dans chaque faciès; il est parfois possible d'avoir une idée des dimensions (épaisseur et largeur) des hétérogénéités appartenant à un faciès donné. On peut alors les quantifier à l'aide de fonctions de distribution.

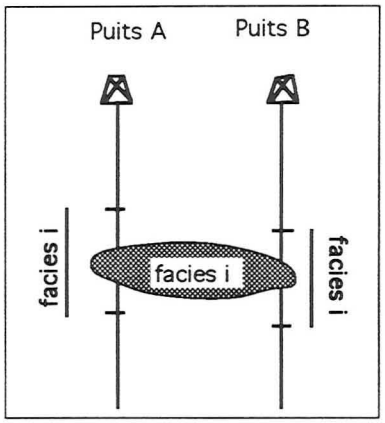


Figure 0.3 Exemple de corrélation entre les puits A et B pour le faciès i.

(b) **Distribution de directions** préférentielles d'allongement des corps à l'intérieur d'un même faciès; les corps géologiques sont souvent allongés suivant une direction privilégiée pour un faciès donné. Il est alors possible de caractériser cette direction à l'aide d'une fonction de distribution.

(c) **Proportions volumiques relatives** des faciès dans le reservoir; ces proportions, souvent issues d' observations au niveau des puits, sont dans la plupart des cas considérées comme incertaines.

(d) **Forme géométrique plus ou moins définie** pour un corps appartenant à un faciès donné; il est parfois possible d'imaginer une forme type d'hétérogénéité pour un faciès donné. Comme nous l'avons dit plus haut, nous considérons que c'est toujours le cas ici.

(e) **Probabilités de corrélation** entre puits pour un faciès particulier; il est difficile d'affirmer que deux puits sont ou ne sont pas corrélés pour un faciès donné. Il peut être alors intéressant d'exprimer une probabilité de corrélation (voir Figure 0.5, page 6).

(f) **Probabilités de répulsion (attraction)** entre corps appartenant ou non au même faciès; on observe parfois que certains types de corps ne se rencontrent qu'en l'absence ou la présence d'autres corps. Il peut s'avérer utile de modéliser ce phénomène.

Il est d'autre part important de noter que nombre de données utilisées dans le domaine de la Géologie sont par nature intuitives, floues ou inquantifiables. Parmi celles-ci, on peut citer les décisions de corrélations entre puits qui demandent une synthèse des données provenant de différentes disciplines (Sismique, Diagraphie, Sédimentologie ...). A cet égard, la définition d'une forme "type" pour une famille d'hétérogénéité relève également de l'imagination et de l'intuition. Notre objectif est de pouvoir prendre en compte cet "Art" du géologue dans la définition des données molles.

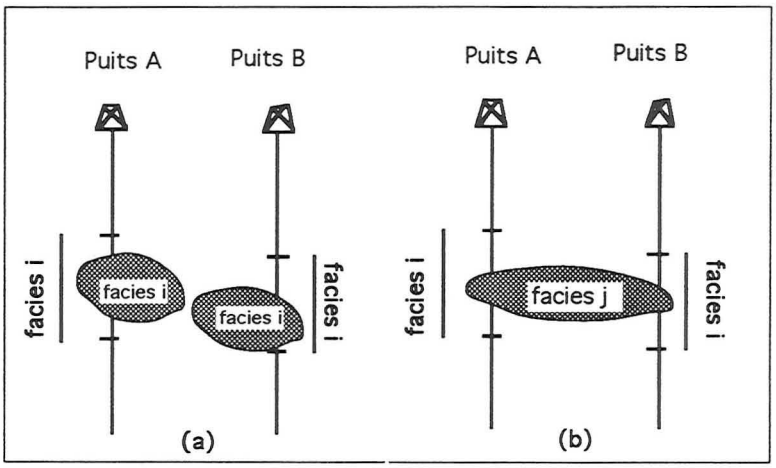


Figure 0.4 Exemples de non corrélation entre les puits A et B, pour le faciès i. (a) aucun des 2 corps de facies i n'intersecte les deux puits. (b) Un même corps intersecte les deux puits, mais il appartient au faciès j.

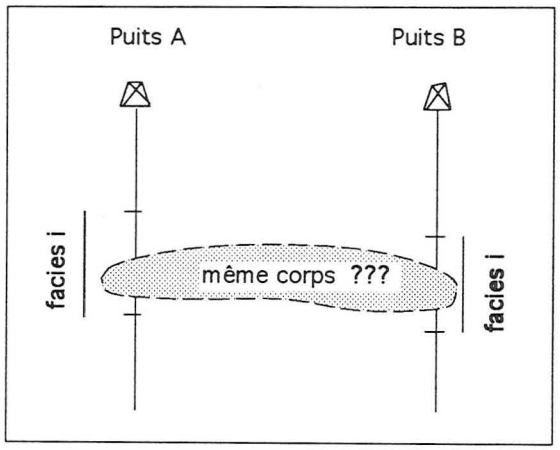


Figure 0.5 Notion de probabilité de corrélation entre les deux puits A et B, pour le faciès i.

0.5 Objectifs

Les objectifs à atteindre sont les suivants:

1. Modéliser **la géométrie** de chaque hétérogénéité au sein du réservoir; ce sont ses contours exacts, en trois dimensions, qu'il convient de caractériser: pour chaque hétérogénéité, il s'agit de définir la surface matérialisant son contour.

2. Obtenir un premier modèle de **la répartition** des hétérogénéités dans le réservoir. Il s'agit de positionner dans l'espace chacune des surfaces précédemment définie.

3. **Conditionner le modèle** aux données "dures" et "molles". Il s'agit de parvenir à une distribution des hétérogénéités dans le réservoir qui respecte au mieux les données. Cette étape garantit la validité du modèle, puisqu'elle signifie que celui-ci honore les hypothèses de départ et **peut** être une représentation de la réalité.

0.6 Brève description bibliographique

Nous reviendrons sur les méthodes stochastiques de modélisation d'hétérogénéités au cours du chapitre 4. Cependant, il est important pour la compréhension de notre démarche de citer les grandes familles de méthodes utilisées jusqu'à présent pour modéliser la distribution des hétérogénéités dans un réservoir pétrolier naturel. Il s'agit des méthodes dites "pixel" d'une part et des méthodes dites "booléennes" d'autre part ([14], [25]).

0.6.1 Les méthodes pixels

Elles constituent la grande majorité des méthodes stochastiques utilisées jusqu'à présent dans le domaine du réservoir pétrolier. Les méthodes pixels appréhendent l'espace comme un ensemble de points. En ces points sont attachées des variables discrètes (indicateur de faciès par exemple) ou continues (perméabilité, porosité ...) ([25],[34],[36]). Elles nécessitent également que l'on possède des données concernant la façon dont les variables attachées en deux de ces points sont liées ([34]). Une modélisation stochastique basée sur une approche pixel se déroule en deux phases:

1. Détermination de la fonction f reliant la valeur de la variable en un point à la valeur de la variable en un point distant d'un vecteur h du précédent[1]. L'expression de la fonction f est généralement issue:
   (a) de modèles mathématiques théoriques;
   (b) de réservoirs de référence dont on estime que les statistiques sont proches de celles du réservoir à étudier. On considère alors que la fonction f utilisée pour le cas de référence peut être appliquée au modèle à étudier.
2. Modélisation par simulation conditionnelle aux données: on respecte alors la valeur de la variable aux points où elle est mesurée. La simulation respecte en général également la fonction f.

Les applications trouvées dans la littérature ([1],[5],[29],[35],[36]) montrent que les méthodes "pixels" sont parfaitement appropriées lorsque l'on s'intéresse à la modélisation de plusieurs attributs en un point, et lorsque l'on dispose de suffisamment de données pour en déduire l'expression de f. Cependant, de part leur nature–elles s'attachent à modéliser des valeurs ponctuelles d'attributs–elles identifient mal les contours des corps. Ces méthodes s'appliquent à un milieu "continu" sans se soucier de discontinuités liées aux frontières géométriques. Ainsi, pour un même jeu de données, les résultats sont identiques que l'on s'intéresse à des données géologiques, médicales ou atmosphériques ....

[1] voir chapitre 4

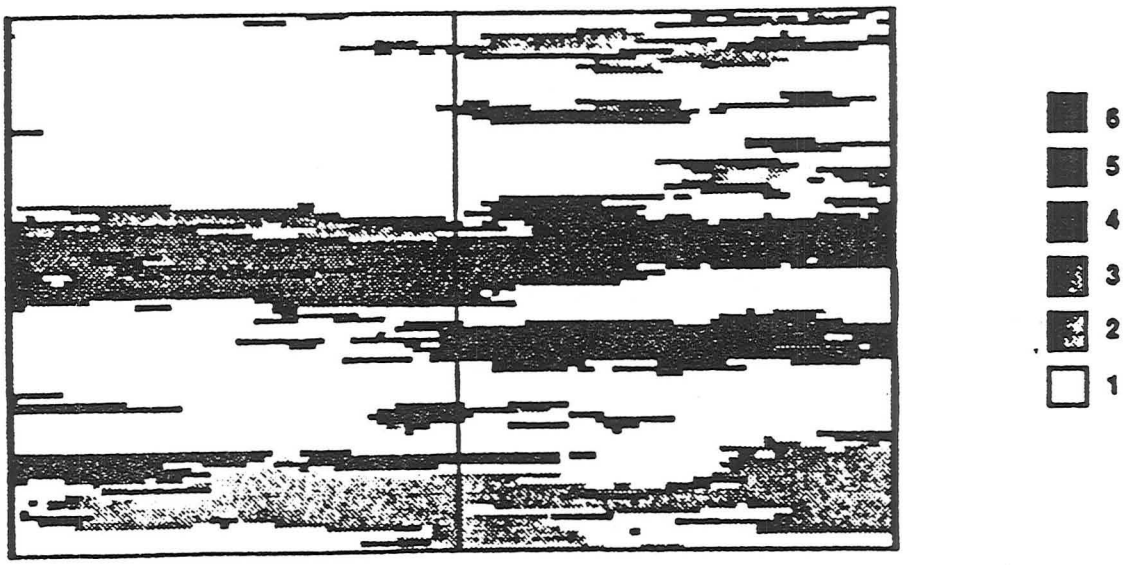


Figure 0.6 Exemple de simulation basée sur une approche pixel.

La Figure 0.6, extraite de [56] est un exemple de simulation stochastique utilisant une approche pixel. On a simulé la distribution d'une variable discrète qui mesure l'appartenance d'un point du modèle à un des six faciès du réservoir. Il apparaît très délicat de distinguer des limites de corps bien définies. En effet, s'il est aisé de mettre en évidence une "zone" où un faciès est plus représenté qu'ailleurs, il devient pratiquement impossible de mettre en évidence des objets distincts dans ce faciès. Il est a fortiori difficile d'en extraire leurs contours. Néanmoins, beaucoup de techniques développées dans le cadre de ces méthodes peuvent être considérées de manière plus générale et être appliquées à la modélisation d'objets. Cela sera développé plus largement au chapitre 4.

0.6.2 Les méthodes booléennes (objet)

Elles considèrent que chacune des unités géologiques présente dans le réservoir possède une forme connue et suffisamment simple pour pouvoir être paramétrisée ([27]). On considère que l'on possède des données statistiques sur les distributions de tailles et de directions d'allongement de ces corps. On connaît également la proportion volumique de chaque faciès dans le réservoir. Une simulation stochastique basée sur une approche booléenne repose sur les étapes générales suivantes ([25],[27],[32]):

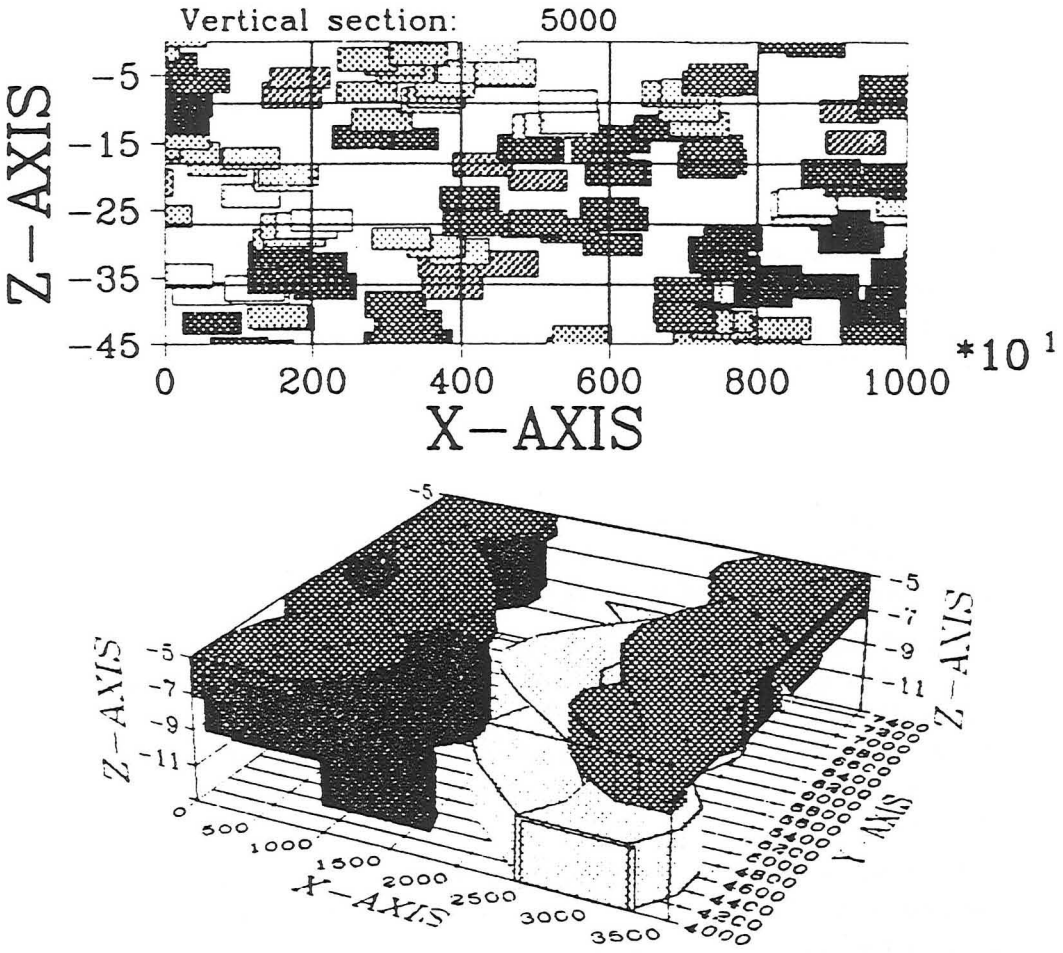


Figure 0.7 Exemple de simulation de faciès basée sur une approche booléenne

1. Choix d'une forme de base par famille d'hétérogénéités au sein d'une librairie d'objets de géométrie prédéfinie. Chacune de ces formes représente la "forme type" d'une hétérogénéité élémentaire au sein d'un faciès donné.

2. Génération d'un certain nombre d'objets élémentaires par famille d'hétérogénéités à partir de la forme de base qui lui correspond. La géométrie des objets générés peut varier au sein du faciès (cela s'effectue généralement par une transformation aléatoire simple de la forme type).

3. Localisation des objets en fonction des données à respecter. Cette étape est généralement assez complexe et est le sujet de nombreuses publications ([27],[32]).

Toute méthode booléènne demande donc, en entrée, qu'on lui définisse quelles sont les formes géométriques de base qui doivent être considérées comme typiques des différents faciès en présence. Ainsi, Haldorsen ([26]) extrait la géométrie des objets type assignés à chaque faciès d'une librairie de formes analytiques simples (parallélépipèdes, formes allongées régulièrement sinueuses). Il dispose également de données empiriques sur les règles de distribution de tailles et d'orientation des hétérogénéités en fonction de leur mode de formation. Les hétérogénéités sont ensuite générées suivant ces règles dans chaque faciès et disposées de manière aléatoire dans le réservoir jusqu'à ce que les proportions volumiques relatives soient respectées. Leur géométrie est définie par une transformation aléatoire des paramètres analytiques des formes géométriqes de base suivant des lois de tirage au hasard données. Le conditionnement aux puits n'est pas assuré.

Pour modéliser la géométrie d'hétérogénéités de réservoir, Henriquez et al ([27]), utilisent des parallélépipèdes dont l'épaisseur et la largeur sont générées suivant une loi de distribution donnée. Cette méthode honore les faciès observés aux puits mais ne prend pas en compte les possibilités de corrélation entre puits.

Damsleth([7]) utilise des boîtes rectangulaires ou des ellipsoides associés à un grand choix de lois de distributions différentes qui décrivent leurs paramètres de dimensions, d'orientation et d'inclinaison. Dans chaque faciès, les objets sont générés suivant ces lois. Le conditionnement des faciès aux puits est assuré et il est possible de spécifier de façon déterministe des corrélations entre puits. MacDonald([32]) utilise le même type d'approche.

Clementsen et al ([4]) considèrent des formes de base de section rectangulaire dont la sinuosité et la taille de la section sont spécifiées par l'utilisateur. Dans chaque faciès, les corps sont générés par une transformation aléatoire de la taille des sections suivant une loi de distribution donnée. Il incorpore à la simulation des règles d'attraction/répulsion entre corps, les corps au sein d'un même faciès s'attirant, ceux appartenant à des faciès différents se repoussant. Le respect des faciès observés aux puits est assuré. Les recherches futures devraient s'attacher à incorporer des possibilités de corrélations entre puits.

Suro-Perez ([53]) utilise des formes paramétrisables, dont la taille dépend d'une distribution donnée par l'utilisateur. Il joue sur la valeur des paramètres pour les ajuster aux puits. Le conditionnement aux puits est assuré mais il n'est pas possible de tenir compte de contraintes de corrélations entre puits.

Ces applications montrent que les méthodes "booléennes" sont appropriées pour caractériser les limites exactes de chaque hétérogénéité. Ainsi, sur l'exemple de simulation booléenne montré à la Figure 0.7 et extraite de [4], la géométrie de chaque corps apparaît clairement. Cependant, le conditionnement aux données est rendu difficile par le fait que les formes élémentaires disponibles sont souvent simples (ellipsoides, boîtes rectangulaires) ou exprimées par des fonctions mathématiques de base ([14], [27],[32]). D'autre part, beaucoup de ces méthodes ([4],[27]) sont fonction du type de dépôt modélisé ([10]). Celui-ci influera sur la librairie de formes mise en place, et sur d'éventuelles règles de répulsion/attraction.

0.7 Analyse

Il semble qu'une approche possible de modélisation d'hétérogénéités de réservoir soit la suivante:

1. Utiliser une approche objet (booléenne) pour modéliser la géométrie de chaque hétérogénéité au sein du réservoir.

2. Considérer ensuite "l'intérieur" de chaque hétérogénéité comme un milieu continu et utiliser une approche pixel pour modéliser la distribution de variables continues (perméabilité par exemple) dans cette hétérogénéité.

Cependant, notre centre d'intérêt a clairement été défini comme la caractérisation du contour exact de chacune des hétérogénéités présente dans le réservoir. C'est donc le premier des deux volets précédents qui est concerné et qui sera traité ici. L'aspect de la modélisation de réservoir consistant à simuler la valeur de variables continues à l'intérieur d'une hétérogénéité dont la géométrie est déjà définie, sort du cadre de cette thèse. Cependant, des solutions possibles seront évoquées dans la partie "Conclusion" de ce manuscrit.

Par ailleurs, en ce qui concerne la modélisation géométrique des hétérogénéités, il paraît logique de nous placer d'emblée dans le cadre d'une approche booléenne de modélisation du réservoir. Dans cette optique, il semble que le développement d'une "bonne" méthode ([14],[27], [32]) passe par le traitement des trois aspects distincts suivants:

1. Définition interactive d'un objet type pour chaque famille d'hétérogénéités. Si l'on souhaite que notre méthode sorte du cadre des méthodes booléennes classiques où la forme de ces objets est (trop?) simple[2], se posent ici les problèmes suivants:

   (a) La modélisation et la paramétrisation d'une surface décrivant le contour en trois dimensions de cet objet.

   (b) L'intégration dans le processus de modélisation de l'intuition et de l'imagination de l'utilisateur.

   Cet aspect du problème est en fait un problème de modélisation de surfaces particulières matérialisant chacune le contour d'une hétérogénéité.

2. On suppose ici que chaque faciès est caractérisé par un objet type, issu de la modélisation précédente. Il faut définir alors les relations entre cet objet et les autres objets contenus dans le réservoir. Cela nécessite:

   (a) La définition du lien de parenté entre un objet caractéristique d'un faciès et les objets appartenant à ce faciès.

   (b) La définition d'un lien génétique, permettant de passer de l'objet type (le parent), à l'objet fils (un des objets représentant une hétérogénéité dans le faciès considéré).

[2] voir section 0.6.2

3. La définition de la modélisation stochastique elle-même. Il s'agit de proposer une méthode mathématique permettant de générer et de positionner dans l'espace du réservoir les objets représentant des hétérogénéités de manière à honorer au mieux les données. Nous nous attacherons en particulier au respect des données de corrélation entre puits.

0.8 Organisation du manuscrit

Le chapitre 1 sera consacré à la présentation d'un outil utilisé pour la modélisation surfacique d'objets naturels de formes complexes limitant un volume fermé dans l'espace. Il s'agit de l'objet *gshape*[3] qui introduit une nouvelle façon de modéliser et de paramétriser des formes complexes en trois dimensions.

Au cours du chapitre 2, nous présenterons une panoplie d'outils développés pour initialiser et modéliser interactivement un objet *gshape*. Nous verrons également comment on peut matérialiser la surface définissant son contour et le volume limité par cette surface.

Le chapitre 3 montrera comment l'objet *gshape*, jusqu'ici général, peut être utilisé pour définir une famille de corps au sein d'un réservoir pétrolier. On y introduira l'objet *rshape*[4], extension du *gshape* à la représentation plus spécifique d'une hétérogénéité faisant partie intégrante d'un modèle de réservoir.

Le chapitre 4 présentera d'abord les concepts, issus des approches pixels, dont nous nous sommes inspirés pour mettre au point une méthode booléenne originale de simulation stochastique. Nous décrirons ensuite cette méthode dans le détail et verrons quels sont les avantages qu'elle présente par rapport aux approches objet classiques.

Enfin, le chapitre 5 sera dédié à l'application des méthodes présentées aux chapitres précédents à trois cas réels. Des études de rapidité de convergence seront également présentées.

Nous tirerons ensuite les conclusions de notre travail et discuterons de son intérêt. Nous dirons également quelles directions de recherche nous envisageons pour l'avenir.

[3] *gshape* signifie *g*éométrical *shape* object
[4] *rshape* signifie *r*éservoir *shape* object

Chapitre 1

Paramétrisation de formes en trois dimensions: l'objet GSHAPE

1.1 Présentation du chapitre

Nous nous éloignerons ici du problème spécifique de la modélisation d'hétérogénéités de réservoir pour aborder celui, plus général, de l'interpolation et de la paramétrisation en trois dimensions d'un objet de forme complexe et limitant un volume fermé dans l'espace. Celui-ci sera représenté par l'objet *gshape*. Nous présenterons dans un premier temps les raisons qui ont conduit à la création de l'objet *gshape*. Nous décrirons ensuite la méthode d'interpolation utilisée pour modéliser et paramétriser cet objet.

1.2 Introduction

Une des limitations inhérentes aux simulations booléennes est la trop grande simplicité des formes élémentaires choisies initialement pour représenter les différentes familles d'objets[1]. Du point de vue mathématique, elles ont généralement une représentation analytique simple et des variations aléatoires de tailles. De plus, il n'y a pratiquement jamais d'intervention extérieure au niveau de l'élaboration des formes de base. Celles-ci sont souvent issues de bibliothèques de formes déjà disponibles, et préprogrammées ([14],[27],[32]). Nous nous proposons de développer un outil intéractif qui soit en mesure de permettre:

1. La saisie en trois dimensions de la surface que l'on imagine comme la plus représentative du contour du corps que l'on désire modéliser. Cet outil doit être simple et suffisamment souple pour permettre la prise en compte de l'intuition du géologue et le manque de données quantifiées quant à la forme à créer.

[1] ceci a été développé à la section 0.6.2

2. La paramétrisation de cette surface de façon à pouvoir modifier la forme saisie à tout moment et de manière simple et naturelle. Celle-ci est en partie subjective et il faudra qu'elle puisse évoluer pour honorer de nouvelles données.

Dans tous les cas, il est important que les paramètres mathématiques qui contrôleront la géométrie de l'objet soient facilement et rapidement modifiables et qu'ils aient une signification "naturelle". La Figure 0.1, donne un aperçu des géométries que l'on veut pouvoir modéliser. Les faciès *chenaux*, *lobes* et *slump* sont composés d'objets de formes compliquées, souvent assymétriques, et difficilement paramétrisables à l'aide de fonctions mathématiques simples. Il est également difficile de les décrire à partir d'un ensemble discret de points de l'espace. Elles ont cependant l'aspect commun suivant:

Elles limitent dans l'espace un volume fermé.

1.3 Méthodes d'interpolation existantes

1.3.1 Méthodes classiques

Différentes méthodes permettent de modéliser interactivement des objets complexes en trois dimensions. Les méthodes classiques de CAO[2], basées sur les nurbs ([50]) ou les interpolations polynomiales de type B_spline ou patches de Bézier ([42], [21], [19], [49]) permettent d'interpoler en trois dimensions un objet dont certains "points de contrôle" sont connus. Ces méthodes sont basées sur une représentation paramétrique polynomiale de la surface modélisée. Une surface de Bézier $B(u,v)$, par exemple, est définie par une série de points de contrôle $\{p_{ij}\}$ formant un réseau rectangulaire $\{p_{11}, p_{12}, \cdots, p_{mn}\}$ (voir Figure 1.1, page 15) tel que:

$$B(u,v) = \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}.B_{i,m}(u).B_{j,n}(v); u,v \in [0,1]$$

avec:

$$\left| \begin{array}{ll} B_{i,m}(u) & = \frac{m!}{i!(m-i)!}u^i(1-u)^{m-1} \\ B_{j,n}(v) & = \frac{m!}{j!(m-j)!}v^j(1-v)^{n-1} \end{array} \right.$$

Les bases mathématiques des méthodes de type NURBS ou B-spline sont proches de celles de Bézier, et leurs caractéristiques sont les mêmes. Ces méthodes produisent des surfaces lisses dont le maillage est régulier et les propriétés mathématiques séduisantes (elles sont ainsi plusieurs fois dérivables). La façon dont les positions des points de contrôle influent sur la position de ceux définissant la surface est résumée dans la Figure 1.2. Pour des raisons de simplification, nous décrivons ici la manipulation d'une ligne dans un espace à trois dimensions mais le principe est le même pour une surface. On observe que:

[2] CAO signifie Conception Assistée par Ordinateur

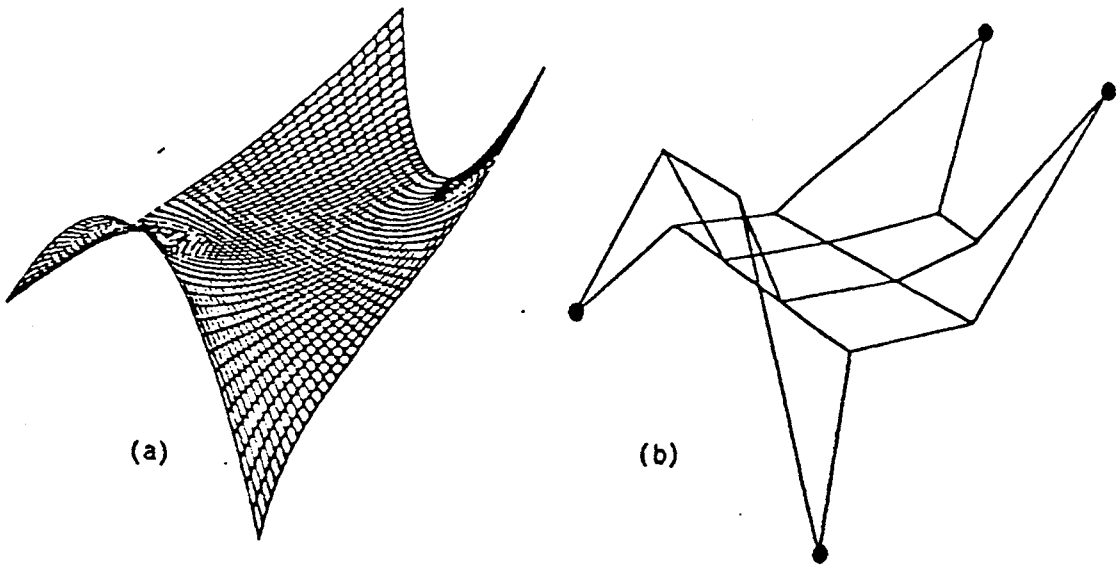


Figure 1.1 (b) points de contrôle p_{ij}, (a) surface de Bézier associée.

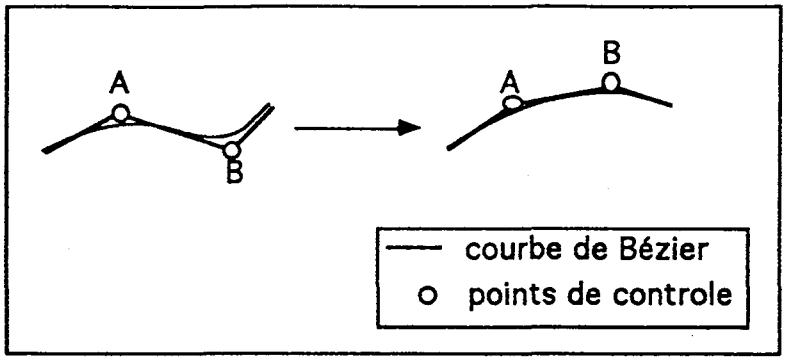


Figure 1.2 manipulation d'une courbe de Bézier.

1. Les points de contrôle n'appartiennent pas à la surface elle-même; la forme souhaitée est obtenue en anticipant l'effet sur la surface d'une modification de leurs positions. Ceci ne pose pas de problème dans la grande majorité des domaines où sont utilisées les surfaces de Bézier. Ainsi, dans l'industrie automobile, la forme exacte de la surface que l'on désire obtenir est connue par avance et il est possible d'en déduire la position des points de contrôle. En revanche, dans le domaine de la géologie, on désire que les (rares) points de données que l'on possède appartiennent à la surface modélisée et fassent office de points de contrôle. C'est ainsi que mise en œuvre dans le cadre de la géologie, une approche de type Bézier, se heurte aux problèmes suivants:

   (a) La surface à interpoler n'étant connue que par la position de points irréguliè-

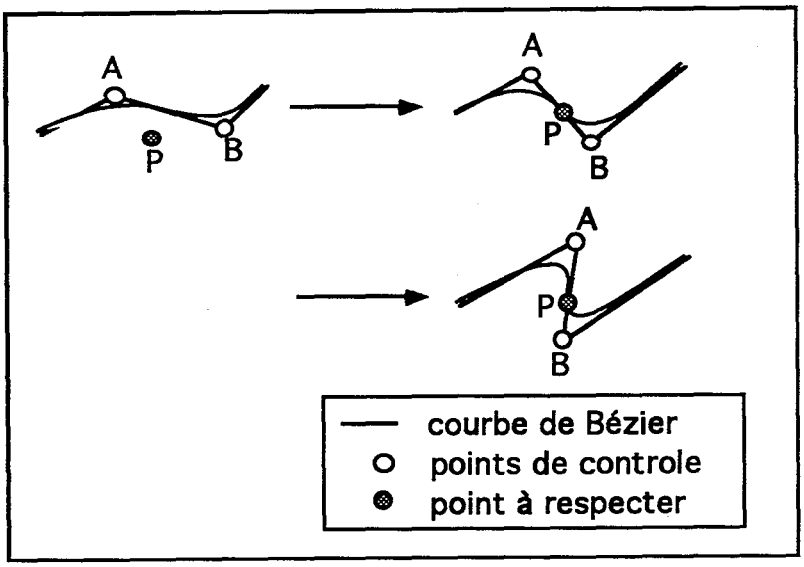


Figure 1.3 ajustement d'une courbe de Bézier à un point.

rement échantillonnés $\{q_1, q_2, \cdots, q_m\}$, leur position doit être fonction de celles des points connus.

$$\{p_{ij} = f(q_1, q_2, \cdots, q_m)\}$$

(b) Il n'y a en général pas unicité de la solution. En effet, si l'on considère la Figure 1.3, on remarque que plusieurs combinaisons des positions des points de contrôle respectent la position du point P.

2. Le maillage est régulier, et la surface continue (i.e, en un seul "morceau"). Or en géologie, il est parfois nécessaire de "raffiner" le maillage à certains endroits ou de considérer qu'une surface est constituée de plusieurs morceaux distincts (c'est le cas lorsqu'on s'intéresse à des surfaces érodées par exemple). On doit alors pouvoir interactivement modifier la topologie d'une surface. Ces modifications topologiques sont très délicates à mettre en œuvre pour des surfaces de Bézier, surtout lorsqu'elles indiquent une modification ou une discontinuité dans le maillage. Leur application obligerait en effet à reconstruire entièrement le modèle (voir Figure 1.4).

3. Modifier la position d'un point de contrôle permet de modifier localement la surface modélisée. Cependant, il devient malaisé d'agir sur celle-ci de manière globale si l'on désire lui donner une forme très différente de celle qu'elle avait au départ. C'est précisément ce que l'on voudrait être capable de faire dans le cas spécifique de la modélisation d'hétérogénéités. On souhaite pouvoir laisser libre cours à l'imagination dans la création de formes élémentaires et il est alors très important de pouvoir les faire évoluer facilement et globalement.

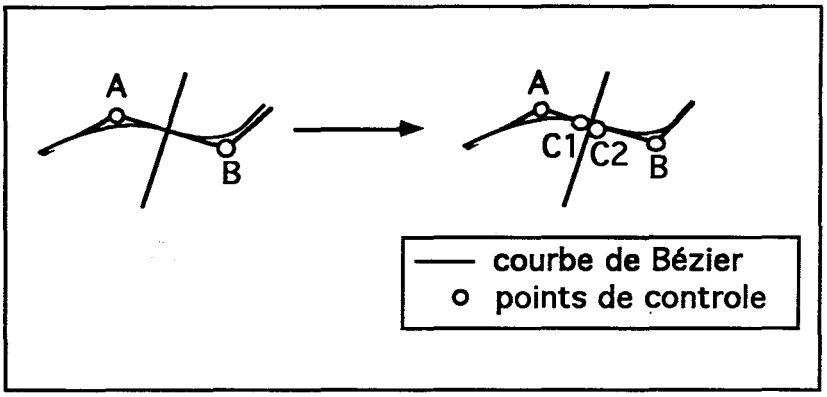


Figure 1.4 rupture de continuité d'une courbe de Bézier.

Pour toutes ces raisons, les méthodes de Bézier semblent mal adaptées au domaine de la géologie.

1.3.2 Méthode *DSI*

Les surfaces modélisées

La méthode *DSI* ([45],[48]) a été conçue pour appréhender les problèmes spécifiques à la géologie en ce qui concerne l'interpolation de surfaces en trois dimensions. Elle autorise en effet l'interpolation de surfaces discontinues, contraintes par des nœuds de contrôle situés sur cette surface d'une part, et par le respect de données pouvant être de natures très différentes d'autre part.

DSI considère que n'importe quel graphe est caractérisé par un certain degré de "rugosité" qu'il est possible de mesurer. C'est le cas par exemple de la surface triangulée présentée dans la Figure 1.5 ou de la ligne polygonale présentée dans la Figure 1.6. La minimisation de ce critère de rugosité conduit à la solution *DSI* qui est en fait le moyen le plus "lisse" de lier entre eux cet ensemble de nœuds. D'un point de vue mathématique, la méthode *DSI* a été proposée dans le but d'interpoler simultanément un ensemble de n fonctions définies en chaque nœud k d'un graphe. Nous définirons par:

$$\begin{cases} \Omega = \{1, 2, \ldots, N\} & : \text{l'ensemble des N nœuds constituant le graphe à interpoler;} \\ N(k) & : \text{le sous-ensemble de } \Omega \text{ décrivant l'ensemble des nœuds} \\ & \quad \text{constituant le voisinage d'un nœud } k \text{ particulier.} \end{cases}$$

Soit:

$$\varphi(k) = \{\varphi^1(k), \varphi^2(k), \ldots, \varphi^n(k)\}$$

L'ensemble des n fonctions $\varphi^\nu(k)$ definies pour tous les nœuds $k \in \Omega$. L'objectif est d'interpoler les valeurs de $\varphi(k), \forall k \in \Omega$. Nous considérons de plus que les voisinages $N(k)$ honorent la condition de symétrie suivante:

$$\alpha \in N(\beta) \Longleftrightarrow \beta \in N(\alpha), \forall(\alpha, \beta) \in \Omega \tag{1.1}$$

Critère de rugosité

Soit φ l'ensemble des $\{\varphi(k), \forall k \in \Omega\}$.
La méthode DSI est basée sur un critère global de rugosité défini comme suit:

$$\begin{cases} R(\varphi) & = \sum_k \mu(k).R(\varphi|k) \\ R(\varphi|k) & = \sum_{\alpha \in N(k)} |v^\alpha(k).\varphi(\alpha)|^2 \end{cases}$$

$R(\varphi|k)$ est un critère de rugosité local défini au nœud k, les $v^\alpha(k)$ sont des coefficients pondérateurs et les $\mu(k)$ des coefficients positifs donnés.
Un des choix les plus simples pour les coefficients $\mu(k)$ et $v^\alpha(k)$ est proposé dans [45]:

$$\left| \begin{array}{lll} v^\alpha(k) & = \begin{cases} -|\Lambda(k)| & \text{si} \quad \alpha = k \\ 1 & \text{si} \quad \alpha \in \Lambda(k) \end{cases} \\ \mu(k) & = 1 \quad \forall k \in \Omega \end{array} \right.$$

avec $\Lambda(k) = N(k) - \{k\}$;

Critère de contraintes

DSI peut prendre en compte des contraintes linéaires C_i en supposant qu'elles puissent s'exprimer de la manière suivante ([45]).

$$C_i(\varphi) = 0$$

1.3.3 La solution DSI

La solution DSI est donnée par l'ensemble des fonctions $\varphi(k)$ minimisant le critère $R^*(\varphi)$ ([5],[11]) tel que:

$$R^*(\varphi) = \sum_{k \in \Omega} \mu(k) R(\varphi|k) + \sum_i \varpi_i^2 . |C_i(\varphi)|^2 \tag{1.2}$$

où les $\{\varpi_i^2\}$ sont des pondérateurs correspondant à l'importance relative donnée aux contraintes $\{C_i\}$.
Il a été prouvé ([45]) que la solution DSI existe et est unique si:

$$\left| \begin{array}{lll} \mu(k) & > 0 & \forall k \\ v^\alpha(k) & > 0 & \forall \alpha \neq k, \alpha \in N(k) \\ v^k(k) & = & -\sum_{\alpha \in N(k), \alpha \neq k} v^\alpha(k) \neq 0 \end{array} \right. \tag{1.3}$$

De plus, si l'on considère un nœud $\alpha \in N(k)$, chaque $\varphi(\alpha)$ est défini à travers un processus itératif qui calcule à chaque itération une nouvelle valeur de $\varphi(\alpha)$:

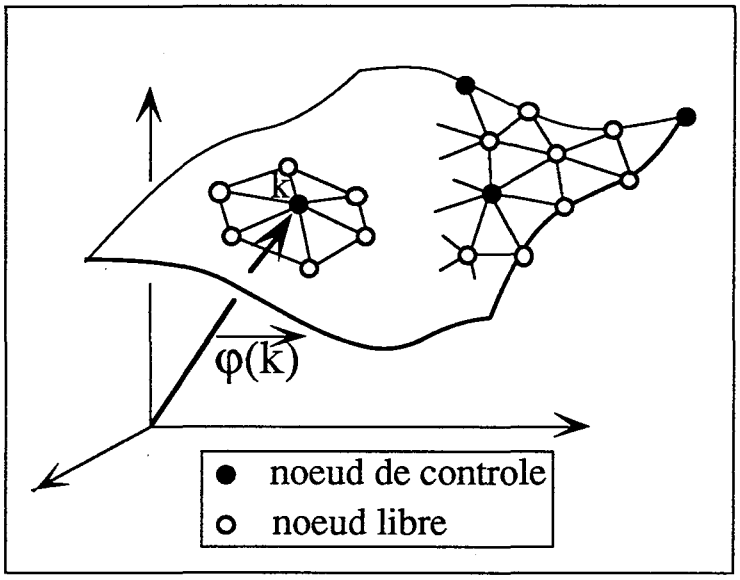


Figure 1.5 graphe représenté sous forme de surface triangulée

$$\varphi(\alpha) = -\frac{\left\{\sum_{k\in N(\alpha)} \mu(k)v^{\alpha}(k).\sum_{\beta\in N(k),\beta\neq\alpha} v^{\beta}(k).\varphi(\beta)\right\} + \sum_{i}\Gamma_i(\alpha)}{\sum_{k\in N(\alpha)} \mu(k)(v^{\alpha}(k))^2 + \sum_{i}\gamma_i(\alpha)} \tag{1.4}$$

où $\Gamma_i(\alpha)$ et $\gamma_i(\alpha)$ dépendent uniquement des contraintes $C_i(\alpha)$ et des poids relatifs ϖ_i. La convergence de ce processus itératif a également été prouvé ([45]). De plus, il apparaît que la formule 1.4 n'utilise que les valeurs $\varphi(k)$ pour $k \in N^2(\alpha)$ defini par:

$$N^2(\alpha) = \bigcup_{\beta\in N(\alpha)} N(\beta)$$

Enfin, il est important de noter ici qu'il a été prouvé que l'interpolation de chacune des fonctions $\varphi^{\nu}(k)$ est indépendante de l'interpolation des autres ([45]).

1.3.4 Utilisation classique de *DSI*

En pratique, *DSI* est utilisé pour modéliser des graphes matérialisés par des lignes polygonales ou des surfaces triangulées. Ces objects géométriques sont définis par les coordonnées

$$\{\varphi^x(k), \varphi^y(k), \varphi^z(k)\}$$

de chacun des nœuds $k \in \Omega$ (voir Figures 1.5 et 1.6). De plus,

1. certains nœuds correspondent à des nœuds de contrôle et ont une position connue dans l'espace à trois dimensions;

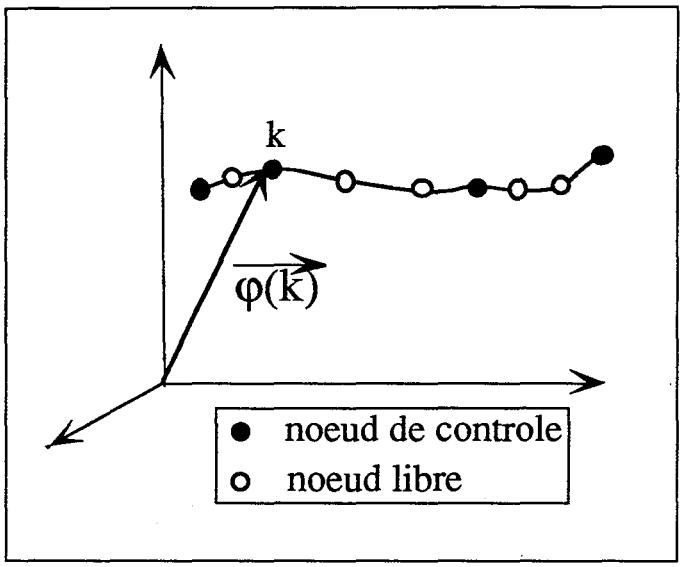


Figure 1.6 graphe représenté sous forme de ligne polygonale.

2. certains nœuds ont une position inconnue qui est calculée par *DSI* durant le processus itératif.

3. n'importe quelle modification de la position des nœuds de contrôle entraîne une modification de la solution *DSI* .

D'autre part,

1. les nœuds de contrôle appartiennent au graphe interpolé.

2. les nœuds de contrôle peuvent être spécifiés indépendamment les uns des autres.

La Figure 1.7 montre une ligne polygonale dont quatre nœuds ont été spécifiés comme nœuds de contrôle. Leurs positions sont donc fixes et invariantes par *DSI* . La solution *DSI* génère une ligne circulaire, qu'elle interprète comme étant la solution la plus "lisse" respectant la position des nœuds de contrôle.

1.3.5 Notion d' "atomes"

L'implémentation pratique de l'algorithme *DSI* repose sur la notion d'*atome*. Celui-ci représente un nœud du graphe Ω et contient l'information nécessaire à *DSI* pour interpoler la position de ce nœud dans l'espace. En effet, la description de l'algorithme *DSI* montre qu'au nœud k de Ω, trois types d'informations différentes sont utilisées. Ces informations sont de nature géométrique d'une part, topologique d'autre part, génétique enfin.

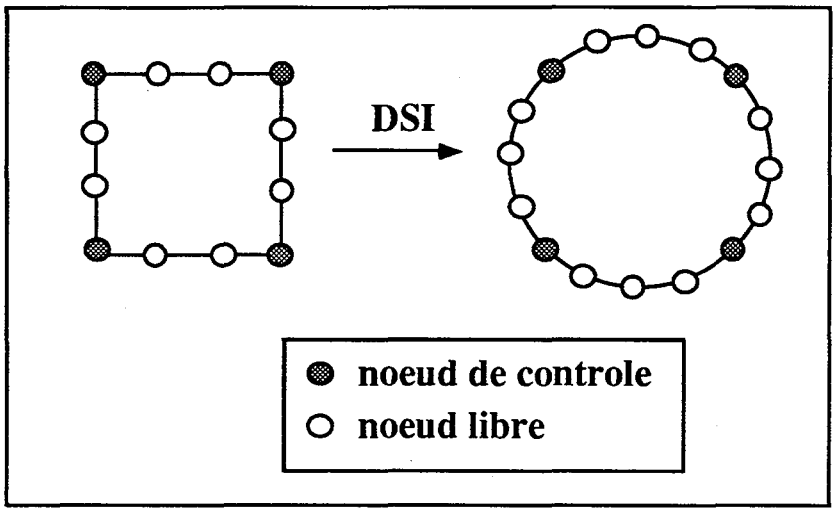


Figure 1.7 effet de l'interpolation DSI sur une ligne polygonale (30 itérations).

1. **Information d'ordre géométrique:** elle correspond à la position (x, y, z) dans l'espace du nœud k.

2. **Information d'ordre topologique:** c'est la spécification du voisinage au nœud k. Cela correspond en fait à la définition de $N(k)$.

3. **Information d'ordre génétique:** c'est la spécification du fait que le nœud k est ou non considéré comme un *nœud de contrôle* pour DSI .

Nous utiliserons la notion d'*atome* pour regrouper ces trois types d'information. Dans la suite de ce manuscrit nous considèrerons qu'un *atome* représente:

1. La position d'un nœud k de Ω;

2. L'information selon laquelle k est un *nœud de contrôle* ou non.

3. La définition de son voisinage $N(k)$. Celui-ci découle en général de la spécification d'un nombre "d'auréoles" autour de k. Les *atomes* voisins de k sont alors ceux définissant ces auréoles. Ainsi, dans la Figure 1.8, $N(k)$ contient respectivement 18 et 4 *atomes* dans le cas de la surface triangulée et de la ligne polygonale.

Par ailleurs, l'ensemble des *atomes* liés à un *atome* est appelé l'ensemble de ses *satellites.* Dans la Figure 1.8, le nombre des satellites de l'*atome* α est de six pour la surface triangulée et de deux dans le cas de la ligne polygonale. L'*atome* apparaît donc comme le vecteur de l'information que DSI utilise au cours de l'interpolation.

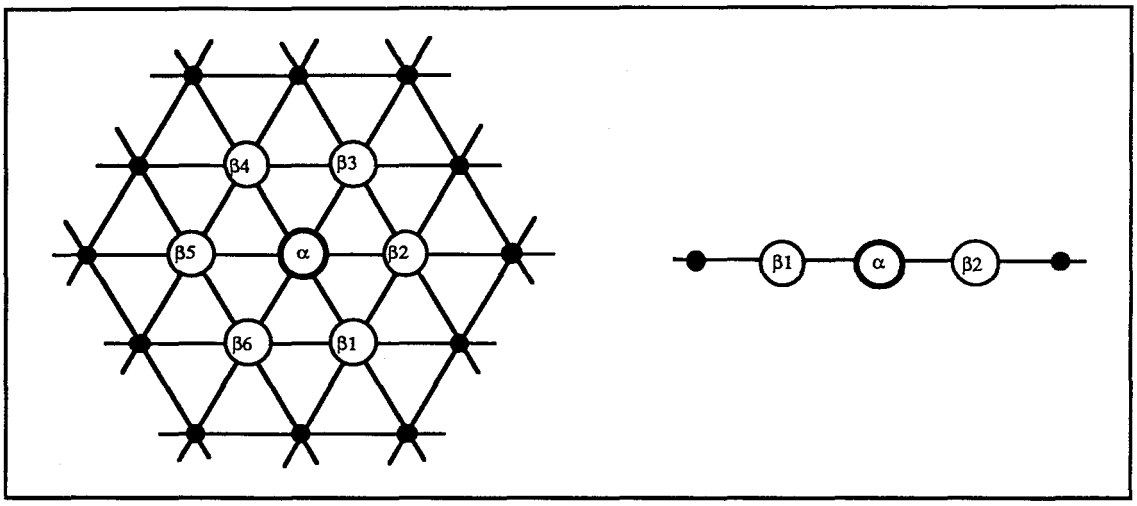


Figure 1.8 Notion de voisinage et de satellites dans le cas d'une surface triangulée et d'une ligne polygonale. Les $\{\beta_i\}$ représentent les *atomes* satellites de l'*atome* α. Le voisinage de α est défini par les *atomes* présents dans les deux premières auréoles autour de α.

1.3.6 Discussion

Reprenons les problèmes évoqués dans l'application des méthodes classiques au domaine de la géologie (voir section 1.3.1):

1. **Position des points de contrôle:** Pour l'interpolateur *DSI* , les nœuds de contrôle appartiennent à la surface à interpoler et c'est par leur manipulation directe que l'on modifie la surface. De plus, si l'on respecte la condition 1.3 sur les coefficients μ et v, à un ensemble de nœuds de contrôle donné ne correspond qu'une seule solution. *DSI* satisfait donc à l'exigence que la surface modélisée soit complètement décrite par la position de ses nœuds de contrôle.

2. **Régularité du maillage:** *DSI* repose sur la définition du *voisinage* en tout nœud du graphe à interpoler. Ce voisinage peut être redéfini **localement**, (au niveau de quelques *atomes* par exemple), ce qui modifie la topologie du graphe (voir Figure 2.9). On peut ainsi installer une discontinuité, sans avoir à modifier la position des nœuds de contrôle.

3. **Modification locale de la surface:** De même que pour les méthodes dites "classiques", la modification de la position d'un nœud de contrôle introduit une modification locale au niveau de la géométrie de la surface modélisée. Il reste malaisé de modifier celle-ci radicalement et globalement. Ainsi, pour obtenir les surfaces représentant les chenaux montrés dans la Figure 1.9, il a fallu plus d'une semaine!

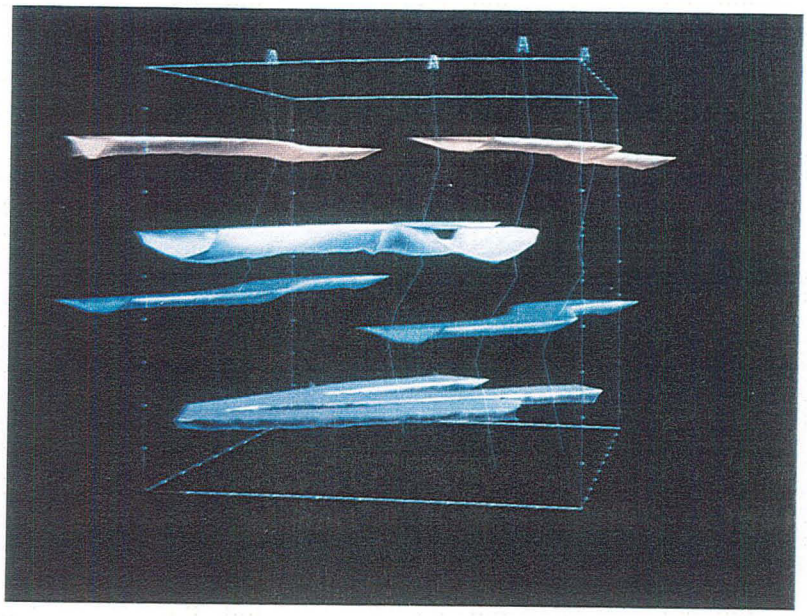


Figure 1.9 Modèle artificiel de chenaux modélisés à l'aide de l'interpolation *DSI* classique.

Ainsi, si l'approche *DSI* paraît adaptée à l'appréhension des problèmes propres à la géologie, elle ne semble pas être suffisamment souple pour modéliser aisément et de manière globale des corps limitant un volume fermé. Cette constatation nous a conduit à développer une nouvelle méthode, basée sur *DSI* , mais qui permet beaucoup plus facilement que l'utilisation classique présentée à la section 1.3.4 de générer des formes complexes en trois dimensions. Cette méthode, basée sur la construction d'un nouvel objet, que l'on appèlera *gshape*[3] est décrite dans la suite de ce chapitre.

1.4 Présentation de l'objet gshape

Nous dirons d'abord ce qui caractérise un objet *gshape*. Nous décrirons ensuite les concepts utilisés pour le développement de la méthode de paramétrisation et d'interpolation d'un *gshape*.

[3] *gshape* signifie *g*eometrical *shape* object

1.4.1 Notion de gshape

Si nous revenons aux schémas de la Figure 0.1, nous pouvons observer que, en plus d'être caractérisées par une surface englobant un volume fermé de l'espace, les hétérogénéités de type *chenal*, *lobe* ou *slump* peuvent être décrites par les trois éléments suivants:

1. Une ligne particulière qui décrit l'allure générale de l'objet, et "autour" de laquelle celui-ci est construit. Cette ligne suit plus ou moins la direction d'allongement de l'objet.
2. Un ensemble de sections planaires situées le long de cette ligne.
3. Une *enveloppe*, surface qui lie les sections entre elles autour de la ligne, et qui limite dans l'espace un volume fermé.

Par la suite, la ligne caractéristique de l'objet sera appelée *backbone*[4], une section planaire particulière sera appelée une *section*. Nous appelerons *gshape* tout objet pouvant être défini par les trois éléments *backbone*, *sections*, *enveloppe* (voir Figure 1.14, page 33). Modéliser un objet de forme complexe et limitant un volume fermé dans l'espace, revient alors à s'intéresser à la géométrie de l'*enveloppe* d'un *gshape*. Cependant, notre description laisse apparaître que celle-ci dépend à la fois de la géométrie du *backbone* et de celle des *sections*. Elle en est, en fait, une *conséquence*. Cela a conduit au principe de travail suivant:

Il est possible de paramétriser l'*enveloppe* d' un *gshape* en paramétrisant séparémment le *backbone* d'une part, et les *sections* d'autre part.

1.4.2 Le backbone

Il s'agit ici de modéliser la "ligne conductrice" autour de laquelle un *gshape* est construit. Nous définissons alors le *backbone* d'un *gshape* par une ligne polygonale constituée d'un ensemble de nœuds liés entre eux (voir Figure 1.10). Ses caractéristiques sont les suivantes:

1. Les nœuds du *backbone* peuvent appartenir à deux catégories:
   (a) **Les nœuds de contrôle:** ils sont considérés comme ayant une position fixe dans l'espace et correspondent, par exemple, à une connaissance a priori de la position d'un nœud du *backbone*. Ils constituent les *paramètres* du *backbone*.
   (b) **Les nœuds libres:** leurs positions peuvent varier dans l'espace et seront, le plus souvent, *interpolées*.
2. La manière dont les nœuds du *backbone* sont liés entre eux décrit, par définition, la *topologie* du *backbone*. Celui-ci apparaît donc comme la définition d'un graphe dont les nœuds libres pourront être interpolés. Au sens de DSI , le voisinage $N(k)$ de chacun des nœuds k du *backbone* est décrit par la façon dont ceux-ci sont liés entre eux. D'autre part, la topologie du *backbone* est limitée par la condition suivante:

[4]colonne vertébrale

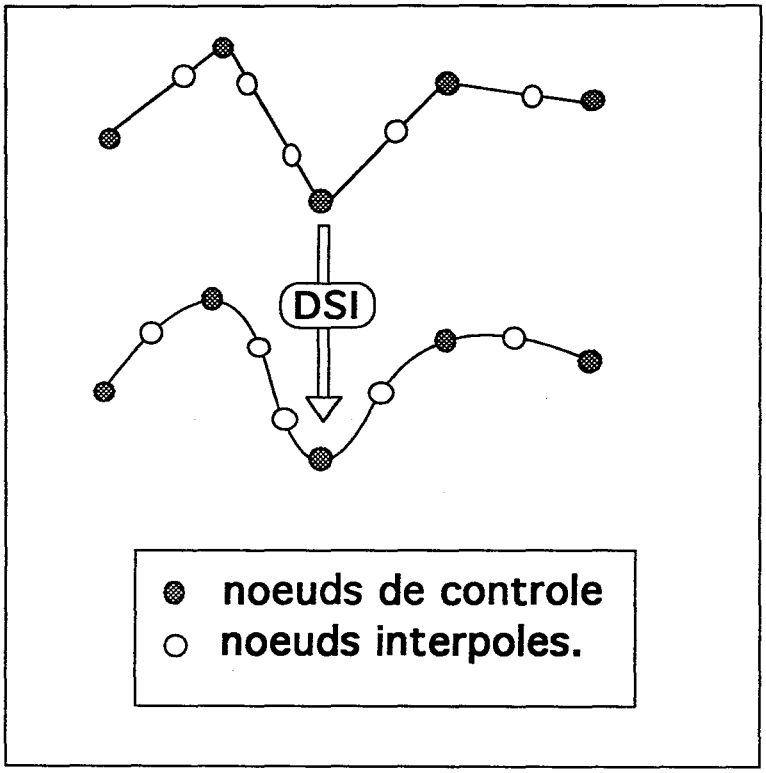


Figure 1.10 Ligne polygonale définissant le *backbone* d'un *gshape.*

Chaque nœud du backbone est, au plus, connecté à deux autres nœuds.

3. Le *backbone* peut également être décrit comme un ensemble de *segments.* On dira que les nœuds (k_i, k_j) décrivent un *segment* du *backbone* si:

$$k_i \text{ et } k_j \text{ sont connectés.}$$

1.4.3 Les sections

On appelle *section* un ensemble de vecteurs liés à un nœud du *backbone* et qui décrit un contour plan autour de ce nœud (voir Figure 1.11). Une *section* est définie par les éléments suivants:

1. Un nœud k du *backbone*;

2. Un ensemble de N vecteurs $\{V_0, \cdots, V_{N-1}\}$ ayant comme origine le nœud k et décrivant le *contour* de la section planaire à laquelle il est associé. Ces vecteurs

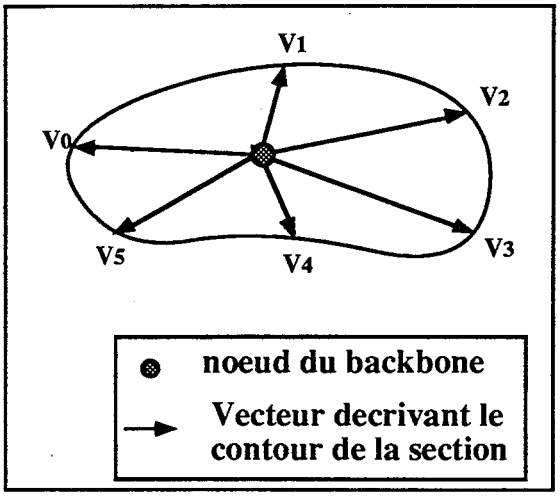


Figure 1.11 Définition d'une section décrite par 6 vecteurs ordonnés.

sont ordonnés[5], chaque vecteur V_i étant caractérisé par son *index i*. Le *contour* de la section est décrit par les points constituant l'extrémité de ces vecteurs pris selon leurs *index* croissants.

On associe ainsi à chaque nœud du *backbone* un ensemble de N vecteurs qui décrivent un contour plan autour de lui. De plus, on peut faire les remarques suivantes:

1. Les *sections* d'un *gshape* peuvent être de natures différentes.

   (a) **Les sections de contrôle** sont fixes. Elles correspondent en fait à des nœuds dont les *contours* associés sont invariants lors de l'interpolation. Ceux-ci sont par exemple issus de connaissance a priori. Les *sections de contrôle* constituent les *paramètres* de l'ensemble des *sections* d'un *gshape*.

   (b) **Les sections libres**. Leurs *contours* peuvent varier et seront en général *interpolés*.

2. Les *sections* d'un *gshape* apparaissent comme la traduction d'une série de "coupes sériées" effectuées à l'intérieur d'un objet. C'est pour cette raison qu'a été formulée, d'emblée une contrainte de planarité.

3. Le nombre N de vecteurs associés à un *contour* **doit** être le même pour toutes les *sections* d'un *gshape*. Nous verrons par la suite que cette caractéristique entraîne des conséquences de simplification lors de la mise en place d'algorithmes tels que triangulation ou tétraédrisation de l'*enveloppe* d'un *gshape*[6].

[5] voir section 1.4.5
[6] voir section 2.6.1 et 2.6.2.

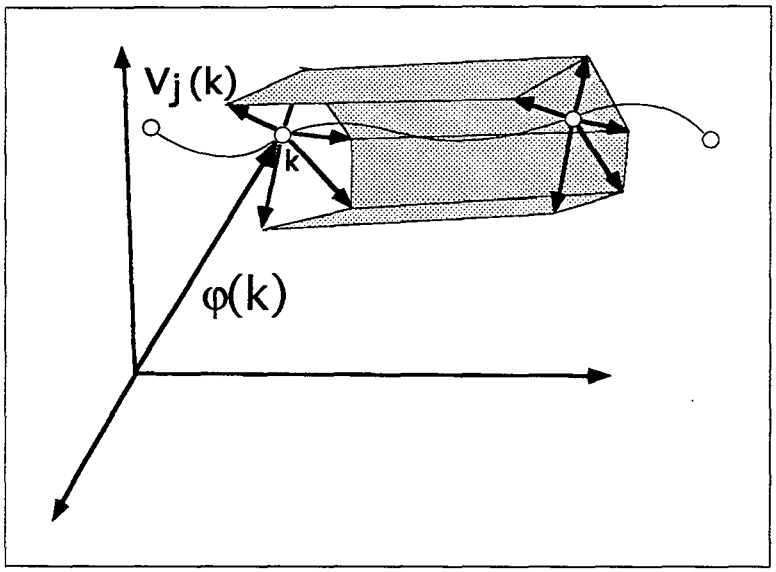


Figure 1.12 Définition de l'*enveloppe* d'un *gshape* dont les *sections* sont définies par 5 vecteurs.

1.4.4 L'enveloppe

Description

Soit un *gshape* dont le backbone est composé de N_b nœuds, et dont chacune des sections est formée de N_v vecteurs. La géométrie du *backbone* est définie par les vecteurs:

$$\{\varphi(k), k \in \Omega\}$$

et celle des *sections* par les vecteurs:

$$\{\{V_j(k), j \in [0, N_v[\}, k \in \Omega\}$$

L'*enveloppe* de ce *gshape* est alors décrite par l'ensemble E des vecteurs de l'espace, et l'ensemble P des points de l'espace tels que:

$$\begin{aligned} E &= \{\ \{\varphi(k) + V_j(k), j \in [0, N_v[\}, k \in \Omega\ \} \\ P &= \{\ p^j(k), j \in [0, N_v[, k \in \Omega\ \} \end{aligned}$$

P est en fait l'ensemble des points extrémité des vecteurs de E. *L'enveloppe* d'un *gshape* peut donc être matérialisée par $N_b \times N_v$ points de l'espace, dont les coordonnées dépendent à la fois de la géométrie du *backbone* et de celle des *sections* (voir Figure 1.12).

1.4.5 Conventions adoptées

Nous adopterons par la suite les conventions suivantes: soit Ω l'ensemble des nœuds du

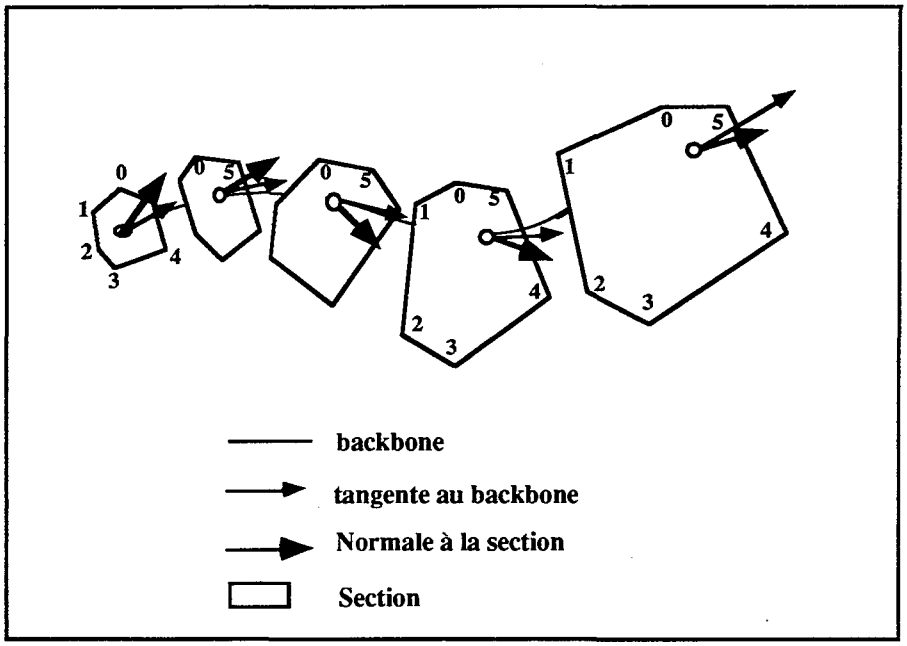


Figure 1.13 Conventions adoptées en ce qui concerne la numérotation des vecteurs, de la *normale* et de la *tangente* associées aux nœuds d'un *gshape* composé de 6 vecteurs par *section*.

backbone d'un *gshape*; pour tout $k \in \Omega$, nous noterons $T(k)$ le vecteur tangent au *backbone* en k et $n(k)$ le vecteur normal à la *section* portée par k, $T(k)$ et $n(k)$ étant toutes deux des notations normalisées. Nous considérons alors que la construction d'un *gshape* obéit aux trois conventions suivantes:

1. Le sens de parcours du *backbone* est donné par l'orientation de $T(k)$ à un nœud k quelconque de Ω. Ce sens est cohérent pour tous les nœuds de Ω.

2. $\forall k \in \Omega, n(k).T(k) \geq 0$

3. Soit k un nœud quelconque de Ω et soit $\{V_i(k), i \in [0, N[\}$ les vecteurs décrivant la *section* en ce nœud; la transformation d'un vecteur V_i en V_j, $\forall j > i$ consiste en une homothétie de centre k suivie d'une rotation dont l'axe est donné par la normale au nœud k.

Le respect de ces conventions permet une indexation cohérente au niveau des vecteurs décrivant une *section*. Le sens de parcours est ainsi le même pour toutes les *sections* d'un *gshape*, et ce sens est reflété par l'orientation de la normale à la *section* et de la tangente au *backbone* à un nœud donné. Ces conventions sont résumées par la Figure 1.13.

1.5 Interpolation d'un "gshape"

1.5.1 Formalisation

Un *gshape*, nous l'avons vu, est composé de deux entités distinctes, le *backbone* d'une part, et les *sections* d'autre part. Soit Ω, l'ensemble des nœuds du graphe matérialisé par le *backbone*, les informations portées par chaque nœud $k \in \Omega$ sont les suivantes:

1. Information sur la position dans l'espace du nœud k: elle est donnée par la fonction:

$$\varphi(k) = \{\varphi^x, \varphi^y, \varphi^z\}$$

où $\varphi^x, \varphi^y, \varphi^z$ sont les coordonnées x, y, z au nœud k dans l'espace à trois dimensions (voir Figure 1.6, page 20).

2. Information sur la géométrie de la *section* associée au nœud k. Elle est donnée par la fonction:

$$\begin{cases} V(k) & = \{V_0^x(k), V_0^y(k), V_0^z(k), \ldots, V_{N-1}^x(k), V_{N-1}^y(k), V_{N-1}^z(k)\} \\ & = \{v^1(k), v^2(k), \ldots, v^{3*(N)}(k)\} \end{cases}$$

où N est le nombre de vecteurs définissant chaque *section*. On attache cette fois un vecteur V de dimensions $3 * N$ au nœud k. Le vecteur$\{v^1(k), v^2(k), \ldots, v^{3*N}(k)\}$ est interprété comme définissant les coordonnées $\{x, y, z\}$ des N vecteurs

$$\{V_0, V_1, \ldots, V_{N-1}\}$$

par rapport au nœud k (voir Figure 1.14).

1.5.2 Notion de "vertèbre"

La notion d'*atome*, telle qu'elle a été présentée à la section 1.3.5 ne permet pas de prendre en compte le fait que l'on interpole deux informations différentes sur le même nœud. Nous avons donc étendu la notion d'*atome* à celle de *vertèbre*. Pour chaque nœud k de Ω, celle-ci ajoute aux informations contenues dans l'*atome*, les informations suivantes:

1. Aux informations d'ordre géométrique, elle ajoute:

   (a) l'ensemble des N vecteurs ordonnés et indexés, définis par rapport à la position du nœud k et qui décrivent, dans l'ordre de leur index croissant, le contour de la section plane autour de k. On parlera alors de *contour* défini par le vecteur $V(k)$, tel que:

$$V(k) = \{V_0^x(k), V_0^y(k), V_0^z(k), \ldots, V_{N-1}^x(k), V_{N-1}^y(k), V_{N-1}^z(k)\}$$

   où $\{V_i^x(k), V_i^y(k), V_i^z(k)\}$ sont les coordonnées du vecteur d'*index i*. On appellera également $V_i(k)$ le *rib*[7] d'*index i* associé à la *vertèbre k*.

[7] *rib* signifie côte en anglais

(b) Un vecteur tangent au *backbone* au nœud k;

(c) Un vecteur orthogonal au plan de la section au nœud k.

2. Aux informations d'ordre génétique, elle ajoute la spécification du fait que le nœud k est ou non une *section de contrôle* pour l'interpolateur.

3. Aux informations d'ordre topologique, elle ajoute la contrainte suivante:

   Le nœud k a, au plus, deux satellites (au sens exposé à la section 1.3.5).

Une *vertèbre* est ainsi le vecteur de l'information supportée par un nœud k de Ω. Puisqu'elle est une extension de l'*atome*, elle contient également l'information topologique nécessitée au cours de l'interpolation. Les deux fonctions à interpoler portées par une *vertèbre* k sont donc les suivantes:

$$\begin{cases} \varphi(k) & = \{\varphi^x, \varphi^y, \varphi^z\} \\ V(k) & = \{V_0^x(k), V_0^y(k), V_0^z(k), \ldots, V_{N-1}^x(k), V_{N-1}^y(k), V_{N-1}^z(k)\} \end{cases}$$

La signification physique des résultats de l'interpolation sur ces fonctions est de deux ordres:

1. **Au niveau local** de la *vertèbre* k: l'interpolation de chacun des deux vecteurs $\varphi(k)$, et $V(k)$ détermine respectivement:

   (a) la position de la *vertèbre* relativement au *backbone*;

   (b) la forme de son *contour*.

2. **Au niveau global** du graphe Ω, l'interpolation de chacun des deux vecteurs détermine respectivement:

   (a) la géométrie du *backbone*; elle est décrite par le vecteur

   $$\varphi = \{\varphi(k), \forall k \in \Omega\}$$

   (b) la géométrie de l'*enveloppe* du *gshape*. Celle-ci est décrite par le vecteur

   $$V = \{V(k), \forall k \in \Omega\}$$

Il y aura donc deux fonctions distinctes responsables de l'interpolation d'un *gshape*. La première interpolera le vecteur φ, la deuxième le vecteur V. Le seul lien entre ces deux interpolations est le fait qu'elles sont supportées par **le même graphe**, et donc par la même topologie. De ce fait, le même voisinage sera considéré en chaque *vertèbre* pour interpoler les deux informations dont elle est porteuse.

Définition d'un gshape.

Un *gshape* est la représentation d'un graphe unidimensionel particulier dont les nœuds sont des *vertèbres*.

Le *backbone* définit la topologie et la géométrie de ce graphe.

1.5.3 Interpolation de la géométrie du *backbone*

Dans la suite de ce manuscrit, nous appelerons Ω, le graphe représenté par un *gshape*. La géométrie du *backbone* est définie par la fonction:

$$\varphi = \{\varphi(k), \forall k \in \Omega\}$$

La méthode d'interpolation utilisée pour interpoler φ correspond à l'utilisation classique de *DSI* décrite à la section 1.3.4. Nous rappelons que l'interpolation s'effectue à travers les étapes suivantes:

1. Une solution initiale est générée;
2. Les positions de certaines *vertèbres* sont fixées. Elles correspondent alors à des *nœuds de contrôle*.
3. *DSI* interpole le vecteur $\varphi(k)$ de chaque *vertèbre* $k \in \Omega$, en tenant compte de la position des *vertèbres* appartenant à son voisinage $N(k)$. Ceci se fait par le procédé itératif présenté à la section 1.3.3.

De plus, il est important de noter que:

1. La solution initiale n'est utilisée que comme outil pour spécifier de manière naturelle la façon dont les *vertèbres* sont liées entre elles, c'est à dire, la *topologie* du *backbone*.
2. La solution finale de *DSI* ne dépend pas de la position des *vertèbres* mais de la manière dont elles sont liées entre elles. Cependant, de la solution initiale dépend la rapidité de la convergence ([45]).
3. La solution *DSI* définit la géométrie du *backbone* compte tenu de sa *topologie*.

1.5.4 Interpolation de la géométrie des *sections*

Introduction

La géométrie de l'ensemble des *sections* associées aux *vertèbres* de Ω est donnée par la fonction:

$$V = \{V(k), \forall k \in \Omega\}$$

Chaque $V(k)$ est cette fois de dimension N. Nous ne sommes donc plus dans le cadre d'une utilisation "classique" de *DSI* (voir section 1.3.4) où les vecteurs interpolés sont

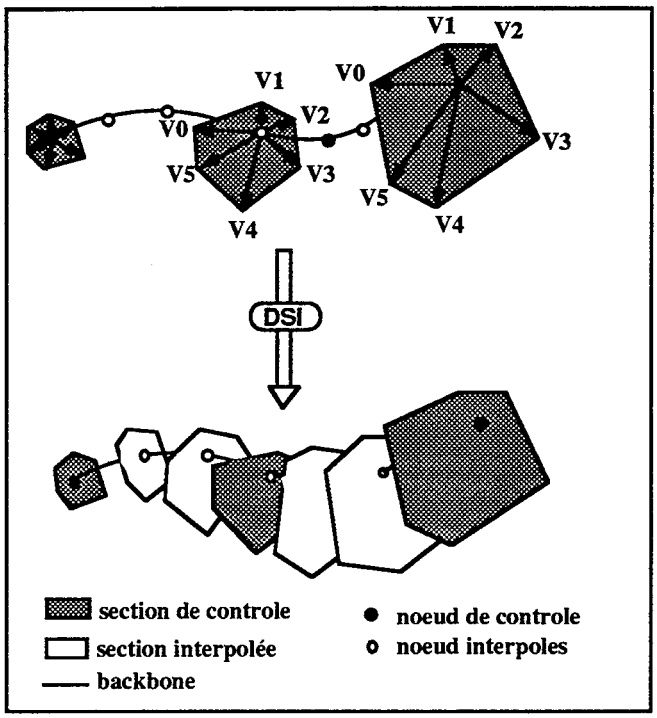


Figure 1.14 Interpolation des *sections* d'un *gshape*.

de dimension 3. Cependant, si nous adoptons une démarche similaire à celle utilisée pour l'interpolation du *backbone*, l'interpolation de V devrait s'effectuer à travers les étapes suivantes:

1. Générer une solution initiale;
2. Fixer le *contour* de certaines *vertèbres*. Elles correspondent alors à des *sections de contrôle*.
3. Interpoler le vecteur $V(k)$ de chaque *vertèbre* $k \in \Omega$, en tenant compte de la position des *vertèbres* appartenant à son voisinage $N(k)$ et dans le cadre d'une simple extension de DSI à N dimensions. Utiliser le procédé itératif présenté à la section 1.3.3.

Cependant, nous nous sommes rendu compte qu'appliquer cette méthode dans le cas d'un vecteur à N dimensions soulevait le problème suivant:

L'interpolation du vecteur $V(k)$ au nœud k ne conduit pas à l'obtention de N *ribs* parfaitement coplanaires.

Or, un *gshape* ne possède par définition que des *sections* coplanaires[8]. Il a donc été nécessaire de contraindre la méthode d'interpolation à prendre en compte le fait que

[8]voir section 1.4.3

l'interpolation des *sections* d'un *gshape* doit conduire à des vecteurs

$$\{V_i(k), i \in [0, N[, \forall k \in \Omega\}$$

coplanaires.

Introduction d'une contrainte de coplanarité

Soit k une *vertèbre* de Ω et soit $n(k)$ sa normale. Soit N, le nombre de vecteurs décrivant son *contour*. Il est possible de spécifier que tout *rib* $\{V_i(k), i \in [0, N[\}$ de k soit orthogonal à $n(k)$, en introduisant la contrainte linéaire suivante:

$$V_i(k).n(k) = 0, \forall i \in [0, N[.$$

avec:

$$\begin{cases} n(k) & = \{n^x(k), n^y(k), n^z(k)\} & : \text{vecteur normal de la } \textit{vertèbre } k; \\ V_i(k) & = \{V_i^x(k), V_i^y(k), V_i^z(k)\} & : \text{le } i^{ieme} \textit{ rib} \text{ du } \textit{contour} \text{ de la } \textit{vertèbre } k; \end{cases}$$

Cela conduit à l'expression:

$$\sum_{\nu=x,y,z} \sum_{\alpha \in \Omega} A_i^\nu(\alpha).V_i(\alpha)^\nu = 0$$

où:

$$A_i^\nu(\alpha) = \begin{cases} n^\nu(k) & \text{si } k = \alpha \\ 0 & \text{sinon} \end{cases}$$

Ce type de contrainte s'exprime donc par une série de N contraintes linéaires du type $C_i(\varphi) = 0, i \in [0, N[$. Ces contraintes s'appliquent à chaque nœud k de Ω et nous avons vu qu'elles pouvaient être prises en compte par DSI lors de l'interpolation ([45]). Elles ont donc été rajoutées à la fonction d'interpolation des vecteurs $\{V_i(k), k \in \Omega\}$ selon la méthode exposée à la section 1.3.2.

Détermination du vecteur normal au plan de la *section*

Un des problèmes qui s'est posé lors de la mise en place de l'objet *gshape* a été celui de la direction du plan des *sections* par rapport à l'axe du *backbone*. Notre première approche a été d'imposer que le plan contenant le *contour* d'une *vertèbre* k soit orthogonal à la tangente en k au *backbone*. Cela a conduit à des résultats corrects lorsque la sinuosité de la courbe n'était pas trop complexe (voir Figure 1.15). Cependant, lorsqu'il s'est agit de modéliser des corps du type de celui présenté dans la Figure 1.16 (b), nous arrivions à des phénomènes "d'intersection" des *contours* qui pouvaient s'avérer gênants lors des matérialisations de l'*enveloppe* (voir sections 2.6.1 et 2.6.2).

Il a donc fallu mettre en place une autre solution de manière à déterminer, en chaque

vertèbre k de Ω, le vecteur $n(k)$ orthogonal au plan de la *section* et qui ne résulte pas d'une décision "arbitraire" mise en place uniquement pour "éliminer" ces phénomènes d'intersection. Pour cette raison, nous avons décidé d'interpoler les *normales* d'une *vertèbre* k, au même titre que sont interpolés les *ribs* $\{V_i(k)\}$ définissant son *contour*. La méthode utilisée est la suivante:

1. On définit le vecteur à interpoler par:

$$n = \{n(k), \forall k \in \Omega\}$$

où $n(k)$ est la *normale* en k.

2. Les paramètres considérés sont les normales $\{n(k)\}$ pour toute *vertèbre* k définie comme *section de contrôle*. Ces normales sont alors invariantes par DSI .

L'algorithme d'interpolation de l'ensemble des *sections* d'un *gshape* est alors le suivant:

tant que le nombre demandé d'itérations de DSI n'est pas atteint

Pour tout $k \in \Omega$

. interpoler sa *normale* $n(k)$;

. interpoler les *ribs* $\{V_i(k), k \in [0, N[\}$ de k en utilisant $n(k)$ comme valeur de la normale à son *contour*. Utiliser cette valeur dans le calcul de la contrainte linéaire associée (voir section 1.3.3).

finpour.

fintantque.

(a)

(b)

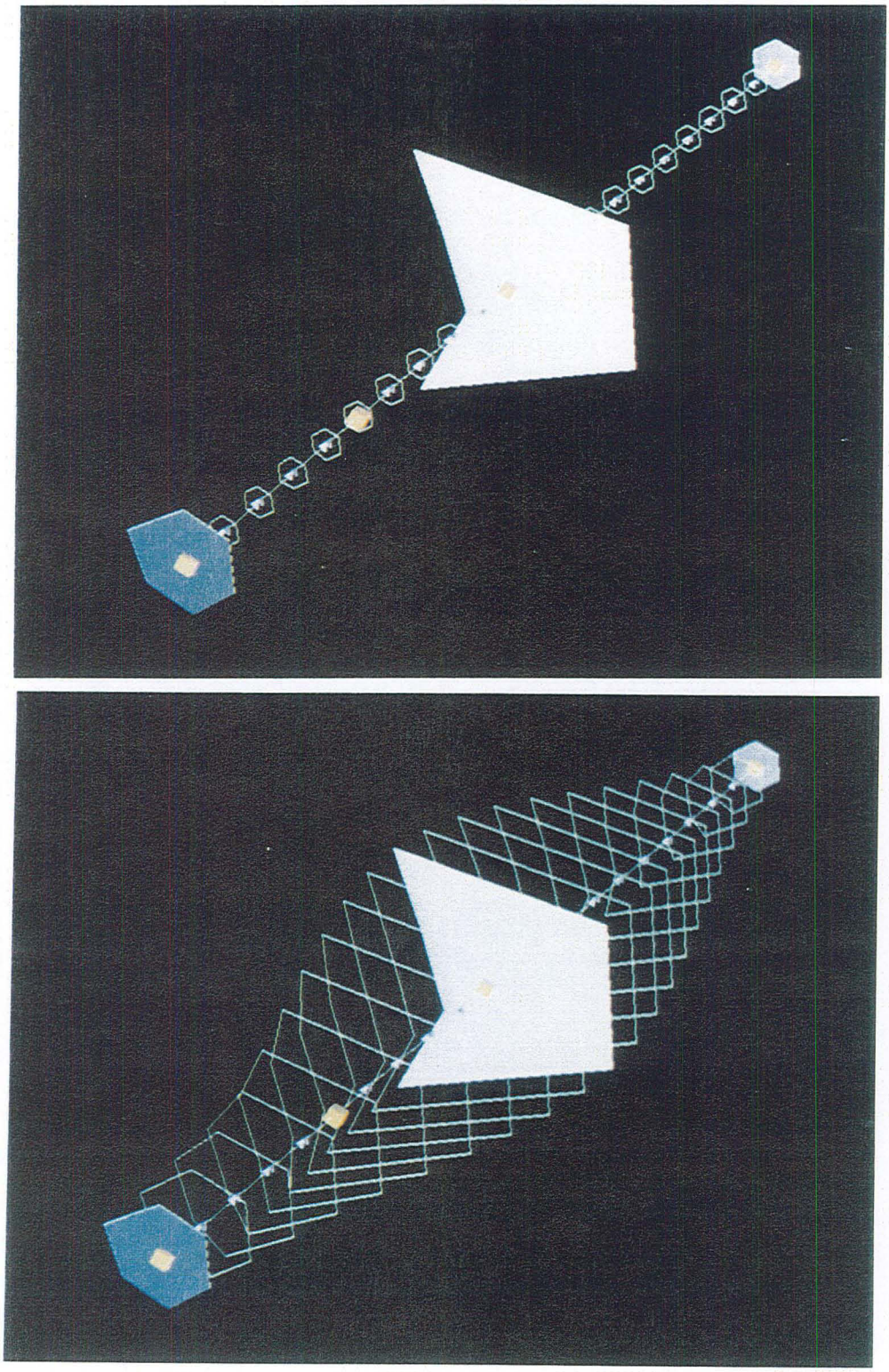
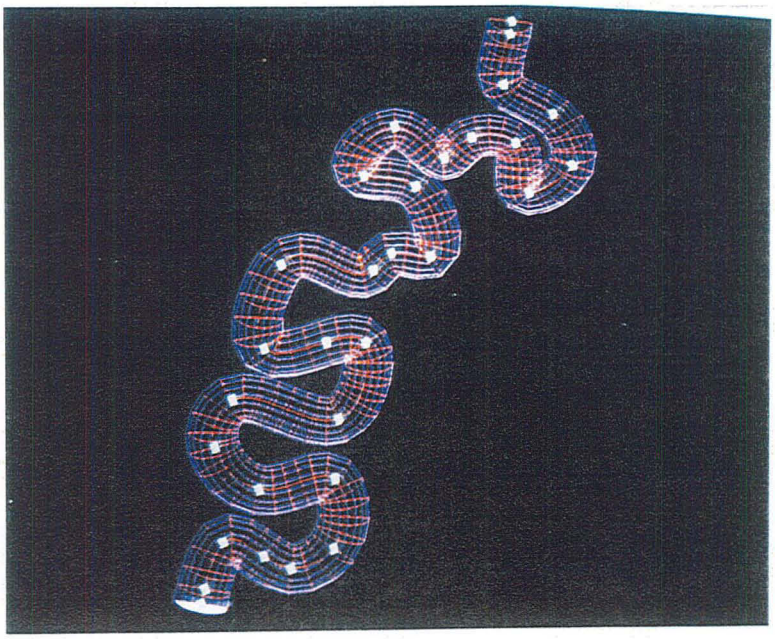


Figure 1.15 (a) *gshape* avant interpolation des *sections* et après interpolation du *backbone*; trois *sections de contrôle* et quatre *nœuds de contrôle* ont été spécifiés.(b) *gshape* après interpolation des *sections*.

(a)



(b)

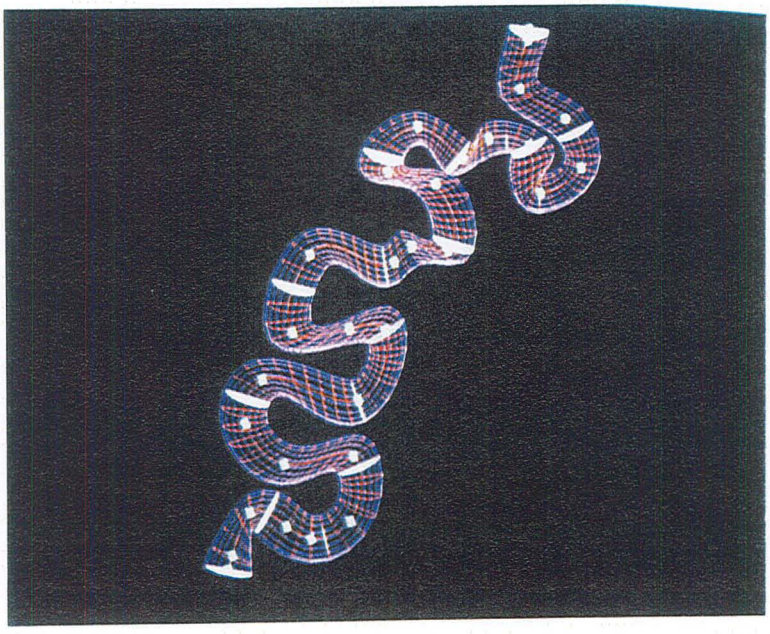


Figure 1.16 (a) *gshape* avant interpolation sur les *sections*. (b) Effet de l'interpolation des *sections* sur ce *gshape*, après spécification de 16 *sections de contrôle*.

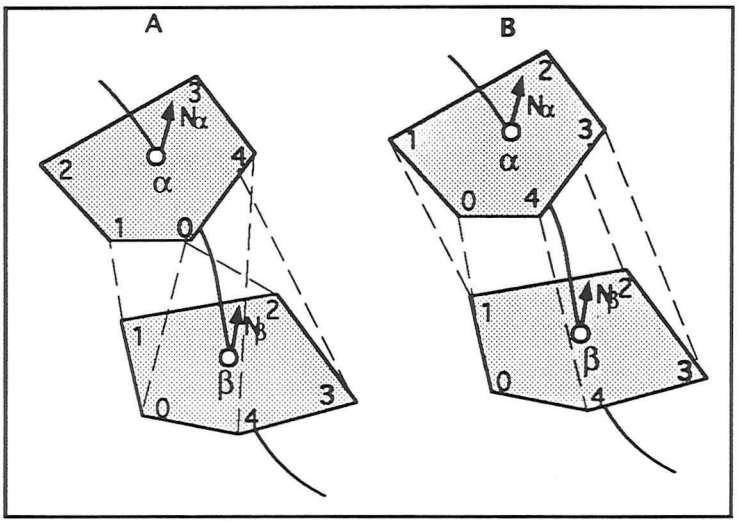


Figure 1.17 Deux *sections* correctement décrites par la numérotation relative de leurs *ribs*. (A) *ribs* corrélées de façon à entraîner une "torsion" de *l'enveloppe*. (B) gshape après spécification que les *ribs* $V_1(\alpha)$ et $V_0(\beta)$ devaient être corrélées.

Les conséquences de l'introduction d'une telle méthode de calcul des *normales* sont les suivantes:

1. La *normale* d'une *vertèbre* est interpolée sans tenir compte de la valeur des *ribs* définissant son *contour*. On évite ainsi les problèmes d'ordre numérique qui pourraient se poser si l'on calculait une *normale* à partir du plan moyen de ce *contour*. En effet, il est possible, après une itération donnée de se retrouver avec un *contour* dégénéré. Le calcul de la normale à un tel plan n'aurait alors aucun sens.

2. L'interpolation par DSI du plan d'une *section* au nœud k conduit à calculer le plan qui fait passer de la manière la plus lisse possible du plan des *sections* voisines au plan interpolé. Cela engendre en pratique un phénomène de "non intersection" des plans qui est très appréciable lorsqu'on modélise un *gshape* aussi sinueux que celui de la Figure 1.16 (a). L'interpolation des *sections* d'un tel *gshape* est donné dans la Figure 1.16 (b).

Remarque

Il est important de mentionner l'importance des relations entre *ribs* appartenant à différentes *vertèbres*. Il faut pour cela revenir sur la notion d'ordre qui apparaît dans la définition du *gshape* et dans les conventions adoptées à la section 1.4.5. Cette notion d'ordre est exigée pour les deux raisons suivantes:

1. Un contour est physiquement décrit par des points qui se succèdent dans un certain ordre. Il est donc nécessaire de numéroter les points qui le définissent afin de

déterminer comment ils doivent être parcourus. A fortiori, l'ordre des *ribs* d'une *vertèbre* doit être spécifié. Nous rappelons que le *contour* d'une *section* est décrit par la courbe joignant l'extrémité des *ribs* $\{V_i\}$, parcourus dans l'ordre croissant des *index* $\{i \in [0, N[\}$.

2. Nous avons vu que le *rib* $V_i(k)$ d'une *vertèbre* k sera interpolé en fonction des *ribs* $\{V_i(\alpha), \alpha \in N(k)\}$, où $N(k)$ est l'ensemble des *vertèbres* voisines de k. Autrement dit, les coordonnées d'un *rib* d'index i d'une *vertèbre* k seront interpolées en fonction des coordonnées des *ribs* d'index i des *vertèbres* voisines de k. Il est donc d'une extrême importance que les *ribs* soient numérotés d'une façon consistante d'une *vertèbre* à une autre.

La première remarque concerne la façon dont les *ribs* sont ordonnés **dans** une *vertèbre*. Elle ne concerne donc que l'ordre **relatif** des *ribs* entre eux et est prise en compte dans les conventions exposées à la section 1.4.5. La deuxième remarque concerne le *gshape* dans son aspect global, et touche à la manière dont l'interpolateur DSI a été implémenté. C'est l'ordre **absolu** des *ribs* qui intervient, c'est-à-dire la valeur exacte du vecteur

$$V(k) = \{v^1(k), v^2(k), \ldots, v^{3*N}(k)\}$$

associé à chaque *vertèbre* k. La Figure 1.17 illustre ce phénomène:

1. L'exemple (A) montre deux *sections de contrôle* parfaitement décrites par l'ordre relatif de leurs *ribs*. Cependant, lors de l'interpolation de *sections* intermédiaires libres, les *ribs* d'index i seront interpolés en fonction des *rib*s d'index i voisins, entrainant une "torsion" de l'*enveloppe* du *gshape*.

2. Pour éviter cet effet de "torsion" (qui peut par ailleurs être désiré), il faut redéfinir les *index* des *ribs* d'une des *sections de contrôle* afin de mettre en correspondance, par exemple, les *ribs* d'*index* 1 et 0 des *sections de contrôle* α et β. Par la suite, nous appelerons cette opération le fait de "corréler" des *ribs* entre elles.

Soient $\{V_i(\alpha), i \in [0, N[\}$ et $\{V_i(\beta), i \in [0, N[\}$ les *ribs* associés aux *vertèbres* α et β. Corréler les deux *ribs* $\{V_{newIndex}(\alpha), newIndex \in [0, N[\}$ et $\{V_{oldIndex}(\beta), oldIndex \in [0, N[\}$ se fait de manière évidente par l'algorithme ci-dessous:

pour tout $V_i(\beta), i \in [0, N[$

. soit $j = (newIndex + i)\%N$

. soit $k = (oldIndex + i)\%N$

. soit $v_j = V_k(\beta)$;

finpour.
pour tout $V_i(\beta), i \in [0, N[$

. $V_i(\beta) = v_i$

finpour.

Le résultat d'une corrélation entre les *ribs* $V_0(\beta)$ et $V_1(\alpha)$ du gshape de la Figure 1.17(A) est présenté dans la Figure 1.17 (B).

1.6 Conclusion

Les avantages de la méthode d'interpolation des *gshapes* reposent sur les remarques suivantes:

1. La décomposition d'un objet en deux éléments physiques distincts *backbone* et *sections*, se traduit par un formalisme mathématique modélisant séparément ces deux éléments, sur le même graphe.

2. Les paramétrisations et les interpolations de ces deux éléments sont indépendantes l'une de l'autre, leur seul point commun étant d'être supportées par le même graphe. L'*enveloppe* d'un objet est ainsi modélisée en paramétrisant le *backbone* d'une part, les *sections* d'autre part. Les points caractérisant cette *enveloppe* décrivent la surface définissant le contour de l'objet et la géométrie du *gshape* associé.

3. Ces deux paramétrisations reposent sur les principes généraux de DSI qui permettent de "figer" la position ou le *contour* d'une *vertèbre* au cours de la modélisation et selon la volonté de l'utilisateur. Elles s'appréhendent de façon naturelle.

4. Relativement peu de paramètres sont nécessaires pour modéliser un objet de forme complexe. Ainsi, si l'on considère l' objet présenté dans la Figure1.16 (b), on remarque que le nombre de paramètres vectoriels de contrôle est de:

   (a) 29 *nœuds de contrôles.*

   (b) 16 *sections de contrôle.*

   45 paramètres vectoriels suffisent donc à décrire une forme qui n'est pas formalisable à l'aide d'une expression analytique simple. Cela est un nombre relativement faible comparé à celui qu'auraient engendré des méthodes classiques de type Bézier ou Bspline.

5. L'objet *gshape* introduit une manière de modéliser une surface de manière **globale**, tout en respectant des données de type "coupe sériée". Il lève donc le problème posé à la section 1.3.6 à propos de l'action locale des interpolateurs dits "classiques", *DSI* compris.

1.7 Résumé du chapitre

La méthode *gshape* est basée sur le fait que beaucoup d'objets naturels, englobant un volume fermé, peuvent se décomposer en deux éléments distincts:

1. le *backbone* de l'objet : C'est une ligne conductrice qui correspond physiquement à une ligne imaginaire servant de "guide" à *l'enveloppe* de l'objet, et autour de laquelle il est construit. Cette ligne est matérialisée par un ensemble de nœuds liés entre eux.

2. les *sections* de l'objet : Elles correspondent physiquement à un ensemble de sections planaires qui coupent entièrement l'objet en passant par un nœud du *backbone*. Elles dessinent un *contour* autour de ce nœud. La plan des *sections* n'est pas quelconque et ne varie que légèrement d'une *section* à une autre.

L'*enveloppe* de l'objet découle de la géométrie des *sections* et de celle du *backbone*. On considère donc qu'il est possible de la modéliser en interpolant le *backbone* d'une part, les *vertèbres* d'autre part. D'un point de vue mathématique, ceci se traduit de la manière suivante:

1. On considère un graphe Ω unidimensionel dont chaque nœud est représenté par une *vertèbre*, porteuse de l'information. Ce graphe sert de support à l'interpolation de deux fonctions distinctes, la fonction $\varphi = \{\varphi(k), \forall k \in \Omega\}$ d'une part, et la fonction $V = \{V(k), \forall k \in \Omega\}$ d'autre part. φ décrit la géométrie du *backbone* et V, celle des *sections*. Les paramètres de l'interpolation sont des *nœuds de contrôle* pour le *backbone* et des *sections de contrôle* pour les *sections*. L'objet *gshape* est en fait une représentation du graphe Ω.

2. La méthode d'interpolation utilisée est basée, dans les deux cas, sur la méthode *DSI* . En ce qui concerne l'interpolation des *sections* elle a été étendue afin de prendre en compte le fait que:

   (a) On interpole des vecteurs à $3 * N$ dimensions où N est le nombre de *ribs* par *contour*;

   (b) Chacun de ces vecteurs est interprété comme N vecteurs coplanaires.

   Le plan d'une *section* est interpolé également, conduisant à une variation régulière de la direction d'une *section* à celle de ses voisines. Les deux interpolations sont indépendantes l'une de l'autre, tant au niveau des paramètres qu'elles nécessitent que de leur définition mathématique. En effet, l'interpolation de V ne dépend pas

de la position des nœuds de Ω. Le seul lien entre ces deux interpolations est le fait qu'elles sont supportées par la même topologie.

3. Physiquement, le *backbone* d'un *gshape* définit
   (a) La topologie de Ω, c'est-à-dire la manière dont les nœuds sont liés entre eux.
   (b) La géométrie des nœuds de Ω, c'est-à-dire la position dans l'espace de chacun des nœuds de Ω. Celle-ci est traduite par la valeur de φ sur Ω.
4. Physiquement, l'*enveloppe* d'un *gshape* représente la surface délimitant les contours d'un objet *gshape* et définit sa géométrie. Elle est déterminée par l'expression de V et φ.
5. Enfin, nous avons mis en place une terminologie particulière où une *vertèbre* représente un nœud de Ω et l'information nécessitée par DSI au cours de l'interpolation. On parlera de plus du *contour* associé à une *vertèbre* et défini par un nombre constant de *ribs*.

Chapitre 2

Outils de Modélisation de l'objet gshape

2.1 Présentation du chapitre

Nous allons présenter dans ce chapitre les outils développés pour modéliser et paramétriser interactivement l'objet *gshape*. Nous présenterons d'abord comment il est possible de prendre en compte une surface de référence dans la définition d'un *gshape*. Nous dirons ensuite quels sont les moyens d'agir sur la géométrie du *backbone* d'une part, des *sections* d'autre part. Nous montrerons enfin comment on peut matérialiser l'*enveloppe* d'un *gshape* une fois qu'il a été modélisé.

2.2 Introduction

Nous avons vu en quoi l'objet *gshape* permet de décrire et de modéliser le contour d'un corps de forme complexe limitant un volume fermé dans l'espace. Il s'agit maintenant de définir les outils indispensables à sa modélisation interactive. Ils doivent permettre:

1. L'entrée de données de type *section de contrôle* ou *nœuds de contrôle*.

2. Des opérations portant plus spécifiquement sur ces données.

3. Des opérations modifiant le *gshape* dans son ensemble.

D'autre part, un certain nombre de propriétés explicitées dans la suite de ce chapitre permettent l'utilisation de l'objet *gshape* dans la représentation de corps géologiques. L'ensemble des ces nouveaux outils est présenté dans la suite de ce chapitre.

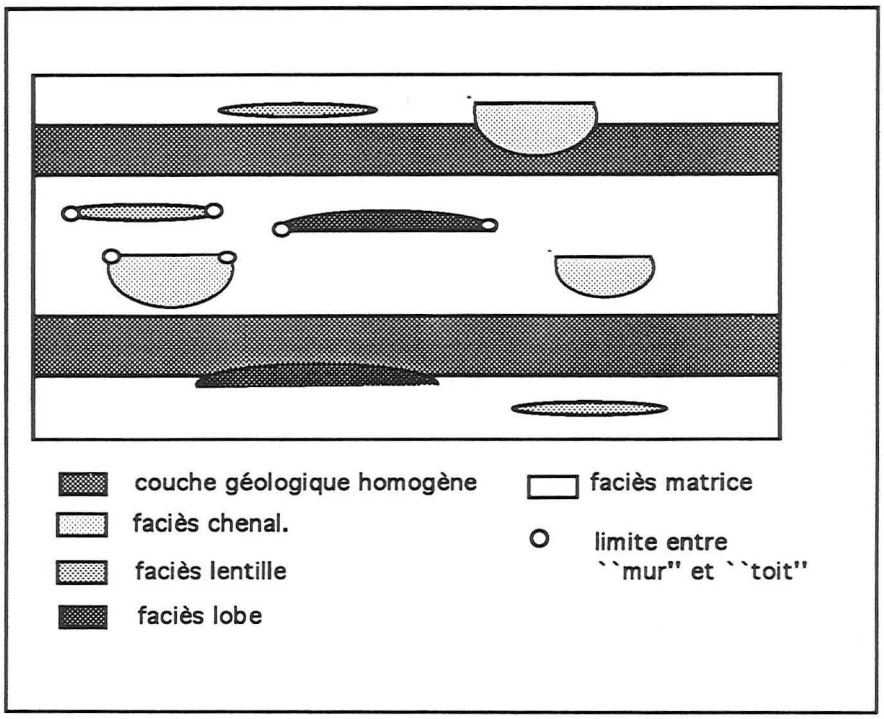


Figure 2.1 Coupe de réservoir contenant les faciès *matrice*, *chenal*, *lobe*, *lentille*.

2.3 Spécification d'une ligne d'horizontalité

2.3.1 Introduction

Dans le cadre d'une modélisation de réservoir pétrolier, les objets de type *gshape* seront utilisés pour caractériser des objets **naturels** dont la génèse obéit à des lois complexes. En ce qui concerne la définition d'un corps géologique, on distingue souvent:

1. Le "mur" du corps géologique: c'est la partie de la surface matérialisant le contour du corps située au *dessous* du plan horizontal au moment du dépôt. Ainsi, si on considère la coupe de réservoir présentée dans la Figure 2.1, on observe que le "mur" des *lobes* est une surface plus ou moins plane et horizontale alors que le "mur" des *chenaux* et des *lentilles* est une surface bombée.

2. Le "toit" du corps: c'est la partie de la surface matérialisant le contour du corps située au *dessus* du plan horizontal au moment du dépôt. Le "toit" des *lobes* et des *lentilles* est alors une surface bombée et celui des *chenaux* est une surface plane.

Cette distinction est très importante lorsque l'on travaille dans un environnement géologique qui n'est plus celui du dépôt. En effet, une surface S représentant un plan horizontal au moment du dépôt peut avoir été déformée sous l'effet de plissements, fractures, ou autres évènements géologiques. Il est alors important lors de la modélisation stochastique,

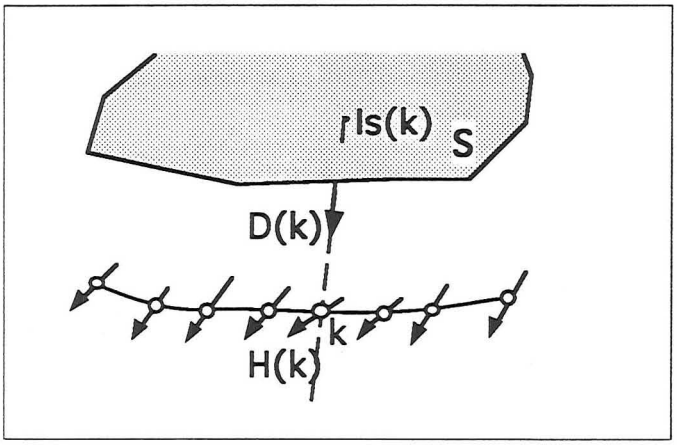


Figure 2.2 Mise en place d'une contrainte de parallélisme entre un *gshape* dont les *vecteurs horizon* $\{H(k)\}$ ont été spécifiés et la surface S, suivant la direction $D(k)$.

de contraindre les hétérogénéités modélisées à "suivre" cette surface S. Cela revient à introduire une "contrainte de parallélisme" entre une hétérogénéité et une surface donnée.

2.3.2 Définition d'un vecteur horizon

La première étape a été de définir, au niveau d'un *gshape*, la limite virtuelle séparant le toit du mur. Nous avons procédé de la façon suivante:

1. Soit une *vertèbre* $k \in \Omega$, porteuse de l'information $\{V_i(k), i \in [0, N[\}$ décrivant son *contour*. On spécifie alors deux *ribs* d'*index* i et j qui définissent un vecteur $H(k)$ tel que:

$$H(k) = V_j(k) - V_i(k)$$

   $H(k)$ est le *vecteur horizon* associé à la *vertèbre* k et est supposé être dans le plan horizontal au moment du dépôt.

2. Au niveau de tout $\alpha \in \Omega$, on définit le *vecteur horizon* par:

$$H(\alpha) = V_j(\alpha) - V_i(\alpha)$$

La surface séparant le toit du mur d'un *gshape* dont les *vecteurs horizons* sont définis est alors la surface passant par tous les *vecteurs horizon*. Un *gshape* dont les *vecteurs horizon* ont été spécifiés est présenté dans la Figure 2.4 (a).

2.3.3 Contrainte de parallélisme dans une direction donnée

Nous considérerons ici les hypothèses de travail suivantes:

1. La position de chaque *vertèbre* k de Ω est projetée sur une surface S suivant une direction qui peut elle-même être une fonction de k et qui est interpolée ou non par DSI . Nous noterons $D(k)$ cette direction.

2. A chaque *vertèbre* k de Ω correspond un point d'impact $I_S(k)$ sur la surface S, selon la direction $D(k)$.

Ces hypothèses sont illustrées par la Figure 2.2. Contraindre un *gshape* à être parallèle à une surface S revient alors à:

1. Contraindre son *backbone* à être parallèle à S. Deux étapes sont nécessaires:

   (a) Calculer la distance moyenne algébrique d séparant la position des *vertèbres* du *backbone* de la surface S:

   $$d = \frac{1}{|\Omega|} \cdot \sum_{\alpha \in \Omega} (\varphi(\alpha) - I_S(\alpha)).D(\alpha)$$

   où $|\Omega|$ est le nombre d'éléments de Ω.

   (b) Translater la position de chaque *vertèbre* k afin qu'elle soit à une distance d du *backbone* selon la direction donnée par $D(k)$:

   $$\varphi(k) = I_S(k) + d.D(k)$$

2. Contraindre les *vecteurs horizon* $H(k)$ de chaque *vertèbre* k à être parallèle à S. Cette étape est plus délicate que la précédente. L'algorithme mis en œuvre est développé ci-après et illustré par la Figure 2.3.

Soient:
- Ω : Le graphe représenté par le *gshape*
- $\{V_i(k), i \in [0, N[\}$: Les *ribs* décrivant le *contour* de la *vertèbre* k
- $T(k)$: la tangente au *backbone* en k
- $H(k)$: le *vecteur horizon* en k

On suppose d'autre part que $H(k).D(k) \neq 0$.

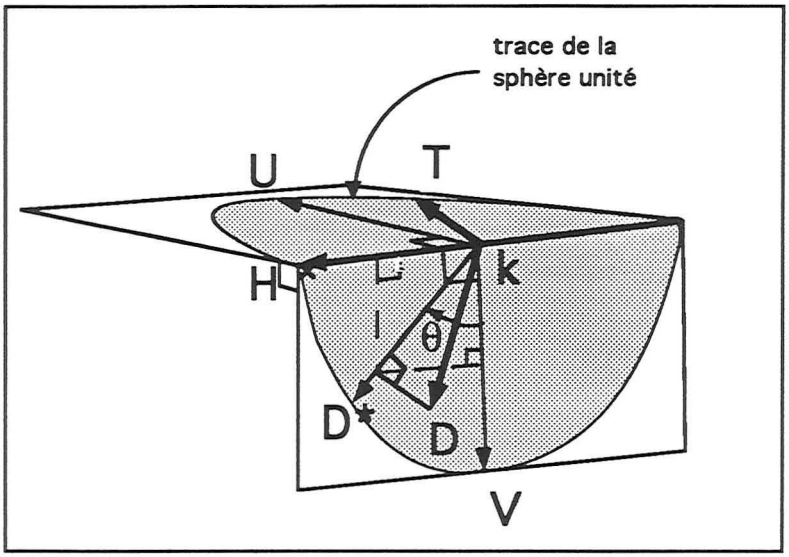


Figure 2.3 illustration de l'algorithme de parallélisation d'un *gshape*. T, D et H sont respectivement la *tangente*, la *direction de projection* et le *vecteur horizon* en k.

pour tout $k \in \Omega$

. Calcul de V, vecteur unitaire perpendiculaire à $T(k)$ et $H(k)$:

$U = \frac{H(k) \wedge T(k) \wedge H(k)}{||H(k) \wedge T(k) \wedge H(k)||}$
$V = \frac{H(k) \wedge U(k)}{||H(k) \wedge U(k)||}$;

. Calcul de $D^*(k)$, projection de $D(k)$ dans le plan $(V, H(k))$:

soit $v = V.D(k)$ et soit $h = H(k).D(k)$;
$D^* = \frac{v.V + h.H(k)}{||v.V + h.H(k)||}$

. Calcul de l'axe et de l'angle de rotation transformant V en D^* et donc, rendant H orthogonal à $D*$:

soit $a = V \wedge D^*$ et soit $A = \frac{a}{||a||}$;
soit $\theta = \arcsin(||a||)$;
si $(V.D^* \leq 0), \theta = -\theta$; finsi.

. Soit $R(A, \theta)$ la rotation d'axe A, d'origine $\{0, 0, 0\}$ et d'angle θ, on transforme les vecteurs $\{V_i(k)\}$ par $R(A, \theta)$.

pour tout $\{V_i(k)\}$

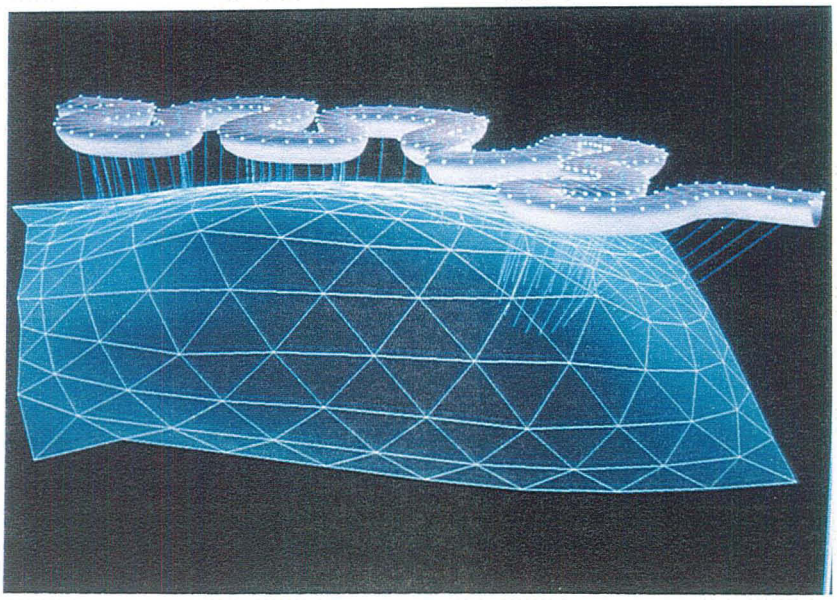
. Soit $V_i'(k)$ la transformée de $V_i(k)$ par $R(A, \theta)$;

. $V_i(k) = V_i'(k)$;

finpour.

finpour.

(a)



(b)

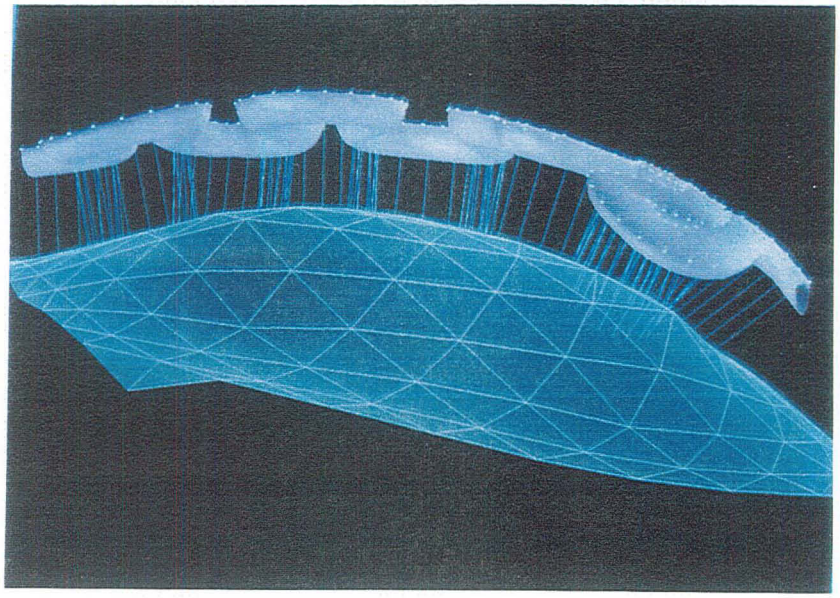


Figure 2.4 (a) *gshape* dont les *vecteurs horizon* ont été spécifiés avant application de la contrainte de parallélisme. (b) *gshape* après application de la contrainte de parallélisme.

La Figure 2.4 (b) présente un exemple de *gshape* contraint à être parallèle à une surface.

2.4 Méthodes d'initialisation d'un *gshape*

2.4.1 A partir d'une ligne polygonale

Cette méthode d'initialisation considère que la modélisation du *backbone* est "première" par rapport à celle des *sections*. Dans cette optique, la modélisation du *backbone* précède toujours celle des *sections*.

On considère que l'on dispose déjà d'une ligne polygonale composée d'*atomes*[1]. Les *nœuds de contrôle* de cette ligne auront été fixés afin d'obtenir la géométrie globale souhaitée du futur *backbone*. La méthode utilisée pour initialiser un *gshape* à partir d'une ligne polygonale est la suivante:

1. initialisation du graphe correspondant au *gshape*. Il s'agit de sa définition topologique.
2. initialisation des *vertèbres* du graphe en fonction des informations contenues dans les *atomes* de la ligne polygonale. En particulier, les *nœuds de contrôle* sont hérités de la ligne. De même, le voisinage d'une *vertèbre* est topologiquement identique à celui de l'*atome* dont elle est issue.
3. initialisation, pour chaque *vertèbre* k, des informations supplémentaires qu'elle contient. Il s'agit de:
   (a) $T(k)$, la tangente orientée au *backbone* en k; Elle est calculée de manière à respecter les conventions citées en 1.4.5.
   (b) $n(k)$, la normale au *contour* en k; elle est initialisée de façon à ce que $n(k) = T(k)$.
   (c) $\{V_i(k), i \in [0, N[\}$, les N *ribs* ordonnés définissant le *contour* de k.
   (d) la spécification que la *vertèbre* k n'est pas une *section de contrôle*.

Nous initialisons les $\{V_i(k)\}$ associés à chaque *vertèbre* k de la manière suivante:

1. Dans le plan orthogonal à $n(k)$, on crée N *ribs* dont les directions varient de façon à ce qu'ils décrivent le contour d'un cercle autour de k et dont l'indexation suit les conventions données en 1.4.5. Le module des vecteurs est le même pour tous les *ribs* et est fonction d'un pourcentage de la longueur totale du *backbone*.
2. Au niveau du *gshape*, les corrélations[2] entre *ribs* sont faites de manière à ce que l'on obtienne une *enveloppe* qui se rapproche le plus possible d'un *tube*.

[1] voir section 1.3.5
[2] voir 1.5.4

Cette méthode est en fait un moyen de donner une solution initiale aux *sections* d'un *gshape*. Il est ensuite possible de modifier et de manipuler le *gshape* ainsi obtenu. Ainsi, le *gshape* présenté dans la Figure 1.15 (a) a été initialisé à partir d'une ligne polygonale. On a ensuite spécifié trois *sections de contrôle* dont on a modifié le *contour*. La Figure 1.15 (b) présente l'*enveloppe* du *gshape* après interpolation des *sections*.

2.4.2 A partir de plusieurs lignes polygonales fermées

On considère cette fois que les *sections* sont modélisées avant le *backbone*.

Description

On suppose que l'on dispose de lignes polygonales fermées. Elles sont par exemple issues de scannérisation de coupes sériées provenant d'une base de donnée quelconque ou de saisies interactives (voir Figure 2.6, page 54 (a)). On considère que pour chaque ligne, les *atomes* qui la décrivent sont indexés et que leur index décrit le sens de parcours de la ligne. La démarche suivie pour créer un *gshape* à partir de lignes fermées est la suivante:

1. Définir la topologie du *gshape*, en matérialisant son *backbone*:
   (a) Création de *vertèbres* aux centres de gravité de chacune des lignes fermées considérées.
   (b) Spécification de la topologie du *backbone*: spécification du voisinage de chaque *vertèbre*.
2. Compléter l'information en chaque *vertèbre* k:
   (a) calcul de la tangente au *backbone* $T(k)$ de manière à respecter les conventions données en 1.4.5.
   (b) extraction de N *atomes* ordonnés, caractéristiques de la forme de la ligne polygonale décrivant le *contour* à lui associer (voir plus bas).
   (c) calcul du plan moyen dans lequel sont contenus ces *atomes*. Projection de leurs positions sur ce plan. Calcul des N vecteurs correspondant à ces nouvelles positions et qui décriront le *contour* de k.
   (d) spécifier la normale à la *vertèbre* $n(k)$ comme étant la normale au plan précédant.
   (e) spécifier que la *vertèbre* est une *section de contrôle* et un *nœud de contrôle* pour *DSI* .
3. Possibilité de parcourir toutes les *vertèbres* et de corréler certains *ribs* entre eux; pour une *vertèbre* k, cela revient à spécifier les *index* des *ribs* $V_i(k)$ en fonction des *index* $V_i(\alpha), \forall \alpha \in N(k)$, où $N(k)$ est l'ensemble des *vertèbres* appartenant au voisinage de k (voir section 1.5.4).

L'étape 2b, qui consiste à extraire, pour chaque ligne , les N *atomes* ordonnés décrivant au mieux sa géométrie est décrite ci-dessous:

Extraction de N atomes d'une ligne polygonale fermée

Nous avons choisi de développer l'algorithme suivant (voir Figure 2.5):

Soient
- L : l'ensemble des M *atomes* $\{L_i, i \in [\,0, M\,[\}$ d'une ligne polygonale fermée. Ces *atomes* sont ordonnés de telle façon que *l'atome* L_{i+1} ait pour voisins les *atomes* L_i et $L_{[i+2]\%M}$;
- f : une fonction retournant l'index d'un *atome* de L telle que $f(L_i) = i$;
- P : l'ensemble des N atomes $\{P_k, k \in [0, N[\}$ de L à extraire. Ces points sont ordonnés de telle façon que:

$$f(P_k) < f(P_{[k+1]\%N}) < f(P_{[k+2]\%N})$$

- $|P|$: Nombre d'élements de l'ensemble P.

On considère que la ligne poygonale est fermée, que $M > N$ et que $|P| = 0$.

. Sélectionner les deux *atomes* de L les plus éloignés.

. Inclure ces deux *atomes* dans l'ensemble P. Ils deviennent alors P_0 et P_1

tant que $(|P| < N)$

Pour chaque paire d'*atomes* $(P_i, P_{[i+1]\%|P|}), i \in [0, |P|[$

Pour $k \in]\, f(P_i), f(P_{[i+1]\%|P|})\,[$

. sélectionner k tel que l'aire du triangle:

$$(P_i, L_k, P_{[i+1]\%|P|}) = A_k$$

soit maximale.

. stocker A_k dans l'ensemble A des aires calculées;

finpour.

. finpour.

. Sélectionner k tel que $A_k, \forall A_k \in A$ soit maximale

. éliminer A_k de l'ensemble A ;

. insérer L_k dans P, et réordonner les $\{P_i\}$ dans l'ordre croissant des index $\{f(P_i)\}$

fintantque.

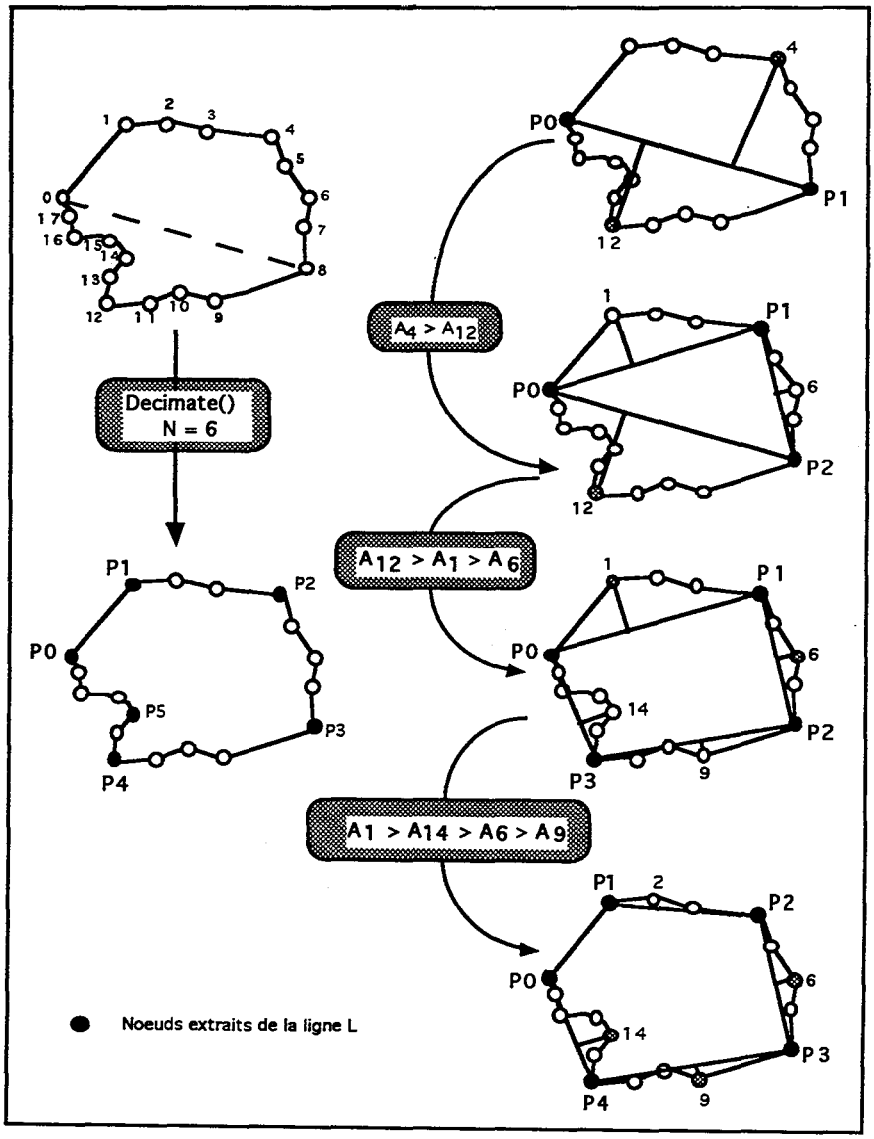


Figure 2.5 Exemple d'application de l'algorithme. Cas où $M = 18$ et $N = 6$.

(a)

(b)

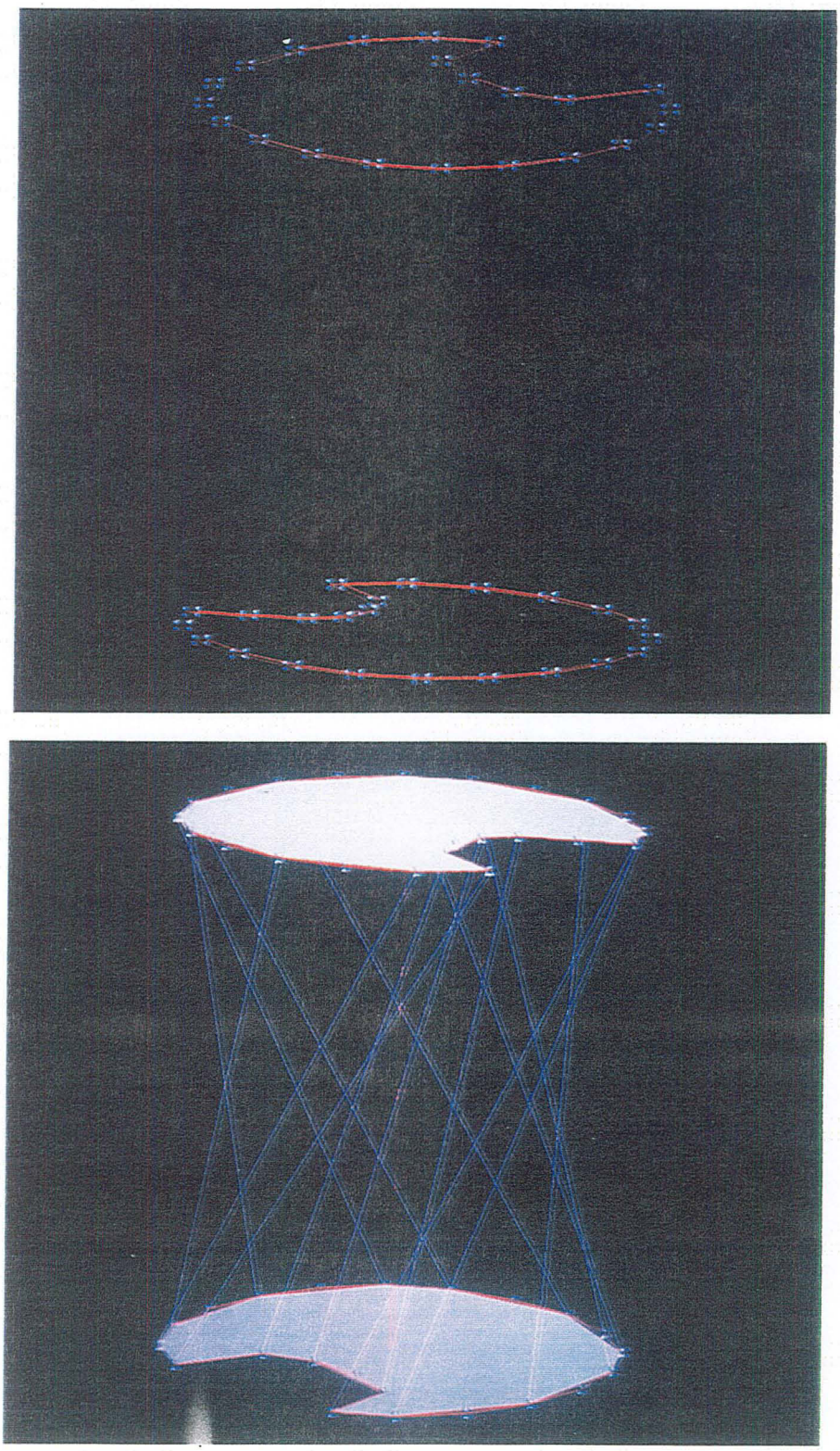
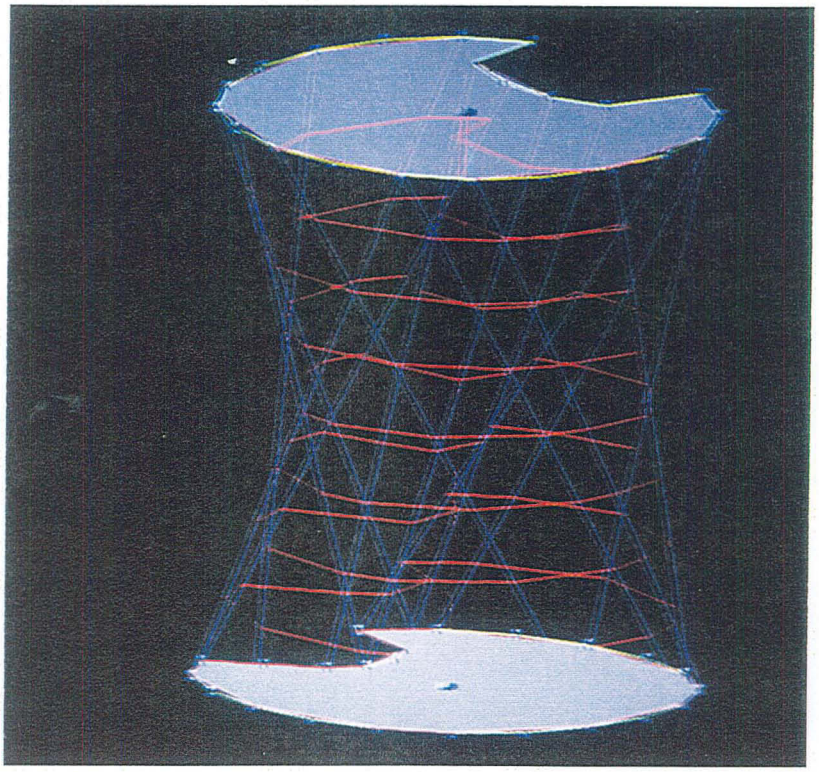


Figure 2.6 (a) Exemple de lignes polygonales fermées utilisées pour initialiser les *sections* de contrôle d'un *gshape* (b) *gshape* initialisé d'après les deux lignes polygonales fermées. Les positions de 15 *atomes* par ligne constituent les extrémités des *ribs* de chaque *section*.

(a)



(b)

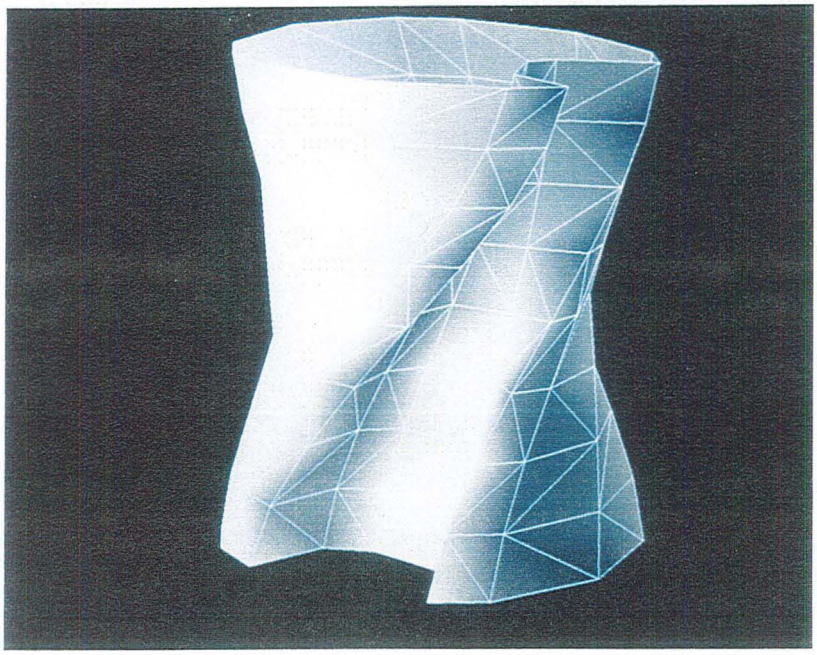


Figure 2.7 (a) *gshape* après une opération de raffinement de maillage. Des *sections intermédiaires* ont été rajoutées (b) matérialisation de l'*enveloppe* d'un *gshape* par une surface triangulée.

On obtient ainsi N *atomes* ordonnés $\{P_0, P_1, \ldots, P_N\}$ qui caractérisent au mieux la géométrie de la ligne polygonale. L'étape 2c de l'algorithme complète ensuite l'information de la *vertèbre k* en calculant les coordonnées des vecteurs dont l'origine est k et qui décrivent le contour formé par les *atomes* $\{P_i, i \in [0, N[\}$. On note cependant l'existence d'un cas particulier lorsque tous les *atomes* dont l'index est compris entre $] f(P_i), f(P_{[i+1]\%|P|}) [$ sont alignés sur le segment formé par les *atomes*:

$$[P_i, P_{i+1}], \forall i \in [0, |P| [$$

Le critère de "l'aire maximale" ne peut alors plus être appliqué. Dans ce cas, il conviendra de le remplacer par celui de "distance maximale" définie en tout nœud k par D_k. Il est défini comme suit:

On sélectionne k tel que la distance euclidienne de L_k à P_i se rapproche le plus de la distance euclidienne de L_k à $P_{[i+1]\%|P|}$.

complexité de l'algorithme:

Cet algorithme est en $O(M^2)$ (à cause de l'étape de selection des deux *atomes* de la ligne les plus éloignés) mais cela n'est pas un handicap car dans la pratique, M est suffisamment petit pour que le calcul puisse s'executer en temps réel.

intérêt de la méthode

1. Il est possible de prendre en compte dans la construction d'un *gshape* des contours non nécessairement planaires, et de complexité arbitraire, comme l'illustre la Figure 2.6 (a). L'algorithme étant étudié pour extraire les *atomes* caractéristiques des contours donnés par les lignes polygonales, leurs géométries seront préservées pour peu qu'un nombre N de *ribs* suffisamment élevé soit utilisé. Un *gshape* initialisé à partir de deux lignes polygonales complexes est présenté dans la Figure 2.6 (b).

2. Il est possible de prendre en compte des vecteurs qui "se croisent" dans la définition d'une *section* d'un *gshape*. Ainsi dans la Figure 2.8, le *contour* de la *vertèbre* est décrit par les sommets de 1 à 12. Cependant, les vecteurs correspondant aux sommets 6 à 10 ne se succèdent pas dans l'ordre logique de ces sommets mais se "croisent".

2.5 Modelisation et déformation d'un gshape

La section précédente illustre les différentes façons d'initialiser un *gshape*. La suite logique consiste à modéliser et à déformer ce *gshape* afin d'imposer à l'*enveloppe* la géométrie désirée. La déformation s'effectue en intervenant sur:

1. La topologie au voisinage de certaines *vertèbres*. Il s'agit de la définition des connections entre *vertèbres*.

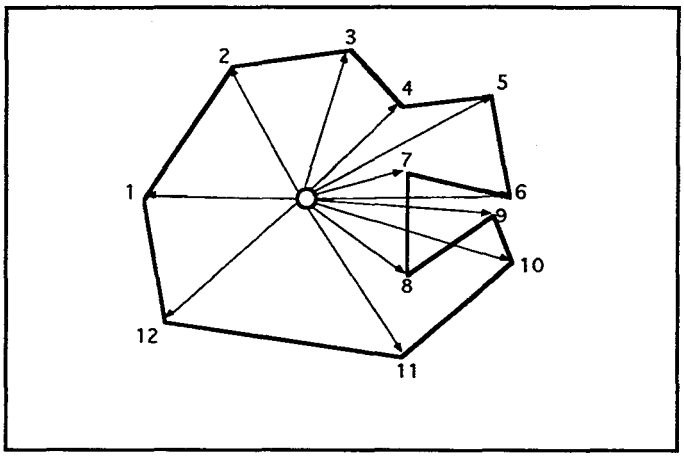


Figure 2.8 Exemple de *section* composée de 12 *ribs* et dont les *ribs* d'index $6, 7, 8, 9, 10$ ne se succèdent pas dans le même ordre que les sommets associés.

2. La géométrie des *vertèbres*. Il s'agit de la définition des *ribs* décrivant le *contour* des *vertèbres*.

3. L'aspect génétique des *vertèbres*. Il s'agit de l'information *section de contrôle* ou *nœud de contrôle*.

2.5.1 En modifiant l'aspect topologique des vertèbres

Comme nous l'avons vu à la section 1.4.4, l'*enveloppe* d'un *gshape* dépend à la fois des résultats de l'interpolation des vecteurs V et φ. Or l'interpolation de ces vecteurs dépend en grande partie de la topologie du *gshape*. Modifier cette topologie est donc un moyen d'agir sur la géométrie de l'*enveloppe* en:

1. Ajoutant des *vertèbres*; cela revient à rajouter des *sections* au *gshape* et donc à "etoffer" le *gshape*. On augmente alors la densité du maillage. Cette opération est décrite dans l'opération *breakSimplex()* de la Figure 2.9. L'interpolation par *DSI* du *backbone* et des *sections* aura alors pour effet de diminuer la "rugosité" de l'*enveloppe* par rapport à la situation précédente.

2. Enlevant des *vertèbres*; l'*enveloppe* est alors définie par un nombre moindre de *sections*. Cette opération est décrite par l'opération *collapseAtom()* de la Figure 2.9. L'interpolation par *DSI* aura alors pour effet d'augmenter la "rugosité" de l'*enveloppe* par rapport à la situation précédente.

3. Modifiant le nombre de voisins d'une *vertèbre* donnée. Cette opération est traduite par les opérations *bridgeSimplex()* et *killAtom()* de la Figure 2.9. Cela revient à agir sur les paramètres pris en considération pour interpoler *backbone* et *sections*.

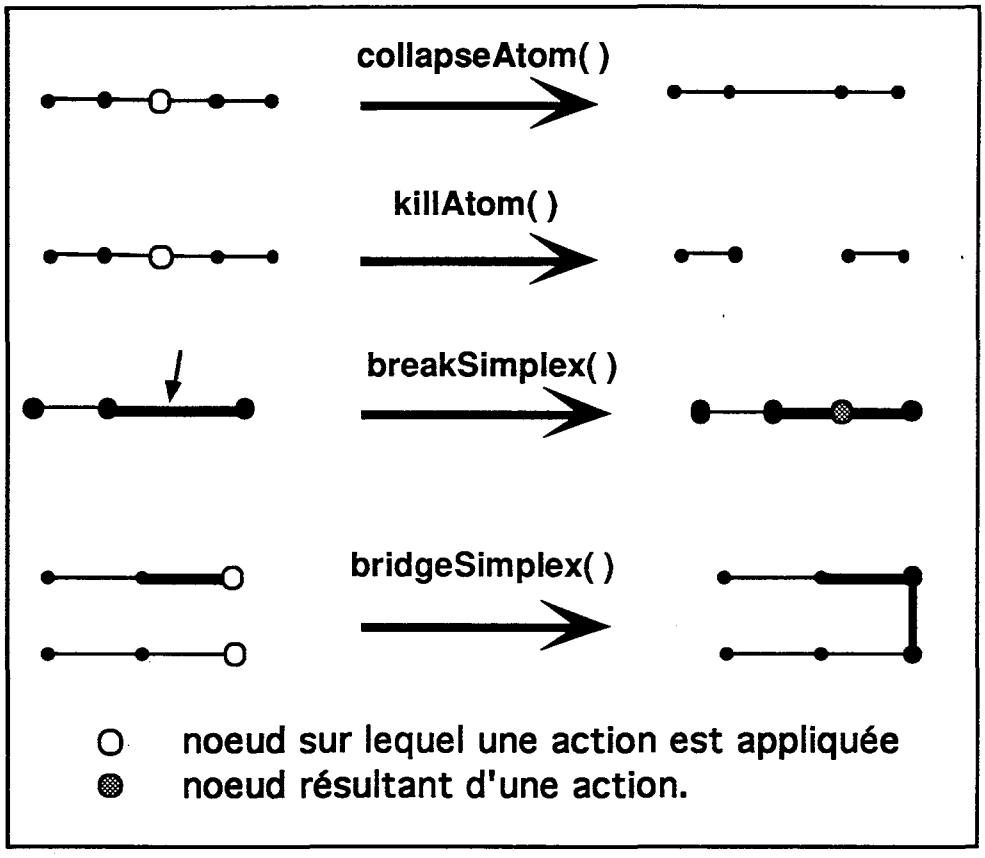


Figure 2.9 Opérations de modification de la topologie du *backbone*.

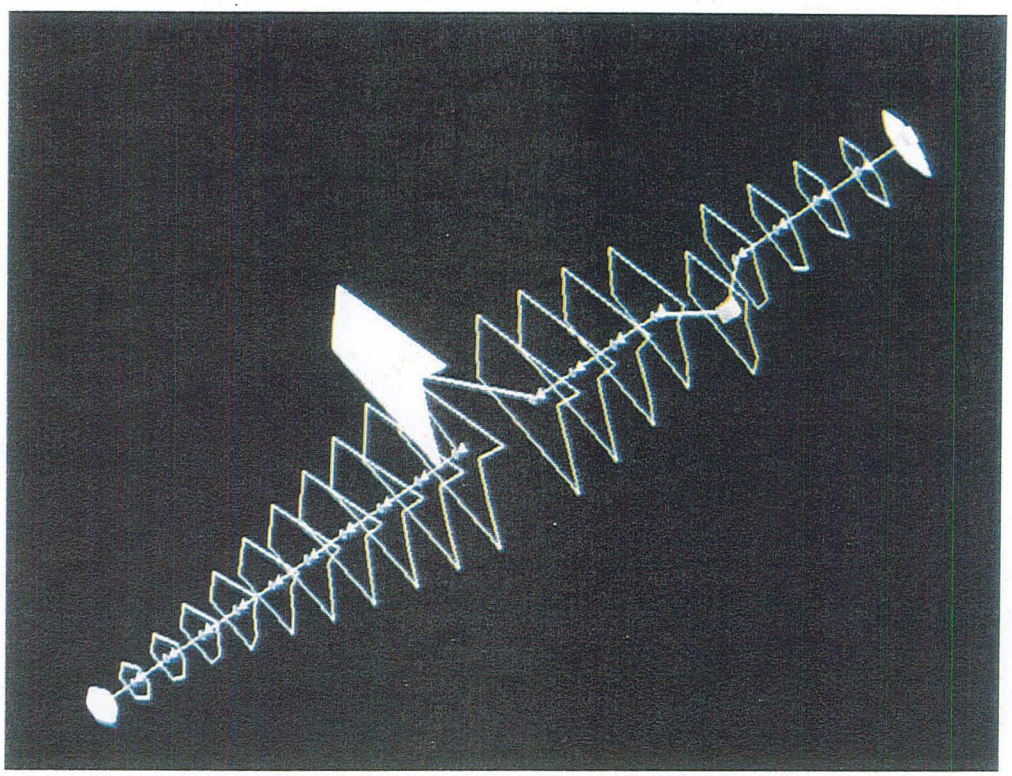
Nous ne rentrerons pas ici dans le détail de ces trois fonctionalités, qui sont des mécanismes généraux de GOCAD. Un exemple de *gshape* où des *sections* intermédiaires ont été rajoutées puis interpolées est cependant donné dans la Figure 2.7 (a).

2.5.2 En modifiant l'aspect géométrique des vertèbres

Modifier la position des vertèbres

En modifiant la position des *vertèbres* d'un *gshape*, on influe sur la géométrie du *backbone*. Ainsi, si les *nœuds de contrôle* sont déplacés, l'interpolation par *DSI* sur le *backbone* modifiera sa géométrie. La géométrie des *sections* ne sera cependant pas affectée, puisque les méthodes d'interpolation sur les *sections* et sur le *backbone* sont indépendantes. Un exemple de *gshape* dont le *backbone* a été réinterpolé après déplacement d'un de ses nœuds de contrôle est présenté dans la Figure 2.10

(a)



(b)

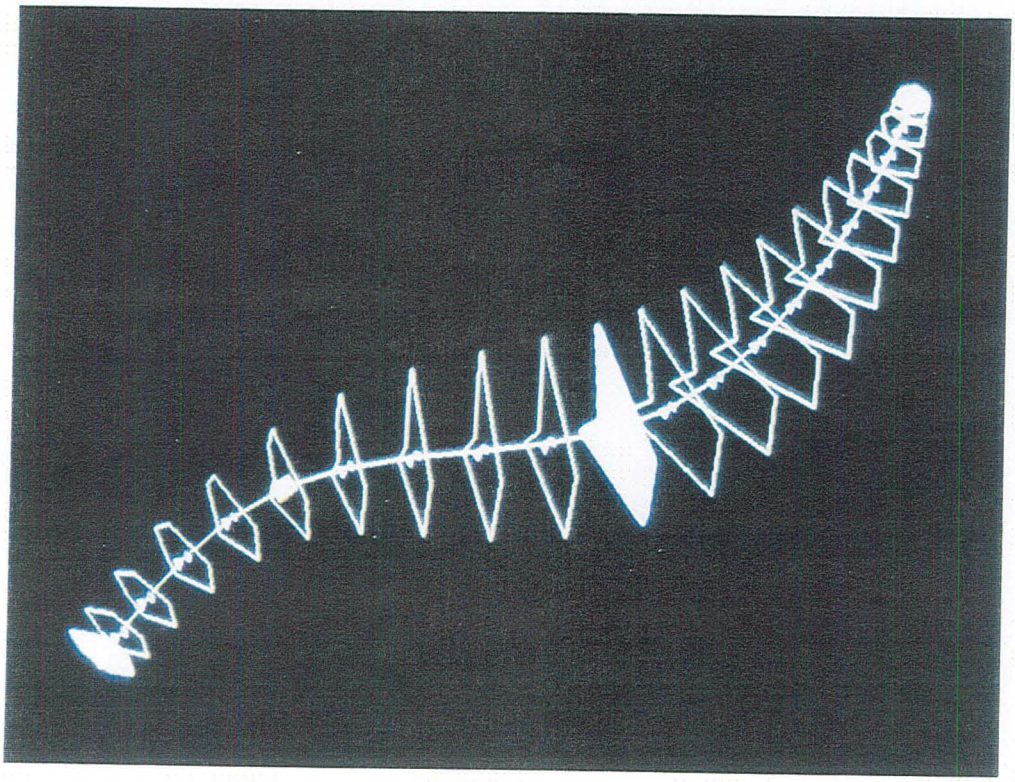


Figure 2.10 (a) *gshape* défini par 4 *nœuds de contrôle* et 3 *sections de contrôle*. (b) réinterpolation du *gshape* après déplacement de deux de ses *nœuds de contrôle*.

Modifier le "contour" des vertèbres

Modifier le *contour* des *vertèbres* d'un *gshape* est un moyen d'influer sur l'*enveloppe* d'un *gshape*. En effet, si une *vertèbre* dont le *contour* est modifié est une *section de contrôle* la géométrie de l'*enveloppe* sera modifiée après interpolation des *sections* par DSI . Plusieurs types d'outils ont été développés pour spécifier et modifier la forme des *sections* d'un *gshape*. Ils sont cités ci-dessous pour une *vertèbre* k donnée.

1. Evolution globale, du *contour*: on applique alors une transformation affine simple (rotation, affinité) à tous les *ribs* $\{V_i(k), i \in [0, N[\}$ de k.

2. Evolution locale du *contour*: on modifie un *rib* particulier de k. Les *sections de contrôle* du *gshape* présenté dans la Figure 1.15 (a) ont été modifiées de cette manière.

3. Attribution d'un *contour* extrait d'une ligne polygonale fermée connue. Cette opération utilise l'algorithme décrit à la section 2.4.2

4. Spécification du fait que le *contour* de k doit être contenu dans un plan dont on connaît un vecteur normal. Il peut ainsi être intéressant de placer une *section* orthogonalement au *backbone*.

5. Spécification d'un *contour* de référence devant servir à initialiser les *contours* de toutes les *vertèbres* d'un *gshape*. Il s'agit de définir une méthode de "propagation" de *contour*.

Les opérations 1 et 2 ne présentent pas de difficultés majeures. Le principe de l'opération 3 a été décrit à travers la section 2.4.2. Nous décrirons par contre plus particulièrement les opérations 4 et 5.

Positionner un contour perpendiculairement au "backbone"

Soit k une *vertèbre* dont la *normale* à la *section* est $n(k)$ et la tangente au *backbone* est $T(k)$. Soient:

$$\{V_i(k), i \in [0, N[\}$$

les N ribs décrivant son *contour*. On suppose connu le *vecteur horizon*[3] $H(k)$.

Le positionnement des $\{V_i(k)\}$ perpendiculairement au *backbone* se ramène au calcul de la rotation transformant $n(k)$ en $T(k)$. Une approche de "force brute" serait de calculer cette transformation; cependant, dans beaucoup de cas, les vecteurs $n(k)$ et $T(k)$ sont confondus ou très proches, rendant le calcul du produit vectoriel $n(k) \wedge T(k)$ numériquement instable.

L'approche que nous préconisons est la suivante (voir Figure 2.11):

[3] voir section 2.3.2

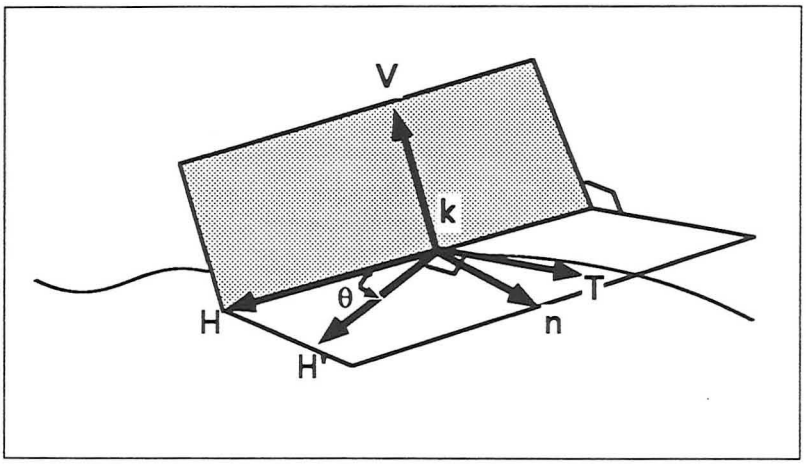


Figure 2.11 Positionner une *section* orthogonalement au *backbone*. T, H et n sont respectivement la tangente, le *vecteur horizon* et la normale de la *vertèbre* k.

. Calcul de $V = H(k) \wedge n(k)$

. Calcul de $H'(k)$, transformée souhaitée de $H(k)$:
$H'(k) = V \wedge T(k)$.

. Calcul de l'axe de la rotation A:

$a = H(k) \wedge H'(k)$
$A = \frac{a}{||a||}$.

. Calcul de l'angle de la rotation $R(A, \theta)$ d'origine $\{0, 0, 0\}$, d'axe A et d'angle θ tel que:

$$\theta = \begin{cases} \arcsin ||\vec{A}|| & \text{si } H(k).H'(k) \geq 0 \\ -\arcsin ||\vec{A}|| & \text{si } H(k).H'(k) \leq 0 \end{cases}$$

. appliquer à tout $V_i(k)$, $R(A, \theta)$.

Le vecteur V est en fait:

1. Inclus dans le plan de la *section* de la *vertèbre*.

2. Orthogonal au *vecteur horizon* $H(k)$. L'axe de rotation A est donc calculé de telle sorte que l'*horizontalité* de $H(k)$ soit préservée au cours de la rotation.

Cette méthode peut s'appliquer dans tous les cas de figure, et ne nécessite aucun test préalable d'orthogonalité entre $T(k)$ et $n(k)$.

Propager un "contour" de référence

Cette opération consiste à "propager" le *contour* d'une *vertèbre* particulière au niveau de toutes les *vertèbres* du *gshape*. En pratique, le *backbone* peut s'avérer très complexe. L'interpolation des *sections* par DSI conduit alors à des variations importantes des *normales* aux *sections* (voir Figure 1.16). De ce fait, le terme de "propager" le *contour* d'une *vertèbre* k ne peut résulter d'une simple copie du *contour* le long des *vertèbres* du *backbone*. Nous effectuons la "propagation" du *contour* de k de la façon suivante:

1. Le *contour* de k est copié aux *vertèbres* satellites de k;
2. au niveau de chaque *vertèbre* k' satellite de k:
   (a) On applique l'opération d'orthogonalisation par rapport au *backbone* (voir section 2.5.2).
   (b) On la "propage" aux *vertèbres* satellites de k';

L'algorithme impliqué est l'algorithme récursif suivant:

Soient:		
	$T(k)$:	la tangente au *backbone* de la *vertèbre* k
	$n(k)$:	la normale au *contour* de la *vertèbre* k
	$S(k)$:	l'ensemble des *vertèbres* définissant les satellites de k
	$\{V_i(k), i \in [0, N[\}$	Les *ribs* de k

ribPropagate(k)

tant qu'il y a des *vertèbres* non traitées:

pour tout $\alpha \in S(k)$

. $V_i(\alpha) = V_i(k), \forall i \in [0, N[$;

. $n(\alpha) = n(k)$;

finpour

. Faire subir à $\{V_i(\alpha), \forall i \in [0, N-1]\}$, la rotation les positionnant orthogonalement au *backbone* (voir section 2.5.2).

. appeler *ribPropagate(α)*

fintantque.

Le gshape de la Figure 1.16 (a) a été produit en propageant, un *contour* prédéfini ayant la forme type d'une coupe transversale de *chenal* (voir Figure 2.1).

2.5.3 En modifiant l'aspect génétique des vertèbres

La manière la plus courante de parvenir à une géométrie désirée de l'*enveloppe* est de modifier l'aspect *génétique*[4] des *vertèbres* d'un *gshape*. Cela revient à préciser et à positionner les *nœuds de contrôle* et les *sections de contrôle* du *gshape*. Les principales étapes conduisant à l'obtention de l'*enveloppe* souhaitée sont les suivantes:

1. Préciser les *vertèbres* considérées comme des *nœuds de contrôle* du *gshape*.
2. Appliquer l'interpolation DSI à la position des *vertèbres* du *backbone* (interpolation du vecteur φ). On définit ainsi la géométrie du *backbone* .
3. Préciser les *vertèbres* considérées comme des *sections de contrôle* du *gshape*.
4. Appliquer l'interpolation DSI aux *ribs* définissant les *contours* des *vertèbres* du *gshape* (interpolation du vecteur V). On obtient ainsi la géométrie de *l'enveloppe*, le plan des *sections* variant de façon progressive (on rappelle que la direction du plan du *contour* d'une *vertèbre* est interpolée également).
5. Reprendre n'importe laquelle des étapes précédentes si l'*enveloppe* exacte souhaitée n'est pas atteinte.

Etant donné l'indépendance totale entre l'interpolation de la géométrie du *backbone*, et celle des *sections*, l'ordre dans lequel sont effectuées les étapes précédentes n'affecte pas le résultat final.

2.6 Matérialisation de "l'enveloppe" d'un gshape

Une fois modélisée, il peut être intéressant de matérialiser l'*enveloppe* d'un *gshape*. Cette matérialisation peut être considérée comme une représentation "figée" du *gshape*. Nous avons mis en place deux méthodes de matérialisation de l'*enveloppe*. La première utilise une représentation surfacique de celle-ci, la seconde une représentation volumique.

2.6.1 Triangulation de l'enveloppe

Soit un *gshape* composé de N_v *ribs* par *vertèbre*, et deux *vertèbres* (α, β) les extrémités d'un *segment* du *backbone*. Soient:

$$\{V^j(\alpha), j \in [0, N_v[\}$$

et:

$$\{V^j(\beta), j \in [0, N_v[\}$$

les *ribs* caractérisant les *contours* des *vertèbres* α et β. Soient de plus:

$$\{\varphi^\nu(\alpha), \nu = x, y, z\}$$

[4] voir section 1.5.2

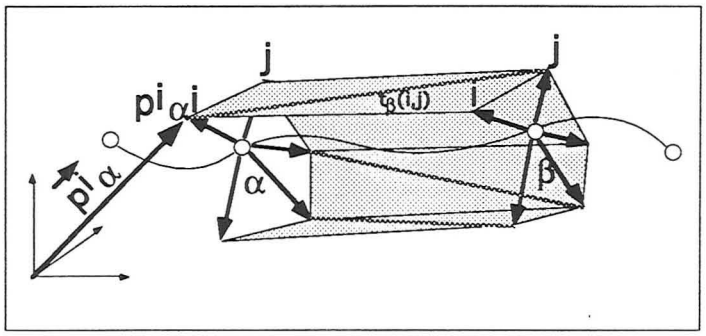


Figure 2.12 Triangulation de l'*enveloppe* d'un *gshape*, entre les *vertèbres* α et β.

et:

$$\{\varphi^{\nu}(\beta), \nu = x, y, z\}$$

les vecteurs décrivant leurs positions. Nous allons montrer que la portion d'*enveloppe* comprise entre ces deux *contours* peut être décrite à l'aide de $N_v \times 2$ triangles. Nous noterons p^i_α le point extrémité du vecteur $V^i(\alpha)$ et $\vec{p}^{\,i}_\alpha = \varphi(\alpha) + V^i(\alpha)$ le vecteur dont l'extrémité est matérialisée par le point p^i_α. Le triangle dont deux des sommets sont composés des points p^i_α et p^j_α sera noté $t_\alpha(i,j)$ (voir Figure 2.12, page 65).

pour tout $i \in [0, N_v[$

. soit $j = (i+1)\%N_v$

. soit $\vec{p}^{\,i}_\alpha = \{\varphi(\alpha) + V^i(\alpha)\}$

. soit $\vec{p}^{\,j}_\alpha = \{\varphi(\alpha) + V^j(\alpha)\}$

. soit $\vec{p}^{\,i}_\beta = \{\varphi(\beta) + V^i(\beta)\}$

. soit $\vec{p}^{\,j}_\beta = \{\varphi(\beta) + V^j(\beta)\}$

. $t_\alpha(i,j) = \{p^\alpha_i, p^\alpha_j, p^\beta_i\}$ et $t_\beta(i,j) = \{p^\alpha_j, p^\beta_i, p^\beta_j\}$

finpour.

La portion d'*enveloppe* comprise entre les *sections* aux *vertèbres* délimitant un *segment* (α, β) du *backbone* est composée des triangles

$$\{\, t_k(i, (i+1)\%N_v), i \in [0, N_v[, k = \{\beta, \alpha\}\,\}$$

et le nombre de ces triangles est bien de $2 \times N_v$. De ce fait, le nombre de triangles nécessaires pour représenter la totalité de l'*enveloppe* d'un *gshape* est égale à:

$$\begin{cases} 2 \times N_v \times N_b & \text{si le } \textit{backbone} \text{ est une ligne polygonale fermée} \\ 2 \times N_v \times (N_b - 1) & \text{si le } \textit{backbone} \text{ est une ligne polygonale ouverte} \end{cases}$$

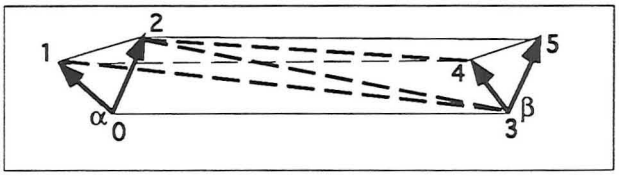


Figure 2.13 Tétraédrisation d'un prisme formé des points (0,1,2,3,4,5).

Deux exemples de *gshapes* triangulés sont présentés dans la Figure 2.7 (b) et la Figure 2.14. Cette dernière présente une *enveloppe* triangulée dont les extrémités ont été formées à l'aide de l'algorithme du "patch" cité en [41]. Nous verrons qu'il est parfois utile de pouvoir représenter un *gshape* sous la forme d'une succession de triangles, notamment en ce qui concerne le calcul de l'intersection entre un *gshape* et un segment.

2.6.2 Tétraédrisation de l'enveloppe

Soit un *gshape* composé de N_v *ribs* par *section*. On suppose d'autre part, que la position des *vertèbres* du *gshape* est située à l'intérieur du *contour* qui lui est associé (voir Figure 2.8). Soient deux *vertèbres* (α, β) les extrémités d'un *segment* du *backbone*. Considérons le prisme formé par les 6 points

$$\{p_i(\alpha), p_j(\alpha), p_i(\beta), p_j(\beta), \varphi(\alpha), \varphi(\beta)\}$$

Comme le montre la Figure 2.13, un tel prisme est composé des trois tétraèdres suivants:

1. Le tétraèdre de sommets $(\alpha, p_i(\alpha), p_j(\alpha), \beta)$;
2. Le tétraèdre de sommets $(p_i(\alpha), p_j(\alpha), \beta, p_i(\beta))$;
3. Le tétraèdre de sommets $(p_j(\alpha), \beta, p_i(\beta), p_j(\beta))$;

La portion d'*enveloppe* comprise entre ces deux *vertèbres* peut donc être décrite à l'aide de $N_v \times 3$ tétraèdres. De ce fait, le nombre de tétraèdres nécessaires pour représenter la totalité de l'*enveloppe* d'un *gshape* est égale à:

$$\begin{cases} 3 \times N_v \times N_b & \text{si le } \textit{backbone} \text{ est une ligne polygonale fermée} \\ 3 \times N_v \times (N_b - 1) & \text{si le } \textit{backbone} \text{ est une ligne polygonale ouverte} \end{cases}$$

Nous verrons que tétraédriser l'*enveloppe* d'un *gshape* peut s'avérer très intéressant, notamment pour initialiser des grilles tridimensionnelles à partir d'un *gshape*[5].

2.6.3 Remarque

Les méthodes mises en place pour trianguler ou tétraédriser l'*enveloppe* d'un *gshape* en

[5]voir partie "Conclusion" de ce manuscrit

trois dimensions sont très simples. Ceci est dû à la contrainte imposée lors de la conception de l'objet *gshape* qui "force" toutes ses *sections* à être définies par le même nombre de *ribs*. Cette contrainte simplifie grandement la mise en œuvre d'algorithmes tels que la tétraédrisation d'un volume. Nous verrons lors de la partie "Conclusion" de ce manuscrit les applications qui peuvent être faites de ces algorithmes.

2.7 Conclusion

La panoplie d'outils présentés dans ce chapitre introduit une grande flexibilité dans la modélisation d'un *gshape*. Ils offrent à l'utilisateur la possibilité de matérialiser une "vue de l'esprit" en intégrant, dans la forme créée, un maximum de son "Art". D'autre part, on peut faire les observations suivantes:

1. L'*enveloppe* d'un *gshape* est très facilement matérialisée par une surface triangulée. L'objectif de modélisation surfacique du contour d'un corps englobant un volume fermé de l'espace est donc atteint. Il est possible de matérialiser le volume lui même à l'aide de tétraèdres.

2. La façon de modéliser un *gshape* est facilement appréhendée par l'utilisateur et correspond en pratique à une caractéristation de deux éléments physiques distincts, le *backbone* d'une part, les *sections* d'autre part.

3. La spécification de *vecteurs horizon* au niveau de chaque *vertèbre* d'un *gshape* permet de prendre en compte le fait que l'on modélise un objet naturel et matérialise une surface de référence qui sert de base à l'application de nombreux algorithmes.

2.8 Résumé du chapitre

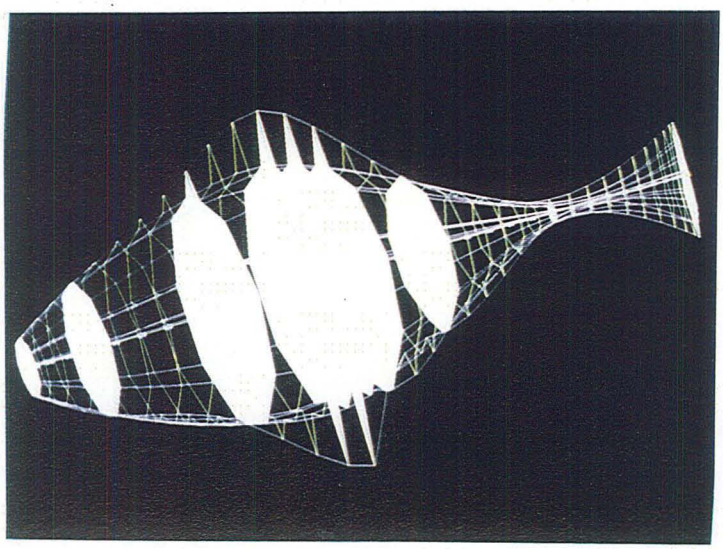
Ce chapitre présente les moyens mis en place pour initialiser, paramétriser et modéliser un objet gshape. Il apparaît que plusieurs approches sont possibles:

1. une approche favorisant la géométrie du *backbone* comme "première" dans la modélisation.

2. une approche favorisant la géométrie des *sections* comme "première" dans la modélisation.

Il est ensuite possible de faire évoluer l'*enveloppe* d'un *gshape* initial en modifiant successivement l'information géométrique, topologique et génétique contenue dans les *vertèbres* du *gshape*. L'*enveloppe* d'un *gshape* est de plus facilement matérialisée à l'aide de triangles pour une modélisation surfacique et à l'aide de tétraèdres pour une modélisation volumique.

(a)



(b)

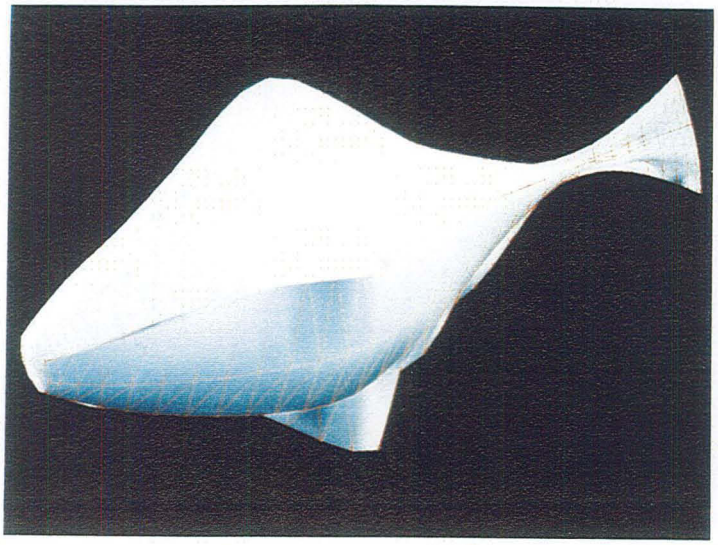


Figure 2.14 (a) *gshape* paramétrisé par deux *nœuds de contrôle* et neuf *sections de contrôle*; (b) matérialisation de l'*enveloppe* par une surface triangulée.

Chapitre 3

Présentation de l'objet "rshape" comme représentant d'une hétérogénéité

3.1 Présentation du chapitre

Nous ferons dans ce chapitre, le lien entre la représentation générale d'un objet limitant un volume fermé dans l'espace, et la représentation spécifique d'une hétérogénéité au sein d'un faciès donné. Nous définirons en effet l'objet *rshape* comme représentant d'une hétérogénéité dans un réservoir. Nous verrons qu'il est une extension d'un objet *gshape* dont la géométrie a été "bruitée" Nous définirons d'abord la nature du "bruit" impliqué, puis nous dirons quel est le lien génétique permettant de passer de l'objet *gshape* à l'objet *rshape*. Nous verrons enfin comment on peut définir la notion de famille d'hétérogénéités.

3.2 Introduction

Le but de ce chapitre est de proposer un moyen de définir l'ensemble des géométries possibles pour une hétérogénéité de réservoir, sans tenir compte de la géométrie des autres hétérogénéités (ce point fera l'objet du chapitre 4). Nous avons défini ce qu'était un *gshape*. Il apparaît comme la représentation générale d'un objet limitant un volume fermé dans l'espace et dont la géométrie est définie par l'ensemble des points composant son *enveloppe*. Cependant, il est probable que des modélisations distinctes d'un *gshape* représentant une hétérogénéité d'un faciès particulier, vont donner lieu à des géométries toutes différentes mais possédant des caractéristiques communes (direction globale d'allongement, degré de sinuosité, tailles ...). Cette incertitude quant à la forme de l'hétérogénéité est dûe à plusieurs facteurs dont les principaux sont les suivants:

1. Toutes les hétérogénéités n'ont pas exactement la même forme au sein d'un faciès donné.

2. De ce fait, les indications que l'on peut donner sur des caractéristiques majeures telles que des directions globales d'allongement ou des tailles, sont très difficilement quantifiables.

3. L'imagination joue un rôle majeur dans la modélisation des corps que l'on trouve dans un réservoir, puisque les données sont très peu nombreuses et incertaines.

Ces raisons font que l'on peut considérer la géométrie d'une hétérogénéité comme un attribut *stochastique*. En effet, plusieurs géométries–chacune honorant des caractéristiques générales– peuvent définir l'*enveloppe* d'une hétérogénéité. Nous présentons dans ce chapitre l'objet mis en place pour définir la notion de *forme stochastique*. Il s'agit de l'objet *rshape*[1], extension de l'objet *gshape* à la représentation spécifique d'une hétérogénéité singulière au sein d'un faciès donné.

3.3 Données prises en compte dans la définition d'un faciès

Nous avons défini la notion de faciès à la section 0.3. Nous aborderons ici la description d'un faciès lorqu'il s'agit d'une famille d'hétérogénéités. Nous avons cité à la section 0.4 les données à prendre en compte dans cette description et nous avons vu que l'on y introduisait les notions de largeur, épaisseur et orientation d'un corps appartenant à un faciès donné. Si l'on considère qu'un objet *gshape* est susceptible de représenter une hétérogénéité de réservoir, il faut tout d'abord définir comment ces paramètres sont calculés pour un tel objet.

3.3.1 Définition des paramètres de tailles et d'orientation

Largeur d'un gshape

Nous définirons la largeur d'un objet *gshape* grâce à la notion de *vecteur horizon* définie à la section 2.3.2. Nous avons vu que chaque *vertèbre* k pouvait être caractérisée par les *index* i et j de deux *ribs* permettant de définir le *vecteur horizon* $H(k)$ tel que:

$$H(k) = V_j(k) - V_i(k)$$

De plus, les *index* i et j sont les mêmes pour toutes les *vertèbres*, ce qui assure une consistance dans l'interpolation des $\{H(k)\}$, pour k décrivant toutes les *vertèbres* du *gshape*. D'autre part, il est logique de considérer que la largeur d'un objet *gshape*–compte tenu du mode de formation d'un corps géologique–se mesure au niveau de chaque *vertèbre* par le module de son *vecteur horizon*.

Soit Ω l'ensemble des *vertèbres* d'un *gshape*, nous définirons alors la largeur l de celui-ci de la manière suivante:

[1] *rshape* signifie *r*éservoir*shape* object

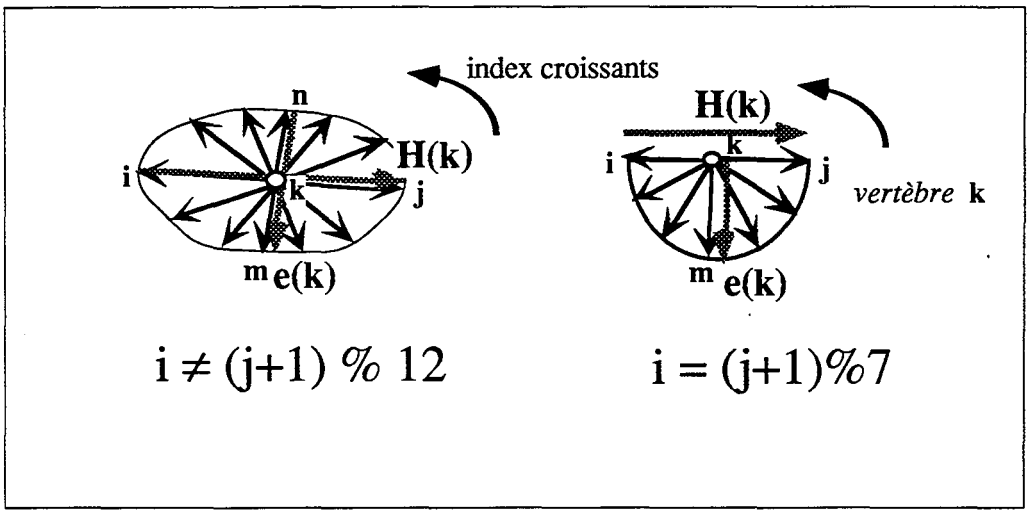


Figure 3.1 Notion d'épaisseur associée à une *vertèbre*. $H(k)$ est le *vecteur horizon* associé à la *vertèbre* k, $e(k)$ est l'épaisseur calculée de k.

pour tout $k \in \Omega$

si k est une *section de contrôle*

Calcul de la largeur l_k de k:

$l_k = ||H(k)||$

finsi.

l = moyenne des $\{l_k\}$.

finpour

La largeur moyenne l d'un *gshape* est donc calculée comme la moyenne des largeurs des *sections de contrôle* du *gshape*.

Epaisseur d'un gshape

L'épaisseur d'un *gshape* sera définie au niveau d'une *vertèbre* k relativement à son *vecteur horizon* $H(k)$.

Soient $i \in [0, N[$ et $j \in [0, N[$ les *index* des deux *ribs* extrémités du *vecteur horizon* $H(k)$ de la *vertèbre* k, l'epaisseur $e(k)$ associée à la *vertèbre* k est calculée de la façon suivante (voir Figure 3.1):

si $i = (j+1)\%N$ ou $(j = (i+1)\%N$

. Calcul des deux index m et n, $\{m, n\} \in [0, N[$ tels que le produit: $(V_m(k) - V_n(k)).H(k)$ soit le plus faible possible.

. $e(k) = V_m(k) - V_n(k)$.

finsi.
sinon

. Calcul de l'index $m \in [0, N[$ tel que: le produit $V_m(k).H(k)$ soit le plus faible possible.

. $e(k) = ||V_m(k)||$.

finsi.

Comme précédemment, l'épaisseur moyenne d'un *gshape* est la moyenne des épaisseurs $e(k)$, k étant une *section de contrôle.*

Rapport $\frac{largeur}{epaisseur}$

Le rapport $\frac{largeur}{epaisseur}$ moyen d'un objet *gshape* se calcule comme étant la moyenne des rapports $\frac{largeur}{epaisseur}$ des *sections de contrôle.*

Direction d'allongement d'un gshape

La direction d'allongement $\vec{D}$ d'un *gshape* se calcule de la façon suivante:

1. Calcul des trois composantes principales du *gshape.* Il s'effectue en considérant tous les points caractérisant son *enveloppe.* Ceux-ci ont été définis à la section 1.4.4.
2. Calcul de $\vec{D}$, tel que $\vec{D}$ = *projection sur un plan horizontal de la plus grande des trois composantes principales.*
3. Normalisation du vecteur $\vec{D}$.

3.3.2 Intégration de ces paramètres

Nous considérerons qu'un faciès contenant des hétérogénéités est caractérisé par:

1. La distribution de caractéristiques de tailles des corps contenus dans le faciès. Nous considérerons que l'on possède:
   (a) Une distribution gaussienne des rapports $\frac{largeur}{epaisseur}$.
   (b) Une distribution gaussienne des épaisseurs.

2. La distribution de directions préférentielles d'allongement des corps au sein du faciès. Dans la suite de ce chapitre nous considérerons que l'on possède une distribution gaussienne des directions d'allongement.

3. Une forme géométrique plus ou moins définie. Dans la suite de ce chapitre nous considérons qu'un objet *gshape* représente la forme la plus probable prise par les hétérogénéités du faciès. Les contraintes imposées pour la définition de l'objet *gshape* sont les suivantes:

   (a) Le *vecteur horizon*[2] est positionné parallèlement à la "largeur" de la forme la plus probable.

   (b) L'épaisseur moyenne du *gshape* correspond à la moyenne de la distribution donnée en 1b.

   (c) Le rapport $\frac{largeur}{epaisseur}$ moyen correspond à la moyenne de la distribution gaussienne donnée en 1a.

   (d) La direction globale d'allongement correspond à la moyenne de la distribution gaussienne donnée en 2.

   Un tel *gshape* sera dorénavant appelé *template* du faciès considéré.

Ainsi, les données quantifiées dont on dispose pour décrire un faciès sont prises en compte dans la définition du *template* associé à ce faciès. Cependant, le reste de la modélisation est de nature intuitive et doit beaucoup à l'imagination et au savoir faire du géologue. Ces deux facteurs peuvent en partie influencer:

1. La géométrie du *backbone*. Le géologue décide de son extension et de la position de ses *nœuds de contrôle*.

2. La géométrie des *sections*. Le géologue décide de la forme donnée aux *sections de contrôle*.

3.4 Définition de quelques transformations de base

Nous allons décrire ici une série de transformations affectant la géométrie d'un *gshape*, tout en respectant certaines caractéristiques de cette géométrie. Elles peuvent être considérées comme des combinaisons des quatre transformations élémentaires suivantes:

1. *translation* aléatoire de l'objet *gshape* au sein d'un volume donné.

2. Modification aléatoire de la taille des *ribs* associés aux *contours* des *vertèbres* du *gshape* par une opération d'*affinité* homogène.

3. *rotation* aléatoire du *backbone* par rapport à un axe vertical;

[2] voir section 2.3.2

4. perturbation aléatoire de la position de ses *nœuds de contrôle* dans un plan donné. Nous appelerons cette transformation élémentaire *déformation.*

Soit un *gshape* donné et soit Ω le graphe défini par celui-ci. Nous rappelons que:

$$\varphi(k) = \{\varphi(k)^x, \varphi(k)^y, \varphi(k)^z\}$$

représente la position de la *vertèbre* $k \in \Omega$ et que

$$\{V_i(k), i \in [0, N[\}$$

sont les *ribs* définissant son *contour*. Les fonctions $\varphi = \{\varphi(k), k \in \Omega\}$ et $V = \{V(k), k \in \Omega\}$ où:

$$V(k) = \{V_0^x(k), V_0^y(k), V_0^z(k), \ldots, V_{N-1}^x(k), V_{N-1}^y(k), V_{N-1}^z(k)\}$$

sont interpolées de manière à définir la géométrie du *backbone* et des *sections* du *gshape*. Le détail des quatre transformations est donné ci-dessous.

3.4.1 Translation

La géométrie dans l'espace d'un *gshape* représentant une hétérogénéité n'est pas déterminée de façon unique. Au contraire, plusieurs localisations peuvent rendre compte des positions d'une telle hétérogénéité. Par cette transformation, on passera d'une position à une autre, les positions successives étant également possibles et générées de manière aléatoire.

Définition

Soit un parallélépipède de l'espace de dimensions $\triangle_u, \triangle_v, \triangle_w$, et soit

$$(O(x_o, y_o, z_o), \vec{u}(x_u, y_u, z_u), \vec{v}(x_v, y_v, z_v), \vec{w}(x_w, y_w, z_w))$$

la base de l'espace associée à ce parallélépipède (voir Figure 3.2). Nous appelerons $(\vec{i}, \vec{j}, \vec{k})$ la base associée aux axes x, y, z. Nous définissons la transformation *translation* de la façon suivante:

1. Sélection d'un point P au hasard dans l'espace délimité par le parallélépipède rectangle:

$$\begin{cases} u_p &= rand(0,1).u_\triangle \\ v_p &= rand(0,1).v_\triangle \\ w_p &= rand(0,1).w_\triangle \end{cases}$$

   où $rand(0,1)$ est un générateur de nombres aléatoires entre 0 et 1.

2. Calcul des coordonnées (x_p, y_p, z_p) de P dans la base $(\vec{i}, \vec{j}, \vec{k})$

$$\begin{cases} x_p &= x_o + u_p.x_u + v_p.x_v + w_p.x_w \\ y_p &= y_o + u_p.y_u + v_p.y_v + w_p.y_w \\ z_p &= z_o + u_p.z_u + v_p.z_v + w_p.z_w \end{cases}$$

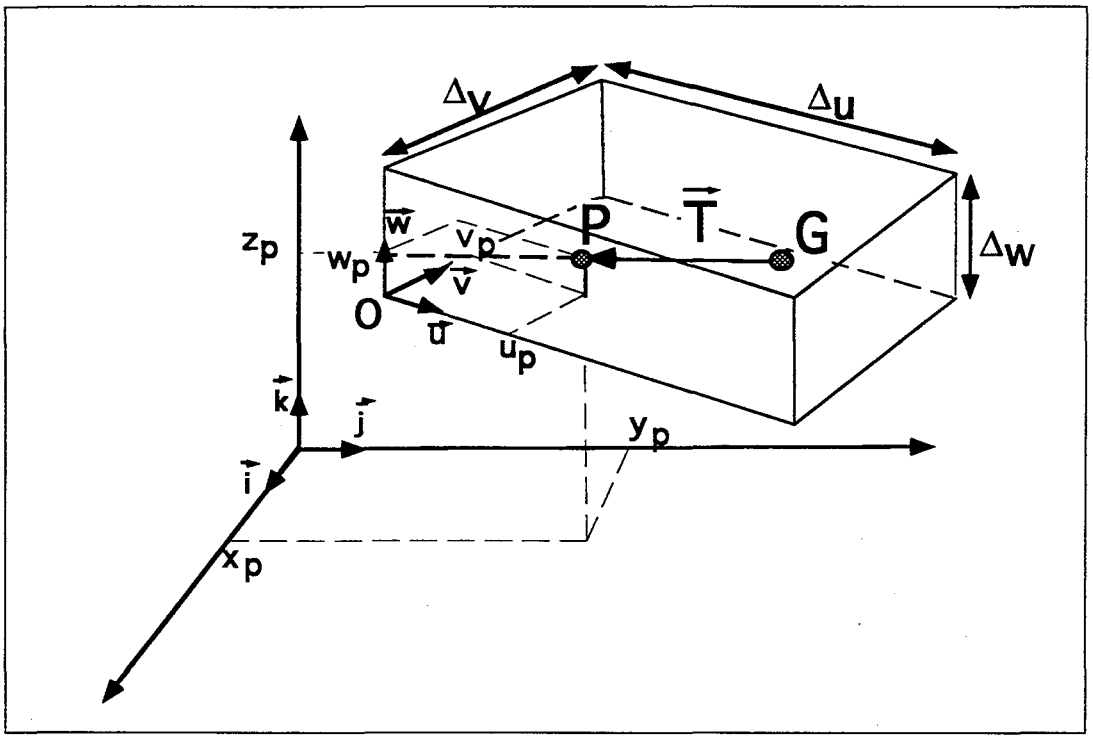


Figure 3.2 Définition de l'opération transformation *translation.*

3. Calcul du vecteur $\vec{T}$ de déplacement amenant le centre de gravité du *gshape* au point P ;
4. pour tout $k \in \Omega$, translater le vecteur $\varphi(k)$ de $\vec{T}$.

Cette opération est illustrée par la Figure 3.2

Paramètres de la transformation

La *translation* est paramétrisée par quatre vecteurs et un point de l'espace, ces paramètres correspondent:

1. Au point *origine* O de la base associée au parallélépipède rectangle.
2. Aux 3 vecteurs de l'espace $\vec{u}, \vec{v}, \vec{w}$ définissant cette base.
3. Au vecteur *dimension* $\vec{\Delta}(u_\Delta, v_\Delta, w_\Delta)$.

Ils décrivent une portion parallélépipédique de l'espace, d'origine O et de dimensions $u_\Delta, v_\Delta, w_\Delta$ selon les directions données par $\vec{u}, \vec{v}, \vec{w}$. Dans la pratique, ils décriront l'extension d'un faciès dans le réservoir, c'est-à-dire l'étendue dans laquelle peut se trouver une hétérogénéité appartenant à ce faciès.

Remarques

On remarque que:

1. Seule la position des *vertèbres* est modifée par cette opération; les vecteurs $\{V(k), k \in \Omega\}$ définissant les *contours* des *vertèbres* étant décrits relativement aux vecteurs $\{\varphi(k), k \in \Omega\}$, il n'est pas besoin de les modifier pour qu'ils soient, eux aussi, translatés.
2. Cette transformation ne déforme pas l'*enveloppe* du *gshape*.

3.4.2 Rotation

Cette transformation a pour objectif de faire varier de manière aléatoire la direction d'allongement d'un *gshape* autour d'une direction donnée.

Définition

Soit: $\vec{D}_{mean}$ un vecteur normé de direction horizontale, donné. Soit $N(0, \sigma_\theta)$ une distribution gaussienne d'écart-type σ_θ et de moyenne 0. Les différentes étapes de cette transformation sont les suivantes:

1. Tirage au hasard d'une orientation θ_{new} suivant la loi $N(0, \sigma_\theta)$.
2. Calcul du vecteur $\vec{D}_{new}$ correspondant à une rotation du vecteur $\vec{D}_{mean}$ d'un angle de θ_{new} autour d'un axe vertical, dans un plan horizontal.
3. Calcul de $\vec{D}$, vecteur normé indicateur de l'orientation actuelle du *gshape*: ceci se fait par le calcul exposé à la section 3.3.1.
4. Calcul de G, centre de gravité du *gshape*.
5. Calcul de l'axe de rotation $\vec{A}$ transformant $\vec{D}$ en $\vec{D}_{new}$
   (a) Si $(\vec{D}.\vec{D}_{new} < 0)$, $\vec{D} = -1.\vec{D}$;
   (b) $\vec{A} = \vec{D} \wedge \vec{D}_{new}$;
6. Soit $R(G, arcsin||\vec{A}||)$ la rotation d'origine G, d'axe vertical et d'angle $arcsin(||\vec{A}||)$:

   pour tout $k \in \Omega$
   pour tout $\{V_i(k), i \in [0, N[\}$
   .Soit $V_i'(k)$, la transformée par R du vecteur $V_i(k) + \varphi(k)$
   .Transformer $\varphi(k)$ par R;
   finpour.
   pour tout $\{V_i(k), i \in [0, N[,\}$ $V_i(k) = V_i'(k) - \varphi(k)$ finpour
   finpour.

Paramètres de la transformation

Les paramètres de la transformation *rotation* sont au nombre de deux:

1. $\vec{D}_{mean}$, direction de référence d'une direction d'allongement.
2. σ_θ, représentant la fourchette des écarts autour de la direction de référence.

Remarques

1. L'opération *rotation* appliquée à un *gshape* transforme à la fois les vecteurs φ et V de Ω. On modifie ainsi la géométrie du *backbone* et celle des *sections*. Il est à noter cependant que cette transformation n'a pas d'action déformante sur l'*enveloppe* du *gshape*, la position absolue des points qui la décrivent étant seule affectée.
2. En géologie, l'"azimut" d'un corps est une mesure de l'angle entre sa direction d'allongement dans un plan horizontal et la direction du Nord. La rotation R–qui s'effectue autour d'un axe vertical–est donc une manière de faire varier aléatoirement l'azimut d'un *gshape* autour d'une direction de référence.

Cette opération est illustrée par la Figure 3.3

3.4.3 Affinité

L'opération *affinité* a pour objectif de modifier de manière homogène et aléatoire la taille des *sections* d'un *gshape*.

Définition

Soit $N(e_{mean}, \sigma_e)$, une distribution gaussienne de moyenne e_{mean} et d'écart-type σ_e. On suppose connus e_{mean} et σ_e. La transformation *affinité* est appliquée à un *gshape* de la façon suivante:

1. tirage de e_{new} tel que: $e_{new} = N(e_{mean}, \sigma_e)$. Si $e_{new} \leq 0$, alors on procède à un autre tirage.
2. Calcul du facteur f d'affinité $f = \frac{e_{mean}}{e_{new}}$.
3. Application de la transformation affine de centre k et de facteur f à toutes les *vertèbres* $k \in \Omega$

   pour tout $k \in \Omega$
   pour tout $V_i(k), i \in [0, N[$
   transformer $V_i(k)$ tel que: $V_i(k) = f.V_i(k)$
   finpour.
   finpour
4. Interpoler $\{V(k), k \in \Omega\}$.

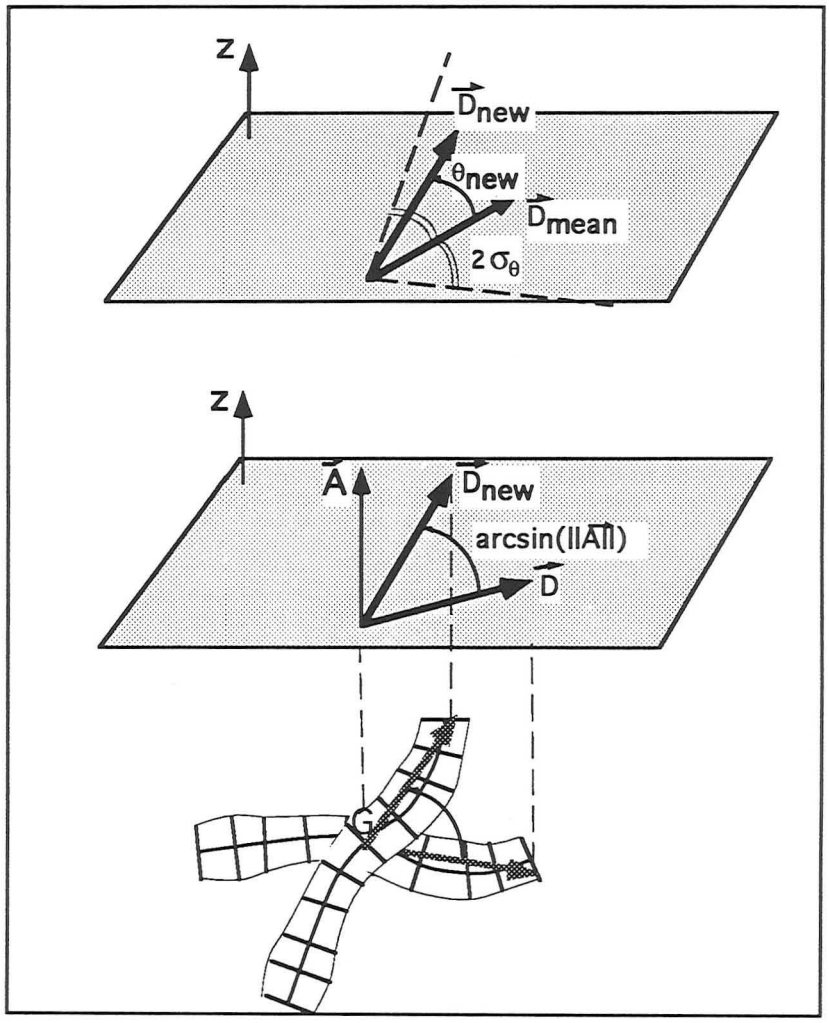


Figure 3.3 Définition de l'opération *rotation*.

Paramètres de la transformation

Les paramètres de la transformation *affinité* sont au nombre de deux, il s'agit de:

1. e_{mean}, moyenne de la distribution gaussienne associée.
2. σ_e, écart-type de la distribution gaussienne associée.

Remarque

1. Seuls les vecteurs $\{V(k), k \in \Omega\}$ sont transformés par l'opération *affinité*, les vecteurs $\{\varphi(k), k \in \Omega\}$ restant inchangés. L'*enveloppe* du *gshape* est ainsi modifiée par une action sur les *sections* et non sur le *backbone*.

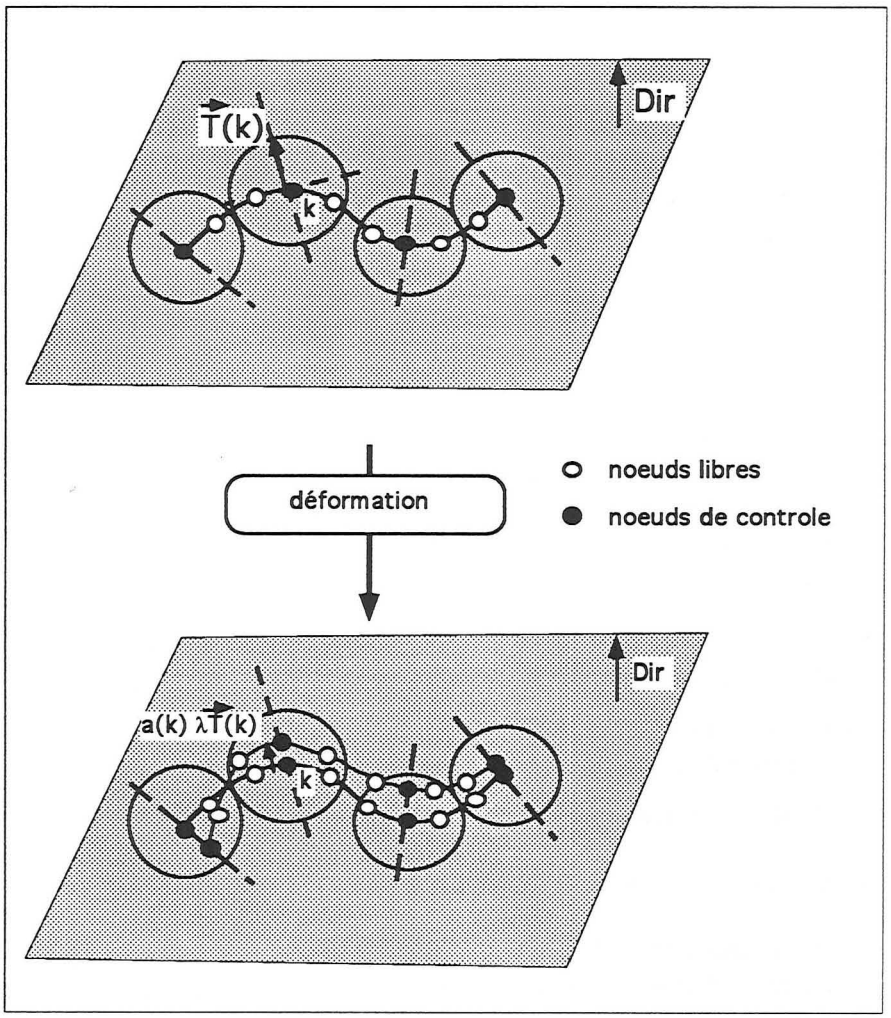


Figure 3.4 Définition des vecteurs déplacements associés à l'opération *déformation*.

2. L'*enveloppe* est déformée de manière non affine à cause de l'opération d'interpolation appliquée au cours de l'étape 4.

3. Il aurait été possible de ne modifier que les tailles des *sections de contrôles*, laissant à l'interpolateur le soin de réajuster la taille des autres *sections*; toutefois, nous avons choisi de changer la taille de **toutes** les *sections* pour des raisons de rapidité de convergence de l'étape 4. En effet, *DSI* converge beaucoup plus rapidement si toutes les *sections* ont, au départ, une taille proche de celle recherchée.

3.4.4 Déformation

L'opération *déformation* a pour but de faire varier la sinuosité du *backbone* en modifiant les positions dans l'espace de ses *nœuds de contrôle* de manière aléatoire.

Définition

Soit Ω_{nc} l'ensemble des *nœuds de contrôle* de Ω et soit $\vec{Dir}$ un vecteur. Nous définirons par $\{\vec{T}(k), \forall k \in \Omega_{nc}\}$ l'ensemble des "vecteurs mouvements" de Ω tels que:

1. $\vec{T}(k)$ est orthogonal au *backbone* au *nœud de contrôle* k;
2. $\vec{T}(k)$ est inclus dans le plan dont un vecteur normal est $\vec{Dir}$
3. $||\vec{T}(k)||$ est égal au rayon du cercle le plus large, centré en k et qui n'intersecte pas les cercles associés aux autres nœuds de Ω_{nc} (voir Figure 3.4, page 79).

$\vec{T}(k)$ caractérise la direction selon laquelle peut se déplacer le nœud de contrôle k, la distance maximale qu'il est autorisé à parcourir selon cette direction étant égale à $||\vec{T}(k)||$.

Soit d'autre part λ un réel $\in [0,1]$ et correspondant à un coefficient de sinuosité. Les différentes étapes de la transformation *déformation* sont:

1. pour tout $k \in \Omega_{nc}$
   (a) Soit $\{\vec{T}(k)\}$ le vecteur "mouvement" associé à k.
   (b) tirer $a(k) \in [-1,+1]$ tel que: $a(k) = rand(0,1) * 2 - 1$ où $rand(0,1)$ est un générateur de nombres aléatoires entre 0 et 1.
   (c) Si $\lambda > 0$, translater le vecteur $\varphi(k)$ du vecteur $a(k).\lambda.\vec{T}(k)$
2. interpoler les vecteurs $\{\varphi(k), \forall k \in \Omega\}$.
3. $\forall k \in \Omega$ transformer les *ribs* $\{V^i(k), i \in [0, N[\}$ de façon à rendre le *contour* qu'ils décrivent orthogonal au backbone. Cette opération se fait par l'opération décrite à la section 2.5.2.
4. Interpoler $\{V(k), \forall k \in \Omega\}$;
5. Appliquer la transformation *Rotation* sur Ω.

Paramètres de la transformation

Les paramètres de la transformation *déformation* sont les suivants:

1. Le coefficient de sinuosité λ. Une valeur de λ égale à 1 sera caractéristique d'un degré de sinuosité maximal alors qu'une valeur inférieure à 1 le restreint. Ainsi, si $\lambda = 1$, un *nœud de contrôle* k pourra se déplacer sur toute la longueur du rayon du cercle de rayon $||\vec{T}(k)||$ (voir Figure 3.4), si $\lambda = 0.5$, il ne pourra se déplacer que sur la moitié de cette longueur.
2. La direction du plan de perturbation des *nœuds de contrôles* $\vec{Dir}$.

Remarques

Les caractéristiques de la transformation *déformation* sont les suivantes:

1. Cette transformation autorise beaucoup de liberté dans le changement de forme de l'*enveloppe*. Celle-ci est cette fois modifiée en intervenant sur la forme attribuée au *backbone* du *gshape* et sur celle attribuée aux *sections*. La modification n'est affine dans aucun des deux cas puisqu'on applique DSI au *backbone* et aux *sections*.

2. Le fait d'orthogonaliser les *sections de contrôle* assure une certaine consistance entre la géométrie du *backbone* et celle des *sections*. En effet, la transformation des *sections* est induite par celle du *backbone* de manière à garder les *sections de contrôle* orthogonales au *backbone*.

3. Les oscillations incorporées dans le *backbone* dépendent à la fois du coefficient λ qui contrôle l'étendue des déplacements, et des distances entre les *nœuds de contrôle* du *gshape*. D'autre part, tous les *nœuds de contrôle* du *gshape* sont perturbés dans un plan défini par la normale $\vec{Dir}$.

4. L'application de la transformation *Rotation* assure que la direction d'allongement du *gshape* "perturbé" reste localisée autour d'une direction de référence.

3.4.5 Remarques

Ces 4 transformations de base ont été mises en place afin de formaliser les différentes réalisations de la géométrie d'un *gshape* représentant une hétérogénéité. Il est évident que la liste des transformations possibles n'est pas exhaustive. De même, d'autres lois de distribution sont susceptibles de rendre compte de la répartition des tailles ou des directions d'allongement d'un corps. Cependant, il s'agit ici de mettre en place un processus général de modélisation de la géométrie des hétérogénéités. Les quatre transformations implémentées présentent l'avantage d'être simples et de correspondre à une première approche de ce que les géologues entendent par "variation de la géométrie d'une hétérogénéité autour d'une forme donnée". Il est entendu que lors de développements ultérieurs, il sera possible–voir nécessaire–d'étendre la liste des transformations proposées.

3.5 Présentation de l'objet Rshape

L'objet *rshape* a été mis en place afin d'intégrer à l'objet *gshape* la possibilité de rendre compte de la multiplicité des géométries pouvant décrire une hétérogénéité appartenant à un faciès donné. On suppose que ce faciès peut être caractérisé par des données du type de celles exposées à la section 3.3.2.

3.5.1 Définition

Un *Rshape* est une extension de l'objet *gshape* à une représentation spécifique de la géométrie d'une hétérogénéité au sein d'un faciès donné. Ses caractéristiques sont les suivantes:

1. Sa géométrie est définie de la même manière que pour un objet *gshape*. Nous utiliserons donc la même terminologie que celle mise en place pour l'objet *gshape*[3]. De même, toutes les opérations définies aux chapitre 1 et 2, et les opérations de base décrites à la section précédente s'appliquent aux *rshapes*.

2. Il contient la liste des transformations susceptibles de lui être appliquées. Il s'agit des 4 opérations aléatoires définies ci-dessus: *translation, affinité, rotation, déformation.*

3. Il contient les paramètres associés à ces 4 opérations.

4. Il contient une *transformation courante* définie comme étant une des opérations *translation, affinité, rotation, déformation.*

De plus, nous dirons qu'un *rshape* r est issu d'un *template* t représentatif d'un faciès f si:

1. La topologie de r est la même que celle de t.

2. La paramétrisation de r est la même que celle de t: les *vertèbres* portant les *sections de contrôle* et les *nœuds de contrôle* de r ont les mêmes positions relatives que celles portant les *sections de contrôle* et les *nœuds de contrôle* de t.

3. Les *vecteurs horizons* de r sont définis par les *ribs* de même *index* que ceux de t[4]

4. Les paramètres associés aux transformations *translation, rotation, affinité, déformation* ont été initialisés en fonction des données décrivant f et mentionnées à la section 3.3.

3.5.2 Initialisation

La procédure d'initialisation d'un *rshape* r depuis un *template* t caractérisant un faciès f est la suivante:

1. **copie de la topologie et de la géométrie de** t: chaque *vertèbre* du *rshape* est initialisée à partir de la *vertèbre* de t correspondante. La géométrie des *contours* de r est copiée de celle de t. La définition des *nœuds de contrôle* et des *sections de contrôle* est copiée de celle de t.

2. **copie des vecteurs horizons de t**: les *vecteurs horizons* de r sont initialisés à partir des index des *ribs* définissant les *vecteurs horizons* de t.

[3]Cette terminologie est essentiellement composée des termes suivants: *backbone, sections, vertèbre, section de contrôle, nœuds de contrôle, ribs, vecteur horizon.*

[4]voir section 2.3.2

3. **initialisation des paramètres.** On initialise les paramètres correspondant aux transformations attachées à r. Ces procédures d'initialisation font l'objet de la section suivante.

4. **application des transformations**: on transforme r par l'ensemble des transformations qu'il supporte: on lui applique dans l'ordre:

   (a) *translation*

   (b) *affinité*

   (c) *rotation*

   (d) *déformation*

   Nous verrons que cette étape de l'initialisation d'un *gshape* est optionnelle.

5. On initialise sa *transformation courante* comme étant la transformation *translation.*

La procédure d'initialisation d'un *rshape* est une façon de construire une réalisation possible de sa géométrie, compte tenu des données caractérisant le faciès auquel il appartient (données de tailles, de directions d'allongement, *template*).

3.5.3 Définition des paramètres des opérations de base

Un *rshape* appartenant au faciès f est caractérisé par sa capacité à se déformer. Dans ce contexte, la valeur des paramètres des 4 transformations de base qu'il supporte doit permettre des déformations qui prennent en compte les éléments suivants:

1. Les caractéristiques du *template* associé au faciès f.

2. Les caractéristiques de tailles et de directions d'allongement se traduisant par les distribution 1a, 1b et 2 de la section 3.3.2. Nous rappelons que les *moyennes* de ces distributions sont représentées par l'épaisseur moyenne, le rapport $\frac{largeur}{epaisseur}$ moyen et la direction d'allongement du *template.*

Nous définissons ci-après les paramètres des 4 transformations *translation, rotation, affinité, déformation.*

translation

Nous rappelons que les paramètres de la *translation* sont au nombre de cinq et correspondent au point *origine* O, aux vecteurs $(\vec{u}, \vec{v}, \vec{w})$ et au vecteur *dimension* $\vec{\Delta}(u_\Delta.v_\Delta, w_\Delta)$. Soit f un faciès dont on considère que l'extension dans l'espace peut être limitée par un parallélépipède rectangle. Les paramètres de la *translation* seront alors initialisés de la manière suivante (voir Figure 3.2):

1. O est le point de l'espace correspondant à l'*origine* du parallélépipède rectangle.

2. $\vec{u}, \vec{v}, \vec{w}$ décrivent les 3 directions principales de ce parallélépipède rectangle.

3. $u_\triangle, v_\triangle, w_\triangle$ sont les dimensions du parallélépipède rectangle dans les directions de $\vec{u}, \vec{v}, \vec{w}$.

On peut remarquer ici qu'aucun des paramètres de la *translation* ne dépend du *template* associé à f. Nous dirons qu'ils sont *template indépendants.*

rotation

Nous rappelons que les paramètres de la transformation *rotation* sont au nombre de deux. Il s'agit de $\vec{D}_{mean}$, direction de référence d'une direction d'allongement et de σ_θ caractérisant l' écart-type autour de cette direction. Ces deux paramètres sont initialisés de la manière suivante (voir Figure 3.3):

1. Calcul de $\vec{D}_{mean}$:

   $\vec{D}_{mean}$ correspond à la direction d'allongement du *template*[5]. Celle-ci sert de direction de référence à toute transformation *rotation* effectuée sur un *rshape* issu de ce *template.*

2. affectation à σ_θ de la valeur de l'écart-type caractérisant la distribution gaussienne des directions d'allongement donnée à la section 3.3.2.

Le paramètre $\vec{D}_{mean}$ dépend du *template* associé à f. Il sera dit *template dépendant.* σ_θ, au contraire est *template indépendant.*

affinité

Les paramètres de la transformation *affinité* sont les deux paramètres de la distribution gaussienne associée, e_{mean} et σ_e. En pratique, ils décrivent les paramètres de distribution de l'*épaisseur* d'un *gshape.* Ils sont calculés de la manière suivante:

1. Affectation à e_{mean} de l'épaisseur moyenne associée au *gshape*[6].

2. Affectation à σ_e de l'écart-type de la distribution sur les épaisseurs mentionnée à la section 3.3.2:

On remarque ici que le paramètre e_{mean} est *template dépendant* alors que σ_e est *template indépendant.*

déformation

Les paramètres de la transformation *déformation* sont le *coefficient de sinuosité* λ et la direction du plan de perturbation $\vec{Dir}$. λ est par défaut initialisé à 1, ce qui correspond à une variation maximale de la sinuosité. $\vec{Dir}$ est par défaut initialisé au vecteur de coordonnée $(0, 0, 1)$, ce qui contraint les nœuds de contrôle à ne se déplacer que dans un plan horizontal. Ces paramètres sont tous deux *template indépendants.*

[5] voir section 3.3.1

[6] ce calcul est expliqué à la section 3.3.1

3.5.4 Discussion

Nous avons supposé jusqu'ici que les paramètres de tailles (épaisseurs, rapport $\frac{largeur}{epaisseur}$) et de directions d'allongement du *template* père d'un *rshape* avaient été initialisés à partir de distributions gaussiennes connues. Cependant, le calcul des paramètres e_{mean} et $\vec{D}_{mean}$ des deux transformations *affinité* et *rotation* fait intervenir des mesures faites directement sur le *template* plutôt que sur les distributions associées. Cette opération, qui peut paraître redondante, présente les avantages suivants:

1. Elle permet à un géologue de modéliser un *template* sans connaître les paramètres e_{mean} et $\vec{D}_{mean}$. Ceux-ci sont extraits du *template* et des valeurs par défaut sont données aux paramètres *template indépendants* σ_e et σ_{rot}.

2. Elle diminue le nombre de données à introduire. En effet, la donnée d'un *template* étant obligatoire, il semble logique de lui attribuer des paramètres de tailles et d'élongation qui sont caractéristiques des distributions. Seuls les écarts-type σ_e et σ_{rot} ont alors besoin d'être numériquement connus.

Par ailleurs:

1. La donnée des écart-types se fait sous forme d'un pourcentage de variation par rapport aux paramètres extraits du *template*. Ainsi, une variation de 10% autour d'une épaisseur moyenne de $5m$, se traduira par un écart-type de $\frac{0.5}{2} = 0.25m$. Cela permet d'introduire les données de distribution de façon indirecte.

2. L'introduction des paramètres *template indépendants* est facultative. Des valeurs par défaut leurs sont alors attribuées.

3.6 Génération d'une famille d'hétérogénéités

Nous considérerons ici le mode de génération d'une famille d'hétérogénéités que nous noterons faciès d'index k. On suppose que celui-ci est composé de n_k *rshapes* dont la génération s'est faite par une des trois procédures suivantes:

3.6.1 Procédure stochastique

On considère que les n_k *rshapes* constituant le faciès sont générés depuis le **même** *template* père et par toutes les étapes de la procédure décrite à la section 3.5.2. Un unique *template* représente alors les caractéristiques essentielles du faciès. Le nombre de *rshapes* n_k à générer est calculé de la manière suivante:

Soient:
- V_{tot} : le volume total du réservoir.
- p_k : la proportion volumique relative du faciès k dans le réservoir.
- t_k : le template associé au faciès k.

1. Calculer le volume v_k du *template* t_k; ce volume est considéré comme le volume moyen d'une hétérogénéité appartenant au faciès k.

2. Calculer n_k, le nombre d'objets *rshape* à produire pour le faciès k, par l'expression suivante:
$$n_k = -\frac{V_{tot}}{v_k} \ln(1 - p_k)$$

3. Générer n_k objets *rshapes* ayant le *template* t_k comme *parent.* Ceci se fait par le processus décrit à la section 3.5.2. Nous rappelons que ceci initialise leur *transformation courante* comme étant la *translation.*

Le calcul de n_k est issu de [54]. Il permet de prendre en considération l'intersection possible de corps distribués au hasard. Pour respecter la proportion volumique p_k d'un faciès, il faudra générer plus de corps qu'en l'absence d'intersections.

3.6.2 Procédure déterministe

On considère que l'*enveloppe* des n_k *rshapes* constituant le faciès d'index k est déjà modélisée à travers n_k *templates* distincts. On crée alors n_k *rshapes* à partir de ces *templates*, mais on n'exécute pas l'étape "application des transformations" présentée à la section 3.5.2. Les *rshapes* fils ont alors la même *enveloppe* que leur *template* père, mais leurs paramètres *templates dépendants* varient. Le respect des proportions volumiques n'est pas vérifié. Un exemple de création d'un tel faciès est présenté à la section 5.5.1.

3.6.3 Procédure mixte

On considère un faciès composé de m_k *rshapes* créés de manière stochastique ou déterministe. On y ajoute alors $n_k - m_k$ *rshapes*, eux-mêmes créés de manière déterministe ou stochastique.

3.6.4 Signification de l'objet rshape

Au niveau de sa géométrie

Le *rshape* apparaît comme un moyen de représenter la géométrie d'une hétérogénéité dont une réalisation existe déjà. Il s'agit à la fois de la géométrie de l'*enveloppe* associée au *rshape*, mais également de sa géométrie dans l'espace, au sein du faciès auquel il appartient. La représentation initiale de la géométrie d'un *rshape* peut être issue de:

1. L'*enveloppe* d'un *template* caractéristique d'un faciès donné. On génère alors un objet *rshape* appartenant à ce faciès à partir de ce *template.* Ce processus est décrit à la section 3.5.2.

2. L'*enveloppe* du *rshape* lui-même. Il possède alors déjà une géométrie que l'on modifie en appliquant la transformation *transformation courante* au *rshape.*

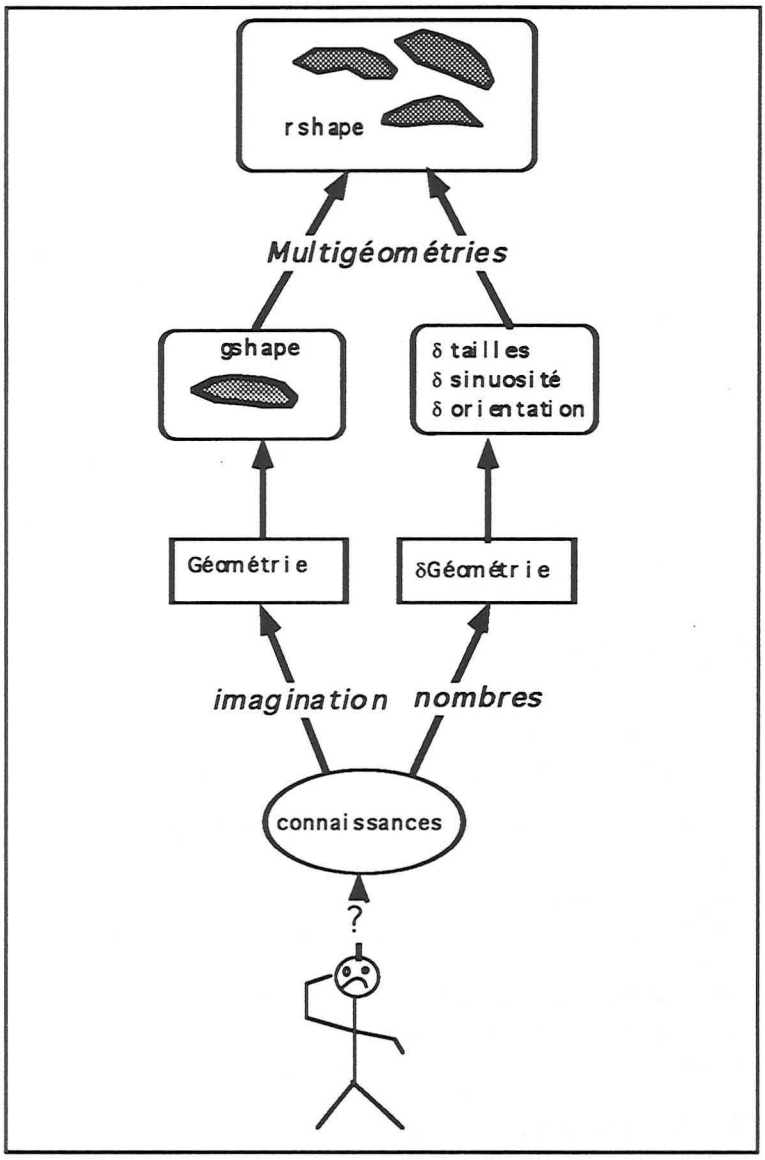


Figure 3.5 Résumé de la notion de *rshape*.

Le *template* permet d'initialiser la géométrie de *rshapes* appartenant à un même faciès et de définir les paramètres associés aux transformations qu'ils peuvent subir. Une fois cette étape exécutée, l'objet *rshape* est "autosuffisant"; il est susceptible d'être transformé par application de sa *transformation courante* et peut produire plusieurs réalisations possibles de sa géométrie, chacune respectant toutes les données citées à la section 3.3. Cette notion est illustrée par la Figure 3.5.

Au niveau de ses paramètres de transformation

Les paramètres des transformations *translation, affinité, rotation, déformation* sont de deux ordres.

1. Les paramètres *template dépendants* d'un *rshape* sont directement issus du *template* utilisé pour initialiser celui-ci. Dans le cas d'une procédure stochastique de création d'un faciès k, un *template* unique est père de tous les *rshapes* appartenant au faciès k. Les paramètres *template dépendants* sont donc nécessairement les mêmes pour tous les *rshapes* du faciès f et nous considèrerons qu'ils ne peuvent être modifiés qu'en caractérisant f par un nouveau *template*, et donc en régénérant tous les *rshapes* appartenant à f.

2. Les valeurs des paramètres *template indépendants* ne sont pas fonction du ou des *templates* associés au faciès f. Pour un *rshape* donné, ces valeurs représentent l'ampleur des déformations autour de la géométrie de leur *template* père. Nous considèrerons que ces paramètres sont modifiables à tout moment.

3.7 Conclusion

Concernant l'introduction de l'objet *rshape* pour rendre compte des différentes géométries possibles pour une hétérogénéité singulière, nous ferons les remarques suivantes:

1. La grande flexibilité de paramétrisation de la géométrie des *rshapes*, directement héritée de celle des *gshapes* permet la mise en place d'opérations aléatoires déformant l'*enveloppe* du *rshape*. De même, l'exploitation du *vecteur horizon* hérité des *gshapes* permet de définir les notions de largeur et d'épaisseur d'une hétérogénéité.

2. Les paramètres des transformations aléatoires intègrent des données caractérisant le faciès auquel appartient un objet *rshape*. La cohérence entre la transformation appliquée et le respect de ces données est en permanence assurée.

3. L'utilisation d'un *template* pour caractériser l'aspect géométrique des hétérogénéités appartenant à un faciès donné apparaît comme un moyen d'intégrer les connaissances intuitives et l' "Art" du géologue, tout en utilisant les données quantifiées qu'il connaît. Cependant, la disponibilité de ces données n'est pas une contrainte, puisque si celles-ci ne sont pas connues, elles sont extraites du *template* modélisé.

4. Le degré de variation de la géométrie d'un *rshape* par rapport à son *template* père est parfaitement contrôlable; en effet, les paramètres *template indépendants* d'un *rshape* donné peuvent être modifiés à tout moment et indépendamment de ceux des autres *rshapes* appartenant au même faciès.

Ainsi, l'objet *rshape* utilise la flexibilité apportée par le mode de paramétrisation de l'objet *gshape* pour définir un ensemble de déformations respectant les données générales décrivant la géométrie des hétérogénéités au sein d'un faciès donné. L'application d'une de ces

transformations à un *rshape* engendre une nouvelle réalisation de sa géométrie qui respecte les données. Cela justifie le terme de "forme stochastique" dont il a été qualifié au début de ce chapitre. La liste des transformations disponibles n'est pas exhaustive.

3.8 Résumé du chapitre

Nous avons d'abord présenté une série de transformations aléatoires mises en place pour décrire les différentes géométries possibles de l'*enveloppe* d'un *gshape* susceptible de décrire une hétérogénéité dans un faciès.

1. Ces transformations respectent les distributions d'épaisseurs, de rapports $\frac{largeur}{epaisseur}$, de directions d'allongement, et un *gshape* type, appelé *template* qui représente la forme la plus probable d'une hétérogénéité au sein d'un faciès donné.

2. Nous avons défini l'objet *rshape* comme une extension de l'objet *gshape* dont la géométrie et les paramètres des opérations *translation, rotation, affinité, déformation* sont initialisés à partir d'un *template* père.

3. Nous avons défini une famille d'hétérogénéités comme un faciès composé d'un nombre fini de *rshapes*. Chacun de ces *rshapes* est initialisé à partir d'un ou plusieurs *templates* caractérisant ce faciès.

4. Chacun des *rshapes* contenu dans un faciès est "autosuffisant" dans la mesure où il peut se déformer et produire une nouvelle réalisation de son *enveloppe* par application de sa *transformation courante*.

90Présentation de l'objet "rshape" comme représentant d'une hétérogénéité

Chapitre 4

Méthode de simulation stochastique utilisée

4.1 Présentation du chapitre

Une manière de représenter des hétérogénéités de réservoir en trois dimensions ayant été proposée au cours des chapitres précédents, nous aborderons ici le problème de la répartition de ces corps dans l'espace de façon à ce que le maximum de contraintes soient respectées. Nous présenterons d'abord les références bibliographiques utilisées dans notre approche, puis nous proposerons notre méthode et étudierons ses caractéristiques principales.

4.2 Introduction

L'objet *rshape* introduit lors du chapitre précédent offre un moyen de générer des réalisations différentes de la géométrie d'une hétérogénéité de réservoir. Cette géométrie est représentée par l'*enveloppe* d'un objet *rshape*. Nous voudrions maintenant mettre en place une manière de générer des réalisations équiprobables de la distribution des hétérogénéités dans l'espace, réalisations devant toutes honorer un certain nombre de contraintes et, notamment, le respect des faciès observés aux puits. Il nous faut pour cela résoudre les problèmes suivants:

1. La génération de modèles équiprobables rendant compte de la distribution des hétérogénéités au sein de leur faciès.

2. La mesure de l'écart entre ces modèles et le modèle réel.

3. L'introduction d'une certaine dose d'interactivité pour permettre au géologue de garder le contrôle sur la production du modèle final.

4.3 Quelques remarques sur les méthodes de simulations stochastiques

Considérons la distribution sur un support A d'un attribut $z(\mathbf{u}), \mathbf{u} \in A$. Les procédés de simulation stochastiques ont pour but de générer des modèles équiprobables de la distribution dans l'espace de $z(\mathbf{u})$ ([10]). La simulation est dite conditionnelle, si les réalisations résultantes honorent les valeurs mesurées ([10],[24]). Soient $Z(\mathbf{u})$ et $Z(\mathbf{u}+\mathbf{h})$ les variables aléatoires régionalisées représentant les valeurs prises par le même attribut aux localisations $\mathbf{u}$ et $\mathbf{u}+\mathbf{h}$, distantes du vecteur $\mathbf{h}$. Les méthodes de simulations stochastiques traditionnelles sont souvent basées sur le respect de:

1. La fonction de répartition choisie pour représenter la distribution de Z. Nous rappelons qu'une fonction de répartition d'une variable aléatoire continue $Z(\mathbf{u})$ s'exprime par ([11]):

$$F(\mathbf{u};z) = Prob\{Z(\mathbf{u}) \leq z\} \in [0,1]$$

2. Une ou plusieurs fonctions, reliant la valeur de la variable $Z(\mathbf{u})$ à celle de la variable $Z(\mathbf{u}+\mathbf{h})$. La plus utilisée de ces fonctions est le variogramme $\gamma(h)$ défini comme:

$$2\gamma(h) = E\{[Z(\mathbf{u}) - Z(\mathbf{u}+\mathbf{h})]^2\}$$

Ces deux types de données s'expriment souvent à l'aide de modèles mathématiques théoriques que l'on suppose adéquats au problème posé. Ces modèles sont généralement issus d'études antérieures et que l'on juge similaires à l'étude en cours. Cette approche est typique des méthodes dites "pixels" qui considèrent qu'un modèle est composé d'un ensemble fini de points (ou de "pixels"). De nombreuses publications illustrent leur diversité et leur richesse ([6], [16]). Cependant, et comme nous l'avons justifié au cours de l'analyse effectuée au cours de la section 0.7, nous nous dirigeons ici plutôt vers une approche booléenne. Nous ne rentrerons donc pas dans le formalisme des méthodes "pixels" qui peut par ailleurs être abordé dans les ouvrages [35], [10], [33]. Cependant, plusieurs volets de la géostatistique, développés dans le cadre des méthodes pixels, semblent pouvoir être extraits de ce cadre spécifique et appliqués de manière plus générale. Il s'agit notamment des domaines suivants:

1. Le formalisme de la géostatistique non paramétrique. Il permet en particulier de s'affranchir de modèles mathématiques ([37],[38]).

2. L'application des méthodes générales de recuit simulé en simulation stochastique ([11]).

Ces deux concepts sont décrits dans les sections suivantes et sont inspirés des descriptions faites par Srivastava ([33]), Journel ([35]) et Deutsch ([11]):

4.3.1 Géostatistique non paramétrique

Nous allons présenter sommairement un volet de la géostatistique appelé "Géostatistique non paramétrique", qui permet par son formalisme de s'affranchir de la nécessité de choisir un modèle théorique pour exprimer la fonction de répartition d'une variable aléatoire. Contrairement à la géostatistique paramétrique qui donne pour priorité l'estimation de la valeur d'une variable en un point, la géostatistique non paramétrique modélise la fonction de distribution de cette variable ([35]). Ce volet de la géostatistique repose sur le formalisme dit des "variables indicatrices", développé par A.G. Journel à l'université de Stanford, Californie ([37],[52], [34]).

Variables indicatrices: présentation

En géostatistique, lorsqu'on estime la valeur aléatoire d'une variable Z en fonction des valeurs $\{z_1, z_2, \ldots, z_n\}$ qui ont été mesurées pour cette variable, on a besoin de pouvoir répondre à la question suivante: "sur l'ensemble des valeurs mesurées, quelle est la proportion de celles situées au-dessous d'un certain seuil?" ([33]). Il suffit pour cela de "compter" combien de données se situent au dessous de cette valeur et de diviser ce nombre par le nombre total n de données. Cette notion de "compter" est traduite par une fonction indicatrice, définie comme:

$$i(z_0; z_j) = \begin{cases} 0 & \text{si } z_0 < z_j \\ 1 & \text{si } z_0 \geq z_j \end{cases}$$

où z_j est la valeur de la variable Z issue de la mesure j et z_0 une valeur seuil ([35]).

On peut alors modéliser la distribution de la variable aléatoire Z par sa fonction de répartition qui s'exprime par:

$$F(z_0|(n)) = P\{Z \leq z_0|(n)\} = \frac{1}{n}\sum_{j=1}^{n} i(z_0; z_j) \in [0,1]$$

La notation $F(z_0|(n))$ est utilisée pour rappeler que la distribution de Z dépend non seulement de la valeur seuil z_0 mais également de l'ensemble des n valeurs qu'elle peut prendre. Ce formalisme des variables indicatrices a été introduit par A.G. Journel et est détaillé dans [35],[37],[38],[10]. Il signifie que l'on mesure "la probabilité qu'une valeur inconnue z soit inférieure ou égale à un certain seuil z_0", comme correspondant à la proportion de valeurs z_j telles que $z_j \leq z_0$. C'est aussi l'expression du fait que la modélisation de la probabilité pour que la valeur d'une variable soit située au dessus d'un certain seuil dépend de la proportion des données où l'on observe une valeur supérieure à ce même seuil.

Le formalisme des variables indicatrices permet donc de transformer, pour n'importe quelle valeur seuil z_0 les valeurs continues de la variable Z en une matrice colonne indica-

trice $I(z_0)$ telle que:

$$I(z_0) = \begin{bmatrix} i(z_0; z_1) \\ i(z_0; z_2) \\ \vdots \\ i(z_0; z_n) \end{bmatrix}$$

On peut également exprimer la fonction de répartition en pondérant différemment les indicatrices, selon qu'on estime que les valeurs auxquelles elles s'appliquent sont plus ou moins "sûres".

$$F(z_0|(n)) = P\{Z \leq z_0|(n)\} = \sum_{j=1}^{n} a_j.i(z_0; z_j) \in [0, 1]$$

où les $\{a_j\}$ sont des poids assignés à l'indicatrice $i(z_0; z_j)$ et tels que

$$\begin{cases} \sum_{j=1}^{n} a_j & = 1 \\ a_j & \geq 0, \forall j \end{cases}$$

Cas des distributions spatiales

Jusqu'à présent n'était considérée que la modélisation de la distribution de variables ayant été mesurées n fois dans les mêmes conditions. Nous considèrerons ici le cas où l'on modèle la distribution d'une variable $z(x)$ à la localisation x en fonction de n mesures $\{z(x_1), z(x_2), \ldots, z(x_n)\}$ localisées en $\{x_1, x_2, \ldots, x_n\}$. On considère que "La probabilité pour que l'inconnue $z(x)$ soit inférieure ou égale à un certain seuil z " est la proportion des données $\{z(x_j)\}$ inférieures à ce seuil.

On peut alors modéliser la fonction de répartition de la variable aléatoire correspondante Z de la façon suivante:

$$F(z; x|(n)) = P\{Z(x) \leq z|(n)\} = \frac{1}{n} \sum_{j=1}^{n} i(z; xj) \in [0, 1]$$

avec:

$$i(z; x_j) = \begin{cases} 0 & \text{si } z < z(x_j) \\ 1 & \text{si } z \geq z(x_j) \end{cases}$$

Dans le cas où il y a dépendance entre les valeurs $z(x)$ et $z(x_j)$ lorsque $x \neq x_j$, on peut exprimer $F(Z; x|(n))$ par:

$$\begin{aligned} F(z; x|(n)) &= P\{Z(x) \leq z|(n)\} \\ &= \sum_{j=1}^{n} a_j(x) i(z; xj) \in [0, 1] \end{aligned}$$

avec $a_j(x) \geq 0, \forall j$ et $\sum_{j=1}^{n} a_j(x) = 1$

Les $\{a_j(x)\}$ sont des coefficients de pondération qui peuvent par exemple servir à mettre un poids plus fort sur les données localisées plus près de x. Dans un premier temps, ce formalisme des variables indicatrices a été utilisé pour simuler des variables continues telles que porosités et perméabilités ([24],[52],[39],[36]). Il a ensuite été étendu aux variables catégoriques (non continues).

Cas des variables catégoriques

Journel ([10]) et Suro-Perez ([56]) attribuent à chaque point du modèle une valeur entière (0 ou 1) en fonction du faciès auquel il appartient. Ainsi, si $L(x)$ est le faciès associé à la localisation x et l_k est un des K faciès présents dans le reservoir, on définit par $I(x; l_k)$ une fonction indicatrice telle que:

$$I(\mathbf{x}; l_k) = \begin{cases} 1 & \text{si } L(x) \in l_k \\ 0 & \text{sinon} \end{cases}$$

Ce formalisme a lui aussi été utilisé pour simuler des distributions de faciès ([1]). Cette application aux variables catégoriques paraît très intéressante, puisque c'est justement une variable catégorique (appartenance ou non à un faciès donné) que nous devons simuler.

Intérêt de la géostatistique non paramétrique

Les aspects essentiels de cette approche sont les suivants:

1. Tous les types de données sont traités de la même façon, et peuvent être formalisés à l'aide de matrices colonne indicatrices. La prise en compte de variables catégoriques telles que le respect des faciès observés est assurée.
2. On évite de formuler des hypothèses sur l'expression de la fonction de répartition ou de fonctions de type variogramme. Il n'est donc pas nécessaire d'extraire leurs expressions de modèles mathématiques théoriques.
3. L'expression des fonctions citées ci-dessus se simplifie énormément lorsqu'on utilise ce formalisme ([35]).
4. On estime la distribution de la valeur d'une variable en un point plutôt que d'estimer cette valeur.
5. Le formalisme binaire rend le calcul des variables indicatrices très efficace du point de vue informatique.

Par ailleurs, on peut dire que ce formalisme paraît général et doit pouvoir être sorti du cadre des méthodes pixel pour être appliqué à l'expression d'une formule prenant en compte des données de variables catégoriques dans le cadre d'une approche booléenne. Nous verrons par la suite comment le formalisme des variables indicatrices a été utilisé pour formaliser nos données de manière simple et efficace.

4.3.2 Méthodes de recuit en simulation stochastique

Rappel

D'une manière générale, les méthodes de recuit[1] font un parallèle avec le domaine de la thermodynamique et la manière dont un liquide ou un métal surchauffé se refroidit et cristallise pour arriver à un état stable. Les molécules se réorganisent alors de façon à atteindre un état qui se stabilise de plus en plus au fur et à mesure que la température du système diminue. La probabilité de distribution des molécules est résumée par la loi de Boltzmann $P\{E\} \sim e^{\frac{-E}{kT}}$, qui exprime le fait que l'énergie d'un système en équilibre à la température T est celle de ses molécules distribuées parmi les différents états d'énergie E. k est la constante dite de Boltzmann et relie, dans la nature, la valeur de la température à celle de l'énergie ([51]). Metropolis ([46]) a simulé le comportement des molécules durant le processus de refroidissement: il a mis en évidence le fait que la probabilité pour qu'un système évolue d'un système d'énergie E_1 vers un système d'énergie E_2 était:

$$\begin{cases} 1 & \text{si } E_2 \leq E_1 \\ p \in]\,0,1\,[& \text{sinon} \end{cases}$$

C'est ce qu'on appelle l'algorithme de Metropolis. Plus généralement, un processus d'optimisation basé sur l'analogie avec des phénomènes thermodynamiques est appelé processus de "recuit simulé".

Définition

Les techniques de recuit, dans les approches stochastiques, sont des méthodes qui optimisent un modèle initial en fonction d'un certain nombre de données à respecter. Un nombre élevé de perturbations est apporté au modèle de départ, et l'on mesure l'équivalent d'une fonction de température après chaque modification apportée.

On définit pour cela une fonction coût, qui mesure la "température" d'une réalisation quelconque. Sa valeur correspond en fait à une mesure de la différence entre des caractéristiques spatiales que l'on veut respecter et celles mesurées au niveau de la réalisation. Une réalisation nouvelle est obtenue en introduisant une perturbation dans la réalisation précédente et en "mesurant" la fonction coût de la nouvelle réalisation. Un critère de décision intervient ensuite pour décider si la seconde réalisation doit être conservée et devenir la réalisation courante ou si, au contraire, elle doit être rejetée. Les étapes générales des méthodes de recuit sont décrites ci-dessous ([11]):

1. Déterminer une expression de la fonction coût O qui tienne compte de toutes les contraintes que l'on souhaite respecter. Soit $Z(\mathbf{u})$ une variable catégorique pouvant prendre une des K valeurs $k = 1, \cdots, K$. Si l'on désire générer la distribution de la variable $Z(\mathbf{u})$ pour chacune des n localisations $\{\mathbf{u_1}, \mathbf{u_2}, \cdots, \mathbf{u_n}\}$, on pourra par

[1] "Annealing" en anglais

exemple considérer une expression de la fonction coût O du type:

$$O = \sum_{j=1}^{n} \sum_{k=1}^{K} |f_{realisation}(\mathbf{u_j}; k) - f_{reference}(\mathbf{u_j}; k)|$$

où $f(\mathbf{u}; k)$ est une fonction de mesure à définir pour la localisation $\mathbf{u}$ et la catégorie k. L'expression ci-dessus mesure la différence entre une réalisation donnée et un état de référence à honorer.

2. Générer une réalisation initiale.
3. Calculer la valeur de la fonction coût pour cette réalisation.
4. Perturber aléatoirement la réalisation par un mécanisme simple.
5. Calculer la nouvelle valeur O' de la fonction coût pour cette nouvelle réalisation.
6. Accepter ou non cette nouvelle réalisation en fonction d'un critère de décision à déterminer. Mettre à jour la valeur O de la fonction coût.
7. Si O est proche de zéro ($f_{realisation}()$ honore $f_{reference}()$), accepter la réalisation comme étant le modèle final et arrêter le processus, sinon retourner à l'étape 4.

Cette démarche générale fait ressortir les points essentiels suivants:

1. Etant donné le nombre probablement très élevé d'itérations que l'on va devoir effectuer, il est essentiel que la fonction coût se recalcule très facilement.
2. La détermination du mécanisme de perturbation aléatoire doit être efficace et permettre de prendre en compte un maximum de combinaisons possibles.
3. La définition d'un facteur de décision autorisant ou non à conserver une réalisation après qu'une perturbation y eut été apportée apparaît comme déterminante pour assurer la convergence de l'algorithme.

Différents facteurs de décision

On parlera de recuit simulé si le facteur de décision utilisé dans l'étape 6 est basé sur une analogie avec la thermodynamique et la probabilité de distribution de Boltzman. Dans ce cas, la probabilité $P\{accept\}$ d'accepter qu'une réalisation mesurée par une fonction coût O_{new} remplace une réalisation dont le coût est O_{old} est:

$$P\{accept\} = \begin{cases} 1 & \text{si } O_{new} \leq O_{old} \\ e^{-\frac{O_{new}-O_{old}}{t}} & \text{sinon} \end{cases}$$

On accepte alors toutes les réalisations résultant de perturbations où $O_{new} \leq O_{old}$ et on accepte les autres avec une certaine probabilité. Cela signifie que l'on génère un nombre aléatoire entre 0 et 1. Si celui-ci est inférieur à la probabilité d'acceptation,

la perturbation est conservée, sinon elle est rejetée. C'est en fait l'application directe de l'algorithme de Métropolis. t est l'équivalent de la température et diminue au fur et à mesure que le processus converge, la spécification de la décroissance de t en fonction du nombre d'itérations effectuées étant généralement empirique. La probabilité d'accepter une perturbation défavorable diminue donc lorsque la température décroît. De plus, de la vitesse à laquelle diminue cette température dépendra le fait que l'on atteigne ou non un état stable. En effet, si elle diminue rapidement, le liquide ou métal se solidifiera dans un état amorphe ou ne possèdera que certaines parties localement cristalisées. Au contraire, si la diminution de température est très progressive, on obtiendra un cristal parfait (état de niveau d'énergie minimal). Enfin, il est important d'ajouter que le but premier des méthodes de recuit simulé est d'éviter les risques de minimum locaux en autorisant de temps en temps une perturbation défavorable.

Un tel processus de recuit simulé est utilisé par C.L Farmer ([20]). Il est appliqué dans le cadre de méthodes pixels et simule des valeurs de faciès en tout point. La fonction coût utilisée prend en compte le respect d'un histogramme et/ou variogramme issu d'un modèle de référence, pour des points situés à certaines distances et selon certaines directions. Cette méthode s'est avérée efficace, tant au niveau du respect des données que du temps de calcul mis pour parvenir à une réalisation.

On parlera de "Maximum A Postériori Variable" (MAP) lorsque la probabilité d'accepter une nouvelle réalisation est définie par:

$$P\{accept\} = \begin{cases} 1 & \text{si } O_{new} \leq O_{old} \\ 0 & \text{sinon} \end{cases}$$

On n'accepte donc jamais une réalisation qui augmente la valeur de la fonction coût. Un des problèmes soulevés par cette approche est la possibilité de se retrouver au sein d'un minimum local dont on ne peut s'échapper. Cependant, on considère que le fait de représenter un maximum de réalisations à partir de conditions initiales différentes génère une forte probabilité de "récupérer" parmi elles celle correspondant à un minimum global ([40]). Doyen([13]) a utilisé la critère du MAP en exprimant la fonction coût à l'aide de deux informations de natures différentes. Les réalisations produites étaient conditionnées aux deux types de données à la fois.

La troisième façon d'envisager le critère d'acceptation est celle du "Threshold Accepting" introduite par Dueck dans [15]: on accepte ici une perturbation défavorable si le changement dans la fonction objective est inférieure à un certain seuil.

$$P\{accept\} = \begin{cases} 1 & \text{si } O_{new} - O_{old} \leq seuil \\ 0 & \text{sinon} \end{cases}$$

Le *seuil* est diminué au fur et à mesure que le nombre d'itérations augmente.

C.V Deutsch ([11]) a appliqué des méthodes classiques de "recuit simulé" à la modélisation des réservoirs pétroliers. L'idée principale était de pouvoir, à tout moment, mesurer la "température" d'une image. Plus celle-ci différait du modèle à atteindre et plus la température était élevée. On procédait ensuite par itérations successives pour diminuer

la température et faire converger le modèle initial vers le modèle souhaité. Les intérêts de cette méthode résident dans le fait que l'expression de la fonction coût utilisée était composée de plusieurs termes indépendants, chacun d'eux rendant compte du respect de données de types différents. On arrive ainsi à honorer simultanément des contraintes de nature et de qualité très différentes (données de puits, de sismique).

Perturbation du système

Dans tous les exemples cités ci-dessus, on fonctionne sur une grille de pixels et le support de l'information est un point. La perturbation est générée en intervertissant en deux points, la valeur de l'information qu'ils supportent.

Intérêts et contraintes des méthodes de recuit

Les principaux intérêts des méthodes de recuit sont les suivants:

1. Bien que les applications effectuées jusqu'alors l'aient été en considérant l'espace comme un ensemble de points (méthodes pixels), les méthodes de recuit ne leur sont pas spécifiques et doivent pouvoir être étendues à des approches booléennes.

2. Contrairement aux méthodes pixels générales, elles ne reposent pas sur l'existence d'une image de départ de laquelle on puisse tirer les statistiques. Elles n'exigent pas non plus le choix d'un modèle théorique pour exprimer des fonctions de type variogramme.

3. L'expression de la fonction coût doit pouvoir permettre de prendre en compte des données de natures très différentes et de pondérer ces données les unes par rapport aux autres.

4. Le critère d'acceptation d'une perturbation est simple et peut être défini de plusieurs manières. Il doit donc être possible de l'adapter afin de trouver une solution en cas de problème de convergence.

D'autre part, il semble que pour une bonne application de ce type de méthode en trois dimensions, il faille respecter les points essentiels suivants:

1. Avoir une expression de la fonction coût qui soit très facilement mise à jour du point de vue du temps de calcul. Celui-ci doit être le plus rapide possible et ne concerner que les endroits de la réalisation qui auront été affectés par la perturbation. Nous dirons qu'elle doit être mise à jour de manière locale.

2. Le phénomène de perturbation doit lui aussi être rapide. Il reste entièrement à définir dans le cadre d'une approche booléenne. En effet, il conviendra de préciser ce que l'on entend par "perturbation", lorsque le support de l'information est un objet à trois dimensions.

4.4 Algorithme de simulation proposé

Nous allons proposer une méthode qui allie les approches booléennes aux méthodes de recuit et au formalisme de la géostatistique non paramétrique.

4.4.1 Contraintes prises en compte

Nous prendrons en compte trois catégories de contraintes différentes:

1. Les contraintes d'ajustement aux puits. Ces contraintes proviennent des données d'observation des faciès aux puits. On considère que ce type de données doit toujours être disponible.

2. Proportions volumiques relatives des différents faciès à modéliser. Ici encore, nous considèrerons que ce type de données est toujours disponible. Il provient généralement de mesures faites aux puits.

3. Les contraintes de corrélation entre puits. Cette notion a été abordée à la section 3.3 et illustrée par la Figure 0.4, page 6 et la Figure 0.3, page 5. Nous avons vu que certains auteurs ([4],[27]) insistent sur la difficulté à prendre en compte de telles contraintes. Ceci est dû au fait qu'il est souvent très difficile d'être affirmatif sur le fait que deux puits soient ou non corrélés pour un faciès particulier. Il apparaît donc intéressant de pouvoir considérer ce type de données comme des données incertaines en les intégrant également en tant que probabilités de corrélation. Nous considérons que ce type de données n'est pas toujours disponible, mais doit être pris en compte lorsqu'il existe.

4. Les contraintes dites "géométriques". Elles correspondent à la donnée d'un *template*[2] par type d'hétérogénéité à modéliser. Nous considèrerons que la donnée de ces *templates* est obligatoire.

4.4.2 Présentation de l'algorithme

Le sytème que nous avons à modéliser est, rappelons le, un système naturel qui s'est formé en un temps très long. Il s'agit d'un réservoir pétrolier où chacune des hétérogénéités est considérée comme un objet distinct. Nous pensons qu'il est possible d'opérer une analogie entre la distribution des hétérogénéités à l'intérieur de ce réservoir et la distribution des molécules à l'intérieur d'un cristal. En conséquence, il semble logique d'envisager une méthode itérative basée sur les méthodes de recuit simulé pour modéliser la façon dont les hétérogénéités se distribuent à l'intérieur du réservoir. Nous proposons l'algorithme suivant:

1. Déterminer une expression de la fonction coût O qui tienne compte des données de puits (respect des faciès aux puits et des données de corrélation entre puits).

[2] voir chapitre 3

2. Déterminer un nombre maximal d'itérations à effectuer, et *une valeur seuil* de la fonction coût.

3. Construire, pour chaque famille d'hétérogénéités, un *template* que l'on considère comme représentatif de la forme, taille, position dans l'espace, d'un corps appartenant à ce faciès.

4. Générer une image initiale, respectant les données de proportions volumiques relatives entre les différents faciès. Chaque famille d'hétérogénéités est composée de *rshapes* issus d'un *template* père.

5. Séléctionner le sous-ensemble Q des *rshapes* du réservoir que l'on souhaite perturber. Ce sous-ensemble comprend généralement tous les *rshapes* du réservoir, mais il peut parfois être restreint (on peut ne vouloir travailler que sur un seul faciès par exemple).

6. Calculer la valeur de la fonction coût O.

7. Pour tout *rshape* de l'ensemble Q:

   (a) *transformer* cet *rshape* en executant sa *transformation courante*[3].

   (b) Calculer la nouvelle valeur de la fonction coût O'.

   (c) Si $O' \leq O$ conserver la nouvelle réalisation, sinon revenir à la précédente (critère du MAP).

8. Si le nombre d'itérations effectuées a atteint le nombre maximal demandé ou que $O \leq$ *valeur seuil*, arrêter le processus, sinon revenir à l'étape 6.

Une itération de l'algorithme de simulation comprend donc les étapes 7 à 8. Par ailleurs, nous appelerons phase de *perturbation* de l'algorithme de simulation un passage par les étapes (a), (b), (c) de l'étape 7. De ce fait, au cours d'une itération, tous les *rshapes* de l'ensemble Q sont perturbés. Ce processus met en évidence les caractéristiques suivantes:

1. Il reflète les mécanismes généraux des approches booléennes qui considèrent des "formes types" pour modéliser la géométrie des hétérogénéités.

2. Il reflète les mécanismes généraux des méthodes de recuit appliquées aux simulations stochastiques et décrits au cours de la section 4.3.2.

3. Les caractéristiques des objets *rshapes* paraissent permettre d'éviter l'écueil des formes élémentaires trop simples (voir section 0.6.2).

Les différentes étapes de cet algorithme vont maintenant être détaillées.

[3] voir section 3.5.1

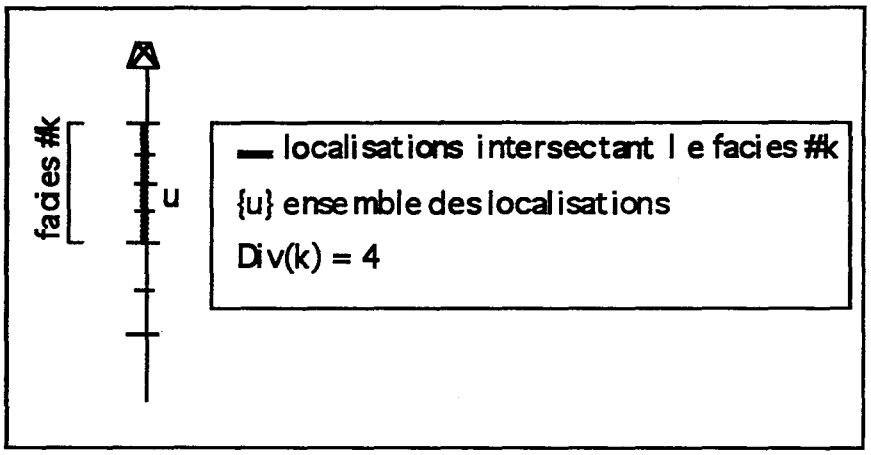


Figure 4.1 Définition des index de discrétisation pour le faciès k. Exemple où $Div(k) = 4$.

4.5 Définition de la fonction coût

4.5.1 Discrétisation des trajets de puits

Soit W, l'ensemble des puits considérés dans le réservoir. Chaque puits est composé de plusieurs "trajets de puits" dont chacun représente un segment n'intersectant qu'un seul faciès. Par ailleurs, on attribue à chacun des K faciès du réservoir un index $k = \{1, 2, \cdots, K\}$. Nous parlerons alors du "faciès d'index k" ou de "faciès k". On peut définir l'ensemble S des localisations définissant le support des données pour ces puits, par une discrétisation de leurs trajets selon la méthode illustrée par la Figure 4.1 et décrite ci-dessous:

1. On assigne à chaque faciès k, un index de discrétisation $Div(k)$;
2. Chaque trajet de puits intersectant un corps appartenant au faciès k est divisé en $Div(k)$ localisations de même longueur.

De ce fait, $Div(k)$ est caractéristique du degré de précision désiré pour chaque faciès. Il peut également apparaître comme une indication du nombre de corps intersectés par un trajet de puits.

4.5.2 Formalisation des données de puits

Soit S l'ensemble des $|S|$ segments $\{\mathbf{u}\}$ supportant les données de puits pour l'ensemble du réservoir et issus d'une discrétisation des trajets de puits. Nous considèrerons au niveau des localisations $\mathbf{u}$ les deux types de données quantifiables suivants:

1. les données correspondant à la caractérisation des différents faciès le long des puits pour les localisations $\{\mathbf{u}\}$;

2. les données correspondant à d'éventuelles probabilités de corrélation entre puits;

Prise en compte des faciès observés

On exprime le fait qu'un segment $\mathbf{u}$ traverse un corps appartenant au faciès k à l'aide de la variable indicatrice suivante:

$$I(\mathbf{u};k) = \begin{cases} 1 & \text{si la localisation } \mathbf{u} \text{ contient (traverse) le faciès } k; \\ 0 & \text{sinon.} \end{cases}$$

Elle traduit l'observation pure et simple d'un type de faciès le long d'une localisation $\mathbf{u}$. Elle correspond donc à une contrainte **dure** (voir section 0.4).

Prise en compte des données de corrélation entre puits

Nous rappelons que l'on appelle "corrélation entre puits" le fait que deux puits traversent le même corps (voir Figure 0.5, page 6). Nous avons signalé précédemment que peu de méthodes pouvaient prendre en compte des contraintes de corrélation entre puits, c'est-à-dire spécifier que deux observations du même faciès le long d'un puits étaient issues du même objet géologique. Cependant, il nous semble intéressant de tenir compte de ce type de données pour plusieurs raisons:

1. Certaines observations au niveau du réservoir permettent de savoir de façon sûre que deux localisations $\mathbf{u}$ et $\mathbf{u}'$ d'un puits **sont** corrélées pour un faciès donné.

2. Certaines observations au niveau du réservoir permettent de savoir de façon sûre que deux localisations $\mathbf{u}$ et $\mathbf{u}'$ d'un puits **ne sont pas** corrélées pour un faciès donné.

3. La prise en compte de ces données de corrélation est d'une extrême importance dans la pratique puisqu'elle va inciter ou non au forage de puits supplémentaires.

4. Il est important d'émettre des hypothèses sur le fait que deux puits soient corrélés afin de pouvoir étudier les conséquences de telles hypothèses. Ces hypothèses peuvent être par exemple mises sous la forme d'une "probabilité de corrélation" pour deux localisations $\mathbf{u}$ et $\mathbf{u}'$.

Plus spécifiquement, nous dirons que deux localisations $\mathbf{u}$ et $\mathbf{u}'$ sont corrélées pour un faciès k, si:

Les deux localisations $\mathbf{u}$ et $\mathbf{u}'$ traversent toutes deux le *même* corps appartenant au faciès k.

Nous exprimons les données de corrélation entre deux localisations $\mathbf{u}$ et $\mathbf{u}'$ à l'aide de la variable indicatrice suivante:

$$I(\mathbf{u},\mathbf{u}';k) = \begin{cases} 1 & \text{si les localisations } \mathbf{u} \text{ et } \mathbf{u}' \text{ contiennent (traversent) le même} \\ & \text{corps de faciès } k; \\ 0 & \text{sinon.} \end{cases}$$

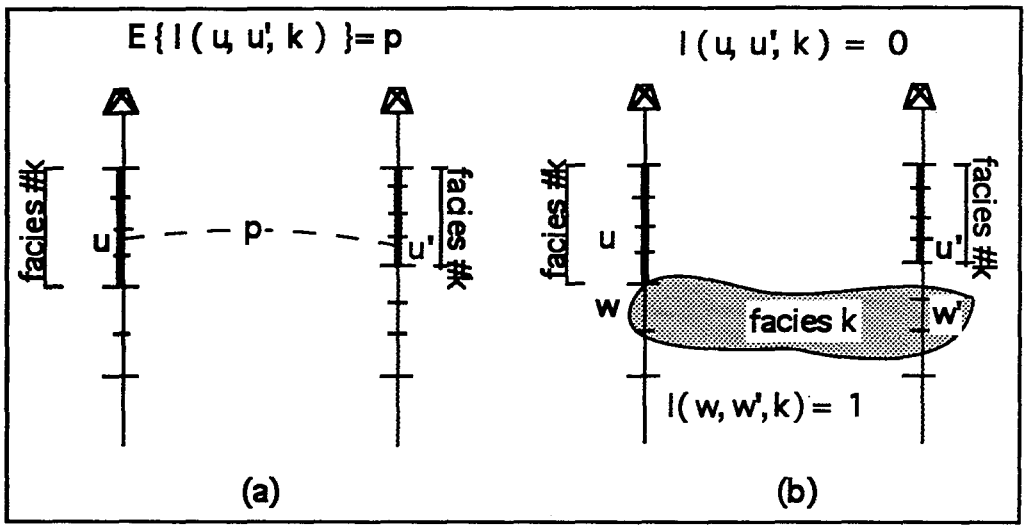


Figure 4.2 (a) Exemple de données de probabilité de corrélation pour le couple $(\mathbf{u}, \mathbf{u}')$ et le faciès k. (b) mesure de la valeur des indicatrices correspondantes pour les couples $(\mathbf{u}, \mathbf{u}')$ et $(\mathbf{w}, \mathbf{w}')$.

Plus généralement, pour permettre de pouvoir prendre en compte des **probabilités** de corrélation, il est plus utile de mesurer l'espérance de cette variable indicatrice. Cela s'exprime de la manière suivante:

$$E\{I(\mathbf{u}, \mathbf{u}'; k)\} = p \in [0, 1]$$

$E\{I(\mathbf{u}, \mathbf{u}'; k)\}$ représente la probabilité pour que les localisations $\mathbf{u}$ et $\mathbf{u}'$ contiennent le *même* objet de faciès k. Lorsque $p = 1$ ou $p = 0$, nous sommes dans le cas de données "dures", lorsque $p \in]0, 1[$, nous sommes dans le cas de données "molles". La Figure 4.2 illustre l'expression des données de corrélation pour les deux couples de localisations $(\mathbf{u}, \mathbf{u}')$ et $(\mathbf{w}, \mathbf{w}')$.

4.5.3 Expression de la fonction coût

La fonction coût O, est, rappelons le, destinée à mesurer la différence entre un état de référence d'un modèle et une réalisation quelconque de ce modèle. Plus la valeur de O est proche de zéro, et plus on est proche de l'état de référence que l'on souhaite atteindre. Dans l'optique d'une analogie avec les processus de recuit, la fonction coût est l'équivalent de l'"énergie" du système. La fonction coût utilisée ici est définie par:

$$O = \frac{1}{2}(O_{adj} + O_{cor}) \tag{4.1}$$

où: O_{adj} est le terme d'*ajustement* de la fonction coût
O_{cor} est le terme de *corrélation* de la fonction coût

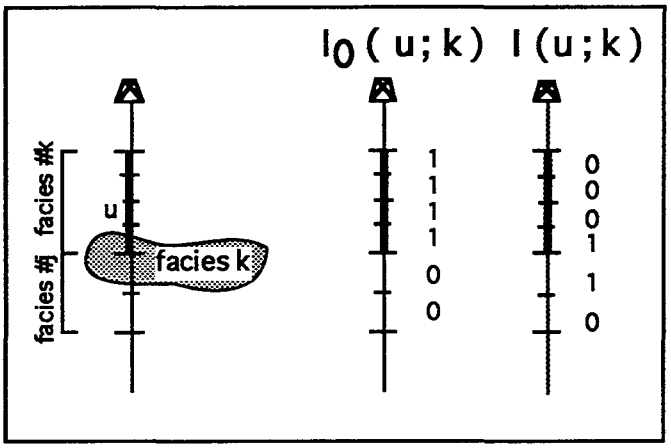


Figure 4.3 Définition des données d'ajustement pour les localisations $\{\mathbf{u}\}$ du puits A

Les deux termes O_{adj} et O_{cor} sont exprimés à l'aide du formalisme de la géostatistique non paramétrique, et sont explicités ci-dessous.

Définition du terme d'ajustement

Pour une réalisation donnée, ce terme mesure la manière dont les observations de faciès aux puits sont respectées. Le terme d'ajustement O_{adj} de la fonction coût est défini pour un nombre total de faciès K et l'ensemble S des localisations de la manière suivante:

$$O_{adj} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{|S|}\sum_{\mathbf{u}\in S}|I(\mathbf{u};k) - I_0(\mathbf{u};k)| \tag{4.2}$$

avec:

$I_0(\mathbf{u};k)$	:	Indicatrice de référence devant être honorée à la localisation $\mathbf{u}$ pour le faciès k.
$I(\mathbf{u};k)$	:	Indicatrice de la réalisation à la localisation $\mathbf{u}$ pour le faciès k.
$\|S\|$	:	nombre d'éléments de l'ensemble S.

De façon pratique, les $\{I_0(\mathbf{u};k)\}$ traduisent quel est le faciès k **observé** dans la réalité au niveau de la localisation $\mathbf{u}$, alors que les $\{I(\mathbf{u};k)\}$ traduisent quel est (sont) le (les) faciès **effectivement présent(s)** à la localisation $\mathbf{u}$ pour une réalisation donnée (voir Figure 4.3). Les indicatrices:

$$\{I_0(\mathbf{u};k), \forall \mathbf{u} \in S, \forall k \in [0, K[\}$$

sont donc l'expression des données de référence et les:

$$\{I(\mathbf{u};k), \forall \mathbf{u} \in S, \forall k \in [0, K[\}$$

sont la mesure de ces données pour une réalisation donnée. Le terme O_{adj} mesure donc la façon dont les données de référence sont honorées. Nous précisons à ce niveau que le

faciès *matrice* du réservoir fait partie intégrante de la définition des K faciès. En effet, le faciès *matrice*[4] est décrit dans les observations aux puits; il est donc pris en compte lors des définitions des $\{I_0(\mathbf{u};k)\}$.

Définition du terme de corrélation

Le terme de corrélation O_{cor} mesure la façon dont les données de corrélation exprimées à travers les variables $E\{I(\mathbf{u},\mathbf{u}';k)\}$ sont respectées pour une réalisation donnée. Soit S_{cor} l'ensemble des $|S_{cor}|$ paires $(\mathbf{u},\mathbf{u}')$ pour lesquelles les données de type $E\{I(\mathbf{u},\mathbf{u}';k)\}$ sont exprimées et soit K_{cor} l'ensemble des $|K_{cor}|$ faciès pour lesquels ces données existent. Le terme O_{cor} de la fonction coût est défini par:

$$O_{cor} = \frac{1}{|K_{cor}|} \sum_{k \in K_{cor}} \frac{1}{|S_{cor}|} \sum_{(\mathbf{u},\mathbf{u}') \in S_{cor}} |I(\mathbf{u},\mathbf{u}';k) - E\{I(\mathbf{u},\mathbf{u}';k)\}| \tag{4.3}$$

avec:

$E\{I(\mathbf{u},\mathbf{u}';k)\}$	: Espérance de la variable indicatrice $I(\mathbf{u},\mathbf{u}';k)$ aux localisations **u** et **u'** pour le faciès k.
$I(\mathbf{u},\mathbf{u}';k)$	: Variable indicatrice de corrélation entre les localisations **u** et **u'** pour le faciès k.
$\|S_{cor}\|$	: nombre d'éléments de l'ensemble S_{cor}.
$\|K_{cor}\|$	: nombre d'éléments de l'ensemble K_{cor}.

Les données de corrélation, lorsqu'elles existent, sont donc exprimées sous la forme d'une probabilité. L'expression 4.3 mesure comment, en moyenne, celles-ci sont respectées. De plus, il faut noter qu'une condition nécessaire pour que $E\{I(\mathbf{u},\mathbf{u}';k)\}$ puisse être définie est que:

$$\begin{cases} I_0(\mathbf{u},k) = 1 \\ I_0(\mathbf{u}',k) = 1 \end{cases}$$

Cette condition traduit le fait que les contraintes d'ajustement et les contraintes de corrélation ne doivent pas être contradictoires. En effet, il est impossible que deux localisations **u** et **u'** où le faciès k n'est pas observé soient corrélées pour k (voir Figure 4.2). Par ailleurs, nous ne prendrons pas en compte le faciès *matrice* dans la définition des $|K_{cor}|$ faciès. En effet, celui-ci ne contenant–par définition–pas d'hétérogénéités, des données de corrélation le concernant ne peuvent pas s'exprimer.

Discussion

1. La valeur de $I(\mathbf{u},\mathbf{u}',k)$ est conditionnée aux valeurs des variables indicatrices $I(\mathbf{u},k)$ et $I(\mathbf{u}',k)$. En effet une condition nécessaire, mais non suffisante, pour que $I(\mathbf{u},\mathbf{u}',k) = 1$ est:

$$I(\mathbf{u};k) = I(\mathbf{u}';k) = 1$$

[4] voir section 0.3

2. Il a été proposé de séparer, dans le terme de corrélation de la fonction coût, les données de corrélation dites "dures" ($E\{I(\mathbf{u},\mathbf{u}';k) = 1$ ou $E\{I(\mathbf{u},\mathbf{u}';k) = 0$), des données de corrélation dites molles ($E\{I(\mathbf{u},\mathbf{u}';k) \in]0,1[$). Cependant, il n'y a pas de raisons théoriques pour considérer différemment ces deux types de données. De plus, nous verrons plus bas que l'existence de données de corrélation dures se révèle être une contrainte très forte au niveau de la simulation et que le processus lui donne souvent priorité par rapport aux autres contraintes.

3. O_{adj} et O_{cor} varient chacun entre 0 et 1. Dans l'expression 4.1, leur contribution à la valeur totale de la fonction coût est donc la même. Il est cependant facile de pondérer la contribution relative de chacun des deux termes en intégrant les coefficients entiers, positifs, a_{adj} et a_{cor} tels que:

$$O = a_{adj}.O_{adj} + a_{cor}.O_{cor} \tag{4.4}$$

avec: $a_{adj} + a_{cor} = 1$, $a_{adj} \geq 0, a_{cor} \geq 0$.

4. Lorsque l'on ne possède pas de données de corrélation, on considère que $O_{cor} = 0$ et que $O = O_{adj}$.

5. L'expression 4.2 pondère chacun des K faciès par un poids de $\frac{1}{K}$. On pourrait introduire une pondération relative différente pour ces faciès en considérant l'expression suivante du terme O_{adj}:

$$\begin{cases} O_{adj} = \sum_{k=1}^{K} \omega_k \frac{1}{|S|} \sum_{\mathbf{u} \in S} |I(\mathbf{u};k) - I_0(\mathbf{u};k)| \\ \sum_{k=1}^{K} \omega_k = 1, \omega_k \geq 0, \forall k \end{cases} \tag{4.5}$$

6. De même, l'expression 4.3 pondère chacun des $|K_{cor}|$ faciès par un poids de $\frac{1}{|K_{cor}|}$. On pourrait introduire une pondération relative de ces faciès en considérant l'expression suivante du terme O_{cor}:

$$\begin{cases} O_{cor} = \sum_{k \in K_{cor}} \omega_k \frac{1}{|S_{cor}|} \sum_{(\mathbf{u},\mathbf{u}') \in S_{cor}} |I(\mathbf{u},\mathbf{u}';k) - E\{I(\mathbf{u},\mathbf{u}';k)\}| \\ \sum_{k=1}^{|K_{cor}|} \omega_k = 1, \omega_k \geq 0, \forall k \end{cases} \tag{4.6}$$

Il faut préciser ici que l'ensemble K_{cor} n'étant pas composé des K faciès du réservoir (le faciès *matrice* n'est pas pris en compte dans les contraintes de corrélations), les poids $\{\omega_k\}$ utilisés dans les expression 4.5 et 4.6 ne sont pas les mêmes.

7. Si on considère les expressions 4.2, 4.3 et 4.1, on constate que le respect des contraintes de corrélation "dures" est avantagé par rapport au respect des contraintes

d'ajustement. En effet, le poids d'une contrainte de corrélation "dure" telle que $I(\mathbf{u}, \mathbf{u}'; k) = 1$ ou $I(\mathbf{u}, \mathbf{u}'; k) = 0$ est beaucoup plus élevé que celui d'une contrainte d'ajustement: son non-respect entraîne une augmentation de O égale à $\frac{1}{2}\frac{1}{|S_{cor}|.|K_{cor}|}$, alors que le non-respect d'une contrainte d'ajustement augmente O de $\frac{1}{2}\frac{1}{|S|.K}$. Or par définition, $|S_{cor}| < |S|$ et $|K_{cor}| < K$.

4.6 Génération du modèle initial

Soit $(K-1)$ le nombre de familles d'hétérogénéités contenues dans le réservoir. La génération du modèle initial se fait en générant $(K-1)$ faciès différents par une des trois procédures (stochastique, déterministe ou mixte) décrite à la section 3.5.2. La procédure stochastique de création de faciès, qui est la plus complexe, engendre les remarques suivantes:

1. Le mode d'initialisation d'un *rshape* d'après leur *template* père lui donne une position aléatoire au sein du faciès auquel il appartient et ceci, grâce à l'opération *translation* qui lui est appliquée durant ce processus (voir section 3.5.2). L'image initiale est donc composée d'objets *rshapes* aléatoirement disposés dans le réservoir.

2. Durant le processus de création de faciès, la *transformation courante* des *rshapes* qu'il contient est initialisée à *translation*. Cela signifie que l'on pense que, au départ, c'est celle qui a le plus de chance de conduire à une diminution rapide de la fonction coût.

3. Différentes initialisations de la fonction $rand(0,1)$ générant des nombres aléatoires entre 0 et 1 engendreront des modèles initiaux différents. Les modèles finals seront donc également différents et équiprobables.

4.7 Description de la phase perturbation

Définition

Lors des procédures de recuit exposées à la section 4.3.2, l'étape de *perturbation*, se produit le plus souvent de la manière suivante:

1. Sélectionner au hasard deux points de la grille de pixel ;

2. interchanger les valeurs des attributs portées par ces deux points.

Une démarche similaire, dans le cas où l'on travaille avec des objets plutôt qu'avec des points, consisterait à:

1. Sélectionner deux *rshapes* au hasard dans le réservoir;

2. Echanger la position de leurs centres de gravité.

Cependant, cette approche aurait le désavantage de manipuler des *rshapes* dont *l'enveloppe* est figée une fois pour toute. Or nous avons vu que les objets *rshapes* étaient conçus pour pouvoir se déformer de manière aléatoire. Ceci veut dire que la géométrie d'un *rshape* peut être considérée comme un attribut stochastique de celui-ci. Elle doit respecter les contraintes suivantes:

1. Les données de distribution de tailles et d'orientation de la famille d'hétérogénéités à laquelle appartient le *rshape*.
2. La topologie, la paramétrisation et la forme globale de son *template* père.

Nous avons tiré parti de la capacité des objets *rshapes* à définir différentes réalisations de leur géométrie pour définir la phase *perturbation* de l'algorithme de simulation. Pour chaque *rshape* perturbé, elle se déroule de la manière suivante:

1. Sélectionner la *transformation courante* du *rshape*.
2. Appliquer cette transformation au *rshape*.
3. En fonction du résultat sur la valeur de la fonction coût:
   (a) mettre à jour l'*enveloppe* du *rshape*.
   (b) mettre à jour la *transformation courante* du *rshape*.
   (c) mettre à jour les paramètres de la *transformation courante*.

Conséquences

La phase *perturbation* de l'algorithme de simulation se caractérise de la manière suivante:

1. La *transformation courante* appliquée à un *rshape* et les paramètres de chacune des transformations aléatoires qui lui sont associées sont dépendants de l'objet auquel ils s'appliquent.
2. *transformation courante* et paramètres associés peuvent être modifiés (généralement de manière déterministe) durant le processus de simulation.

Il est donc possible que la phase *perturbation* appliquée à deux *rshapes* distincts n'engendre pas le même type de transformation sur ces objets, et ceci même s'ils sont issus d'un *template* identique. L'ampleur de ces transformations peut également varier. Ainsi, un *rshape* dont on considère qu'il est "bien placé" pourra se voir limiter à un tout petit déplacement (autour d'un puits par exemple) lors de l'opération *translation*, alors que d'autres pourront évoluer dans tout l'espace du réservoir. De ce fait, le contrôle de l'étape de "mise à jour" des transformations appliquées à chaque *rshape* est très importante. On peut supposer qu'il obéisse à des règles telles que:

1. Tant que la fonction coût O n'a pas "suffisamment" décrue, laisser la transformation courante à sa valeur *translation* défaut:

2. Si O est suffisamment basse, ou reste à une valeur plancher, on peut influer sur la forme de l'*enveloppe* des *rshapes* en changeant la valeur de sa *transformation courante.*

3. Le modèle peut être affiné en limitant l'ampleur des transformations par modification des paramètres non *template* dépendants. On peut de cette façon contrôler l'espace dans lequel se déplace un *rshape* ou les écarts autorisés par rapport aux valeurs moyennes issues du *template.*

4. Les opérations ci-dessus peuvent être aussi bien réalisées de façon globale que locale. Dans le premier cas on considère que tous les *rshapes* au sein d'un même faciès sont traités de la même manière. Dans le second cas, on s'autorise à traiter différemment certains *rshapes.*

Cependant, ces règles restent très subjectives et sont là encore, liées à l'intuition et au degré de précision désiré. Dans l'état actuel de nos travaux, les deux opérations

1. Modification de la *transformation courante* d'un *rshape.*

2. Modification des paramètres des transformations d'un *rshape.*

relèvent d'une décision *déterministe*, puisque subjective. Un élément de décision peut être de considérer les évolutions de la fonction coût durant les dernières itérations.

4.8 Mise à jour de la fonction coût

Nous avons vu que, pour qu'un processus fondé sur les principes des méthodes de recuit soit efficace, il fallait que la fonction coût puisse être très rapidement mise à jour, et ceci, de manière locale. Si l'on considère l'expression mathématique du terme d'ajustement de la fonction coût O, la mettre à jour "localement" signifie que:

> A chaque phase *perturbation* de l'algorithme de simulation, on ne recalcule que les valeurs $I(\mathbf{u};k)$ qui auront été susceptibles d'être modifiées.

De plus, nous avons dit que dans certains cas, les valeurs de $I(\mathbf{u},k)$ et $I(\mathbf{u}',k)$ pouvaient permettre d'assigner la valeur 0 à $I(\mathbf{u},\mathbf{u}';k)$ (voir section 4.5.3). Il sera donc intéressant de calculer la valeur de O_{adj} (et donc de mettre à jour les variables indicatrices $\{I(\mathbf{u},k)\}$) avant de calculer O_{cor}.

4.8.1 Intersection entre un rshape et les localisations du réservoir

Soit $W = \{W^j, j \in [0,W[\}$ l'ensemble des W puits du réservoir, chaque W^j supportant les localisations du réservoir telles que $W^j = \{u_i^j\}$. Nous appelerons B^j le parallélépipède le plus petit contenant W^j et B^r le parallélépipède le plus petit contenant r. L'algorithme qui calcule quelles sont les localisations du réservoir qui intersectent r se déroule de la

manière suivante:

pour tout puit $W^j \in W$

si B_r et W^j s'intersectent

pour tout $u_i^j \in W^j$

si le segment u_i^j intersecte B_r

. soient $t = \{t_i, i \in [1, M]\}$ l'ensemble des M triangles décomposant l'*enveloppe* de r[5]

pour tout $t_i \in t$

. calcul de l'intersection entre le segment u_i^j et le triangle t_i

. si l'intersection est non nulle stocker u_i^j dans l'ensemble des localisations intersectant r finsi.

finpour.

finsi.

finpour.

finsi.

finpour.

Nous remarquons que l'opération la plus coûteuse, le calcul de l'intersection entre un triangle et un segment, n'est effectuée que lorsque cela est absolument nécessaire. En effet, le test d'intersection avec la plus petite boîte contenant le *rshape* permet d'écarter les cas où ce calcul est inutile.

4.8.2 Mise à jour du terme d'ajustement

Nous rappelons que le terme de *matrice* est utilisé pour décrire l'"encaissant" du réservoir. De ce fait, on définit $I(\mathbf{u};matrice)$ de la manière suivante:

$$I(\mathbf{u}, matrice) = \begin{cases} 1 & \text{si aucun } rshape \text{ n'est localisé en } \mathbf{u} \text{ ;} \\ 0 & \text{sinon.} \end{cases}$$

La variable indicatrice $I(\mathbf{u}, matrice)$ est donc affectée par la position des *rshapes* dans le réservoir. Elle devra de ce fait être recalculée à chaque fois qu'une indicatrice:

$$\{I(\mathbf{u}, k), k = 1, 2, \cdots, K-1\}$$

aura pu être affectée par la phase *perturbation* appliquée à un *rshape* de faciès k.

Soient:
- $S = \{u_i, i \in [1, |S|]\}$: l'ensemble des $|S|$ localisations du réservoir.
- K : le nombre de faciès total dans le réservoir.
- m : l'index associé au faciès *matrice*.
- L_{u_i} : la liste des *rshapes* intersectés par la localisation u_i.

Soit d'autre part, une réalisation dont on connaît la valeur du terme O_{adj} et soient δO_{old} et δO_{new} deux variables tampons initialisées à 0.

Nous noterons E^r_{old} l'*enveloppe* d'un *rshape* r avant *perturbation*, et E^r_{new}, son *enveloppe* après *perturbation*. Lorsqu'un *rshape* r appartenant au faciès l est perturbé par le processus de simulation, la nouvelle valeur de O_{adj} devient $O_{adj} + \delta O_{adj}$. Si l'on considère l'expression 4.2 de O_{adj}, on peut alors calculer la valeur algébrique de δO_{adj} de la façon suivante:

pour tout $u_i \in S$

Mise à jour des localisations intersectant E^r_{new}:

si $\mathbf{u_i}$ intersecte E^r_{new}:

si $\mathbf{u_i}$ n'intersecte pas E^r_{old}

. $\delta O_{old} = \delta O_{old} + |I(\mathbf{u_i}; l) - I_0(\mathbf{u_i}; l)| + |I(\mathbf{u_i}; m) - I_0(\mathbf{u_i}; m)|$

. $I(\mathbf{u_i}, l) = 1$;

. $I(\mathbf{u_i}, m) = 0$;

. $\delta O_{new} = \delta O_{new} + |I(\mathbf{u_i}; l) - I_0(\mathbf{u_i}; l)| + |I(\mathbf{u_i}; m) - I_0(\mathbf{u_i}; m)|$

. Ajouter r dans la liste $L_{\mathbf{u_i}}$ des *rshapes* intersectés par $\mathbf{u_i}$.

sinon $L_{\mathbf{u_i}}$ et $I(\mathbf{u_i}, k), k \in [1, K]$ sont à jour.

finsi.

sinon

Mise à jour des localisations qui intersectent E^r_{old}:

si u_i intersecte E^r_{old}

. retirer r de la liste L_{u_i} des *rshapes* intersectés par $\mathbf{u_i}$.

. $.\delta O_{old} = \delta O_{old} + |I(\mathbf{u_i}; l) - I_0(\mathbf{u_i}; l)| + |I(\mathbf{u_i}; m) - I_0(\mathbf{u_i}; m)|$

. $I(\mathbf{u_i}; m) = \begin{cases} 1 & \text{si } L_{u_i} = \emptyset \\ 0 & \text{sinon.} \end{cases}$

. $I(\mathbf{u_i}, l) = \begin{cases} 0 & \text{si } L_{u_i} \text{ ne contient aucun } \textit{rshape} \text{ de faciès } l \\ 1 & \text{sinon;} \end{cases}$

. $.\delta O_{new} = \delta O_{new} + |I(\mathbf{u_i}; l) - I_0(\mathbf{u_i}; l)| + |I(\mathbf{u_i}; m) - I_0(\mathbf{u_i}; m)|$

sinon $L_{\mathbf{u_i}}$ et $I(\mathbf{u_i}, k), k \in [1, K]$ sont à jour.

finsi.

finsi.

finpour

La valeur algébrique de δO_{adj} est alors égale à:

$$\frac{1}{|S|}\frac{1}{K}(\delta O_{old} - \delta O_{new})$$

et $O_{adj} = O_{adj} + \delta O_{adj}$. Dans cet algorithme, l'étape la plus délicate consiste à calculer l'intersection de l'enveloppe d'un *rshape* r avec une localisation $\mathbf{u_i} \in S$. Le détail de ce calcul a été donné à la section précédente.

4.8.3 Mise à jour du terme de corrélation

On considère que le terme O_{adj} de la fonction coût a été mis à jour par le processus exposé à la section précédente, les variables indicatrices:

$$\{I(\mathbf{u},k), \mathbf{u} \in S, k \in [1,K]\}$$

le sont alors également ainsi que les listes:

$$\{L_u, \mathbf{u} \in S\}$$

des *rshapes* intersectés par les localisations $\mathbf{u}$. Soit une réalisation dont on connaît la valeur du terme O_{cor}. Lorsque l'*enveloppe* E^r_{old} d'un *rshape* r appartenant au faciès l est perturbée en E^r_{new}, la nouvelle valeur de O_{cor} devient $O_{cor} + \delta O_{cor}$. Soient δO_{old} et δO_{new} deux variables tampons initialisées à 0. Soient d'autre part:

$$S_{cor} = \{(\mathbf{u_i}, \mathbf{u_j}); i,j \in [1,|S|]; i \neq j\}$$

l'ensemble des $|S_{cor}|$ paires de localisations et K_{cor} l'ensemble des $|K_{cor}|$ faciès pour lesquels $E\{\mathbf{u_i}, \mathbf{u_j}, k\}$ est défini. Si on considère l'expression 4.3, on peut alors calculer la valeur algébrique de δO_{cor} de la manière suivante.

pour tout $(\mathbf{u_i}, \mathbf{u_j}) \in S_{cor}$

. $\delta O_{old} = \delta O_{old} + \left[I(\mathbf{u_i}, \mathbf{u_j}, l) - E\{I(\mathbf{u_i}, \mathbf{u_j}, l)\}\right]$;

Si il y a possibilité de corrélation:

si $I(\mathbf{u_i}, l) = I(\mathbf{u_j}, l) = I_0(\mathbf{u_i}, l) = I_0(\mathbf{u_j}, l) = 1$

si $L_{\mathbf{u_i}}$ et $L_{\mathbf{u_j}}$ contiennent r alors $I(\mathbf{u_i}, \mathbf{u_j}, l) = 1$

sinon $I(\mathbf{u_i}, \mathbf{u_j}, l) = 0$

finsi.

finsi.

. $\delta O_{new} = \delta O_{new} + \left[I(\mathbf{u_i}, \mathbf{u_j}, l) - E\{I(\mathbf{u_i}, \mathbf{u_j}, l)\}\right]$;

finpour.

On peut alors calculer δO_{cor} tel que:

$$\delta O_{cor} = \frac{1}{|S_{cor}|.|K_{cor}|}(\delta O_{new} - \delta O_{old})$$

et la nouvelle valeur de O_{cor} telle que: $O_{cor} = O_{cor} + \delta O_{cor}$.

On remarque ici que la mise à jour n'est pas faite localement mais globalement. Ceci n'est cependant pas gênant puisque les calculs ont déjà été effectués lors de la mise à jour du terme O_{adj} et qu'il s'agit ici d'additionner des variables indicatrices et non de recalculer des intersections.

4.9 Mise à jour de la réalisation courante

On considère une réalisation courante contenant le rshape r appartenant au faciès l. Soit O la mesure de la fonction coût et E^r_{old} l'*enveloppe* de r avant que la phase *perturbation* soit appliquée. Soit E^r_{new}, son *enveloppe* après application de la *perturbation*. Si on considère l'expression 4.1, la mise à jour de la réalisation courante, après la phase de *perturbation* se produit de la façon suivante:

1. Calcul de δO_{adj}. Cette opération met également à jour les variables indicatrices correspondant aux contraintes d'ajustement et les listes $\{L_{\mathbf{u_i}}, \mathbf{u_i} \in S\}$ des *rshapes* intersectés par les localisations $\{\mathbf{u_i}\}$.
2. Calcul de δO_{cor}. Cette opération met également à jour les variables indicatrices correspondant aux contraintes de corrélation.
3. Calcul de O_{new} telle que: $O_{new} = O + \frac{1}{2}(\delta O_{adj} + \delta O_{cor})$
4. si $(O_{new} \leq O)$, on garde la nouvelle réalisation de l'*enveloppe* E^r_{new} et O_{new} devient la valeur courante de la fonction coût.
5. sinon:
   (a) Retour à l'ancienne géométrie de l'*enveloppe* de r, E^r_{old}.
   (b) Mise à jour de δO_{adj} et δO_{adj}, ce qui revient à recalculer les anciennes valeurs des variables indicatrices.
   (c) $O = O_{new} + \frac{1}{2}(\delta O_{adj} + \delta O_{cor})$
6. finsi.

4.10 Intégration d'une contrainte de parallélisme

Nous rappelons que les objets *rshapes* partagent toutes les propriétés des *gshapes*. Ils héritent donc du *vecteur horizon* défini au niveau du *template* dont ils sont issus. On peut, au niveau de la simulation, spécifier que tous les *rshapes* appartenant au même faciès doivent être parallèles à une surface donnée. Cette spécification de parallélisme se fait au sens exposé à la section 2.3.3. La spécification d'une telle contrainte est alors intégrée à l'algorithme de simulation de la manière suivante:

1. Fixer le nombre *nbIter* d'itérations qui seront effectuées sans prendre en compte de contraintes de parallélisme.

2. Effectuer *nbIter* itérations au niveau de l'algorithme de simulation "classique" exposé à la section 2.4.2.

3. Pour toutes les familles d'hétérogénéités pour lesquelles une contrainte de parallélisme est définie, appliquer l'algorithme exposé à la section 2.3.3 à chaque *rshape* contenu dans le faciès.

4. Mise à jour de la fonction coût.

5. Retour à l'étape 2.

La possibilité de respecter des contraintes de parallélisme entre une surface donnée et une famille d'hétérogénéités autorise les modélisations dans des espaces qui ont été déformés après le dépôt des hétérogénéités (on ne prend toutefois pas en compte ici le problème des failles). La surface S est alors une "surface temps" dont on sait qu'elle était horizontale au moment du dépôt et qui a été modifiée par la suite.

4.11 Discussion

1. Par rapport aux méthodes de recuit simulé classiques, on remarque ici que c'est le critère du MAP qui est utilisé pour accepter une réalisation perturbée. Nous avons dit qu'un des problèmes de cette approche était le danger de ne représenter qu'un "minimum local". Cependant, nous n'avons pas eu l'impression de constater ce type de problème lors de l'application de la méthode (voir chapitre 5). Si toutefois celui-ci apparaîssait ultérieurement, il semble a priori facile de remplacer le critère du MAP par celui de Métropolis ou du Threshold Accepting (voir section 4.3.2).

2. De même, nous ne modélisons pour l'instant que des distributions gaussiennes au niveau des tailles et des directions d'élongation. Cependant, il serait possible de prendre en compte n'importe quel type de distribution programmée.

3. L'étendue des *perturbations* autorisées pour un *rshape* peut, une fois encore, paraître limitée. La liste des opérations de base proposée pourra toutefois être allongée ultérieurement. L'implémentation présentée à l'annexe A prend en compte cette possibilité.

4. Les variations latérales de faciès ne se font pas ici en respectant la valeur de fonctions de type variogramme. Cependant, nous intégrons:

   (a) les directions globales d'élongation des corps;

   (b) les contraintes de corrélation entre puits.

5. Contrairement à d'autres méthodes, ([4]) nous ne prenons pour l'instant pas en compte de possibilité d'interaction entre corps du type attraction-répulsion. On peut imaginer qu'une telle contrainte puisse être introduite:

   (a) en intégrant un terme supplémentaire à la fonction coût;

(b) en procédant de manière analogue à la contrainte de parallélisme, c'est à dire qu'après un nombre donné d'itérations, on procède à un réagencement des corps dans l'espace;

(c) en intégrant un nouveau type de *perturbation* associée à l'algorithme de simulation.

Cela impliquerait la mise en place de calculs rapides d'intersections ou de distances entre corps, ce qui est loin d'être évident lorsque l'on travaille avec des objets de formes complexes et non traduites par une expression analytique simple. Une approche possible pourrait être de considérer des opérations entre tétraèdres puisque, nous l'avons vu, un objet *rshape* se décompose facilement en un ensemble de tétraédres (voir section 2.6.2).

6. Le nombre n_k de *rshapes* dans chaque faciès k ne varie pas au cours de la simulation. Or on aurait pu imaginer que celui-ci soit recalculé de manière à honorer complètement les proportions relatives de chaque faciès en prenant en compte les intersections entre corps. Encore une fois, ceci est une opération complexe. La formule donnée par Stoyan ([54]) est donc utilisée pour procéder à la génération d'un faciès de manière stochastique[6]. De plus, il semble que le respect "absolu" des proportions volumiques ne soit pas une nécessité pour les géologues puisque celles-ci sont des données incertaines, très souvent déduites de l'observation de deux ou trois forages....

4.12 Conclusion

Par rapport aux approches booléennes traditionnelles et dont certaines ont été décrites à la section 0.6.2, nous ferons les remarques suivantes:

1. La méthode proposée non seulement honore les observations aux puits, mais également d'éventuelles probabilités de corrélation entre puits.

2. La méthode proposée est générale dans le sens où l'on n'est pas dépendant d'une librairie d'objets de formes prédéfinies comme c'est le cas dans beaucoup de méthodes booléennes ([7],[26],[32],[27]). Au contraire, l'utilisateur est maître de la forme type envisagée et la modélise lui-même. On n'est donc pas limité à un environnement géologique particulier.

3. La fonction coût utilisée pour mesurer la différence entre la réalisation courante et les données à respecter est simple et aisément mise à jour. De plus, le fait qu'elle soit composée de plusieurs termes autorise une grande flexibilité d'évolution. En effet, on peut penser qu'il sera par la suite possible d'intégrer d'autres termes mathématiques rendant compte du respect de types de données différents.

[6] voir section 3.6.1

4. Une contrainte de parallélisme entre une surface donnée et des *rshapes* appartenant à un faciès particulier peut être définie. Cela peut éviter d'avoir à "horizontaliser" le modèle (c'est à dire reconstituer le modèle au moment du dépôt) avant de procéder à la simulation ([1]).

4.13 Résumé du chapitre

Nous avons présenté dans ce chapitre les bases théoriques qui ont servi d'appui à la mise en œuvre de notre méthode de simulation stochastique. Celle-ci repose sur le principe général des méthodes de recuit: on part d'un modèle initial que l'on perturbe de façon à diminuer son énergie globale. Cette énergie est exprimée à l'aide d'une fonction coût qui mesure l'écart entre une réalisation du modèle et les données à respecter. Dans notre algorithme de simulation, nous avons exprimé la fonction coût en utilisant le formalisme des variables indicatrices dont les caractéristiques sont les suivantes:

1. Il autorise, par son formalisme binaire, la prise en compte de données de natures très différentes qui sont dans le cas de données dures, exprimées sous forme de 0 et de 1.

2. Il permet d'exprimer de façon très simple les observations aux puits et les données de corrélation entre puits.

3. Il permet de mettre à jour facilement et de manière locale, la fonction coût.

On a pu ainsi adapter les méthodes de recuit à une approche booléenne du problème. En effet, nous considérons que chaque hétérogénéité est représentée par un objet *rshape* au sein du réservoir et que le moteur de la phase *perturbation* de l'algorithme de simulation est l'application d'une des quatres opérations de base *translation, rotation, affinité, déformation* à chaque *rshape* du réservoir. Il est également possible de contraindre une famille d'hétérogénéités à rester "parallèle" à une surface donnée au cours du processus de simulation. Les intérêts de la méthode sont les suivants:

1. Ses principes sont simples et facilement appréhendés;

2. L'entrée des données est simple, intéractive, et aisément modifiable.

3. L'algorithme produit des réalisations équiprobables qui tendent à respecter à la fois les données d'ajustement aux puits, et celles de corrélation entre puits.

4. Les géométries des hétérogénéités sont bien définies les unes par rapport aux autres, et bien que similaires, ne sont pas toutes identiques au sein d'une même famille.

5. C'est la déformation de la géométrie des objets qui sert de base au processus de *déformation* du processus.

Chapitre 5

Application à des cas réels

5.1 Présentation du chapitre

Nous présenterons dans ce chapitre trois exemples différents d'application de la méthode de simulation stochastique de distribution d'hétérogénéités proposée au chapitre précédent. Nous détaillerons et interprèterons les résultats pour chacun d'eux, tant au niveau qualitatif que quantitatif (contraintes informatiques). Enfin, nous discuterons ces résultats et en tirerons les conclusions.

5.2 Introduction

Trois exemples d'application sont traités ici. Le premier est effectué sur un réservoir essentiellement composé de deux familles d'hétérogénéités (*chenal* et *lobe*) et où le faciès *matrice* est largement majoritaire par rapport à ceux-ci. La simulation s'effectue à l'échelle du réservoir.

Le deuxième exemple de simulation a pour but de représenter un modèle régional d'environnement de dépôt plutôt que d'étudier ce qui se passe aux puits. La taille des corps présents dans le réservoir est beaucoup plus grande que celle des corps du premier exemple. Nous simulerons indépendamment deux séquences de dépôt différentes comprenant respectivement 4 et 5 familles d'hétérogénéités. La proportion du faciès *matrice* est beaucoup moins élevée que pour l'exemple précédent.

Le troisième exemple concerne un modèle de réservoir contenant deux familles d'hétérogénéités, le faciès *lobe* et le faciès *chenal*, tous deux composés de peu de corps. On considère ici que l'on part d'une représentation déterministe d'un modèle, afin d'étudier les variations géométriques possibles autour de ce modèle.

Nous considérerons dans tous les cas que la simulation se situe dans un environnement correspondant à celui du dépôt, c'est-à-dire que les données ont été "horizontalisées" avant d'être traitées. Ceci est une pratique courante dans l'industrie pétrolière, et il est souvent difficile d'avoir accès à d'autres types de données. La contrainte de "parallélisation par rapport à une surface", détaillée à la section 2.3.3 ne sera donc pas mise en pratique. Par

ailleurs, les expressions utilisées pour le calcul de la fonction coût sont celles correspondant aux équations 4.1, 4.2 et 4.3 et l'ensemble Q des *rshapes* perturbés dans le processus de simulation[1] contient, à chaque fois, tous les *rshapes* du réservoir.

5.3 Premier exemple

Ce premier exemple a été complètement décrit dans la littérature ([1], [58]). Il s'agit d'un réservoir de dimensions $1750m \times 1650m \times 92m$ contenant deux familles d'hétérogénéités correspondant au faciès *chenal* et au faciès *lobe*. Le faciès *matrice* est largement majoritaire. Les données sont issues de six puits composés au total de vingt six trajets de puits représentant chacun un segment où l'on observe de façon continue le même faciès. Ainsi, dans la représentation de la Figure 5.1 (a). les trajets de puits intersectant des chenaux sont en rouge, ceux intersectant des lobes sont en vert, et ceux intersectant la matrice sont en jaune.

5.3.1 Description des faciès

Table 5.1 Exemple 1: valeurs moyennes empiriques associées aux faciès *lobe* et *chenal*.

	chenal	*lobe*	*matrice*
largeur/épaisseur	11	71	
épaisseur	9	8	
orientation	145E	145E	
longueur (m)	1000	800	
proportion	0.11	0.05	0.84

Un objet type, caractéristique des faciès *chenal* et *lobe* a été décrit par les géologues. Les valeurs moyennes correspondant à leurs caractéristiques de tailles et d'orientation sont données dans le tableau 5.1: comme nous l'avons vu au cours des chapitres précédents, nous représentons chaque famille d'hétérogénéités par un *template*. Celui-ci est un objet *gshape*, modélisé par l'utilisateur, et dont les caractéristiques de tailles (épaisseur et rapport $\frac{largeur}{epaisseur}$) et d'orientation reflètent les valeurs moyennes de ces attributs pour les faciès qu'ils représentent.

template associé au faciès "chenal"

Celui-ci est représenté dans la Figure 5.1 (b) avec un facteur d'échelle de 10 en z. Ses paramètres de tailles et d'orientation correspondent à celles relevées dans le tableau 5.1. Les caractéristiques géométriques du *template* sont les suivantes:

[1] voir section 4.4.2

1. Il est paramétrisé par 4 *sections de contrôle* et 4 *nœuds de contrôle.* Ces *nœuds de contrôle* donnent au *backbone* une sinuosité relativement peu prononcée.
2. On a attribué au *contour* de ses *sections de contrôle* la forme type d'une coupe transversale de *chenal* (voir Figure 2.1, page 44).
3. Son *vecteur horizon*[2] est initialisé de manière à limiter le *toit* du *mur* du *chenal* (voir Figure 5.1 (b)).
4. Son inclinaison par rapport à l'horizontale est pratiquement nulle.

template associé au faciès "lobe"

Celui-ci est également représenté dans la Figure 5.1 (b) et ses paramètres de tailles et d'orientation issus du tableau 5.1. Ses caractéristiques géométriques sont les suivantes:

1. Il est paramétrisé par 4 *sections de contrôle* et 2 *nœuds de contrôle.* Ces *nœuds de contrôle* donnent au *backbone* une sinuosité nulle.
2. On a attribué au *contour* de ses *sections de contrôle* une forme elliptique, dont la taille décroît au fur et à mesure que l'on s'éloigne du centre de gravité du *template.*
3. Son *vecteur horizon* est initialisé de manière à limiter le *toit* du *mur* du *lobe.* Celui-ci est défini comme étant l'axe horizontal des *contours* elliptiques. (voir Figure 5.1 (b)).
4. Son inclinaison par rapport à l'horizontale est pratiquement nulle.

Paramètres de transformation associés aux faciès

Comme nous l'avons vu au cours du chapitre 3, la géométrie d'un *rshape* au sein d'un réservoir varie autour de celle de son *template* père. L'ampleur de cette variation est spécificiée par les paramètres *template indépendants* associés aux transformations aléatoires *translation, rotation, affinité, déformation.* Ces paramètres ont été décrits à la section 3.5.3 et nous reprendrons ici les notations utilisées à cette occasion. Le tableau 5.2 donne la valeur associée à chacun des paramètres de transformation pour les faciès *chenal* et *lobe.*

Paramètres de discrétisation

Nous rappelons que les paramètres de discrétisation de la simulation correspondent aux valeurs de $\{Div(k)\}$ pour k décrivant les index associés aux faciès *lobe, chenal, matrice.* Ils sont utilisés pour définir le nombre de localisations associées à chaque trajet de puits intersectant le faciès k (voir section 4.5.2). Ceux-ci sont donnés dans le tableau 5.3

On remarque que l'on discrétise chaque trajet de puits en deux localisations. Le nombre de trajets de puits étant égal à 28, cela engendre un nombre de localisations $\{\mathbf{u}\}$ égal à

[2] voir section 2.3.2

Table 5.2 Exemple 1: paramètres des transformations aléatoires associés aux faciès *lobe* et *chenal.*

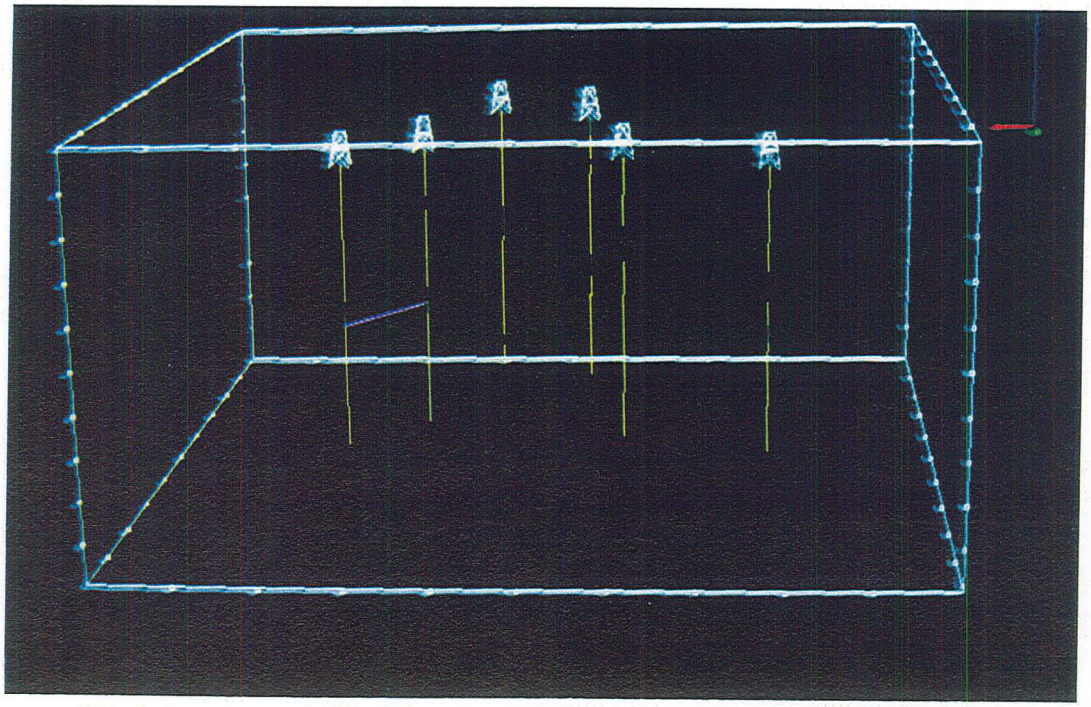
transformations		*chenal*	*lobe*
translation	O	{0,0,0}	{0,0,0}
	u,v,w	selon x,y,z	selon x,y,z
	$\triangle_u$ (m)	1750	1750
	$\triangle_v$ (m)	1650	1650
	$\triangle_w$ (m)	92	92
déformation	*dir*	selon z	selon z
	λ	0.5	0.25
rotation	σ_θ	5°	5°
affinité	σ_e (m)	0.25	0.75
transformation courante		*translation*	*translation*

Table 5.3 Exemple 1: paramètres de discrétisation pour k décrivant les faciès *chenal, lobe, matrice*

k	*chenal*	*lobe*	*matrice*
$Div(k)$	2	2	2

$26 * 2 = 52$. De ce fait, le nombre de variables indicatrices $\{I_0(\mathbf{u}; k)\}$ à honorer est de $52 * 3 = 174$.

(a)



(b)

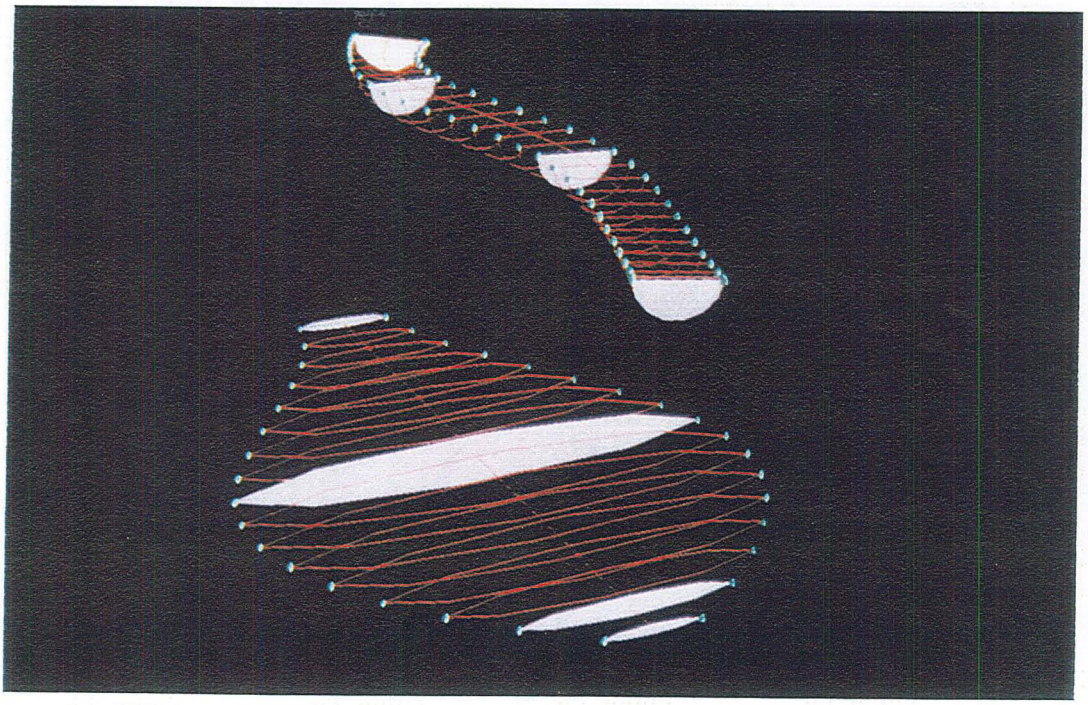
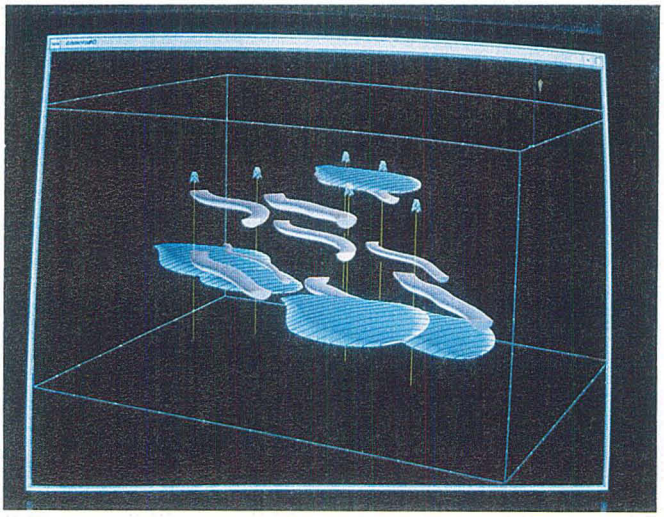
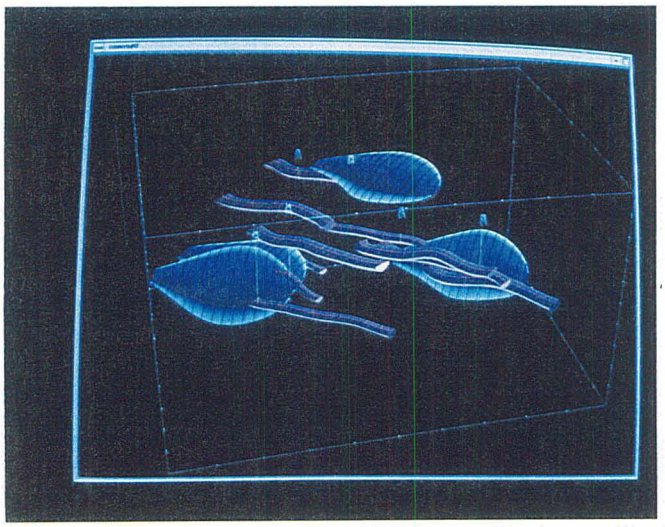


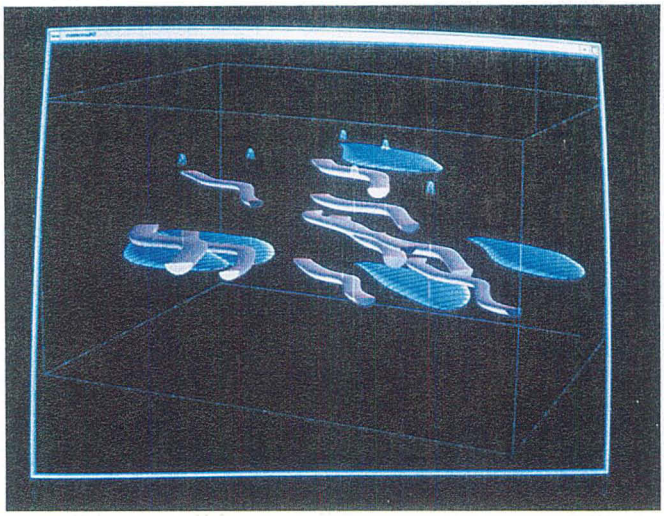
Figure 5.1 Exemple 1: (a) 6 puits représentant les données de départ; une probabilité de corrélation égale à 0.8 est spécifiée entre deux localisations. (b) *templates* associés au faciès *lobe* et *chenal*; les extrémités des *vecteurs horizons* sont en bleu pâle. Echelle de 10 en z.



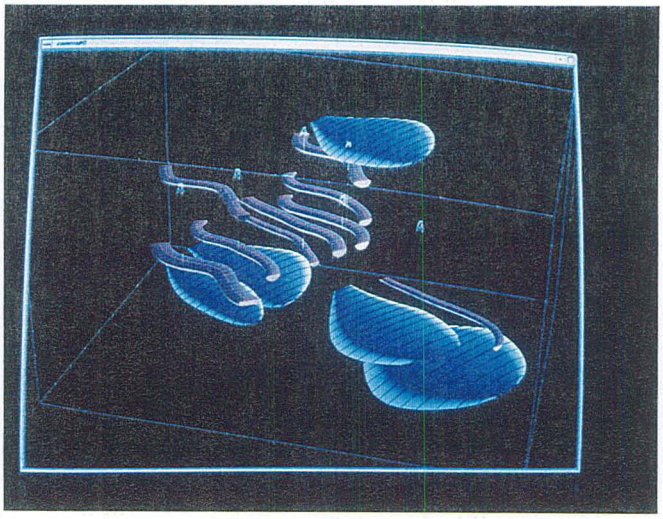
réalisation#1



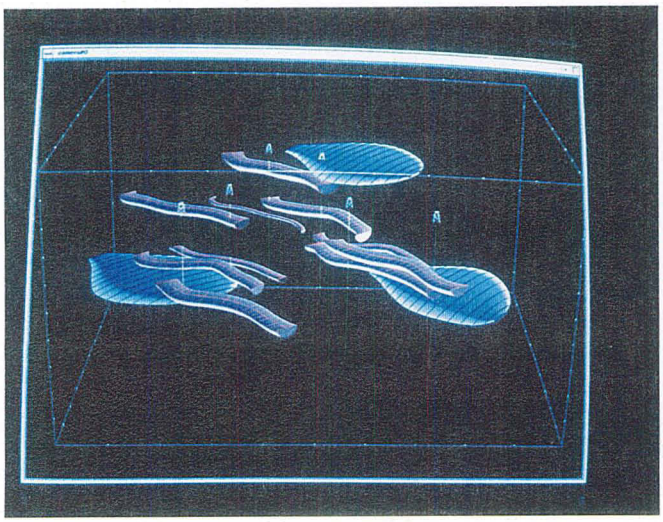
réalisation#2



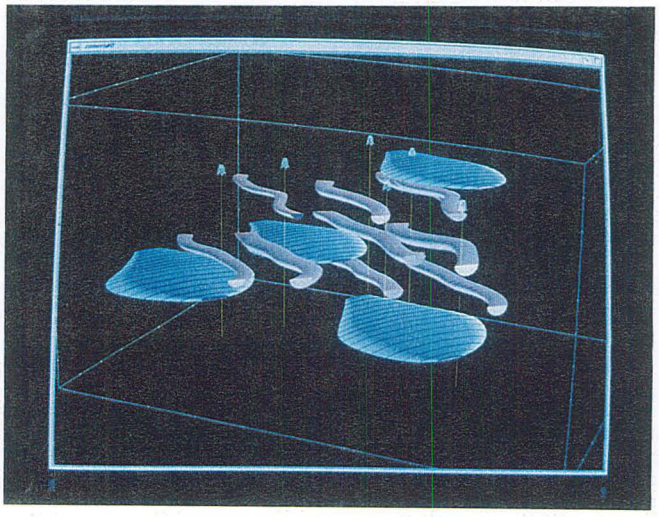
réalisation#3



réalisation#4



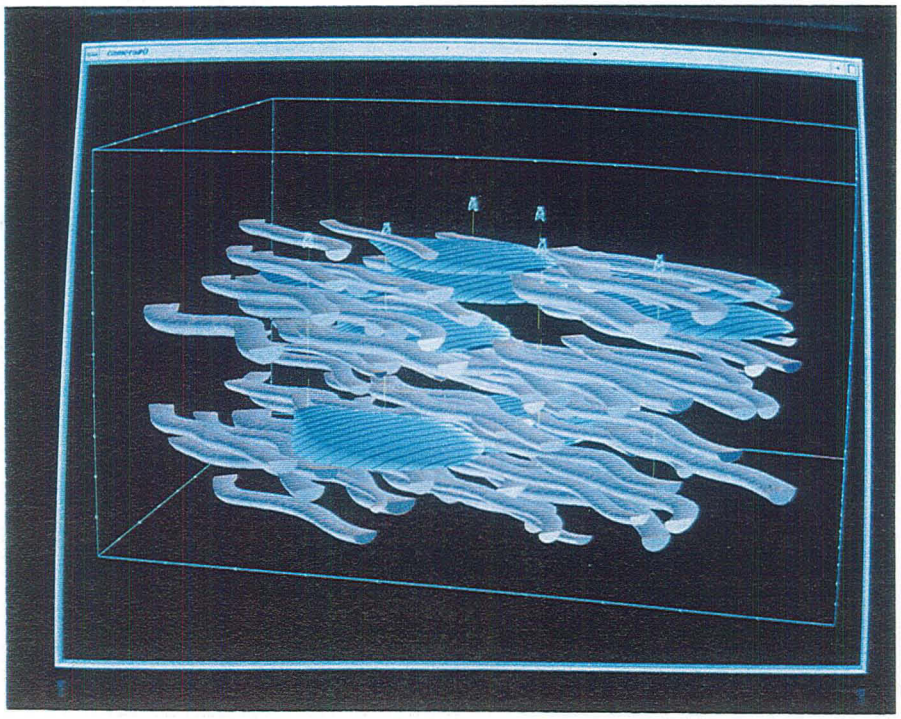
réalisation#5



réalisation#6

Figure 5.2 Exemple 1: présentation de 6 réalisations différentes; la réalisation 1 est contrainte par une probabilité de corrélation entre deux de ses localisations égale à 0.8. Echelle de 10 en z.

(a)



(b)

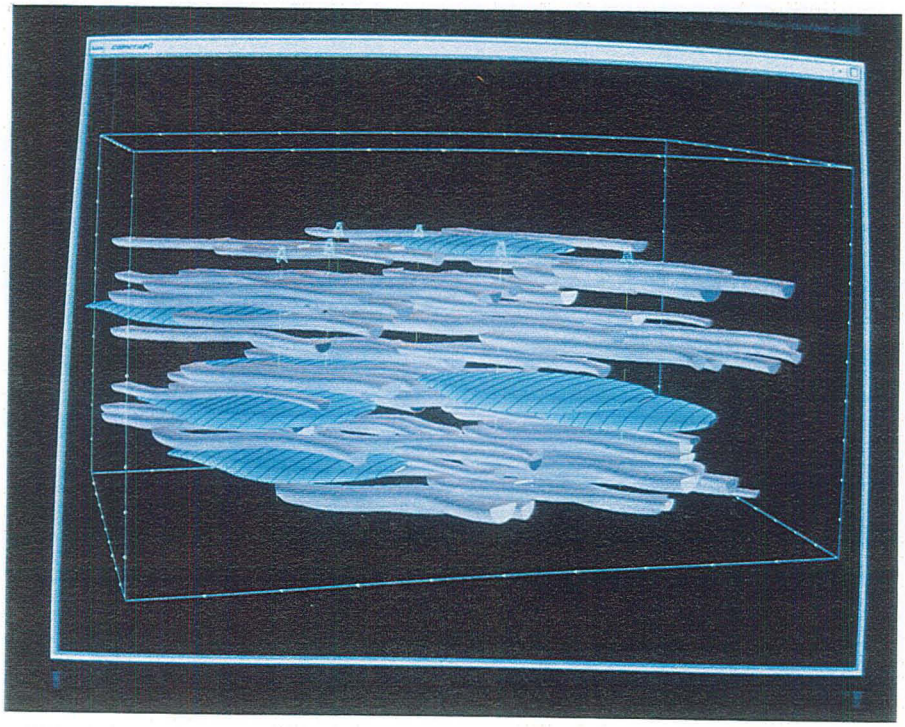


Figure 5.3 Exemple 1: (a) présentation du modèle initial correspondant à la réalisation 6. (b) Présentation du modèle final correspondant à la réalisation 6. Echelle de 10 en z.

Table 5.4 Exemple1: tableau de résultats: $\%I_0$: % d'indicatrices honorées; *Nbcor*: nombre de paires de localisations corrélées; *Nblobe*: nombre de lobes intersectés par au moins deux puits; *Nb chenal*: nombre de chenaux intersectés par au moins deux puits

	$\%I_0$	*Nbcor*	*Nb lobe*	*Nb chenal*
realisation 1	98.3	2	1	0
realisation 2	99.4	2	1	0
realisation 3	99.4	1	1	0
realisation 4	98.9	0	0	0
realisation 5	99.8	1	1	0
realisation 6	98.3	0	0	0

5.3.2 Résultats

Des simulations ont été effectuées dans les deux cas de figure suivants:

1. Avec des données de corrélation. Celles-ci ont été spécifiées entre les deux localisations mises en évidence de la Figure 5.1 (a).

2. Sans données de corrélation.

Par ailleurs, la génération de chacun des faciès *lobe* et *chenal* s'est effectuée suivant la procédure stochastique détaillée à la section 3.6.1. La Figure 5.2 présente le résultat de six simulations différentes après execution de 500 itérations (seuls les objets intersectant les puits sont montrés). La réalisation 1 a été contrainte par une probabilité de corrélation égale à 0.8 entre les deux localisations mises en évidence dans la Figure 5.1 (a). Des exemples de modèle initial et final complets sont présentés dans les Figures 5.3 (a) et 5.3 (b).

Le pourcentage de variables indicatrices $I(\mathbf{u};k)$ égales à leurs valeurs de référence $I_0(\mathbf{u};k)$ est donné dans le tableau 5.4. On y donne aussi le nombre de paires de localisations $\{\mathbf{u_i},\mathbf{u_j}\}$ pour lesquelles $I(\mathbf{u_i},\mathbf{u_j};k)=1$ pour k décrivant les catégories associées au faciès *chenal* et *lobe*.

Géométrie des corps

134 chenaux et 7 lobes ont été générés dans le modèle initial présenté dans la Figure 5.3 (a)). Si l'on considère les *enveloppes* des objets *rshapes* correspondants, on observe que:

1. Aucun des *rshapes* du réservoir ne possède une *enveloppe* identique à celle des autres *rshapes*. Cependant elles "ressemblent" toutes à celle du *template* dont ils sont issus. Ceci est dû à la manière dont est initialisé un *rshape* depuis son *template* (voir section 3.5.2). Les quatre opérations aléatoires *translation, rotation, affinité, déformation*

sont appliquées successivement au *rshape* dont la géométrie a été calquée de celle de son *template*, engendrant des résultats à chaque fois différents.

2. En ce qui concerne la géométrie dans l'espace de ces *rshapes*, on remarque que le modèle initial et le le modèle final présentés dans la Figure 5.3 (a) et la Figure 5.3 (b) représentent des corps uniformément répartis dans l'espace du réservoir. Cela est dû:

   (a) Aux paramètres de la transformation aléatoire *translation* qui décrivent **tout** l'espace du réservoir, il est donc normal que le modèle initial soit composé de corps uniformément répartis dans l'espace.

   (b) Au phénomène *perturbation* de notre algorithme de simulation qui exécute la *transformation courante* associée aux objets, ici la *translation*. Or, le critère d'acceptation d'une *perturbation* est le critère du MAP[3] défini par:

$$P\{accept\} = \begin{cases} 1 & \text{si } O_{new} \leq O_{old} \\ 0 & \text{sinon} \end{cases}$$

   où O_{new} et O_{old} sont les valeurs de la fonction coût avant et après *perturbation*. Ceci signifie que même lorsque la valeur de la fonction coût ne change pas, la *translation* est effectuée. Or c'est ce qui se passe dans la plupart des *perturbations* , lorsqu'on déplace un *rshape* n'intersectant aucun puits, à une localisation où, là encore, il n'intersecte aucun puits. C'est cela qui contribue à créer une répartition des corps homogène.

Convergence du modèle

On peut analyser les résultats du tableau 5.4 de la manière suivante:

1. La proportion de variables indicatrices $I\{\mathbf{u}; k\}$ égales à leur valeur de référence $I_0\{\mathbf{u}; k\}$ pour k décrivant les index associés aux faciès *lobe, chenal, matrice* et $\mathbf{u}$ décrivant l'ensemble des 52 localisations est très élevée. Cela signifie que, au niveau des puits, on retrouve des objets *rshape* appartenant au faciès observé. Or la *transformation courante* (*translation*) est restée la même tout le long du processus et on aurait pu penser que le degré de liberté n'était pas suffisamment grand pour parvenir à un degré d'ajustement élevé. Ce phénomène peut s'expliquer par le fait que dans le modèle de départ, l'ensemble des *enveloppes* possibles pour la population des *lobes* et des *chenaux* est représenté dans le réservoir (ceci grâce à l'étape d'initialisation). Le système parvient donc à trouver le "meilleur objet" pour la "meilleure" place.

2. Lorsque les données de corrélation entre puits ne sont pas spécifiées (réalisations 2 à 6), on remarque que deux réalisations sur cinq génèrent un objet *lobe* qui corrèle entre deux puits. De plus, l'expérience montre que cette corrélation intervient **toujours** pour les mêmes paires de localisations. Cela signifie que des corrélations entre

[3] voir section 4.3.2

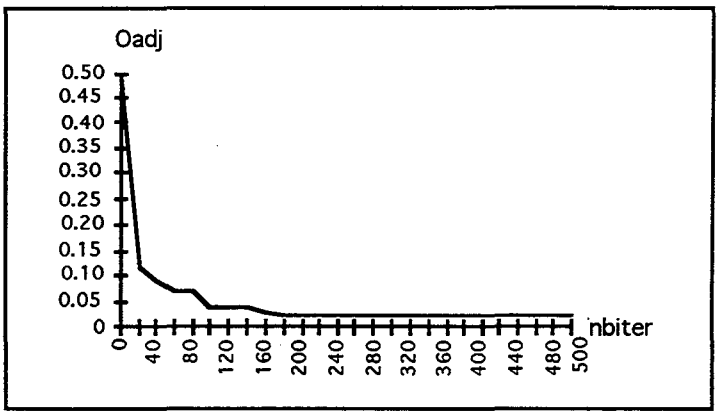


Figure 5.4 Exemple 1: comportement du terme d'ajustement de la fonction coût: graphe de $O_{adj} = f(\textit{nombre d'iterations})$, pour un nombre maximal d'itérations de 500.

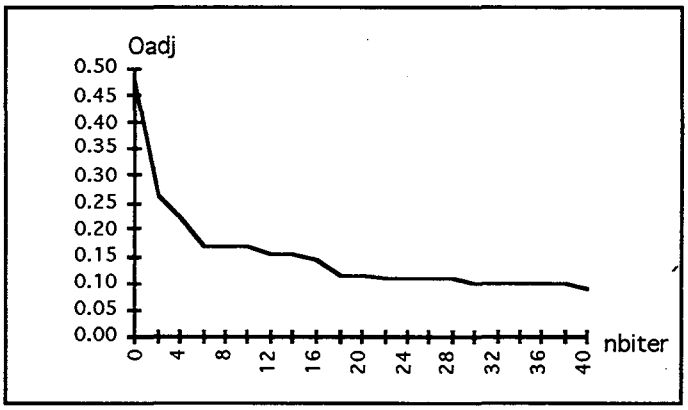


Figure 5.5 Exemple 1: comportement du terme d'ajustement de la fonction coût: graphe de $O_{adj} = f(\textit{nombre d'iterations})$, pour un nombre maximal d'itérations de 40.

puits **peuvent** se produire mais que lorsque cela intervient, c'est toujours entre les mêmes paires de localisations. D'autre part, lorsqu'on introduit une probabilité de corrélation égale à 0.8 pour ces mêmes localisations et pour le faciès *lobe*, celle-ci est honorée. Cependant, si une probablité de corrélation égale à 1 est introduite pour d'autres paires de localisations, on observe que celle-ci n'est pas honorée. Ceci montre qu'il y impossibilité **physique** pour que des corrélations interviennent pour d'autres paires de localisations (en tous cas lorsque la *transformation courante* est initialisée à la *translation*).

Nous avons enfin étudié la rapidité de décroissance de la fonction coût O en fonction du nombre *nbiter* d'itérations effectuées (voir Figure 5.4, 5.5 et 5.6):

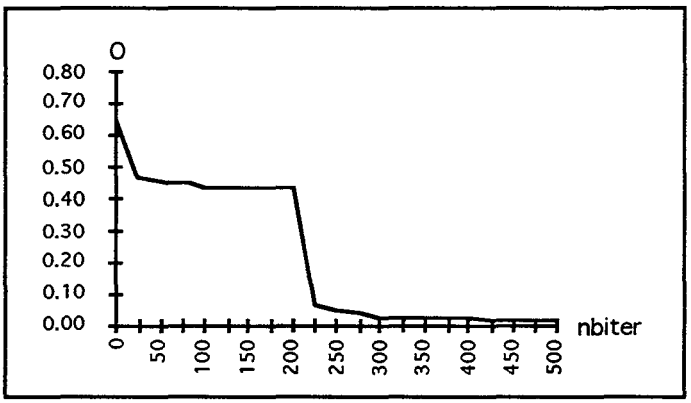


Figure 5.6 Exemple 1 : graphe de $O = \frac{1}{2}(O_{adj} + O_{cor}) = f(\textit{nombre d'iterations})$, pour un nombre maximal d'itérations de 500.

1. Le terme d'ajustement O_{adj} décroît sous la forme d'une exponentielle décroissante, il tend donc rapidement vers zéro. Cela s'explique en partie par le fait, que dès la première itération, les corps qui intersectaient les puits à des localisations où ils n'étaient pas observés sont déplacés, diminuant ainsi rapidement le terme O_{adj} de la fonction coût. De plus, puisqu'une itération parcourt ici tous les *rshapes* du réservoir, plus il y a de *rshapes* dans le réservoir, et plus une itération a de chances de produire une réalisation dont le coût est moindre que pour la précédente.

2. Lorsqu'on observe le comportement de la fonction coût $O = O_{adj} + O_{cor}$, celle-ci décroît brusquement au moment où les contraintes de corrélations sont honorées. En effet, à poids relatif égal des deux termes O_{adj} et O_{cor}, des probabilités élevées de corrélations ou de non corrélation engendreront des variations plus grandes de la fonction coût, et ceci, parce qu'elles sont en général moins nombreuses (voir explication page 107).

5.3.3 Caractéristiques informatiques

temps cpu

Le temps de calcul d'une réalisation issue de 500 itérations est de 12 minutes et 38 secondes sur une station de travail de type hp9000 700/50. Ce temps de calcul est directement proportionnel au nombre d'itérations demandé.

place mémoire

Sur le même type de machine, 10 Mo de mémoire sont nécessaires pour charger le modèle et faire tourner la simulation. La taille de mémoire utilisée ne varie pas au cours de la simulation. Nous rappelons ici que 141 corps étaient présents dans cet exemple.

5.3.4 Conclusion

Ce premier exemple permet de valider la méthode de simulation mise en place. Il montre qu'il est possible de prendre en compte des données de puits et des données de corrélation. Ces données sont d'autre part honorées à un fort pourcentage dans un temps relativement court par rapport à d'autres méthodes de simulation. De plus,

1. Il convient à nouveau de signaler que les objets *rshapes* peuvent, comme les objets *gshapes*, être modifiés et modélisés interactivement. La panoplie d'outils présentés au chapitre 2 peut donc servir à modifier **localement** une réalisation afin d'obtenir une réalisation plus spécifique de l'utilisateur. Il est ainsi possible de parvenir à honorer les observations de faciès aux puits à 100%.

2. Il aurait été intéressant d'étudier plus en détail comment se comporte le modèle lorsque l'on dispose de beaucoup de données de corrélation. Cependant, il est difficile de se procurer des modèles réels où plusieurs d'entre elles puissent être spécifiées. Avis aux amateurs...

5.4 Deuxième exemple

Il nous avait été demandé dans un premier temps de modéliser un réservoir de dimensions "classiques", c'est-à-dire environ $2000m \times 2000m \times 10m$. Cependant, les données de tailles des corps contenus dans ce réservoir laissaient apparaître que ceux-ci étaient énormes par rapport aux dimensions du réservoir. Cela signifiait qu'une simulation à cette échelle n'engendrerait que 4 ou 5 corps dans le modèle final. Dans ce cas, une approche déterministe, plutôt que stochastique semblait plus appropriée. Il nous a donc été demandé de modéliser un environnement géologique à l'échelle régionale (de l'ordre de $10000m \times 10000m \times 20m$) dépassant très largement les limites du réservoir, tout en prenant en compte les données émanant de ce réservoir. L'idée était de "zoomer" sur le réservoir, une fois le modèle régional obtenu. La simulation a été faite sur deux séquences de dépôt distinctes, chacune d'elles étant caractérisée par des faciès différents. Les dimensions de la première séquence sont de $18400m \times 27200m \times 15.3m$ et celle de la deuxième de $18400m \times 27200m \times 10.20m$. Seuls trois puits ont été forés et l'on ne ne dispose pas de données de corrélation entre puits.

5.4.1 Description des faciès

A l'échelle régionale, deux environnements différents ont été étudiés. Nous les appelerons "séquence 1" et "séquence 2".

Description de la séquence 1

4 familles d'hétérogénéités nous ont été décrites dans cette séquence. Il s'agit des faciès suivants:

1. *Chenaux et Lobes Estuairiens*, abrégé *fac A*. Un objet appartenant au faciès *fac A* comporte une partie que l'on peut assimiler à un *chenal* et une partie que l'on peut assimiler à un *lobe*. Le *template* associé à ce faciès est présenté dans la Figure 5.7 (a) (couleur bleu clair). 5 nœuds de contrôle et 5 sections de contrôle consituent ses paramètres.

2. *Bancs Bioclastiques*, abrégé *fac B*. Le *template* associé à ce faciès est présenté dans la Figure 5.7 (a) (bleu marine). Il est également paramétrisé par 5 sections de contrôle et 5 nœuds de contrôle.

3. *Estran Vaseux*, abrégé *fac C*. Le *template* associé à ce faciès est présenté dans la Figure 5.7 (a) (bleu pâle). Il est paramétrisé par 3 nœuds de contrôle et 3 sections de contrôle.

4. *matrice*, légèrement majoritaire par rapport à l'ensemble des faciès précédents.

Les *templates* ci-dessus ont été modélisés en fonction des caractéristiques de tailles données dans le tableau 5.5 et de croquis dessinés par les géologues. De plus, tous ces faciès sont observés aux puits.

Table 5.5 Exemple 2: valeurs moyennes empiriques associées aux faciès de la séquence 1

	fac A	*fac B*	*fac C*	*matrice*
largeur/épaisseur	300	800	700	
épaisseur (m)	8	2.5	3	
orientation	EW	NS	NS	
longueur (m)	25000	15000	3000	
proportion	0.29	0.016	0.12	0.574

Table 5.6 Exemple 2: valeurs moyennes empiriques associées aux faciès de la séquence 2.

	fac D	*fac E*	*fac F*	*fac G*	*matrice*
largeur/épaisseur	1000	500	1250	50	
épaisseur (m)	3	10	4	2	
orientation	NS	NS	EW	EW	
longueur (m)	12000	25000	5000	3000	
proportion	0.28	0.30	0.15	0.05	0.22

Description de la séquence 2

5 familles d'hétérogénéités sont présentes dans cette séquence. Elles concernent les 5 faciès suivants:

1. *cordons oolitiques*, abrégé *fac D*. Le *template* correspondant est en bleu marine sur la Figure 5.7 (b) et comporte 5 nœuds de contrôle et 5 sections de contrôle.
2. *plateau supratidal*, abrégé *fac E*. Le *template* correspondant est en vert sur la Figure 5.7 (b). Ses sections de contrôle, au nombre de 4, ont été définies comme étant triangulaires, la pointe du triangle étant orientée vers l'ouest. Ses nœuds de contrôle sont également au nombre de 4.
3. *Delta de Marée*, abrégé *fac F*. Le *template* correspondant est en vert pâle sur la Figure 5.7 (b). Il est paramétrisé par 3 sections de contrôle et 2 nœuds de contrôle.
4. *chenaux*, abrégé *fac G*. Le *template* correspondant est en bleu marine sur la Figure 5.7 (b). Il est paramétrisé par 4 sections de contrôle et 4 nœuds de contrôle. Ses sections de contrôle ont la forme type d'une section transversale de *chenal* (voir Figure 0.1).
5. *matrice*, minoritaire dans cette séquence également.

Il est important de préciser que le faciès *fac G* **n'est pas** observé aux puits. La connaissance de ce type d'environnement géologique laissant cependant entendre qu'il est présent, les géologues l'ont intégré dans la description des faciès . Ici encore, tous les *templates* de la Figure 5.7 (b) ont été modélisés en fonction de croquis de géologues et des valeurs moyennes de tailles et d'orientation données dans le tableau 5.6.

5.4.2 Paramètres des transformations aléatoires

Les notations ont été reprises de celles exposées à la section 3.5.3.

Les tableaux 5.7 et 5.8 donnent les valeurs de chacun des paramètres des transformations aléatoires associées aux *rshapes* consituant les faciès des séquences 1 et 2. On remarque que les paramètres associés à la transformation *translation* ne sont pas identiques pour tous les faciès. Cela signifie que l'on restreint l'espace dans lequel les objets *rshapes* peuvent se déplacer en fonction du faciès auquel ils appartiennent. Contrairement à l'exemple précédent, cette espace n'est pas l'espace complet de l'étude.

5.4.3 Paramètres de discrétisation

Les paramètres de discrétisation $\{Div(k)\}$ sont donnés dans le tableau 5.9 et 5.10 pour k décrivant les index associés aux faciès *fac A, fac B, fac C, matrice* pour la séquence 1 et *fac D, fac E, fac F, fac G, matrice* pour la séquence 2.

Table 5.7 Exemple 2: paramètres des transformations aléatoires associés à la séquence 1.

transformations		*fac A*	*fac B*	*fac C*
translation	u,v,w	selon x,y,z	selon x,y,z	selon x,y,z
	O	(4500,-12500,-15)	(-5420,-12500,-15)	(2500,-12500,-15)
	$\triangle_u$ (m)	13900	7850	15900
	$\triangle_v$ (m)	27200	27200	-12500
	$\triangle_w$ (m)	15	15	-15
déformation	*dir*	selon z	selon z	selon z
	λ	1	0.5	0.25
rotation	σ_θ	11°	11°	11°
affinité	σ_e (m)	0.7	0.2	0.25
transformation courante		*translation*	*translation*	*translation*

Table 5.8 Exemple 2: paramètres des transformations aléatoires associés à la séquence 2.

transformations		*fac D*	*fac E*	*fac F*	*fac G*
translation	u,v,w	selon x,y,z	selon x,y,z	selon x,y,z	selon x,y,z
	O	(-5420,-12500,-26)	(-5420,-6500,-23)	(-5420,-12500,-26)	(1400,-12500,-26)
	$\triangle_u$ (m)	6500	18400	6500	1300
	$\triangle_v$ (m)	27500	21200	27200	27200
	$\triangle_w$ (m)	10	7	10	10
déformation	*dir*	selon z	selon z	selon z	selon z
	λ	1	1	1	0.50
rotation	σ_θ	11°	10°	20°	11°
affinité	σ_e (m)	0.25	0.75	0.3	0.25
transformation courante		*translation*	*translation*	*translation*	*translation*

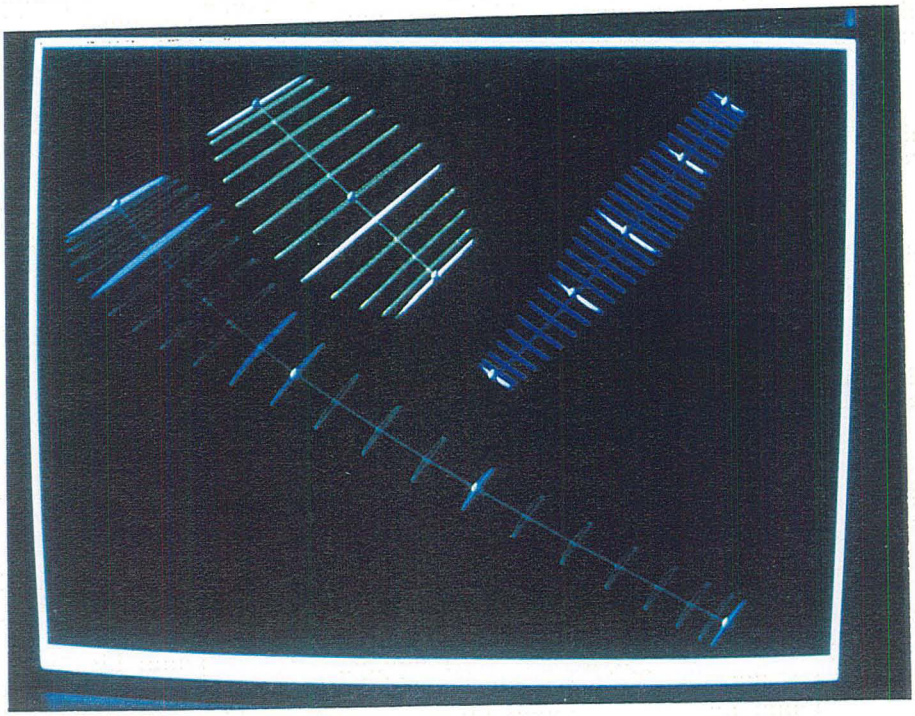
Table 5.9 Exemple 2: paramètres de discrétisation pour k décrivant les faciès *fac A, fac B, fac C, matrice.*

k	*fac A*	*fac B*	*fac C*	*matrice*
$Div(k)$	3	3	3	3

Table 5.10 Exemple 2: paramètres de discrétisation pour k décrivant les faciès *fac D, fac E, fac F, fac G, matrice*

k	*fac D*	*fac E*	*fac F*	*fac G*	*matrice*
$Div(k)$	3	3	3	3	3

(a)



(b)

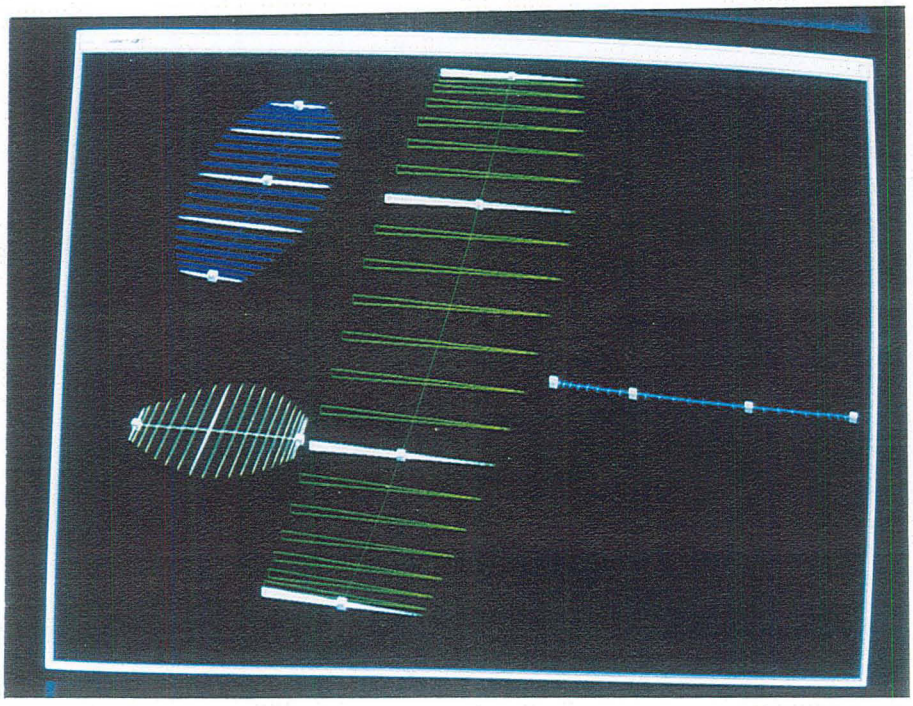
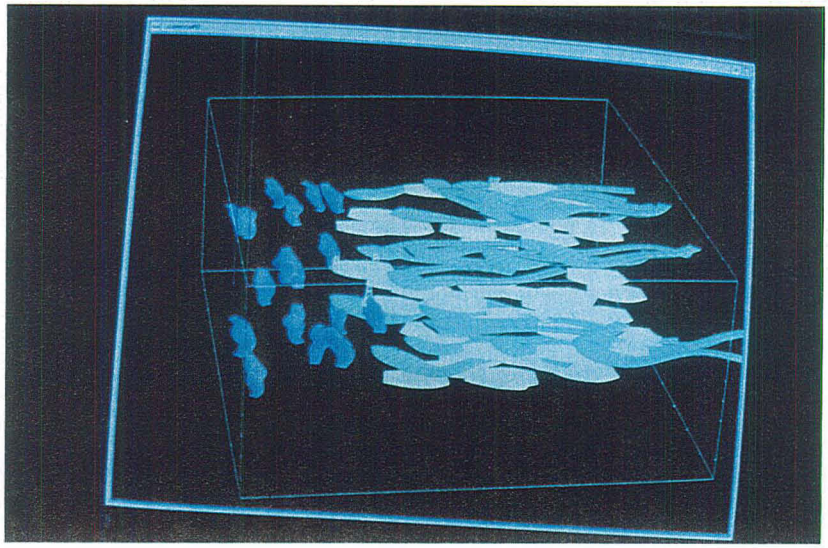
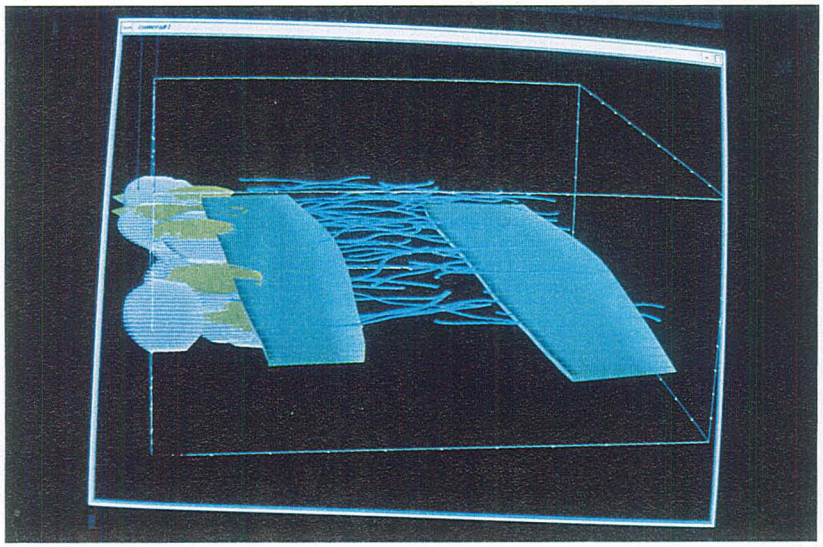


Figure 5.7 Exemple 2: (a) *templates* associés à la séquence 1: bleu clair: *fac A*, bleu marine: *fac B*, bleu pâle: *fac C*. (b) *templates* associés à la séquence 2: bleu marine:*fac D*, vert: *fac E*, vert pâle:*fac F*, bleu marine:*fac G*. Echelle de 30 en z.

(a)



(b)



(c)

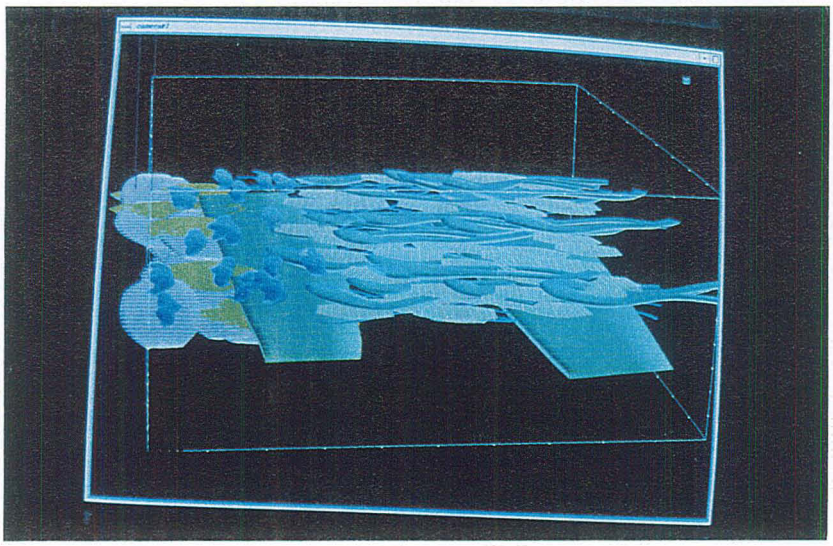


Figure 5.8 Exemple 2: (a) modèle final de la séquence 1. (b) modèle final de la séquence 2. (c) superposition des séquence 1 et 2. Echelle de 30 en z.

5.4.4 Résultats

Deux modèles finals représentant la séquence 1 et la séquence 2 sont présentés dans la Figure 5.8 (a) et la Figure 5.8 (b). Chacun des faciès présent dans la séquence 1 et la séquence 2 a, ici encore, été généré par la procédure stochastique de création de faciès présentée à la section 3.6.1. Le nombre de *rshapes* générés est de 15 pour la première séquence (3 *fac A*, *4 fac B* et 5 *fac C*) et de 99 pour la deuxième séquence (18 *fac D*, 2 *fac E*, 16 *fac F* et *63* chenaux). La Figure 5.8 (c) présente une superposition de ces deux séquences. Parmi toutes les réalisations effectuées, ces modèles ont été sélectionnés parce qu'ils satisfaisaient le plus les géologues. C'est donc un critère subjectif plutôt que qualitatif qui est intervenu ici.
L'étude au niveau de la convergence de la fonction coût est similaire à la précédente, nous ne la referons donc pas à ce niveau. D'autre part, et par rapport à l'exemple précédent, les modèles de la Figure 5.8 engendrent quelques observations supplémentaires:

1. On remarque nettement une régionalisation des corps en fonction du faciès auquel ils appartiennent. Cela est dû à une différentiation des paramètres $O, \triangle_u, \triangle_v, \triangle_w$ attribués à la transformation aléatoire *translation* en fonction de ces faciès.

2. Le faciès *fac G*, non présent aux puits, est cependant représenté (aucun des *rshapes* qu'il contient n'intersecte de puits).

place mémoire

La place mémoire nécessaire pour exécuter une simulation de 500 itérations sur la séquence 1 est de 1.5 Mo. Celle nécessaire sur la séquence 2 est de 3 Mo. La différence par rapport à l'exemple précédent provient du nombre moins élevé de *rshapes* contenus dans les modèles.

5.4.5 Conclusion

Cet exemple a permis de mettre en évidence un mode d'utilisation de nos outils complètement différent de celui que nous avions vu précédemment. Il introduit une manière nouvelle de considérer la modélisation de la géométrie des réservoirs. En effet:

1. L'important ici n'est pas tant le respect des données aux puits, mais la production d'un modèle régional qui permette au géologue de vérifier ou d'infirmer ses hypothèses quant à l'environnement de dépôt dans lequel se trouve le réservoir. En effet, le modèle présenté est le résultat de plusieurs essais, essais concernant par exemple, la forme des *templates* ou le degré de sinuosité des chenaux du faciès *fac G*.

2. Les *enveloppes* des *templates* modélisés sont beaucoup plus complexes que celles des *templates* de l'exemple précédent. Cet exemple montre donc que les objets *gshape* et les outils développés pour les générer sont appropriés pour modéliser les hétérogénéités de réservoir de forme complexe.

3. Il est important de comprendre que là encore, du déterminisme peut être introduit en modifiant la position ou l'*enveloppe* de certains *rshapes*.

4. L'étape suivante consisterait à changer d'échelle de travail et à étudier l'espace du réservoir autour des puits, les corps déjà simulés étant maintenant "figés" pour la simulation. D'autres faciès, à l'échelle du réservoir, pourraient être introduits et leur distribution simulée afin de produire des réalisations à l'échelle du réservoir, et non plus à l'échelle régionale. Cela impliquerait de réinterpréter les données de puits afin de prendre en compte la description de ces nouveaux faciès.

5.5 Troisième exemple

Ce troisième exemple diffère des précédents dans le sens où l'on introduit un modèle déterministe comme modèle de départ de la simulation. En effet, on considère ici un réservoir de dimensions $2244m \times 4641m \times 28m$ dans lequel trois faciès sont présents, le faciès *lobe*, le faciès *chenal* et le faciès *matrice*. On dispose d'autre part, de données provenant de 8 puits. Le problème posé est le suivant:

1. On désire partir d'un croquis de géologue positionnant 4 chenaux de manière déterministe, et honorant les données de puits. Ces chenaux passent par plusieurs puits.

2. On désire simuler entièrement le faciès *lobe*.

3. On désire produire des réalisations différentes de ce modèle, et notamment de la sinuosité des chenaux. Une probabilité de corrélation égale à 1 a été donnée entre les deux réalisations mises en évidence dans la Figure 5.9 (b).

5.5.1 Production d'un modèle de départ

La production d'un modèle initial est faite de la manière suivante:

1. Introduction d'un modèle déterministe de 4 *templates* honorant les données de puits et modélisés suivant des croquis de géologues.

2. Générer le faciès *chenal* à partir des 4 *templates* précédents et par la procédure déterministe de création de faciès[4]. On ne passe alors pas par la dernière étape de l'initialisation d'un *rshape* depuis son *template* père. Les transformations aléatoires *translation, affinité, rotation, déformation* ne sont pas appliquées et la géométrie des *rshapes* crées est identique à celle de leur *template* père.

3. Générer le faciès *lobe* à partir du *template* qui lui est associé (voir Figure 5.9 (a)). On utilise une procédure stochastique[5] classique de création de faciès.

Le modèle initial ainsi produit est présenté dans la Figure 5.9 (b).

[4]voir section 3.6.2
[5]voir section 3.6.2

Table 5.11 Exemple 3: paramètres des transformations aléatoires associés aux transformations *translation* et *déformation* pour les faciès *lobe* et *chenal.*

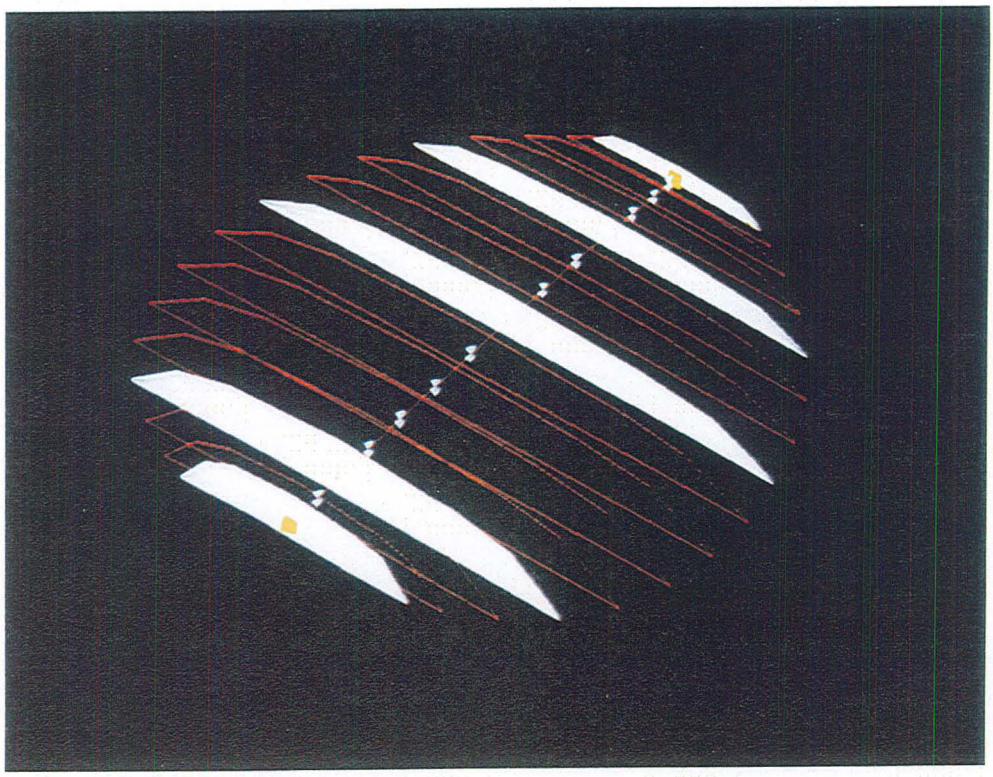
transformations		*chenal*	*lobe*
translation	O		{0,0,0}
	u,v,w		selon x,y,z
	$\triangle_u$ (m)		2244
	$\triangle_v$ (m)		4641
	$\triangle_w$ (m)		28
déformation	*dir*	selon z	
	λ	1	
transformation courante		*perturbation*	*translation*

5.5.2 Paramètres des transformations

Nous considérons que les valeurs moyennes des lobes et des chenaux sont celles extraites du *template* qui leur est associé, et qui a été modélisé par le géologue. De plus, la *transformation courante* associée au faciès *lobe* est la *translation*, la *transformation courante* associée au faciès *chenal* est la *déformation.* Les paramètres associés à ces deux transformations sont donnés dans le tableau 5.11.

Nous voudrions faire remarquer ici que les paramètres de chacun des *rshapes* présent dans le réservoir peuvent être différents les uns des autres. Les paramètres *template dépendants* de chacun des chenaux le sont d'ailleurs, puisque initialisés depuis des *templates* différents.

(a)



(b)

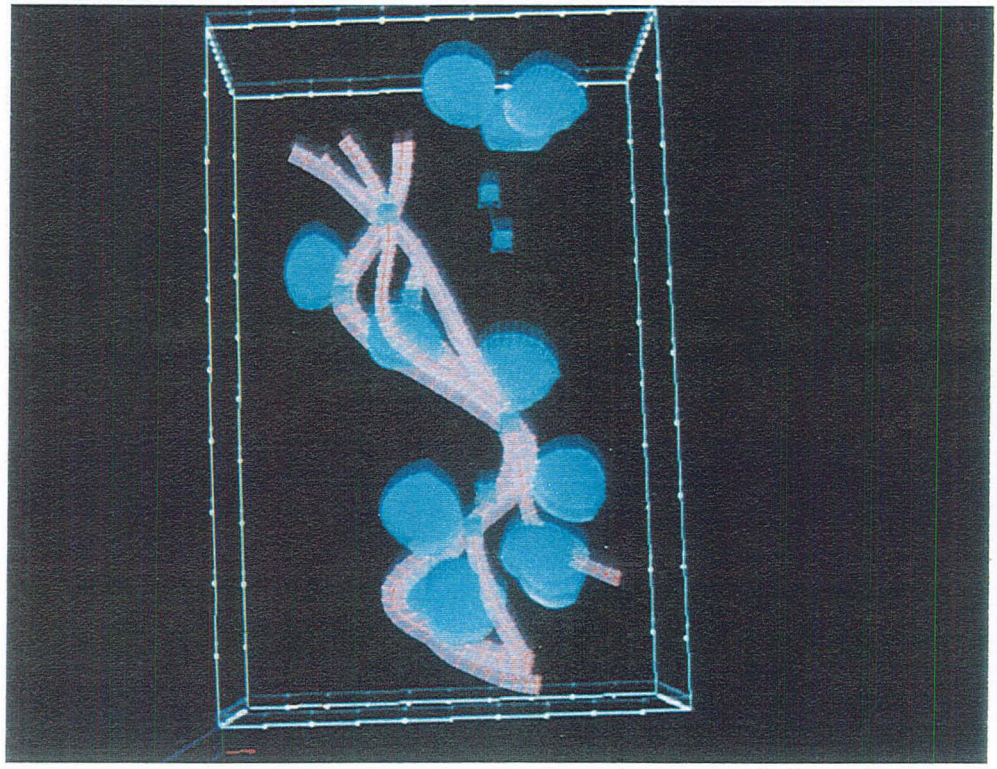


Figure 5.9 Exemple 3: (a) *template* associé au faciès *lobe*. (b) modèle initial de simulation; une probabilité de corrélation de 1 est précisée entre deux localisations; le faciès *chenal* a été produit de manière déterministe. Echelle de 20 en z.

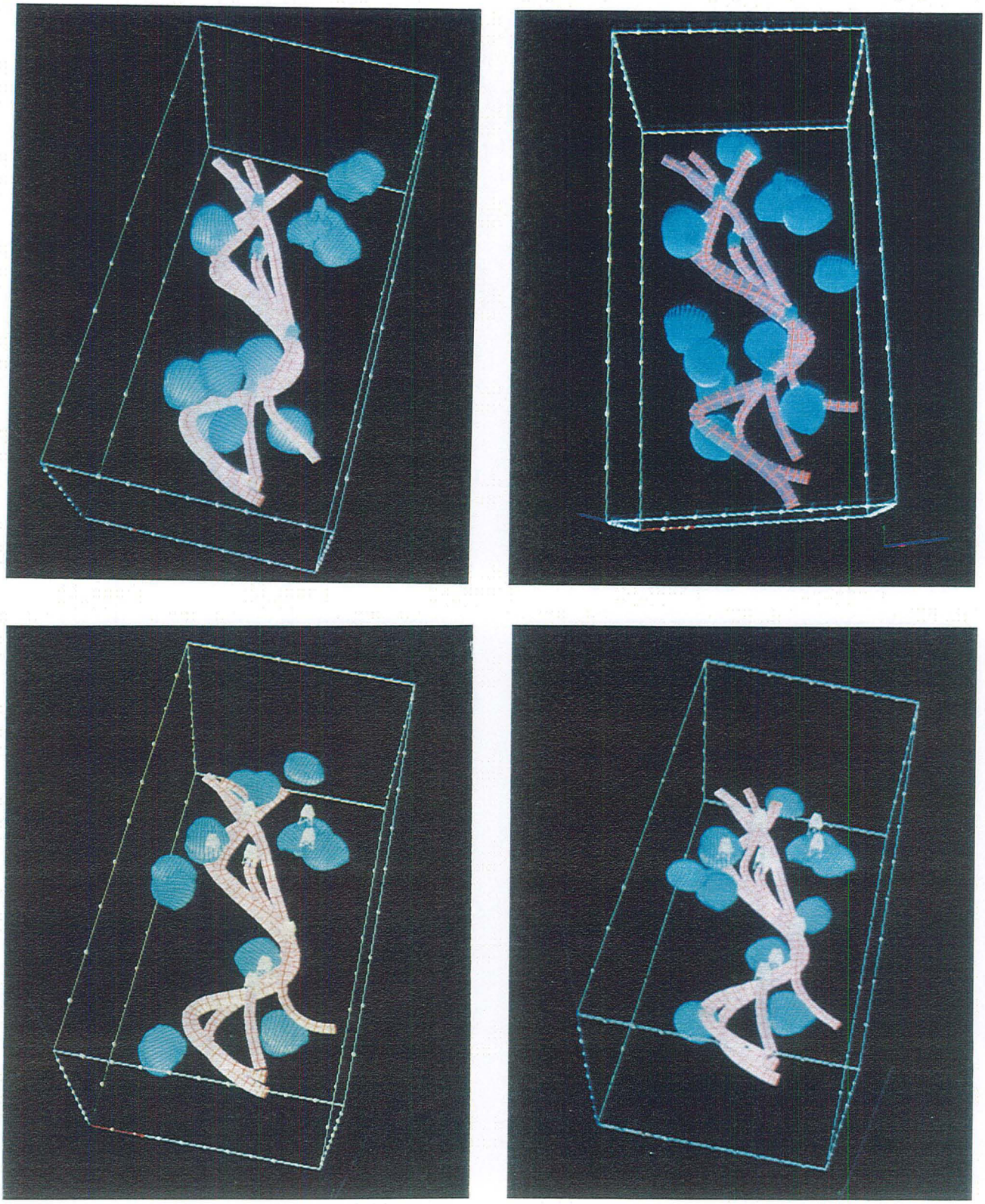


Figure 5.10 Exemple 3: présentation de 4 réalisations différentes. La *transformation courante* des chenaux est initialisée à *déformation*, celle des lobes à *translation*. Echelle de 20 en z.

5.5.3 Résultats

Quatre réalisations différentes sont présentées dans la Figure 5.10. On remarque que la phase *perturbation* appliquée au faciès *chenal* engendre une variation légère de la sinuosité des chenaux alors qu'elle génère un déplacement des lobes. Les différentes réalisations honorent maintenant toutes les données de puits, et non plus seulement celles relatives au faciès *chenal*. On remarque que les données de corrélation entre puits pour le faciès *lobe* sont également honorées.

5.5.4 Conclusion

Cet exemple a montré comment on pouvait introduire du déterminisme dans la construction d'un modèle de départ. De plus, il met en évidence le fait qu'un modèle initial pouvait n'être déterministe que de manière locale, certains faciès pouvant être simulés de manière stochastique. D'autre part, on montre ici l'application de *transformations courantes* différentes pour les *rshapes* contenus dans le faciès *lobe* et ceux contenus dans le faciès *chenal*.

5.6 Discussion

Dans les trois exemples présentés, on peut mettre en évidence des caractéristiques communes spécifiques de la manière dont notre méthode de simulation stochastique fonctionne. Celles-ci sont détaillées ci-dessous.

1. Concernant la modélisation des *templates*, on remarque que les *sections de contrôle* et *les nœuds de contrôle* sont souvent situés sur les mêmes *vertèbres* lorsque le *backbone* du *template* est sinueux. Il paraît en fait tout à fait logique de contraindre les mêmes *vertèbres* pour paramétriser la géométrie du *backbone* et des *sections*. En effet, leur interpolation par DSI étant supportée par le même graphe, il est logique de penser que les *vertèbres* "clé" pour la définition de la géométrie de l'*enveloppe* d'un *rshape* seront les mêmes pour le *backbone* et les *sections*.

2. On aurait pu craindre que l'utilisation du critère du MAP[6] comme critère d'acceptation d'une *perturbation* fasse apparaître des problèmes de convergence. Cela n'ayant pas été jusqu'à présent le cas, il n'a pas été remplacé par celui de Métropolis. Cependant, celui-ci pourrait facilement être intégré à l'algorithme de simulation dans le cas où des problèmes interviendraient.

3. Nous avons vu que les exemples présentés ici ne prenaient pas en compte de contraintes d'interaction "entre" objets. Les *rshapes* placés entre les puits sont donc très souvent déplacés d'une itération a une autre, lorsqu'ils n'introduisent pas une augmentation de la fonction coût. Une voie importante d'une recherche future serait de pouvoir introduire des contraintes entre corps par l'intermédiaire d'un terme supplémentaire dans l'expression de la fonction coût. Il pourrait être intéressant par

[6] voir section 4.3.2

exemple d'introduire des contraintes de "non intersection" des corps. Cela impliquerait de pouvoir rapidement calculer l'intersection en trois dimensions d'objets aussi complexes que des *gshapes*. Cependant, le fait que ceux-ci soient discrétisables en éléments de volume tétrahédriques[7] peut constituer une voie intéressante.

5.7 Conclusion

Les trois exemples présentés ici ont mis en évidence les possibilités d'application des travaux présentés dans ce manuscrit. Ces applications peuvent concerner trois types d'utilisations différentes:

1. Utilisation "classique". On ne dispose que des données de puits et d'un *template* par famille d'hétérogénéités. On génère alors plusieurs modèles stochastiques de départ que l'on perturbe pour qu'ils respectent au mieux les données de puits. Les différentes réalisations peuvent ensuite être comparées afin de déterminer quelles sont leurs caractéristiques communes et où il convient de rechercher de nouvelles données.

2. Utilisation "visuelle". Il s'agit ici, plus que de respecter les données aux puits, d'arriver à une représentation globale d'un modèle qui permette aux utilisateurs de réfléchir sur la nature de l'envionnement géologique dans lequel se trouve le réservoir. On suppose que plusieurs faciès cohabitent dans des régions différentes de l'espace étudié.

3. Utilisation "déterministe". Tout ou partie d'un modèle de départ est disponible et l'on désire y introduire du stochastique afin d'étudier les variations autour de ce modèle. Les paramètres des différents *rshapes* au sein d'un même faciès peuvent alors varier en fonction de la *perturbation* que l'on désire individuellement leur appliquer.

D'autre part, il est important de signaler que ces trois utilisations ne sont pas exclusives les unes des autres, mais, au contraire, complémentaires. On peut supposer par exemple que l'on travaille d'abord à l'échelle régionale afin de déterminer quelles parties du réservoir peuvent être simulées indépendamment les unes des autres. On procède ensuite à une simulation stochastique "classique" dans chacune de ces régions. Enfin, il est possible de raffiner "localement" le modèle par une utilisation déterministe et interactive. D'autre part, la mise en application de nos méthodes montre que celle-ci est:

1. Rapide: l'exemple de simulation le plus coûteux utilise un temps cpu tout à fait raisonnable pour ce type d'application.

2. Econome en mémoire. Là encore, l'exemple le plus coûteux nécessite une place mémoire inférieure à ce qui est généralement disponible dans la plupart des configurations de base (16Mo).

[7] voir section 2.6.2

5.8 Résumé du chapitre

Ce chapitre présente trois exemples très différents d'utilisation de la méthode de simulation stochastique de distribution d'hétérogénéités proposée au cours du chapitre 4. L'étude de ces exemples montre que la production d'une réalisation se fait rapidement et ne demande pas de capacités de mémoire centrale exeptionnelle. Des remarques et suggestions sont faites ensuite quant à l'orientation à donner aux recherches afin d'améliorer encore la méthode. Il pourrait être intéressant d'introduire des contraintes d'interaction entre corps, et, éventuellement, de remplacer le critère du MAP utilisé comme critère d'acceptation d'une *perturbation*, par celui de Métropolis.

Conclusion

Les travaux présentés au cours de cette thèse traitent deux grands types de problèmes:

1. La modélisation interactive de la géométrie de corps naturels de forme complexe.
2. L'application des principes généraux de cette modélisation à la représentation de la distribution d'hétérogénéités dans les réservoirs pétroliers.

Modélisation de corps naturels

La mise en place de l'objet *gshape* a permis de développer de nouveaux concepts de modélisation de corps naturels. On considère en effet que la géométrie d'un objet est le résultat de deux interpolations indépendantes, celle d'une "ligne conductrice" (*le backbone*) d'une part, et celle de sections planaires attachées à cette ligne (les *sections*) d'autre part. Cela permet une modélisation globale de la géométrie de l'objet et introduit une flexibilité qui permet d'intégrer l'intuition de l'utilisateur à cette modélisation. D'autre part, la méthode d'interpolation, basée sur DSI , considère que les différents paramètres d'interpolation appartiennent au *gshape* lui même, ce qui, en Géologie, est primordial. Enfin, une fois modélisé, on peut passer à une représentation surfacique (sous forme de triangles) ou volumique (sous forme de tétraèdres) de l'*enveloppe* d'un *gshape*.

Modélisation d'hétérogénéités

L'utilisation de l'objet *gshape* pour représenter la géométrie d'un faciès donné au sein du réservoir a ouvert des voies tout à fait nouvelles dans le domaine de la modélisation de réservoir. En effet, et contrairement aux approches objets (booléennes) traditionnelles, on peut maintenant considérer qu'une famille d'hétérogénéités est représentée par un objet de forme complexe et dont la géométrie n'est décrite ni à travers une expression analytique simple, ni à travers un ensemble de points de l'espace. La flexibilité de paramétrisation des *gshapes* est utilisée pour générer des objets *rshapes* appartenant à une famille d'hétérogénéités donnée et dont la géométrie pourra par la suite être modifiée de manière aléatoire et de façon à respecter les principales caractéristiques du faciès auquel ils appartiennent. Cela a permis de proposer une approche objet originale de modélisation stochastique de réservoir, qui, en alliant le formalisme de la géostatistique non paramétrique à celui des méthodes de recuit permet d'intégrer:

1. Le respect des faciès observés aux puits;
2. Les probabilités de corrélation entre puits, pour un faciès donné.
3. La génération de modèles pouvant présenter une grande complexité du point de vue de leur géométrie.

Si le premier point est souvent pris en compte dans les méthodes booléennes traditionnelles, les deux derniers ne le sont généralement pas. Notre approche introduit donc des possibilités nouvelles en ce qui concerne la définition des contraintes à respecter.

Recherches futures

Le travail présenté ici met en place des concepts nouveaux dans le domaine de la modélisation de la géométrie d'un ensemble de corps naturels. Il faut maintenant s'attacher à les valoriser complètement en intégrant des éléments tels que:

1. Calculs d'intersections entre objets de type *gshape*.
2. Prise en compte d'interactions entre objet de type *rshapes* appartenant ou non au même faciès.
3. Développement d'autres types de transformations aléatoires pour les objets *rshapes*.

De plus, et comme nous l'avons dit dans l'introduction, une suite logique de nos travaux serait de pouvoir simuler, "à l'intérieur" de l'*enveloppe* de chacun des objets *rshapes* d'une réalisation donnée, la distribution d'une variable continue (perméabilité par exemple). Des travaux ont été entrepris dans ce sens. Ceci ne seront pas détaillés ici mais un des premiers résultats est présenté dans la Figure 5.11. On y voit la représentation d'une grille $100 \times 100 \times 100$ initialisée à l'aide d'une des réalisations du premier exemple présenté au chapitre 5. La décomposition de l'objet *gshape* en éléments de volume tétraédrique a été utilisée de la manière suivante:

1. Attribuer en chaque nœud de la grille une valeur correspondant à l'index associé au faciès *matrice*.
2. Pour chaque *rshape* du modèle:
   (a) séléctionner les nœuds de la sous-grille contenus dans le plus petit parallélépipède rectangle contenant le *rshape*.
   (b) pour chaque tétraèdre du *rshape*
      i. calculer quels sont les nœuds de la sous-grille contenus dans ce tétraèdre.
      ii. attribuer à ces nœuds une valeur correspondant à l'index du faciès auquel appartient ce *rshape*.

Ceci permet de visualiser un modèle de manière plus claire que ce qui se faisait jusqu'alors. De plus, cela ouvre des perspectives nouvelles en ce qui concerne la modélisation stochastiques de variables continues. En effet, nous avons maintenant une représentation "pixel" d'un réservoir dont la géométrie est déjà modélisée. Rien n'empêche alors d'utiliser le large éventail des méthodes pixels pour simuler des variables continues à l'intérieur des zones de la grille où la variable correspondant à l'index du faciès est constante.

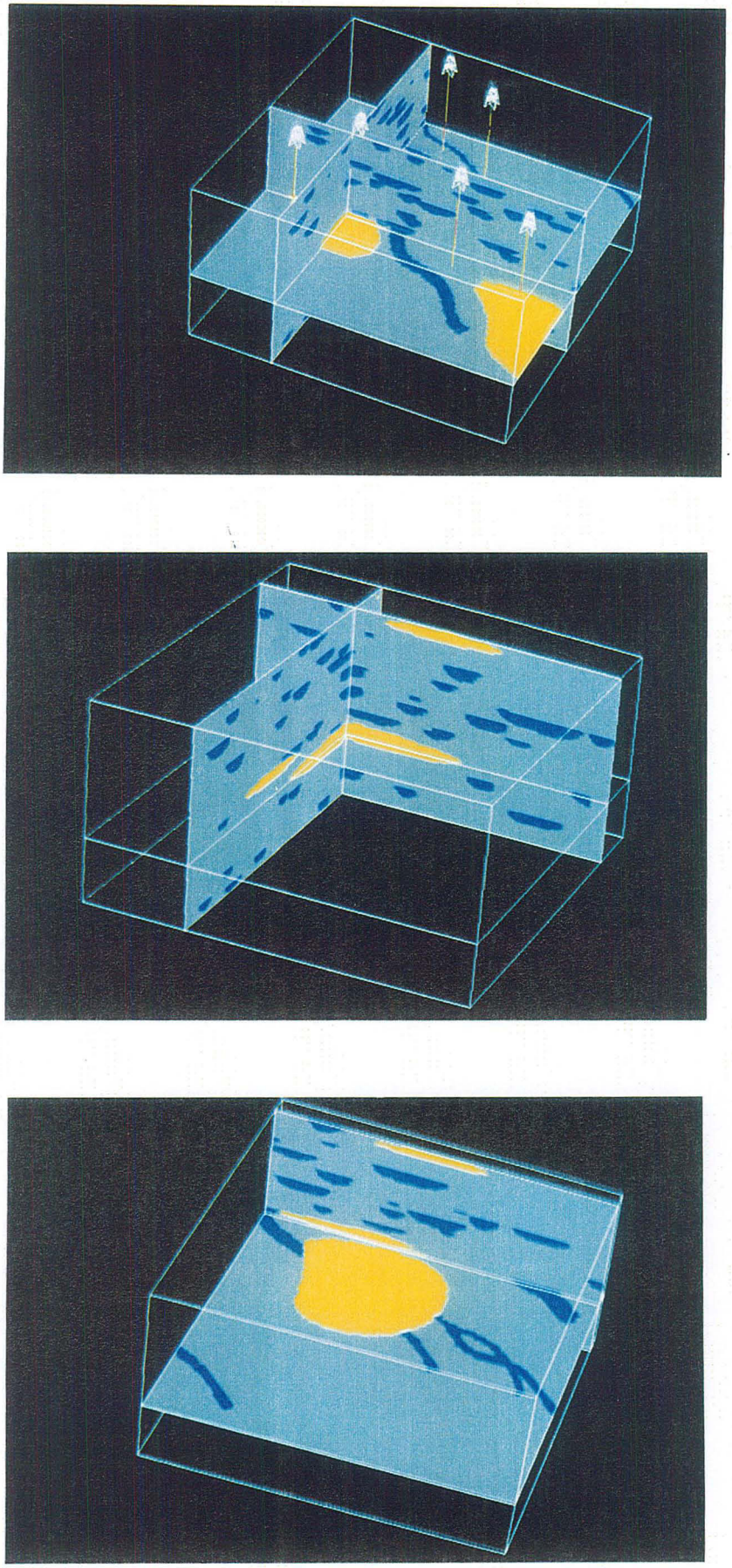


Figure 5.11 Différentes représentations d'une grille $100 \times 100 \times 100$ initialisée à l'aide d'une des réalisations issue de la modélisation de l'exemple 1 présenté au chapitre 5.

Appendice A

Présentation d'une implémentation en pseudo C++

La méthode de simulation stochastique présentée au cours des chapitres précédents, a été entièrement développée dans le cadre de la base de données de GOCAD. Celle-ci est élaborée en language C, et utilise des concepts proches de ceux mis en oeuvre lors de conceptions orientées objet. Nous décrirons dans un premier temps, quelles sont les structures utilisées pour représenter les objets fondamentaux de l'espace à trois dimensions. Nous reviendrons ensuite sur quelques notions spécifiques à GOCAD telles que les concepts d'*atomes* et de *topologie*. Nous décrirons enfin l'architecture du système développé dans le cadre de cette thèse.

A.1 Description de quelques structures de base

Nous allons décrire ici les structures fondamentales de la base de données de GOCAD. Celles-ci ont été implémentées en C et sont à la base de tout développement ultérieur. Nous dirons d'abord comment nous représentons les éléments de base de l'espace à trois dimensions tels que point, ligne, portion de l'espace. Nous définirons ensuite quelles sont les principales structures décrivant les notions de *topologie* et de *graphe*.

A.1.1 Outils géométriques de base

- ```
  struct POINT3 {
    float x ;
    float y ;
    float z ;
  }POINT3 ;
  ```

  La structure `POINT3` représente la notion de point de l'espace. Celui-ci est décrit par ses trois composantes `x,y,z`. Elle sert également à la représentation d'un vecteur dans l'espace. `x,y,z` sont alors les composantes de ce vecteur.
  ```

- ```
  typedef struct LINE3
          {
            POINT3   origin;
            POINT3   dir   ;
            double   s     ;
          } LINE3 ;
  ```

  La notion de ligne est représentée par la structure LINE3. `origin` et `dir` sont respectivement l'origine et le vecteur directeur de la ligne. s est utilisé pour décrire le point $\vec{P}(s)$ localisé sur la ligne et tel que:

$$\vec{P}(s) = \vec{origin} + s.\vec{dir}$$

- ```
  typedef struct BOX3
          {
            POINT3  min;
            POINT3  max   ;
          } BOX3 ;
  ```

  La structure `BOX3` est utilisée afin de représenter une portion de l'espace délimitée par un parallélépipède rectangle dont les faces sont parallèles aux axes $\{\vec{ox}, \vec{oy}, \vec{oz}\}$ Les champs `min` et `max` sont tels que pour un point $\vec{p}$ de l'espace, et une instance `box3` de la classe `BOX3`:

$$\vec{p} \in box3 \Longleftrightarrow \begin{cases} box3.min.x & \leq p.x & \leq box3.max.x \\ box3.min.y & \leq p.y & \leq box3.max.y \\ box3.min.z & \leq p.z & \leq box3.max.z \end{cases}$$

A.1.2 Classes associées à la notion de graphe

Les structures qui vont être présentées à ce niveau sont actuellement écrites en langage C au sein de GOCAD. Leur implémentation est cependant faite dans un style qui emprunte au C++ l'idée fondamentale de l'abstraction et de l'encapsulation des données ([22],[2]). Nous avons donc décidé de les décrire directement en pseudo $C++$, dans un souci de concision et de généricité de l'implémentation. Pour des raisons évidentes de clareté, seule l'interface publique des différents objets sera présentée et expliquée.

Représentation des nœuds du graphe

```
typedef struct VRTX{
   POINT3 pos ;
   long movable ;
 }VRTX ;
```

Dans la base de donnée GOCAD, le *vertex* étend la notion de point de l'espace à la notion de point appartenant à un graphe de l'espace. Un *vertex* décrit la géométrie d'un nœud appartenant à un graphe, dont la position pourra être interpolée par *DSI* . Les champs `pos` et `movable` de la structure `VRTX` définissent le *vertex* de la façon suivante:

1. **pos** est la position spatiale du *vertex*.
2. **movable** définit le comportement des *vertex* lors de l'interpolation DSI . En particulier, si *movable* = 0, le *vertex* est un *nœud de contrôle* pour DSI .

La définition de la topologie sur un graphe passe par la définition de la notion d'*atome*. Celle-ci a été décrite à la section 1.3.5 et est illustrée par la classe ATOM décrite ci-dessous:

```
class ATOM {
 public:
              ATOM() ;
 virtual      ~ATOM();
 POINT3       getVrtxAtom()  const;
 void         setVrtxAtom(const POINT3&);
 void         setAtom( const POINT3&) ;

 int          nbSat()     const;
 ATOM&        sat(int i)  const;

 void         setControlNode();
 void         unsetControlNode();
 void         runDsiGeoAtom() ;

 //  rest of the class omitted

};
```

où, si atome est une instance de la classe ATOM:

1. ATOM() et ~ATOM() sont les constructeurs et destructeurs associés à la classe ATOM.
2. *getVrtxAtom()* retourne la position de l'instance vertex du type VERTEX associée à atome. Sa position dans l'espace peut être fixée par la méthode *setVrtxAtom()*.
3. *nbSat()* retourne le nombre de satellites d'un atome. Le satellite n^o i est accessible par la méthode *sat(*int i*)*.
4. *runDsiGeoAtom()* lance une itération de l'interpolation DSI sur la position du *vertex* contenu dans atome.
5. *setControlNode()* et *unsetControlNode()* contrôlent le comportement du vertex contenu dans atome lors de l'interpolation *dsi*.

Présentation de la classe GOBJ

```
class GOBJ {

        public:
```

```
                        GOBJ();
virtual                 ~GOBJ();

virtual void            load     (stream&);
virtual void            save     (stream&) const ;
virtual int             operator >> (stream&) const ;
virtual int             operator << (stream&);

// rest of the class omitted
};
```

La classe GOBJ est la classe de base pour tous les objets GOCAD et définit les méthodes virtuelles de base qui seront redéfinies au niveau des objets de plus haut niveau. Celles-ci sont:

1. *load()* et *save()*. Elles sont utilisées pour le chargement et la sauvegarde binaire d'un *gobj*.
2. Les opérateurs << et >>. Ils sont utilisés pour le chargement et la sauvegarde en ASCII d'un GOBJ.

GOBJ() et ~GOBJ() sont les destructeurs et constructeurs de la classe GOBJ;

Représentation d'un graphe

De nombreux objets tri-dimensionels manipulés par GOCAD (lignes polygonales, surfaces triangulées, volumes tétraédriques) sont basés sur des graphes contraints. Les contraintes sont d'ordre topologique et interdisent l'existence de certaines branches du graphe. La classe ATOMIC décrit de manière générique des graphes constitués par un réseau d'*atomes* interconnectés. Elle est définie de la manière suivante:

```
class ATOMIC: public GOBJ {

public:
                ATOMIC();
  virtual      ~ATOMIC();

  int           nbAtom () const;
  BOX3          getBox () const;
  ATOM&         getAtom(int) const;

  void          setDsiNbIter(int);
  void          runDsiGeo();

  virtual void computeBox();

  // Redefinition des methodes virtuelles
```

```
    virtual void     load     (stream&);
    virtual void     save     (stream&) const ;
    virtual int      operator >> (stream&) const ;
    virtual int      operator << (stream&);

    protected:

    virtual void addAtom( ATOM& atom );
    virtual void remAtom( ATOM& atom );


  // rest of the class omitted
};
```

Pour toute instance `atomic` de la classe `ATOMIC`, la description de l'interface publique de cette classe est la suivante:

1. `ATOMIC()` et `~ATOMIC()` sont respectivement les constructeurs et destructeurs de la classe `ATOMIC`.

2. La méthode *nbAtom()* retourne le nombre d'`atome` contenus dans *atomic*, et la méthode *getAtom(*`int i`*)* retourne son i^{ieme} `atome`.

3. *runDsiGeo()* lance l'interpolation des positions dans l'espace des `vertex` de tous les `atome` contenus dans `atomic`.

4. *setDsiNbIter()* permet de fixer le nombre d'itérations de DSI qui sera effectué à chaque itération de *runDsiGeo()*

5. *getBox()* est utilisée pour accéder au plus petit élément de type `BOX3` qui contient `atomic`. Il est calculé grâce à la méthode virtuelle *computeBox()*.

6. *addAtom()* et *remAtom()* sont des méthodes virtuelles responsables de gérer l'ajoût ou le retrait d'un `atome` au sein d'un `atomic`.

Représentation d'un graphe de dimension 1

Nous dirons qu'un graphe est de dimension 1 lorsque ses nœuds sont liés entre eux par **un** segment. Le représentant de ce graphe particulier est alors une ligne polygonale. Les nœuds d'un tel graphe, seront des *atomes* particuliers, contraints par la condition suivante:

- Le nombre de satellites d'un *atome* appartenant à un graphe de dimension 1 est, au maximum, égal à deux.

Nous appelerons ces *atomes* spécifiques des *atompline*. La classe `ATOMPLINE` qui les représente dérive de la classe ATOM et est telle que:

```
class ATOMPLINE : public ATOM {
   public:
              ATOMPLINE();
     virtual ~ATOMPLINE();
     POINT3&  getTangent() const;

    // rest of the class omitted
} ;
```

La classe **ATOMPLINE** ajoute à sa classe de base **ATOM** la méthode publique *getTangent()* qui permet d'accéder à la valeur de la tangente à la ligne en un **atompline**, instance de la classe **ATOMPLINE**. L'**atomic** à une dimension, contenant des **atompline** est appelé *pline*[1]. La classe correspondante, **PLINE** est définie par:

```
class PLINE : public ATOMIC {
   public:
       PLINE();
       virtual ~PLINE();

       void updateTangent();


       int      getNbSeg();
       ATOM*[2] getSeg(int);

       void addAtomPline(ATOMPLINE&);
       void remAtomPline(ATOMPLINE&);

       virtual void split();
       virtual void unsplit();

    // Redefinition des methodes virtuelles


       virtual void   load     (stream&);
       virtual void   save     (stream&) const ;
       virtual int    operator >> (stream&) const ;
       virtual int    operator << (stream&);


    // rest of the class omitted

};
```

[1] *pline* signifie *polygonal line*

Pour toute instance `pline` de la classe `PLINE`, l'interface publique de la classe `PLINE` est définie de la manière suivante:

1. *getNbSeg()* retourne le nombre de *segments* contenus dans `pline`, un *segment* étant défini par deux *atomes* connectés.
2. *getSeg(*`int i`*)* retourne le i^{eme} *segment* de `pline`.
3. *updateTangent()* calcule la valeur des tangentes à `pline` au niveau de ses `atomepline`. Le calcul se fait de façon à respecter la condition mentionnée sur les vecteurs $T(k)$ d'un *atome* k, donnée à la section 1.4.5.
4. Les méthodes *addAtomPline()* et *remAtomPline()* gèrent l'ajoût et le retrait d'un `atompline` dans `pline`.
5. Les méthodes virtuelles *split()* et *unsplit()* sont responsables des changements topologiques de `pline`. En particulier, elles appellent les méthodes *breakSimplex()* et *collapseAtom()* décrites dans la figure 2.9.

A.2 Description des structures propres aux "gshapes"

A.2.1 Description des classes associées aux "vertèbres" et aux "ribs"

Nous avons vu au chapitre 1 qu'un *gshape* était un graphe unidimensionel particulier dont les nœuds sont des *vertèbres* composées de *ribs*. Il semble alors logique de dériver la classe `VERTEBRE`, représentant d'un nœud du graphe représenté par un *gshape*, de la classe `ATOMPLINE`, représentant les nœuds d'un graphe d'une ligne polygonale. De plus, la classe `RIB` associée à un vecteur $V_i(k)$ décrivant le *contour* associé à une *vertèbre* est définie comme dérivant de la classe `POINT3`. Les descriptions des deux classes `VERTEBRE` et `RIB` sont données ci-dessous:

```
class RIB:public POINT3 {
    public :
             RIB();
             RIB( const POINT3& );
             RIB( const RIB& );
     virtual~RIB();

     void    set( const POINT3&);
     bool    isHflag() const;
     void    setHflag() ;
     void    unsetHflag() ;

   // rest of the class omitted
     };
```

```
class VERTEBRE : public  ATOMPLINE {

    public:

                VERTEBRE();
                VERTEBRE(int);
                VERTEBRE(VERTEBRE&);
     virtual   ~VERTEBRE();

     void       set( const PLINE& );
     void       set( const VERTEBRE& );
     void       setOrtho();
     void       setRib(int i, const POINT3&) ;
     RIB&       getRib(int i) const;

     void       setControlSection();
     void       remControlSection();
     POINT3&    getNormal() const;

     POINT3&    getHorizVector() const;

     void       runDsiSecVer();
     void       runDsiNormal();

     // rest of the class omitted
    };
```

Nous définissons l'interface publique de la classe RIB, pour toute instance rib de la classe RIB de la façon suivante:

1. RIB() est le constructeur associé à la classe RIB.
2. RIB(const POINT3&) et RIB(const RIB&) sont les constructeurs d'un rib à partir d'un POINT3 et d'un autre rib.
3. ~RIB() est le destructeur associé à la classe RIB
4. *set(*const POINT3&) permet de spécifier la position spatiale de rib.
5. *isHflag()* permet de savoir si oui ou non rib est l'extrémité d'un *vecteur horizon*[2].
6. *setHflag()* et *unsetHflag()* permettent de spécifier si rib est ou non l'extrémité d'un *vecteur horizon.*

Les principales méthodes publiques supportées par la classe VERTEBRE peuvent être regroupées dans les trois types de fonctionalités suivants: pour toute instance vertebre de la classe VERTEBRE,

[2]voir section 2.3.2

1. méthodes liées à la description de `vertebre`:
   (a) `VERTEBRE(int nbRib)` est le constructeur d'une `vertebre` composée de `nbRib` `rib`. `VERTEBRE(VERTEBRE& vertebre)` est le constructeur d'une `vertebre` à partir d'une autre `vertebre`.
   (b) `~VERTEBRE()` est le destructeur associé à la classe `VERTEBRE`.
   (c) *getRib(*`int i`*)* retourne le `rib` d'*index i* de `vertebre`.
   (d) *getHorizVector()* retourne la valeur du *vecteur horizon* associé à `vertebre`.
2. Méthodes liées à l'interpolation de `vertebre`:
   (a) *setControlSection()* et *remControlSection()* spécifient si `vertebre` est une *section de contrôle* ou non.
   (b) *getNormal()* retourne la valeur de la *normale* au *contour* de `vertebre`.
   (c) La méthode *runDsiSecVer()* lance une itération de l'interpolation par DSI des `rib` de `vertebre`.
   (d) *runDsiNormal()* lance une itération de l'interpolation par DSI de la valeur de la *normale* à `vertebre`.
3. Méthodes de spécification de la géométrie de la vertèbre:
   (a) *set(* `const PLINE& pline`*)* initialise les `rib` de `vertebre` en fonction du contour donné par la ligne polygonale `pline`; cette méthode utilise l'algorithme décrit à la section 2.4.2.
   (b) *set(* `const VERTEBRE& ver` *)* initalise les `rib` de `vertebre` en fonction des `rib` de la `vertebre ver`. Cette méthode utilise l'algorithme décrit à la section 2.4.2.
   (c) *setOrtho()* met à jour la valeur des `rib` de telle sorte que la *normale* de `vertebre` et sa *tangente* soient confondues. Cette méthode utilise l'algorithme décrit à la section 2.5.2.

A.2.2 Description de la classe associée aux *gshapes*

Nous définissons la classe `GSHAPE`, représentant d'un graphe unidimensionnel dont les nœuds sont des `vertebre`, comme dérivée de la classe `PLINE`: sa description est la suivante:

```
class GSHAPE : public PLINE {

    public:

    GSHAPE() ;
    GSHAPE( GSHAPE&);
    virtual ~GSHAPE();
```

```
    void loadFrom( const PLINE&, int nbRib);
    void loadFrom( const PLINE *[], int nbPline, int nbRib );
    int  getNbRib() const;
    void runDsiSec();
    void vertebrePropagate( const RIB& );
    int  correlRib( const VERTEBRE&, const VERTEBRE&,int i,int j );
    void updateNormal();

    void  setHorizVector(int i, int j);

    void addVertebre(VERTEBRE&)
    void remVetrebre(VETREBRE&);


   // Redefinition des methodes virtuelles

    void computeBox();
    void split();
    void unsplit();

    void load  (stream&) ;
    void save  (stream&) const;
    int  operator >> (stream& ) const ;
    int  operator << (stream& );


    // rest of the class omitted
};
```

L'interface publique de toute instance gshape de la classe GSHAPE est la suivante:

1. GSHAPE() est le constructeur associé à la classe GSHAPE.

2. GSHAPE(GSHAPE&) est le constructeur d'un gshape depuis un autre GSHAPE.

3. ~GSHAPE() est le destructeur associé à la classe GSHAPE.

4. *loadFrom(*const PLINE * pl, int nbRib*)* initialise gshape avec nbRib rib par vertebre et un *backbone* initialisé depuis pl. Cette méthode est décrite à la section 2.4.1.

5. *loadFrom(* const PLINE** pl, int nbpline, int nbRib*)* initialise *gshape* avec nbRib rib par vertebre à partir de nbplines pline. Chacune de ces pline matérialise un *contour* de vertebre. Cette méthode est décrite à la section 2.4.2.

6. *getNbRib()* permet d'accéder au nombre de rib composant les vertebre de gshape.

7. *runDsiSec()* interpole toutes les `rib` d'un `gshape` par *DSI* . Celle-ci appelle les méthodes *runDsiVer()* et *runDsiSecNormal()* sur toutes les `vertebre`. Le nombre d'itérations effectuées peut être fixé par la méthode *setDsiNbIter()* héritée de la classe `ATOMIC`.

8. *UpdateNormal()*. Cette méthode calcule la valeur des *normales* des `vertebre` et met à jour les *index* associés aux `rib` de `gshape` de telle sorte que les conventions décrites à la *section* 1.4.5 soient respectées.

9. *correlRib()* spécifie une corrélation entre deux `rib` appartenant à des `vertebre` différentes. Cette méthode est décrite à la section 1.4.5.

10. *setHorizVector(*`int i, int j`*)* spécifie que *tous* les `rib` d'index `i` et `j` de `gshape` sont des extrémités de *vecteurs horizon.*

11. Les méthodes *addVertebre()* et *remVertebre()* gèrent l'ajoût et le retrait de `vertebre` dans un `gshape`.

A.3 Description des structures propres à la charactérisation de réservoir

Nous avons jusqu'ici décrit les structures propres à la modélisation des objets *gshapes.* Ces structures sont générales et ne sont pas adaptées au problème spécifique de la simulation d'hétérogénéités de réservoir. Nous présentons ici les classes qui ont été développées pour traiter les informations nécessaires à la modélisation stochastique des réservoirs pétroliers.

A.3.1 Structures liées au support des données de puits

Indicatrices porteuses de l'information

Pour supporter l'information de type $I(\mathbf{u}; k)$, nous avons défini la classe suivante:

```
enum INDValue{ OFF = 0, ON =1 }

class INDICATRICE {

public:
          INDICATRICE();
  virtual ~INDICATRICE();

  INDvalue getValue( int  ) const;
  INDValue getRef(int ) const;

  void setValue(int, const INDValue&) ;
```

```
    void setRef( int,  const INDValue&);

    // rest of the class omitted
};
```

L'interface publique de toute instance `indicatrice` de la classe `INDICATRICE` est décrite par les méthodes suivantes:

1. `INDICATRICE()` est le constructeur associé à la classe `INDICATRICE`; `~INDICATRICE()` est son destructeur.
2. *getValue(*`int k`*)* permet d'accéder à la valeur 0 ou 1 de `indicatrice` pour le faciès k.
3. *getRef(*`int k`*)* permet d'accéder à la valeur 0 ou 1 de référence de `indicatrice` pour le faciès k.
4. *setValue() et setRef()* permettent de spécifier la valeur actuelle et la valeur de référence de `indicatrice` pour un faciès donné.

localisation des données

Nous avons défini la classe `LOC` pour représenter une localisation singulière **u** supportant les données de type $I_0\{\mathbf{u}; k\}$. Celle-ci est définie de la manière suivante:

```
class LOC {
   public:
                    LOC();
   virtual          ~LOC();
   void             load(const POINT3&, const POINT3&, int);
   INDICATRICE&     getInd()  const;
   POINT3&          getTop()  const ;
   POINT3&          getWall() const ;
   int              getIndex() const;
   void             setInd( const INDICATRICE& );
   void             setTop(POINT3&);
   void             setWall(POINT3&);
   void             setIndex(int) ;

   // rest of the class omitted

   };
```

Soit **u** une instance de la classe `LOCALISATION`. L'interface publique associée à cette classe est la suivante:

1. `LOC()` est le constructeur associé à la classe `LOC`.
2. `~LOC()` en est le destructeur.

3. *load(*`POINT3& wall, POINT3& top, int k`*)* permet d'initialiser une `localisation` **u** dont les extrémités sont spécifiées par les `POINT3 top` et `wall` et qui contient le faciès dont l'index est `k`.
4. *getInd()* permet d'accéder à l'`indicatrice` associée à la localisation **u**.
5. *getTop()* et *getWall()* permettent d'accéder aux `point3` extrémités du segment **u**.
6. *getIndex()* permet d'accéder à l'*index* associé au faciès intersecté par **u**.
7. *setInd()* permet de spécifier quelle est l'`indicatrice` associée à **u**.
8. *setTop()* et *setWall()* permettent de spécifier quels sont les `POINT3` extrémité du segment **u**.
9. *setIndex()* permet de spécifier quel est l'*index* associé au faciès intersecté par **u**.

D'autre part, nous avons défini l'objet *well* comme étant un ensemble de localisations **u** et de trajets de puits *wseg*. Ceux-ci sont caractérisés par la classe `WSEG` définie de la manière suivante:

```
class WSEG {
    public :
                WSEG();
        virtual ~WSEG();
        POINT3& geTop() const;
        POINT3& getWall() const;
        int     getIndex() const
        void    setTop( const& POINT3 );
        void    setWall( const& POINT3 );
        void    setIndex( int ) ;

        // rest of the class omitted

    };
```

L'interface publique d'une instance `wseg` de la classe `WSEG` est la suivante:

1. `WSEG()` et `~WSEG()` sont les constructeurs et destructeurs associés à la classe `WSEG`.
2. *getTop()* et *getWall()* permettent d'avoir accès aux `POINT3` extrémités de `wseg`.
3. *setTop()* et *setWall()* permettent de spécifier les extrémités de `wseg`.
4. *getIndex()* donne accès à l'index caractérisant le faciès intersecté par `wseg`.
5. *setIndex()* permet de spécifier l'index du faciès intersecté par `wseg`.

La classe WELL est alors définie de la manière suivante:

```
class WELL : public GOBJ {

public:          WELL() ;
                 WELL(WELL&);
        virtual ~WELL() ;

         void  loadFrom( const WSEG& );
         void  loadFrom( int, int ) ;
         int   getNbWseg() const;
         int   getNbLoc() const;
         WSEG& getWseg(int) const;
         LOC&  getLoc(int) const;
         BOX3& computeBox() ;

        // Redefinition des methodes virtuelles

         void load  (stream&) ;
         void save (stream& ) const;
         int  operator >> (stream&) const ;
         int  operator << (stream&);

        // rest of the class omitted

        };
```

L'interface publique associée à toute instance `well` de la classe `WELL` est la suivante:

1. `WELL()` et `WELL(well&)` sont les constructeurs associés à la classe `WELL`.
2. `~WELL()` est son destructeur.
3. *loadFrom(* `const WSEG&` *)* ajoute un `wseg` dans `well`.
4. *loadFrom(* `int Div, int k` *)* gère l'ajout d'une `localisation` dans `well` à partir de l'index de discrétisation *Div* donné pour le faciès d'index[3] `k`
5. *getNbWseg()* et *getNbLoc()* retournent le nombre de `loc` et de `wseg` contenus dans `well`.
6. *getWseg(*`int i`*)* permet d'accéder au i^{eme} `wseg` de *well.*
7. *getLoc(*`int i`*)* permet d'accéder à la i^{eme} *loc* de *well.*
8. *computeBox()* permet de calculer le plus petit élément de type `BOX3` contenant `well`.

[3]cette notion a été exposée à la section 4.5.2

Supports des données de corrélation entre puits

La classe **INDCOR** a été mise en place pour représenter les variables indicatrices $\{I(\mathbf{u}, \mathbf{u}', k)\}$ rendant compte des corrélations entre deux localisations $\{\mathbf{u}\}$ et $\{\mathbf{u}'\}$ pour le faciès k. Elle est définie de la façon suivante:

```
enum INDValue{ OFF = 0, ON =1 };

class INDCOR {

      public:

                INDCOR();
       virtual ~INDCOR();

       LOC& getFirstLoc() const;
       LOC& getSecondLoc() const;
       int  getFaciesIndex() const;

       float    getRefValue() const;
       INDValue getValue() const;

       void     setFirstLoc( const LOC&);
       void     setSecondLoc(const LOC&);
       void     setRefValue(float) ;

       void     setValue(INDValue&);

       // rest of the class omitted

   };
```

Les méthodes publiques associées à toute instance `indcor` de la classe `INDCOR` sont les suivantes:

1. `INDCOR()` et `~INDCOR()` sont les constructeurs et les destructeurs de la classe `INDCOR`.
2. *getFirstLoc()* et *getSecondLoc()* retournent les deux `localisation` **u** et **u'** pour lesquelles `indcor` est définie.
3. *setFirstLoc()* et *setSecondLoc()* permettent de les spécifier.
4. *getFaciesIndex()* retourne l'index k pour lequel `indcor` est définie.
5. *getValue()* et *getRefValue()* retournent la valeur actuelle et la valeur de référence associée à `indcor`.
6. *setValue()* et *setRefValue()* permettent de les spécifier.

A.3.2 Support des données des objets "rshapes"

Nous avons vu au chapitre 3 que les objets *rshapes* étaient une extension de l'objet *gshape* à la représentation spécifique d'une hétérogénéité de réservoir. Nous avons donc dérivé la classe **RSHAPE** de la classe GSHAPE. Celle-ci est définie comme suit:

```
class RSHAPE;

class SHAPETRANSFORM {

    public:

              SHAPETRANSFORM();
     virtual ~SHAPETRANSFORM();

     void TRANSFORM(RSHAPE&);

     protected:

     virtual rtransform(RSHAPE&) = 0;
    };

class RSHAPE : public GSHAPE {

 public:

          RSHAPE() ;
          RSHAPE(RSHAPE&);
  virtual ~RSHAPE() ;

  void             load( const GSHAPE&, int );
  int              getFaciesIndex() const;
  SHAPETRANSFORM&  getNextTrans() const;
  void             setNextTrans(SHAPETRANSFORM&) ;

  void             ApplyNextTrans() ;

  void             store()
  void             reset();
  bool             IsIntersecting(LOC&) const;

  // redefinition des methodes virtuelles

  void load         (stream&) ;
  void save         (stream&) const;
  int  operator >> (stream& ) const ;
  int  operator << (stream& );

  // rest of the class omitted
```

```
};
```

L'interface publique de toute instance `rshape` de la classe `RSHAPE` est la suivante:

1. `RSHAPE()` et `RSHAPE(rshape&)` sont les constructeurs de la classe `RSHAPE`.
2. `~RSHAPE()` est le destructeur de la classe `RSHAPE`.
3. *getFaciesIndex()* permet de connaitre le faciès auquel appartient `rshape`.
4. *loadFrom(*`const GSHAPE& gshape, int k` *)* initialise `rshape` à l'aide de l'information contenue dans *gshape* et lui attribue le faciès d'index `k`. `gshape` sert alors de *template* au faciès d'index *k*.
5. La méthode *setNextTrans(*`SHAPETRANSFORM& shapetransform`*)* spécifie `shapetransform` comme *transformation courante* associée à `rshape`. Dans l'état actuel de l'implémentation la méthode *rtransform()* peut définir les 4 transformations de base *rotation, translation, affinity, déformation* vues au chapitre 3.
6. La méthode *getNextTrans()* permet d'accéder à la *transformation courante.*
7. *ApplyNextTrans()* engendre les actions suivantes:
   (a) Applique la méthode *store()* qui stocke l'*enveloppe* de `rshape` telle qu'elle se trouve avant toute transformation.
   (b) Applique la *transformation courante* à `rshape`.
8. La méthode *reset()* remet l'*enveloppe* de `rshape` dans l'état stocké par le dernier appel à la méthode *store()*.
9. La méthode *IsIntersecting(* `LOC& loc`) est utilisée pour savoir si oui ou non `loc` intersecte *rshape*

Nous voyons ici que `SHAPETRANSFORM` apparait comme la classe de base des classes définissant les opérations qui peuvent être appliquées à un *rshape*. Une de ces classe dérivée, la classe `SHAPETRANSLATION` est donnée ci-dessous:

```
class SHAPETRANSLATION{

   public:
              SHAPETRANSLATION();
      virtual ~SHAPETRANSLATION();

      void setOrigin(const POINT3&);
      void setDimension(const POINT3&);
      void setBasis( const POINT3&, const POINT3&, const POINT3& );
```

```
rtransform(RSHAPE&) = something;

// rest of the class omitted

};
```

1. `SHAPETRANSLATION()` et ~SHAPETRANSLATION() sont les constructeurs et destructeurs de cette classe.

2. *setOrigin(), setDimension()* et *setBasis()* permettent de spécifier les paramètres d'une instance `shapetranslation` de la classe `SHAPETRANSLATION`. Ceux-ci correspondent aux 5 paramètres de l'opération *translation* donnés à la section 3.5.3.

3. `rtransform()` est redéfinie de manière à ce que son application engendre la transformation *translation.*

A.4 Notion de réservoir

Nous avons jusqu'ici appelé "réservoir" une portion parallélépipédique de l'espace, contenant différentes familles d'hétérogénéités distribuées au sein du faciès *matrice.* Une "famille" d'hétérogénéités est considérée comme un ensemble de *rshapes* dont l'*index* est commun et que l'on souhaite considérer comme un élément homogène du réservoir. Nous avons donc développé la notion de *groupe* d'objets. Celle-ci est définie par la classe suivante:

```
class GROUP : public GOBJ {
   public:
                    GROUP();
      virtual       ~GROUP();

      void          add( const GOBJ&) ;
      void          rem( const GOBJ&) ;
      int           nbGobj() const;
      GOBJ_t&       getObj(i) const;
      int           groupOfKindOf() const;

      // redefinition des methodes virtuelles

      void load  (stream&) ;
      void save  (stream&) const;
      int  operator >> (stream& ) const ;
      int  operator << (stream& );

      // rest of the class omitted

};
```

D'autre part, la notion de groupe satisfait la contrainte suivante:

- Une instance *groupe* de la classe GROUP ne peut contenir que des objets de même type; on parlera de groupe homogène

En conséquence, un *groupe* d'objets de type *groupe* est un groupe homogène et peut être utilisé pour grouper des objets appartenant à des classes différentes. L'interface publique de toute instance group de la classe GROUP est la suivante.

1. `GROUP()` et `~GROUP()` sont les constructeurs et les destructeurs associés à la classe `GROUP`.
2. *add()* et *rem()* gèrent l'ajoût et le retrait d'un objet de type `gobj` dans `group`.
3. *nbGobj()* retourne le nombre d'objets de type `gobj` contenus dans `group`.
4. *getObj(*`int i`*)* donne accès au i^{eme} `gobj` contenu dans `group`.
5. *groupOfKindOf()* retourne le type des objets contenus dans un `group`.

On représente alors la notion de *réservoir* contenant K faciès (dont le faciès *matrice*) comme un `group` d'objets de type `GROUP`. Les `group` qu'il contient sont les suivants:

1. Un `group` de *wells*;
2. $(K-1)$ `group` de `rshape`, chacun de ces $(K-1)$ `group` contenant des `rshape` appartenant au même faciès.

La classe `reservoir` associée est définie de la manière suivante:

```
class RESERVOIR : public GROUP {

 public:

             RESERVOIR(GROUP& wellgroup);
    virtual  ~RESERVOIR();

     void     load( const GROUP&);
     void     loadFacies( const GSHAPE&, int );
     void     loadFacies( const GROUP&);
     void     perturb  ();
     void     perturb (GROUP&);

     int      getNbFacies() const ;
     int      getNbWell() const;
     int      getNbLoc() const;
     int      getNbIndCor() const;

     LOC&     getLoc( int ) const;
     INDCOR&  getIndCor( int ) const;
```

```
    void      setPrecision( int, float ) ;
    void      setWeight( int, float ) ;
    void      setOrigin( const POINT3& );
    void      setDimension( POINT3& ) ;
    void      setBasis( const POINT3&, const POINT3&, const POINT3& );
    void      setProportion( int, float ) ;

    void      setCorrel( const LOC&, const LOC&,float );
    void      remCorrel( const LOC&, const LOC&);

    void      setNextTrans( SHAPETRANSFORM&, int );

    float     computeAdjustmentTerm();
    float     computeCorrelationTerm();
    float     computeCost();

    // Redefinition des methodes virtuelles


    void load  (stream&) ;
    void save  (stream&) const;
    int  operator >> (stream& ) const ;
    int  operator << (stream& );
     .
    void      writeResult (const stream&);

    // rest of the class omitted

};
```

Nous allons décrire l'interface publique associée à la classe `RESERVOIR`. Pour toute instance `reservoir` de cette classe, celle-ci est la suivante:

1. `RESERVOIR()` et `RESERVOIR(group& wellgroup)` sont les constructeurs de la classe `RESERVOIR`. `~RESERVOIR()` est son destructeur.
2. *load(*`GROUP& groupwell`*)* initialise `reservoir` en fonction du group de `well` `groupwell`.
3. *loadFacies(* `GSHAPE& gshape, int k` *)* génère un faciès d'index `k` à partir du *template* `gshape` par la procédure stochastique de création de faciès[4].
4. *loadFacies(* `GROUP& rshapeGroup` *)* définit le *groupe* de *rshape* `rshapeGroup` comme étant un faciès de `réservoir`.
5. *getNbLoc()* retourne le nombre de `loc` contenues dans `reservoir`.
6. *getNbIndCor()* retourne le nombre de `indcor` contenues dans `reservoir`.

[4] voir section 3.6.1

7. *getLoc(*`int i`*)* retourne la i^{eme} `loc` de `reservoir`.
8. *getIndCor(*`int i`*)* retourne le i^{eme} `indcorl` de `reservoir`.
9. *setCorrel(*`LOC&u1 , LOC&u2, float p`*)* spécifie que les deux `localisation` **u1** et **u2** sont corrélées avec une probabilité de p.
10. *remCorrel(*`LOC&u1 , LOC&u2`*)* supprime toute donnée de corrélation entre **u1** et **u2**.
11. *setNextTrans()*, spécifie quelle est la *transformation courante* associée à tous les `rshape` d'un même faciès.
12. *computeAdjustmentTerm()* et *computeCorrelationTerm()* calculent les valeurs de O_{adj} et O_{cor} pour l'ensemble des *loc* contenues dans `reservoir`.
13. *perturb ()* lance le procédé de simulation sur tout `reservoir`.
14. *perturb(*`GROUP& `*)* lance le procédé d'interpolation sur un `group` de `rshape`.
15. *getNbIter()* permet d'avoir accès au nombre d'itérations qui sera effectué au cours de la simulation.
16. *setNbIter()* permet de fixer ce nombre.
17. La méthode *writeResult()* écrit dans un flot de sortie un certain nombre de résultats statistiques.

A.5 Conclusion

La hierarchie décrite ci-dessus reflète les caractéristiques essentielles des idées présentées au cours de ce manuscrit. Elle implémente d'une manière logique les relations entre des objets dont la signification a été développée dans cette thèse. Ainsi, la classe `gshape` hérite de toutes les caractéristiques de la classe `pline` et, en particulier de

1. la définition de sa *topologie* et des méthodes nécessaire pour la modifier.
2. sa *tangente*;
3. la méthode d'interpolation de la géométrie du *backbone*.

La classe `rshape`, hérite de toutes les caractéristiques de la classe `gshape` et, en particulier de:

1. tout ce qui concerne la définition de la géométrie;
2. les méthodes d'interpolation spécifiques aux *sections*;

Y sont rajoutés les éléments nécessaires à la spécification du fait qu'un *rshape* représente une hétérogénéité. Le graphe de dérivation illustrant la hierarchie entre les principaux objets est présenté dans la Figure A.1.

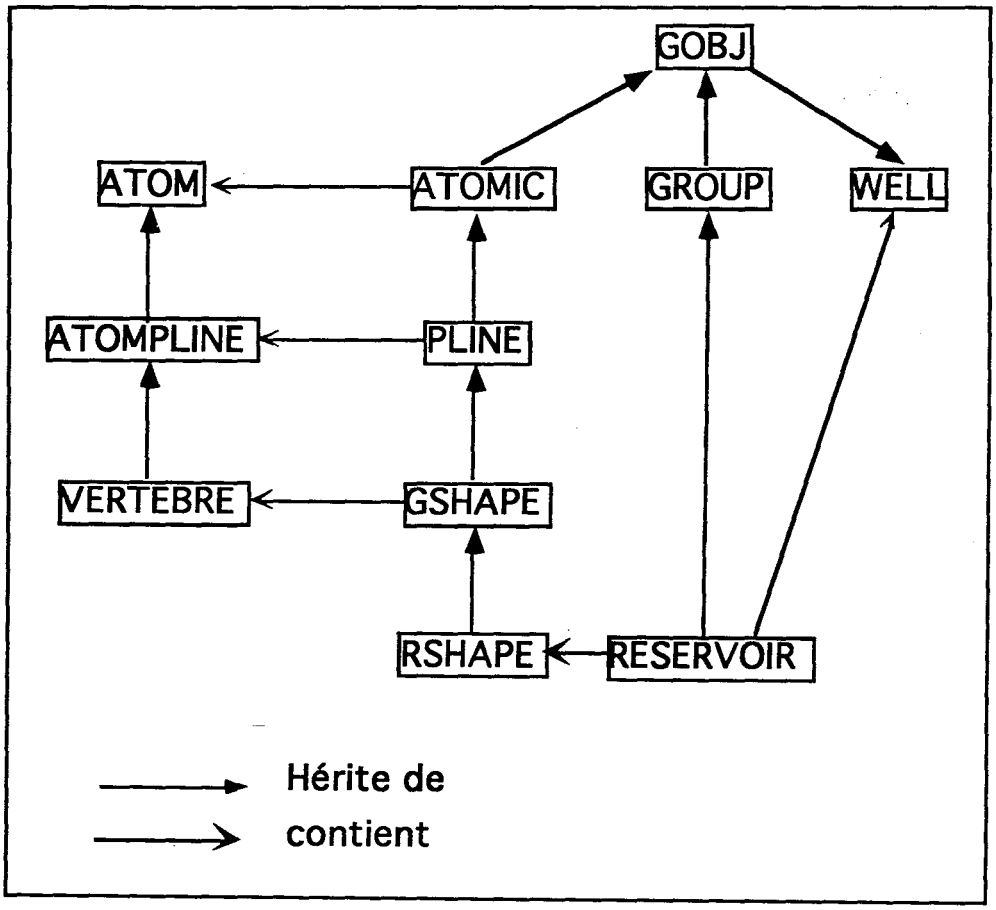


Figure A.1 Hierarchie entre les principaux objets

Lexique

backbone	:	ligne polygonale consituée d'un ensemble de nœuds liés entre eux.
gshape	:	représentation d'un graphe unidimensionnel dont la topologie et la géométrie sont définies par un *backbone*.
vertèbre	:	représentation mathématique d'un nœud de *gshape*. Une vertèbre définit la position dans l'espace de ce nœud, sa topologie, et la forme du *contour* associé à ce nœud.
sections	:	ensemble des contours plans associés à chaque nœud d'un *gshape*.
contour	:	contour plan associé à une *vertèbre* de *gshape*. Un *contour* est décrit par N *ribs* coplanaires
rib	:	un des vecteurs formant le *contour* d'une *vertèbre* et décrit relativement à la position de cette vertèbre.
enveloppe	:	surface obtenue en liant entre elles toutes les *sections* d'un *gshape* autour de son *backbone*.
nœud de contrôle	:	*vertèbre* ayant une position fixe et invariante.
section de contrôle	:	*vertèbre* ayant un *contour* fixe et invariant.
$n(k)$	:	vecteur normal au *contour* associé à la *vertèbre* k.
$T(k)$	:	vecteur tangent au *backbone* en la *vertèbre* k.
$\{Div(k)\}$	:	index de discrétisation associés aux faciès k.
$\{\omega(k)\}$	:	poids relatifs associés aux faciès k.
O	:	valeur de la fonction coût.
O_{adj}	:	terme d'ajustement de la fonction coût.
O_{cor}	:	terme de corrélation de la fonction coût.
$I(\mathbf{u};k)$	:	variable indicatrice d'ajustement associée au faciès k et à la localisation $\mathbf{u}$.
$I(\mathbf{u},\mathbf{u}';k)$	:	variable indicatrice de corrélation associée au faciès k et aux localisations $\mathbf{u}$ et $\mathbf{u}'$.
S	:	ensemble des localisations des données d'ajustement associées au réservoir.
S_{cor}	:	ensemble des paires de localisations où des données de corrélation sont définies.
famille d'hétérogénéités	:	faciès composé d'hétérogénéités de même type et pouvant être caractérisé par un *template*.

matrice	:	faciès qui n'est pas considéré comme une famille d'hétérogénéités.
template	:	*gshape* père utilisé pour définir une famille d'hétérogénéités.
rshape	:	représentant d'une hétérogénéité au sein d'un faciès donné. Un *rshape* est issu du *template* définissant ce faciès.
transformation courante	:	transformation associée à un *rshape* et égale à une des quatre transformations aléatoires translation, rotation, affinité, déformation.
V_{tot}	:	volume total du réservoir.
p_k	:	proportion relative du faciès k dans le réservoir.
t_k	:	*template* associé au faciès k.

Bibliographie

1. F. Alabert, G. Massonat (Septembre 1990). *Heterogeneity in a Complex Turbiditic Reservoir: Stochastic Modelling of Facies and Petrophysical Properties.* Paper SPE 20604, presented at the 1990 Annual Technical Conference and Exhibition, New Orleans.

2. G.Booch (1992). *Conception orientée objets et applications.* Addison-Wesley.

3. Y.Chipot, Y.Huang, P.Jacquemin, J.L. Mallet (Janvier 90). *Présentation du programme* GOCAD *et premiers résultats.* Journées AFCET GROPLAN, BIGRE, num.67, pages 75-85.

4. R.Clementsen, A.R Hurst, R.Knarud and H. Omre (Juin 89). *A Computer Program for Evaluation of Fluvial Reservoir.* presented at the 1989 Intl Conference on North Sea Oil and Reservoirs, Trondheim, Norway.

5. G.R. Cross, A.H. Jain (Janvier 1983). *Markov Random Field Texture Models.* IEEE Transations on pattern analysis and machine intelligence, vol. PAMI-5, $N°1$.

6. A.J Da Costa e Silva (1985). *A New Approach to the characterization of Reservoir Heterogeneity Based on the Geomathematical Model and Kriging Techniques.* Paper SPE 14275, presented at the 1985 Annual Technical Conference and Exhibition, Las Vegas.

7. E. Damsleth, C.B Tjolsen, K.H Omre, H.H Haldorsen (Septembre 1990). *A Two stage Stochastic Model Applied to a North Sea Reservoir.* Paper SPE 20605, presented at the 1990 Annual Technical Conference and Exhibition, New Orleans.

8. K.Dautenhan Wyatt, S.K. Towe, J.E. Layton, S.B. Wyatt, D.H. Von Seggern, C.A. Brockmeier (Octobre 1992). *Ergonomics in 3D depth Migration.* Expanded Abstracts on Technical Programs. 1993 Annual SEG meeting, New Orleans.

9. R. Della Malva,M.C.Williams,T.J.Kunz (Octobre 1992). *3-D Model Construction of Overthrust Geology for structural Imaging Of Seismic by Migration.* Technical Programs-Expanded Abstracts. 1993 Annual SEG meeting, New Orleans.

10. C.V. Deutsch and A.G. Journel (1992). *GSLIB:Geostatistical Software Library.* Oxford University Press, New York, NY.

11. C.V. Deutsch (Juin 92). *Annealing techniques applied to Reservoir Modeling and the Integration of geological and engineering (well test) data.* Phd thesis.

12. P.A. Dowd (1991). *A review of recent developments in geostatistics.* Computers & Geosciences, Volume 17, Nb 10, pp1483-1498.

13. P.Doyen and T Guidish (Mai 1990). *Seismic discimination of lithology: a Bayesian approach.* Geostatistics Symposium in Calgary.

14. O. Dubrule, (March 1988). *A Review of Stochastic Models for Petroleum Reservoirs.* paper presented at the 1988 British Soc. Reservoir Geologists Meeting on Quantification of Sediment Body Geometries and their Internal Heterogeneities, London.

15. G.Dueck and T.Scheuer (1990). *Threshold Accepting:a general purpose optimization algorithm appearing superior to simulated Annealing.* Journal of Computational Physics, 90:161-175.

16. L.M. Falt, A.Henriquez, L.Holden,H.Tjelmeland (Mai 1991). *Moheres, a program system for simulation of reservoir architecture and properties.* 6th European symposium in Stavanger, Norway.

17. G.Farin, (1988). *Curves and Surfaces for Computed Aided Geometric Design. A practical guide.* Academiv Press, Inc.

18. G. Farin (1987). *Geometric Modeling: algorithms and new trends.* Society for Industrial and Applied mathematics.

19. G. Farin (1986). *Triangular Bernstein-Bezier patches.* Comput. Aided Geom. Des.

20. C.Farmer (1991). *The mathematical Generation of Reservoir Geology.* Oxford University Press, New York, NY.

21. P.N. Georgiades, D.P. Greenberg (Janvier1992). *Locally Manipulating the Geometry of Curved Surfaces.* IEEE Computer Graphics and Applications.

22. K.E.Gorlen, S.M.Orlow, P.S.Plexico (1991). *Data abstraction and object-oriented programming in C++.* JOHN WILEY & SONS.

23. J.L Guiziou, J.L Mallet (Juin 1992). *On developing an advanced 3D macro-models estimation tool on top of the* GOCAD *modeler.* EAEG meeting 54, Paris. Technical Programme and Abstract of Papers.

24. J.J Gòmez-Hernàmdex and R.M.Srivastava. *ISIM3D: An ansi-c three-dimensional multiple indicator conditional simulation program.* Computers & Geosciences 16(4):395-408.

25. H.H. Haldorsen and E. Damsleth (Avril 1989). *Stochastic Modeling.* SPE Distinguished Author Series.

26. H.H Haldorsen, C.J MacDonald (Septembre 1987). *Stochastic Modeling of Underground Reservoir Facies (SMURF).* Paper SPE 16751, presented at the 1987 Annual Technical Conference and Exhibition, New Orleans.

27. A.Henriquez, K.J.Tyler,A.Hurst (Septembre 1990). *Characterization of Fluvial Sedimentology for Reservoir Simulation Modeling.* SPE formation Evaluation, 1990.

28. D.Hearn, M.P. Baker (1986). *Computer Graphics.* PRENTICE-HALL, Englewood Cliffs, New Jersey..

29. T.A Hewett, R.A. Behrens (Septembre 1990). *Conditional Simulation of Reservoir Heterogeneities with Fractals.* SPE formation Evaluation, 1990.

30. Y.Huang (Juin 1990). *Modélisation et manipulation des surfaces triangulées.* Phd Thesis.

31. W.M Hsu, J.F. Hughes, H. Kaufman (Juillet 1992). *Direct Manipulation of Free-Form Deformations.* Computer Graphics, 26(2). Proceedings of SIGGRAPH '92.

32. A.C.MacDonald, T.H.Hoye, P.Lowry, T.Jacobsen, J.O Aasen, A.O.Grindheim (Avril 1992). *Stochastic flow unit modelling of a North Sea coastal-deltaic reservoir.* FIRST BREAK Vol 10, No 4.

33. E.H Issaks and R.M Srivastava (1989). *An Introduction to Applied Geostatistics.* Oxford Press.

34. A.G. Journel (Septembre 1990). *Geostatistics for Reservoir Characterization* Paper SPE 20750, presented at the 1990 Annual Technical Conference and Exhibition, New Orleans.

35. A.G. Journel(1989). *Fundamentals of Geostatistics in Five Lessons.* American Geophysical Union.

36. A.G. Journel, F.G. Alabert (Septembre 1988). *Focusing on Spatial Connectivity of Extreme-Valued Attributes: Stochastic Indicator Models of Reservoir Heterogeneities.* Paper SPE 20750, presented at the 1988 Annual Technical Conference and Exhibition, Houston.

37. A.G.Journel (1983). *Non-parametric estimation of spatial distributions.* Math Geology, 15(3):445-468.

38. A.G. Journel and Donato Posa (1990). *Characteristic Behaviour and Order Relations for Indicator Variograms.* Math. Geology, 22(8):1011-1026.

39. A.G. Journel and E.H. Isaaks (1984).. *Conditional Indicator Simulation: Application to a Saskatchewan Uranium Deposit.* Math.Geology, 16(7):685-718.

40. S.Kirkpatrick, C.Gelatt Jr. and Mr.Vecchi (Mai 1983).. *Optimization by Simulated Annealing.* Science, 220(4590):671-680.

41. P.Lavest and Y.Chipot. *Building Complex Horizons for 3D seismics.* Technical Program and Expanded Abstract,1993 Annual SEG meeting, Washington.

42. M. Lounsbery, S. Mann, T. DeRose (Septembre 1992). *Parametric Surface Interpolation.* IEEE Computer Graphics and Applications.

43. J.L Mallet,R.Cognot,S.Cases (Juin 1993). *3D Interpolation of geophysical parameters.The* GOCAD *approach..* EAEG meeting 55, Stavenger. Technical Programme and Abstract of Papers.

44. J.L Mallet, P.Le Melinaire (Juin 1992).. *Modelling complex faults. The* GOCAD *approach..* EAEG meeting 54, Paris. Technical Programme and Abstract of Papers.

45. J.L. Mallet, (Avril 1992). *Discrete Smooth Interpolation in geometric modeling.* Computer Aided Design Journal, Vol24, No.4.

46. N.Metropolis, A.Rosenbluth, M.Rosenbluth, A.Teller and E.Teller (Juin 1953). *Equation of State calculations by fast computing machines.* J.Chem.Phys.,21(6):1087-1092.

47. P.Nobili, J.L. Mallet, Y.Huang (Septembre 1990). *Using Simplex for 3-D Two Point Ray Tracing on Veri Complex Surfaces.* Expanded Abstracts on Technical Programs. 1990 Annual SEG meeting, San Francisco.

48. J.L. Mallet, (1989). *Discrete Smooth Interpolation.* Assoc. for Comp. Machinery, Transactions on Graphics, Vol.8, No.2.

49. M.E. Mortenson, (1985). *Geometric Modelling.* John Wiley.

50. L. Piegl, (Janvier 1992). *On NURBS:A Survey.* IEEE Computer Graphics and Applications.

51. W.Press, B.Flannery,S.Teukolsky and W.Vetterling (1986). *Numerical Recipies.* Cambridge University Press, New York, NY.

52. V.Suro-Pérez and A.G.Journel (1991). *Indicator Principal Component Kriging.* Math.Geology, 23(5):759-788.

53. V.Suro-Pérez (1991). *A boolean Approach for Modeling Heterogeneous Reservoirs.* In Report4, Stanford Center for Reservoir Forecasting, Stanfore, CA.

54. D.Stoyan, W.Kendall and J.Mecke (1987). *Stochastic Geometry and its applications, page 72.* John Wiley & Sons, New York, NY.

55. V.Suro-Perez (Mai 1991). *Generation of a turbiditic reservoir: the Boolean alternative.* Report4, Stanford Center for Reservoir Forecasting, Stanford, CA.

56. V.Suro-Perex and A.J Journel (1990). *Stochastic Simulation of Lithofacies for Reservoir Characterization.* Proceedings of the 2^{nd} europeen conference on the mathematics of oil recovery,pp 3-10.Paris.

57. W. Welch, A. Witkin (Juillet 1992). *Variational Surface Modeling.* Computer Graphics, 26(2). Proceedings of SIGGRAPH '92.

58. L. Wietzerbin, J.L. Mallet (Octobre 1993). *Parameterization of Complex 3D heterogeneities: A new CAD approach.* Paper SPE 26423, presented at the 1993 Annual Technical Conference and Exhibition, Houston.

59. L. Wietzerbin, J.L. Mallet (Septembre 1993). *Modeling complex 3D heterogeneities with the Discrete Smooth Interpolation method.* Expanded Abstracts of the technical programs, 1993 Annual SEG meeting, New Orleans.

60. L. Wietzerbin, J.L. Mallet (Juin 1992). *A method for fitting three dimensional shapes to real reservoir data.* EAEG meeting 54, Paris. Technical Programme and Abstract of Papers.

Index

E

F

G

H

I

L

M

N

O

P

R

S

T

V

W

INSTITUT NATIONAL
POLYTECHNIQUE
DE LORRAINE

AUTORISATION DE SOUTENANCE DE THESE DU DOCTORAT DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

o0o

VU LES RAPPORTS ETABLIS PAR :
Monsieur JOURNEL André, Professeur, Université Stanford USA,
Monsieur ARQUEZ Didier, Professeur, Université Besançon,
Monsieur SCHOTT René, Professeur, CRIN/Université Nancy I.
Le Président de l'Institut National Polytechnique de Lorraine, autorise :

NANCY BRABOIS
2, AVENUE DE LA
FORET-DE-HAYE
BOITE POSTALE 3
F - 5 4 5 0 1
VANDŒUVRE CEDEX

Mademoiselle WIETZERBIN Liliane

à soutenir devant l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE, une thèse intitulée :
"Modélisation et paramétrisation d'objets naturels de formes complexes en trois dimensions.Application à la simulation stochastique de la distribution d'hétérogénéités au sein des réservoirs pétroliers"

en vue de l'obtention du titre de :

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Spécialité : **"INFORMATIQUE"**

Fait à Vandoeuvre le, **18 Février 1994**
Le Président de l'I[illegible]
M. LUCIUS

INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE NANCY

TEL. 33/83.59.59.59