Modélisation de la structure interne des réservoirs de type fluviatile : application sur un site de stockage de gaz en aquifère

Pascal Lavest

Institut
National
Polytechnique
de Lorraine

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

THESE

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

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

Géosciences

par

Pascal LAVEST

Modélisation de la structure interne des réservoirs de type fluviatile. Application sur un site de stockage de gaz en aquifère

Soutenue le 21 Juin 1996

Nicolas	CHEIMANOFF	Rapporteur
Michel	PERRIN	Rapporteur
Jean-Jacques	ROYER	Rapporteur
Jean-Laurent	MALLET	Examinateur
Liliane	WIETZERBIN	Examinateur
François	VERDIER	Examinateur

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, pour m'avoir accueilli au sein de son équipe de recherche. Grâce au consortium Gocad qu'il dirige, ses étudiants bénéficient de conditions de travail exceptionnelles, aussi bien au niveau du matériel dont ils disposent, que des contacts qu'ils peuvent établir au cours de leurs recherches.

Mes remerciements vont également à la société Gaz-de-France, et plus particulièrement à François Verdier, Robert Argiolas, Alain Variéras et Liliane Wietzerbin, pour leur disponibilité, et pour m'avoir permis d'accéder à un jeu de données avec lequel j'ai pu tester et valider les méthodes que j'ai développées.

Je remercie également M. Bayer, M. Cheimanoff et M. Royer de l'honneur qu'ils me font en acceptant d'être rapporteurs de cette thèse.

Je dis aussi un grand merci à toutes les personnes du L.I.A.D pour leur aide, en particulier à Madame Cugurno pour son dévouement et son efficacité, et à Pierre pour avoir accepté de relire ce manuscrit.

Résumé

La modélisation géologique des réservoirs de type fluviatile est un domaine dans lequel de nombreuses techniques ont été développées, qui correspondent chacunes à un contexte particulier, souvent déterminé par la nature et la densité des données disponibles.

Dans le cadre de cette thèse, nous nous situons par rapport à l'étude d'un site de stockage de gaz naturel de type aquifère. La densité et la diversité dont nous disposons ici nous permettent d'envisager la construction d'un modèle déterministe.

Pour cela, nous avons développé de nouveaux outils de modélisation, qui se regroupent autour de trois thèmes:

- la modélisation géométrique et la caractérisation volumique de séquences stratigraphiques. C'est le sens de la création d'un nouvel objet, que nous avons baptisé le *gstack*.

- la création d'un outil de type grille régulière, auquel nous avons associé de nouvelles fonctionnalités. Celles-ci permettent, dans le cadre d'un projet de modélisation géologique complexe où plusieurs types de maillages différents sont utilisés à différentes étapes de ce projet, d'intégrer l'ensemble des résultats obtenus sur un support de travail commun.

- le développement d'un nouvel algorithme d'interpolation, basé sur la méthode D.S.I, qui permet de contraindre par un champ d'anisotropies non uniforme l'estimation d'une propriété physique sur une grille régulière.

Ces outils ont été testés et validés par la construction du modèle géologique d'un site de stockage en aquifère. Ce travail nous a également permis de montrer comment il était possible d'intégrer sur un seul modèle plusieurs sources d'informations différentes (données sismiques, données de puits, connaissance du géologue, etc...).

Abstract

The modeling of fluvial reservoirs is a study area for which many technics have been developped, all related to a specific context, defined by be nature and the density of data available.

Our framework is the study of a natural gas underground storage unit, where the amount of data available enables us to build a deterministic model.

For that purpose, we have developped new modeling tools, that can be grouped in three different parts:

- The geometrical modeling and volumic characterization of stratigraphical sequences, through a new object, that we called the *gstack*.
- The creation of a regular-grid type object, on which we implemented new functionnalities. These allow us, in the frame of a complex geological modeling project where differents types of mesh are used at differents steps, to integrate the results of each of these steps on one single object.
- the development of a new interpolation algorithme, based on the D.S.I method, which allows to constrain the estimation of a physical property on a regular grid, by a non-constant anisotropy field.

These tools have been tested and validated by the contruction of a geological model of a natural gas storage reservoir. This work also let us show how it was possible to integrate different types of information (seismic data, well data, knowledge of the geologist, etc...) in one model.

Table des matières

Introduction

La modélisation géologique des réservoirs pétroliers est une des premières étapes de la chaîne des opérations qui permettent de faire des estimations de réserves en place, des prévisions de production à long terme, et ainsi de mesurer les conséquences économiques des stratégies d'exploitation mises en place.

Au sein du projet Gocad ([7]), dans lequel s'inscrit ce travail de thèse, de nombreux travaux ont été entrepris dans ce domaine et souvent menés à bien, qui ont permis d'apporter des solutions cohérentes et adaptées à certains problèmes spécifiques de la géologie.

La modélisation de la géométrie et de la structure des sédiments fluviatiles fait partie de cette famille de problèmes. Les sédiments de type fluviatile, bien qu'ils ne représentent qu'une très faible partie de l'ensemble des dépôts sédimentaires, ont en effet sucité un intérêt croissant dans la communauté scientifique durant les deux dernières décennies pour leur rôle en tant que roches réservoirs d'hydrocarbures ou en tant qu'aquifères.

Cet intérêt a conduit à la publication de nombreux articles et ouvrages dont nous proposons une synthèse sommaire dans le premier chapitre de ce travail. Cette synthèse s'articule autour de trois thèmes principaux: la description géologique des sédiments fluviatiles, leur caractérisation à l'aide d'outils mathématiques et les différentes méthodes de modélisation des réservoirs fluviatiles.

L'étude de ce dernier thème nous a permis de comprendre que l'utilisation de chacune de ces méthodes n'avait de sens que dans des contextes bien précis, essentiellement déterminés par la nature et la densité des données disponibles sur le cas d'étude considéré.

Dans le cadre de ce travail de thèse, le contexte dans lequel nous nous situons est celui d'un réservoir de stockage souterrain de gaz naturel de type aquifère exploité par la compagnie Gaz-de-France. Nous présenterons donc la nature des informations dont nous disposons, ce qui nous permettra de définir les enjeux de la construction du modèle géologique.

Nous procéderons alors à une présentation complète des nouvelles méthodes de modélisation que nous avons mises en place pour parvenir aux objectifs fixés.

Ces méthodes, dont le champ d'application déborde souvent du cadre de la modélisation des

réservoirs de type fluviatile, se regroupent en trois thèmes principaux:

- La modélisation des séquences stratigraphiques et la caractérisation géométrique de la notion de couche géologique, qui a donné lieu à la création d'un nouvel objet dans la base de données des objets Gocad: le *gstack* («geological stack»).

- La mise en place de nouvelles fonctionnalités qui permettent de convertir le résultat de calculs effectués sur des maillage irréguliers (surfaces triangulées, solides tétraédrisés, etc...) sous forme de grille régulière. En effet, en géologie, certains types de maillage sont mieux appropriés pour certaines méthodes et le besoin d'un support de travail commun s'était très vite manifesté.

- La mise au point d'une nouvelle méthode d'interpolation, qui est une extension de la méthode D.S.I (Discrete Smooth Interpolation) ([10],[36],[37]), et qui permet de tenir compte de considérations d'anisotropie, ce qui est un enjeu important dans la modélisation des réservoirs fluviatiles.

L'application de ces nouvelles techniques sera illustrée à la fin de cette thèse par une description complète de la méthodologie employée pour la construction du modèle géologique du réservoir de stockage sur lequel nous avons travaillé. Cette description sera suivie d'une présentation des résultats obtenus.

Chapitre 1

Synthèse bibliographique

1.1 Description géologique des dépôts de type fluviatile

1.1.1 Introduction: le système fluvial

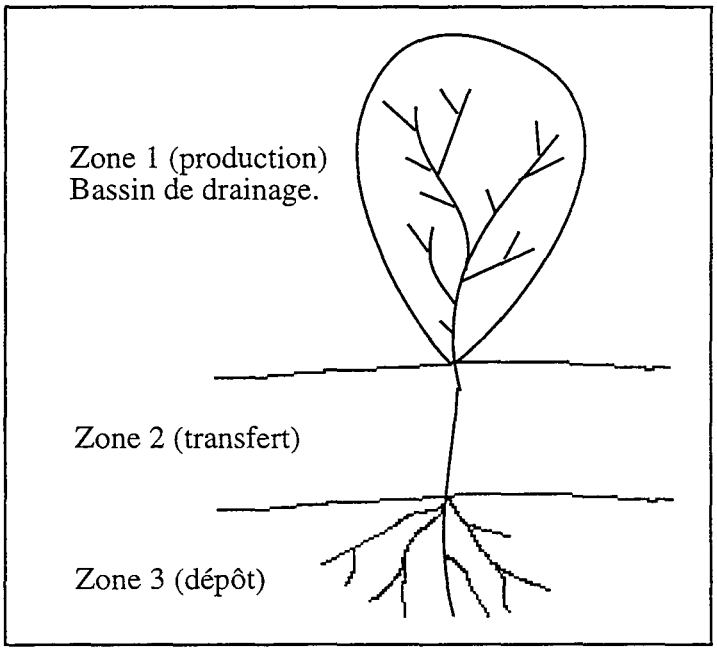


FIGURE 1. Le système fluvial.

Comme illustré sur la Figure 1, les dépôts de type fluviatile ([44]) proviennent des bassins de drainage des cours d'eau. La production de ces bassins est controlée par des facteurs climatiques (niveau de précipitations), des facteurs diastrophiques (la déformation de la croûte terrestre) et la capacité d'érosion des matériaux en place.

Les matériaux érodés sont ensuite transférés par les cours d'eau vers une zone de dépôt, dont l'activité varie en fonction de la subsidence et du niveau eustatique. Les dépôts de type fluviatile se caractérisent par le fait que leur zone de dépôt est située en amont des talus continentaux et des zones de deltas.

1.1.2 Les éléments de classification des dépôts fluviatiles

La classification la plus répandue ([15],[43]) concernant ces types de dépôt distingue essentiellement quatre familles qui diffèrent par deux caractéristiques: la sinuosité des formes de dépôt et leur multiplicité, c'est à dire le fait que celles-ci soient disposées sous forme de chenaux individualisés ou qu'elles forment des réseaux interconnectés plus ou moins complexes.

Ces quatre familles sont les suivantes:

1. les fans alluviaux, plutôt rectilignes et très individualisés.
2. les méandres, relativemant sinueux et bien individualisés.
3. les réseaux en tresse, plutôt rectilignes et très ramifiés.
4. les systèmes anastomosés, très sinueux et très ramifiés.

Les méandres et les réseaux en tresse sont les formes de dépôts les plus souvent rencontrées.

1.1.3 Les systèmes de type méandre

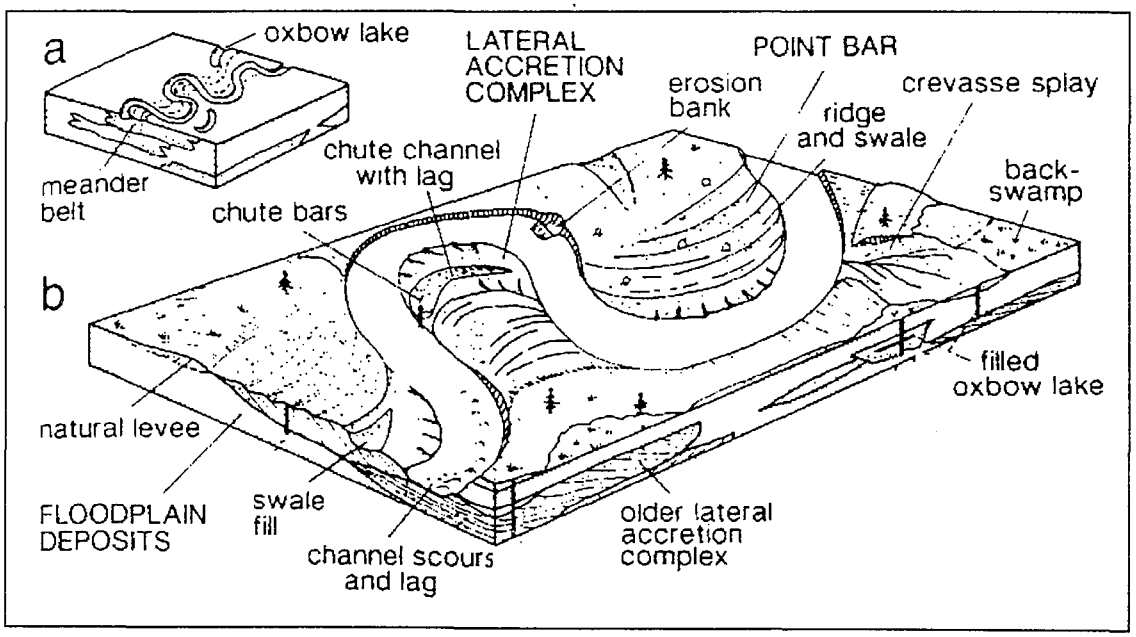


FIGURE 2. Les systèmes de type méandre.

Ces systèmes se présentent sous la forme de chenaux étroits, mais de sinuosité élevée. La proportion en sable des sédiments va de 20 à 40%. S'ils sont associés à une large plaine d'innondation, leur étendue est souvent restreinte à une zone appelée ceinture de méandre.

On distingue un certain nombre de formes de sédimentation dans ce type de dépôt:

1. Le plancher du chenal (channel floor): Celui-ci est généralement composé de sédiments grossiers déposés au cours de périodes de pic d'écoulement, et se présente sous la forme de barres allongées.
2. les barres de méandre (point bars): Elles s'accumulent dans les parties internes des courbures des rivières, alors que les matériaux des rives externes sont érodés. Tout ceci tend à accroître la sinuosité des cours d'eau, jusqu'à ce que ceux-ci se court-circuitent et forment ainsi des segments de chenaux abandonnés. Les ensembles ainsi formés sont souvent regroupés sous le terme de «complexe d'accrétion latérale» ([11]), et produisent des dépôts à dominante sableuse avec un granoclassement positif.
3. les bouchons argileux (clay plugs): Ils sont le produit du remplissage des segments de méandres abandonnés par des matériaux argileux issus de la plaine d'innondation avoisinante. Le mécanisme de remplissage de ces chenaux ainsi que celui du dépôt par accrétion latérale sont illustrés dans la Figure 3.

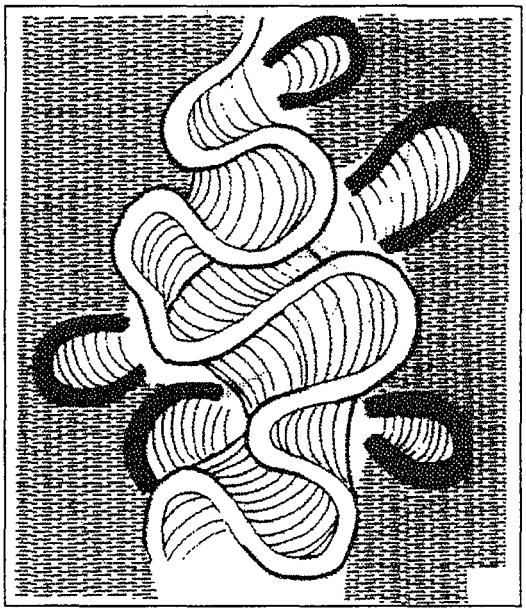


FIGURE 3. Dépôt par accrétion latérale et bouchons argileux.

4. les épanchements de crevasse et levées (crevasse splays and levees): Ces structures se forment dans un contexte d'écoulement modéré, où des dépôts sableux s'accumulent sur les flancs des rivières, formant ainsi des levées. Ces levées, qui constituent alors des sortes de digues naturelles, peuvent se voir détruites par le courant de la rivière. Des épanchements de crevasse se répandent alors dans la plaine d'innondation avoisinante.
5. les dépôts de plaine d'innondation (floodplain deposits): Ils s'accumulent lors de phases d'innondation sporadiques, et se présentent sous forme de laminations de taille millimétrique de matériaux très fins (siltes et boues).

1.1.4 Les dépôts de type réseau en tresse

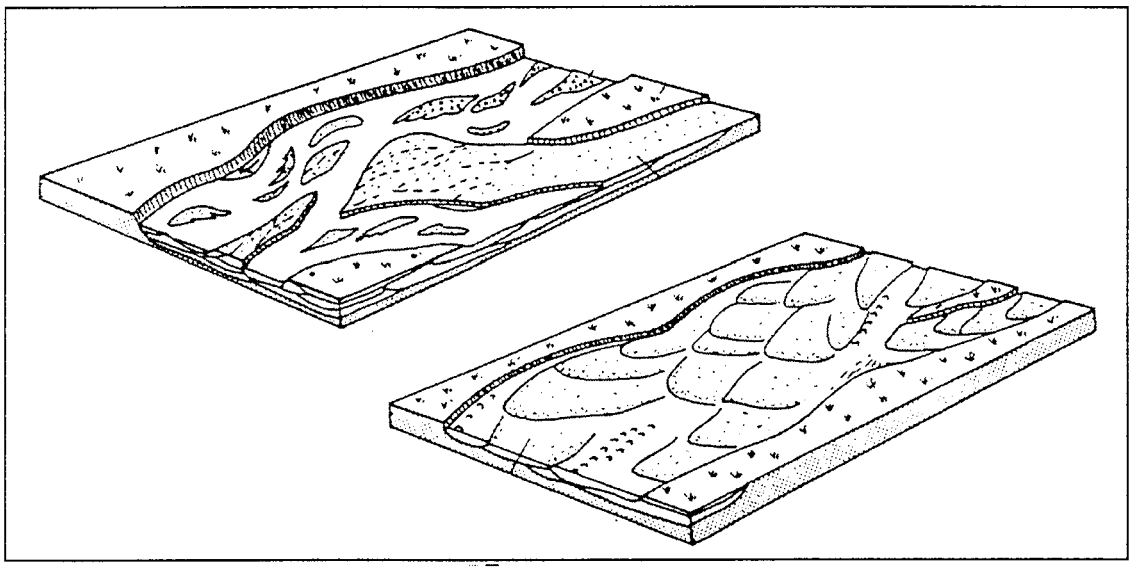


FIGURE 4. Les réseaux en tresse

Les réseaux en tresse, contrairement aux méandres, possèdent une sinuosité réduite. Ils sont constitués de plusieurs chenaux interconnectés séparés par des barres et des îles. Ils occupent ainsi un lit de largeur importante et de faible profondeur.

Ces systèmes se développent près de zones de fort relief, qui délivrent des quantités relativement importantes de débris, de gravelles et de sables. Le faible pourcentage en particules fines (généralement sources de cohésion pour les sédiments) rencontré dans ce type de dépôt expliquerait la mobilité et la capacité à se diviser des chenaux.

Les différents chenaux peuvent être de taille variable, mais l'ensemble possède toujours un rapport largeur/épaisseur très élevé (de 50:1 à 500:1).

Le mécanisme de dépôt le plus fréquent est l'accrétion latérale, mais le retravail continuel des matériaux déposés à l'intérieur de la ceinture des chenaux et le déplacement de ces chenaux génèrent des structures sédimentaires complexes au sein des corps sableux.

1.1.5 Les autres modes de dépôt: fans alluviaux et réseaux anastomosés

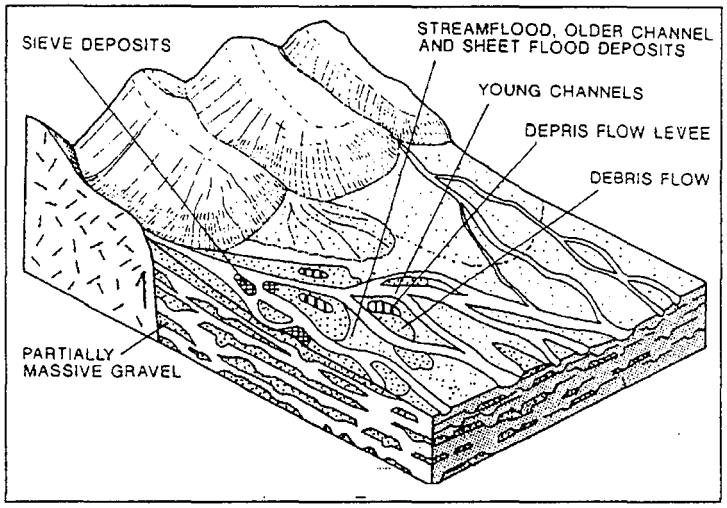


FIGURE 5. Les fans alluviaux.

Les fans alluviaux sont des amas de sédiments en forme de cône qui se forment dans les zones de piémont au débouché de vallées étroites dans les zones de plaine (voir Figure 5).

La taille de la plupart des fans alluviaux est de l'ordre du kilomètre mais peut être parfois supérieure à 50 kilomètres.

Ils sont composés de deux types de dépôt:

- Des dépôts de courant qui sont le faciès prédominant de ces structures.
- Des dépôts gravitaires, aussi regroupés sous le terme de «debris flow», qui sont des matériaux de type grossier.

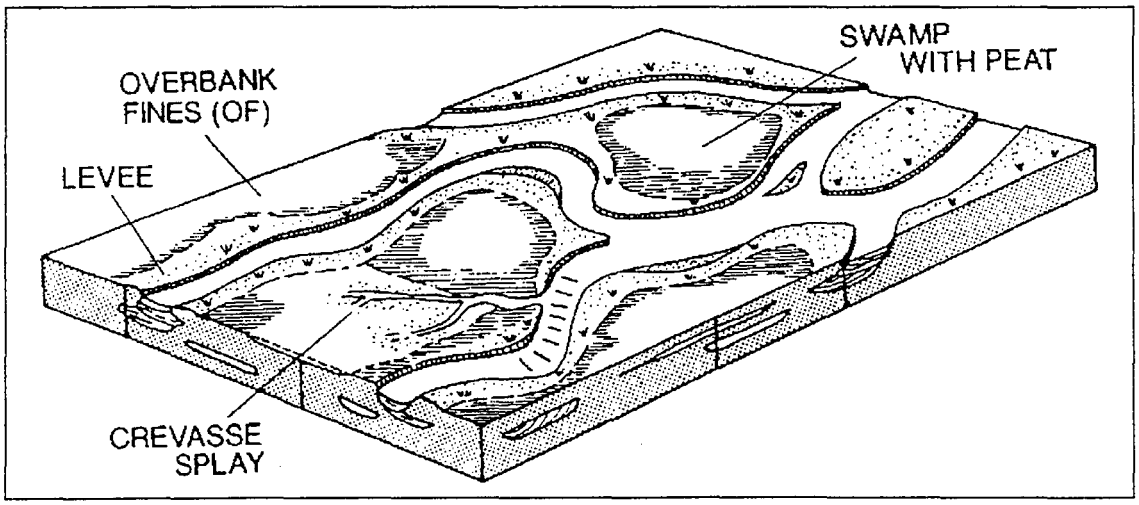


FIGURE 6. Les réseaux anastomosés.

Les réseaux anastomosés se présentent sous la forme d'un réseau interconnecté de chenaux

relativement sinueux, et résultent de l'élévation du niveau de dépôt dans les parties les plus en aval de la zone de sédimentation des rivières. Il en résulte la formation par aggradation verticale dans des zones de faible pente de rubans sableux relativement étroits et épais.

1.2 Caractérisation mathématique des dépôts de type fluviatile

La caractérisation des dépôts fluviatiles anciens se fait souvent à partir de la connaissance des systèmes actuels. Un certain nombre de paramètres morphologiques et pétrophysiques permettent de caractériser ces dépôts.

La direction d'alongement principal

En premier lieu, le premier paramètre le plus souvent pris en compte est la direction d'alongement principal des corps chenalisés. Cette direction est plus facile à déterminer dans le cas des réseaux en tresse qui sont relativement rectiligne que des méandres qui sont eux beaucoup plus sinueux.

Le rapport largeur/épaisseur

En second lieu, on considère également le rapport largeur/épaisseur (width/depth ratio) des objets. Ce paramètre intervient dans la plupart des systèmes de classification car il est très caractéristique des différentes familles de formes de sédimentation que nous avons vues précédemment. Ainsi, le rapport largeur/épaisseur des méandres est très inférieur à celui des réseaux en tresse, qui peut prendre des valeurs allant de 50:1 à 500:1.

La sinuosité

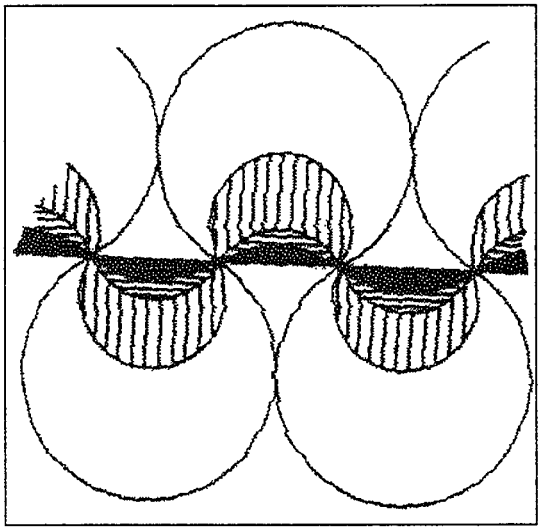


FIGURE 7. Définition de la sinuosité des chenaux.

La sinuosité ([33]) des chenaux se définit comme le rapport de la longueur des chenaux par la longueur de la vallée dans laquelle ceux-ci se développent. Un chenal rectiligne aura donc une sinuosité égale à 1. Afin d'illustrer cette notion, nous avons représenté dans la Figure 7

quatre formes de chenaux idéales correspondant à des valeurs de sinuosité de 1.07, 1.35, 2.12 et 5.24, ce qui correspond à des variations angulaires respectives de 75, 150, 225 et 300 degrés.

La sinuosité est un élément essentiel de classification de corps chenalisés. En effet, le terme de méandre ne peut s'appliquer que pour des formes de sinuosité supérieure à 1.5. A l'inverse, la sinuosité des systèmes de type réseau en tresse est beaucoup plus faible, et se situe généralement entre 1.1 et 1.2 .

La multiplicité

Dans les systèmes fluviatiles à chenaux multiples, la multiplicité des chenaux se définit comme le nombre moyen d'ilôts (aussi désignés sous le terme de tresses) rencontrés sur une distance égale à la longueur d'onde moyenne du cours d'eau. Trois systèmes à chenaux multiples sont représentés dans la Figure 8, dont la multiplicité est respectivement égale à 0.5, 2 et 6.

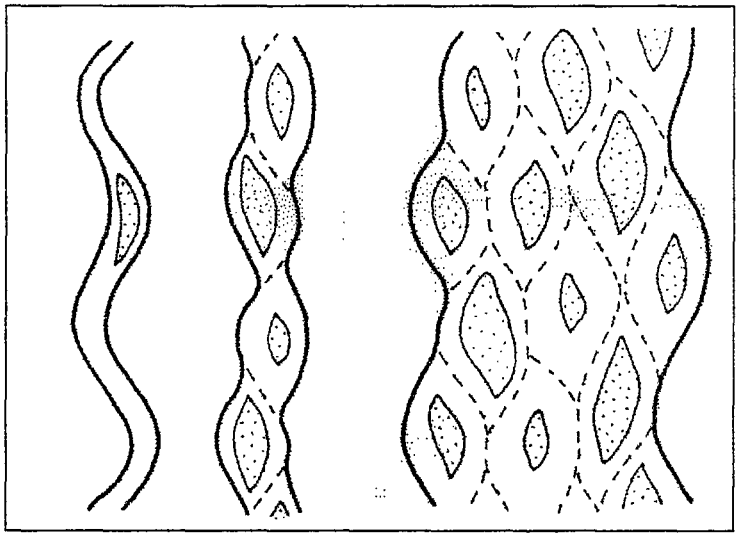


FIGURE 8. Définition de la multiplicité des chenaux.

Ce paramètre, s'il est très important pour classifier les systèmes fluviatiles actuels, n'est pas d'un grand intérêt pour caractériser les sédiments chenalisés, car son estimation est très difficile à réaliser dans ce cas.

Les paramètres pétrophysiques

Le pourcentage en silt-argile (silt-clay percentage) et le rapport sable/fines (sand/shale ratio) sont des caractéristiques essentielles des dépôts fluviatiles. Le pourcentage en silt-argile rend compte de l'importance des fines dans l'ensemble des dépôts. Le rapport sable/fines rend compte de l'importance relative des sédiments grossiers (les sables) par rapport aux sédiments à granulométrie faible (les fines).

Ces deux quantités traduisent la même idée de quantifier de manière globale la granulomé-

trie des dépôts et sont reliés par la relation suivante:

$$\frac{100 - P_{sa}}{P_{sa}} = \frac{S_a}{S_h}$$

où P_{sa} désigne le pourcentage en silt/argiles et S_a/S_h le rapport sables/fines.

Un autre paramètre qui est fréquemment employé pour décrire les dépôts fluviatiles est la porosité des matériaux sableux.

Certains travaux portent également sur l'étude de la distribution des intercalations argileuses dans les dépôts fluviatiles.

D'autres paramètres reviennent également fréquemment dans la littérature, qui n'ont cependant d'intérêt que dans le cadre de l'étude des milieux de dépôt actuels. On peut citer à titre d'exemple la vitesse du courant, la pente moyenne du milieu de dépôt, le débit annuel moyen des cours d'eaux, etc...

Outre son intérêt descriptif, l'étude des milieux de dépôt actuels n'a cependant pas qu'un intérêt descriptif car elle a permis d'établir des relations mathématiques ([17]) entre les différents paramètres présentés ci-dessus. Ces relations, qui sont avant tout des lois statistiques obtenues par régression linéaire, ne sont bien entendu pas à prendre au pied de la lettre mais montrent qu'il existe des corrélations très importantes entre tous ces paramètres.

	Approche 1	Approche 2
Pourcentage en siltes-argiles (%). (M)	M (calculé)	
Rapport largeur/épaisseur (F)	$F = 255*M^{1.08}$	F = W/D (width/depth) (mesurées)
Sinuosité (P)	$P = 0.94*M^{-0.25}$	$P = 3.5 * F^{0.27}$
Débit annuel moyen (cfs) (Qm)	$Qm^{0.38} = W*M^{0.39}/37$	$Qm = W^{2.43}/(18*F^{1.13})$
Pente du chenal (ft/mi) (S)	$S = 60*M^{1.08}*Qm^{0.32}$	$S = 30*(F^{0.95}/W^{0.98})$
Longueur d'onde des méandres (ft). (L)	$L = 1890*Qm^{0.34}/M^{0.74}$	$L = 18*(F^{0.53}W^{0.69})$

FIGURE 9. Relations statistiques entre les différents paramètres des chenaux.

On distingue ici deux approches. Dans la première, les différents paramètres sont obtenus à partir de l'estimation du pourcentage en silt-argile (M) des dépôts et de la largeur moyenne W

des corps chenalisés. La seconde approche repose sur l'estimation préalable du rapport largeur/ épaisseur, noté F.

1.3 Revue des différentes méthodes de modélisation des réservoirs de type fluviatile

1.3.1 Introduction

Il existe plusieurs méthodes ([14],[23]) de modélisation des réservoirs de type fluviatile, faisant appel à de nombreuses disciplines mathématiques. Le plus souvent, c'est la densité et la nature des données disponibles qui décident de la méthode qui va être employée pour modéliser un réservoir.

En effet, il faut savoir que, dans la majorité des cas, l'espacement moyen entre les puits sur une zone d'étude ne permet pas de prédire avec exactitude l'architecture d'un réservoir, ce qui justifie l'usage de méthodes probabilistes ou stochastiques. Ces méthodes peuvent générer une multitude de réalisations équiprobables d'un modèle de réservoir. Toutes ces réalisations honnorent les données disponibles, et sont validées ou non par la suite à l'aide des mesures d'exploitation des réservoirs.

Il existe deux familles de méthodes stochastiques: les méthodes booléennes ou méthodes à base d'objets, et les méthodes pixel ou méthodes de simulation continues.

Nous présenterons également à la fin de ce chapitre une revue sommaire des méthodes déterministes qui sont utilisées pour la modélisation des réservoirs pétroliers.

1.3.2 Les méthodes de simulation à base d'objets ou méthodes booléennes

Ces méthodes ([8],[24]) sont utilisées dès les premières étapes de l'étude d'un réservoir lorsque les données sont relativement rares.

Elles visent, comme leur nom l'indique, à décrire un réservoir sous la forme d'un ensemble d'objets dont il est possible de définir plusieurs familles. Dans le cas de réservoirs fluviatiles, les objets en question sont des corps sableux, des chenaux, des épanchements de crevasse, des intercallations argileuses ou encore des failles.

Le peu de données disponible permet néanmois d'établir des lois statistiques sur la distribution, l'orientation et les dimensions des objets considérés.

Ces lois statistiques sont le point de départ des modeleurs stochastiques booléens qui, à partir de là, vont générer des distributions de corps sableux de forme aléatoire à des emplacements aléatoires.

Il est à noter que ces méthodes peuvent intégrer des connaissances géologiques extérieures sur la forme des objets et leur distribution, qui seraient par exemple issues de l'étude d'affleurements correspondant à des milieux analogues de dépôt.

Ce type de modélisation s'effectue généralement en deux étapes:

1. La reconstitution des corps sableux interceptés aux puits à partir des lois statistiques sur leur orientation, leur rapport largeur/épaisseur, etc...
2. La génération des corps sableux entre les puits suivant les mêmes lois statistiques, jusqu'à ce que l'on atteigne une proportion en sable globale donnée, qui aura été estimée au préalable à partir des puits.

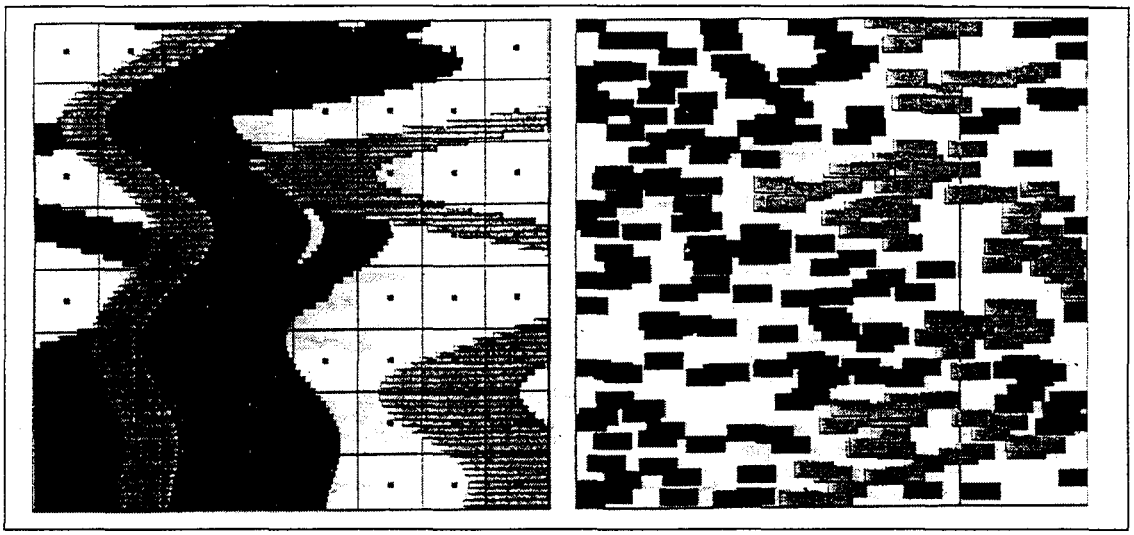


FIGURE 10. Exemple de modèle généré par une simulation à base d'objets (vu d'en haut, puis vu en coupe).

1.3.3 Les méthodes de simulation continues ou méthodes pixel

Ces méthodes sont utilisées lorsqu'une modélisation plus détaillée des corps sableux est rendue possible par une densité de puits plus importante que dans le cas des méthodes booléennes, sans que celle-ci ne permette de modéliser les objets de manière déterministe.

On distingue généralement trois familles de méthodes de simulation continue: les simulations conditionnelles, les méthodes fractales et les méthodes de Markov.

Les simulations conditionnelles

Ces méthodes ([45]) sont utilisées sur des paramètres continus (perméabilités, porosités) mais aussi sur des variables discrètes telles que des fonctions indicatrices de faciès.

La première étape de la simulation consiste à transformer la variable à modéliser sous la forme d'une variable aléatoire gaussienne normalisée, sur laquelle s'effectuera la simulation conditionnelle.

Une méthode de simulation conditionnelle très employée prend pour principe que la variable Y à estimer est la somme de deux quantités:

$$Y(x, y) = Y_k(x, y) + R(x, y)$$

où $Y_k(x, y)$ est le résultat de l'estimation de la variable Y par un krigeage simple et R(x,y) un terme résiduel aléatoire gaussien obtenu par tirage aléatoire, à partir d'une loi gaussienne

normale indépendante de Y, de variance égale à la variance de la variable $Y_k(x, y)$.

Les méthodes fractales

Ces méthodes ([26]) peuvent être utilisées pour générer des distributions 2D de porosités/ perméabilités, et débutent par une analyse du rapport R/S, qui permet de calculer une dimension fractale à partir des données verticales issues de chaque puits.

La méthode prend ensuite pour principe que la dimension fractale horizontale est la même que la dimension verticale calculée, et calcule une dimension fractale moyenne à partir des dimensions fractales obtenues sur chaque puits.

La génération des porosités ou des perméabilités se fait ligne par ligne de manière dichotomique au niveau des points milieux entre deux puits, auxquels sont affectés des valeurs qui, comme dans le cas des simulations conditionnelles, sont la somme de deux quantités:

- un terme qui est la moyenne pondérée des valeurs des deux puits.
- un terme résiduel qui est obtenu par tirage à partir d'une loi gaussienne, dont la variance est calculée à partir de la dimension fractale moyenne déterminée précédemment.

Cette nouvelle valeur est alors considérée comme une donnée et permet de générer deux nouvelles valeurs par la même méthode, et ainsi de suite...

Les méthodes de Markov

Ces méthodes ([12]) fonctionnent sur des variables discrètes (des indices de faciès par exemple) et reposent sur les probabilités conditionnelles.

Elles supposent deux choses:

- l'adoption d'une loi de probabilité conditionnelle qui exprime la probabilité qu'un nœud ait une valeur donnée connaissant les valeurs de ses voisins.
- l'adoption d'une loi de probabilité combinée qui exprime la probabilité que n nœuds de grilles prennent n valeurs (x_1, ..., x_n).

L'objectif des méthodes de Markov est de générer des distributions de faciès qui maximisent cette probabilité combinée. Pour cela, on procède en deux étapes:

- la génération d'une distribution alétoire d'indices de faciès, qui respecte les proportions respectives de chaque famille de faciès.
- l'optimisation de cette solution initiale par un algorithme de type Metropolis vers une solution qui maximise la loi de probabilité combinée.

Ces méthodes sont intéressantes d'un point de vue théorique car elles sont plus facilement rattachables à des concepts géologiques que les modèles à base de variogrammes ou les modèles fractals, mais posent un problème de taille, car elles ne permettent pas jusqu'à présent d'honnorer les données de puits.

1.3.4 Les méthodes déterministes

Dans certains contextes assez spécifiques, tels que l'exploration minière ou sur certains réservoirs pétroliers en phase de production, il est possible que les données de puits permettent de construire des modèles déterministes.

Par ailleurs, la construction de modèles géométriques à partir de données sismiques interprétées, fait par définition appel à des méthodes déterministes.

On distingue trois familles de méthodes déterministes:

- Les méthodes issues des techniques de C.A.O ([4],[18],[19],[20],[38],[39]) utilisées dans l'industrie mécanique, qui sont essentiellement basées sur des interpolations polynomiales de type Bézier, Bsplines et NURBS. Ces méthodes permettent de générer des modèles surfaciques de bonne qualité dans des cas de géométries relativement simples mais s'avèrent particulièrement inadaptées pour des géométries complexes telle que les surfaces faillées.

- Les méthodes issues de la géostatistique ([28],[29]), c'est à dire essentiellement le krigeage et ses dérivés, qui sont très fréquemment utilisées. Les seuls problèmes rencontrés sont liés aux limites en terme de taille des jeux de données utilisables et à l'impossibilité de traiter le problème des discontinuités dans les modèles générés.

- Les techniques récentes dérivées de la méthode D.S.I, qui connaissent un succès grandissant et sur lesquelles nous reviendrons à de fréquentes reprises tout au long de cette thèse, qui s'inscrit dans ce cadre.

1.4 Les enjeux de la modélisation des réservoirs dans le contexte du site de stockage en aquifère

Comme nous venons de le voir, c'est avant tout la distribution, la nature et la densité des données qui détermine le choix de telle ou telle méthode pour construire un modèle géologique de réservoir.

En ce qui concerne le cas du réservoir de stockage en aquifère qui constitue notre support de travail, nous disposons de données de plusieurs types:

- un ensemble d'une trentaine de puits répartis régulièrement sur une zone d'étude d'environ 8*8 kilomètres. Tous ces puits ont été décrits sous la forme d'une séquence de faciès.

- Un ensemble de données sismiques 2D et 3D qui ont permis d'établir des cartes de certains horizons de la séquence.

- Enfin, une étude sédimentologique a permis d'effectuer des corrélations relativement fiables entre les puits et de décomposer la séquence du réservoir sous la forme d'une succession d'unités indépendantes. Chaque unité a pu être identifiée comme appartenant à un type de dépôt donné (méandre ou réseau en tresse) et les paramètres descriptifs de chacunes de ces unités ont pu être estimés (direction d'alongement principal, sinuosité, rapport largeur/ épaisseur, etc...).

Dans le cas des réseaux en tresse, qui sont généralement des objets de taille kilométrique, l'emploi de méthodes déterministes paraît tout à fait envisageable. Dans le cas des méandres, qui sont de taille plus réduite et qui sont plus complexes dans leur structure, mon opinion est

moins tranchée et seuls des tests nous permettront de répondre à cette question.

Le résultat de ces tests sera exposé dans le chapitre 5 de cette thèse dans lequel est détaillée la méthodologie que nous avons choisie pour modéliser le site de stockage.

Les trois chapitres qui vont suivre sont consacrés aux méthodes que nous avons mises en place pour construire un modèle qui permette d'intégrer l'ensemble des données que nous venons de présenter plus haut. En effet, cette capacité à tenir compte de plusieurs sources d'informations constitue, à mon sens, le critère essentiel de qualité d'un modèle.

Chapitre 2

Modélisation de séquences stratigraphiques: l'objet *gstack*

2.1 Introduction

De nombreux dispositifs sédimentaires se présentent sous la forme d'un empilement de couches géologiques partageant une «histoire tectonique» commune. La Figure 11 montre quelques exemples de ce type de structure.

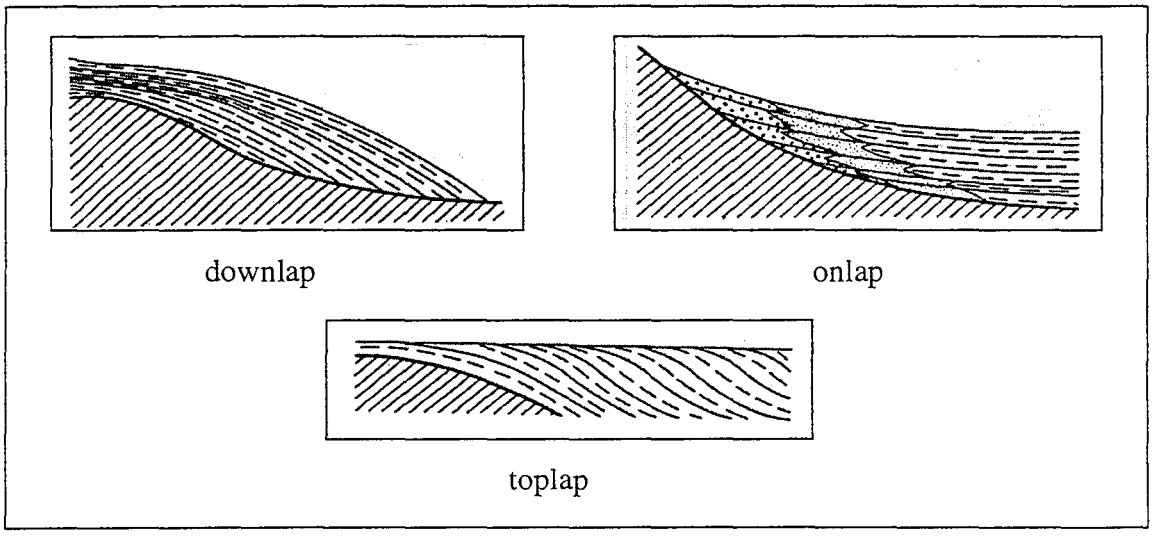


FIGURE 11. Exemples de dispositifs sédimentaires.

L'agencement des couches les unes par rapport aux autres dans de tels dispositifs est contrôlé par des paramètres sédimentologiques, tels que la variation du niveau marin, la pente du talus sur lequel se déposent les sédiments, etc...

La manière la plus flexible de modéliser des objets surfaciques en géologie se fait par l'utili-

sation des surfaces triangulées ([27],[31],[35],[52]). Il est donc logique de modéliser des empilements de couches géologiques tels que ceux que nous venons de voir sous la forme d'empilements de surfaces triangulées. Ces surfaces partageant la même topologie, il ne paraît pas utile de stocker en plusieurs exemplaires cette information pour chacune des couches.

Ces considérations nous ont amené à envisager la création d'un nouvel objet: le *gstack*.

Il nous a également semblé intéressant de concevoir de nouveaux mécanismes d'interpolation interactifs, qui permettent de paramétriser de manière relativement simple ces types de dispositifs sédimentaires.

Les problèmes liés à la géométrie de l'objet étant résolus, nous verrons ensuite que le *gstack* est non seulement un objet surfacique, mais aussi un objet volumique. Nous verrons en effet qu'il est possible de caractériser le volume délimité par cet objet sous la forme d'un ensemble de tétraèdres.

2.2 Description de l'objet gstack

2.2.1 Notion de surface de référence

Une surface triangulée est constituée, comme tout maillage irrégulier, de deux types d'informations:

- des informations géométriques, dans lesquelles sont stockées les coordonnés des nœuds du maillage.
- des informations topologiques, dans lesquelles sont stockées les informations relatives aux connexions entre les nœuds du maillage. Chaque nœud possède ainsi un voisinage, composé de l'ensemble des nœuds du maillage auquels il est connecté. Dans le cas des surfaces triangulées, ces connexions s'organisent de telle sorte que l'objet peut être décomposé en triangles élémentaires, qui s'agencent les uns aux autres par leurs arètes.

Dans le cas de l'objet *gstack*, ces informations topologiques sont stockées sur un niveau particulier de l'objet, que nous appelons le niveau de référence, ou la surface de référence.

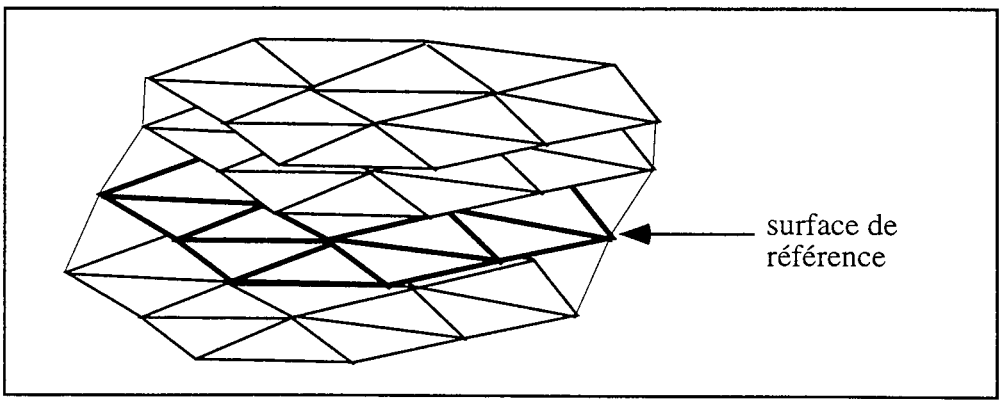


FIGURE 12. Notion de *gstack*.

Cette surface de référence possède toutes les caractéristiques des surfaces triangulées utili-

sées dans le cadre du projet Gocad. Il est notamment possible de définir deux types de nœuds:

- Les nœuds de contrôle dont la position est fixe et qui correspondent à une connaissance sur la géométrie de certains points de la surface.

- Les nœuds libres dont la position peut varier dans l'espace et sera déterminée par interpolation des positions des nœuds de contrôle.

Il est également possible d'utiliser sur cette surface tous les outils d'interpolation disponibles au sein du modeleur Gocad, à savoir:

- La possibilité d'ajuster la géométrie de la surface à un ensemble de points extérieurs à cette surface.

- La possibilité de contrôler le déplacement de ses bords de façon à traiter le problème des surfaces faillées ([1],[2]).

- La possibilité d'estimer la répartition d'une propriété physique sur cette surface par la méthode D.S.I ([34]).

2.2.2 Notion de *pile*

On appelle *pile* l'ensemble des informations liées à un nœud de la surface de référence. .

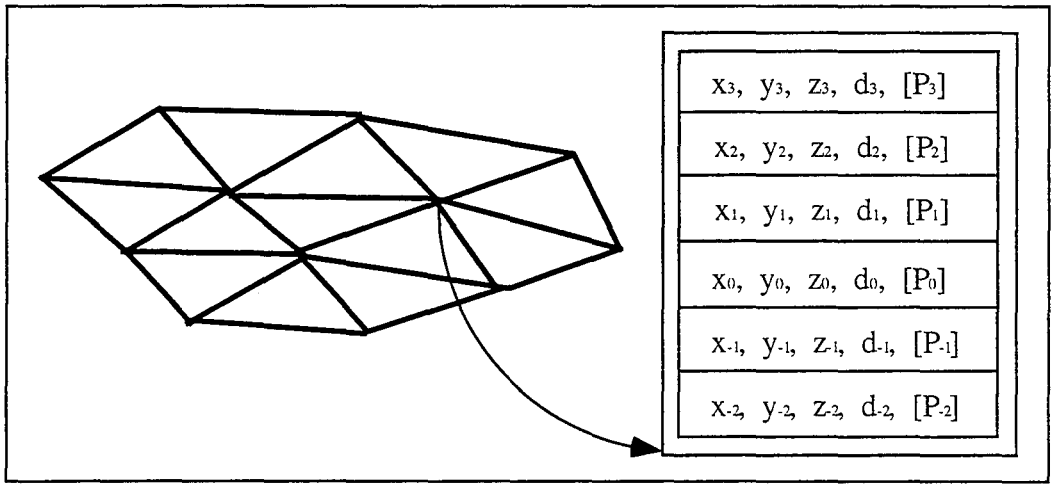


FIGURE 13. Notion de pile.

Ces piles sont des tableaux, de taille égale au nombre de niveaux du *gstack*. Les entrées de ce tableau contiennent les informations suivantes:

- un quadruplet de coordonnées x,y,z,d.

- un tableau de taille variable [P] destiné à stocker les valeurs de propriétés physiques liées à chaque nœud de l'objet.

Les quatre coordonnées x,y,z,d sont destinées à stocker les informations sur la géométrie de l'objet. Le niveau i de cet objet est donc connu à partir des informations topologiques de la surface de référence et des informations géométriques des $i^{èmes}$ entrées des piles de chaque nœud.

Il existe deux modes de stockage de ces informations géométriques: le mode absolu et le

mode relatif.

Codage absolu des informations géométriques

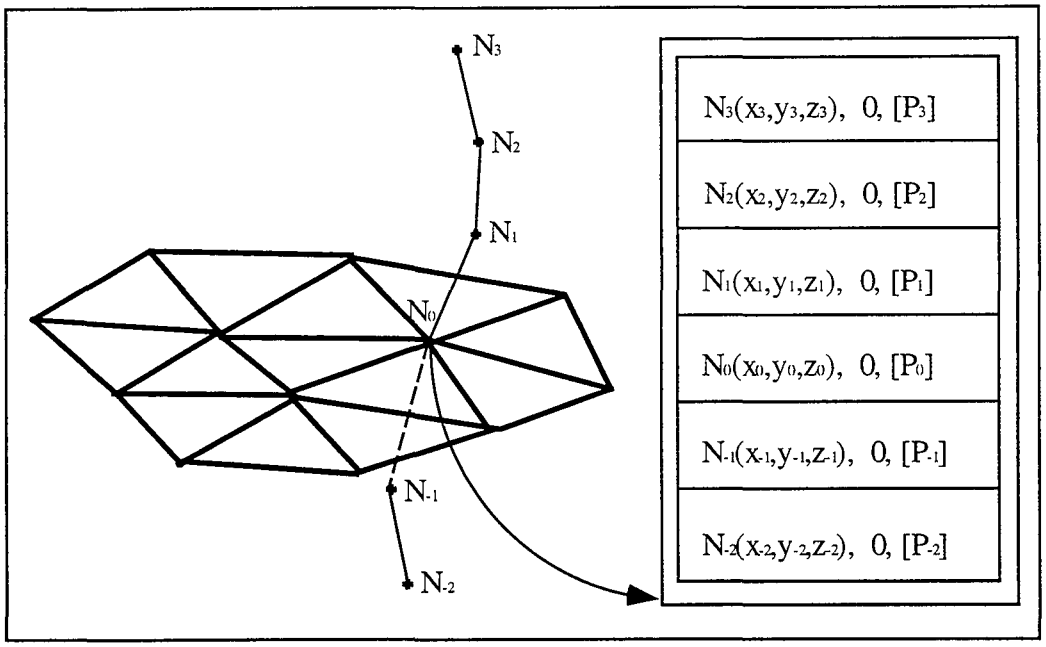


FIGURE 14. Codage absolu des informations géométriques de la pile.

Les coordonnées des nœuds des différents niveaux sont exprimées dans le même repère que ceux de la surface de référence et sont stockées dans les variable x,y et z des entrées de la pile. Le paramètre d n'est pas utilisé.

Codage relatif des informations géométriques

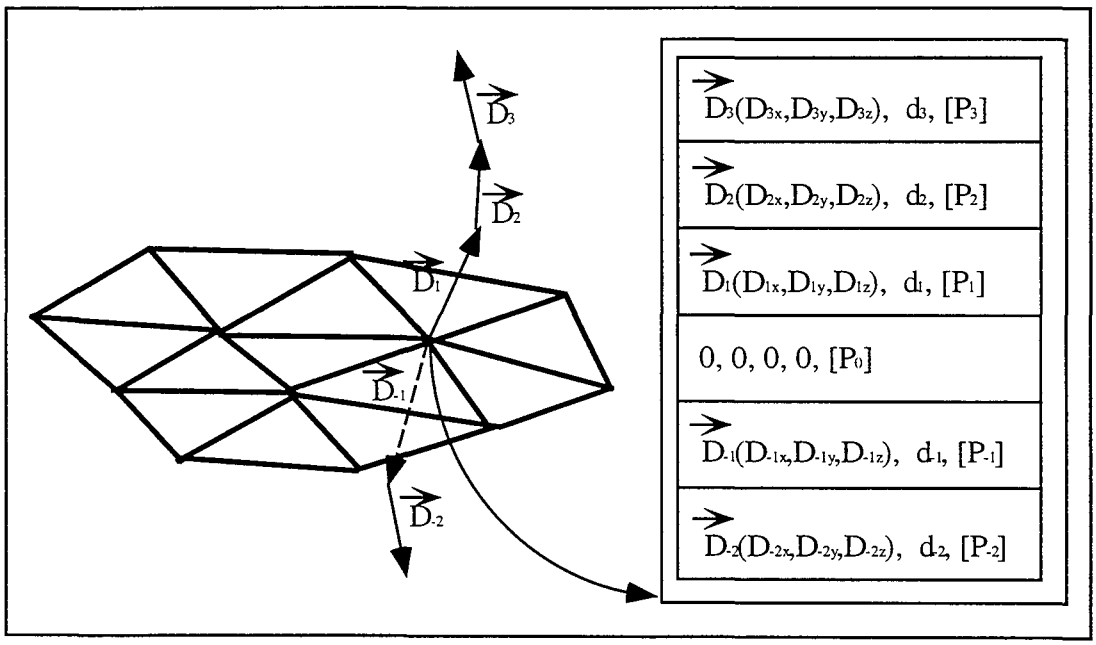


FIGURE 15. Codage relatif des informations géométriques de la pile.

Les positions des nœuds des différents niveaux sont stockés de manière relative par rapport à la position du nœud de la surface de référence. Les informations géométriques stockées à l'entréeE/P courant 1 du tableau correspondant au niveau de référence prennent toutes la valeur 0.

Chacun des vecteurs $\vec{D_i}$ représentés sur la figure est stocké à l'aide de quatre coordonnées:

- les coordonnées Dix, Diy, Diz du vecteur $\vec{D_i}$ normé.

- la norme di du vecteur $\vec{D_i}$.

2.2.3 Fonctionnalités de base associées à l'objet *gstack*

Changement de la surface de référence

La surface de référence d'un *gstack* est un niveau particulier de cet objet qui contient les informations topologiques communes à tous les autres niveaux de cet objet.

Le positionnement du niveau de référence dans la pile des niveaux n'est pas figé et peut être modifié en fonction des besoins de l'utilisateur.

Si les informations des piles sont stockées de manière absolue, cette opération se limite à modifier la variable qui spécifie la position du niveau de référence dans la pile.

Si les informations sont stockées en mode relatif, le changement de niveau de référence suppose de modifier le contenu des piles de façon à exprimer les informations géométriques des piles de manière relative par rapport à ce nouveau niveau de référence.

Passage du système de codage absolu au système relatif et vice-versa

Certaines applications de l'objet gstack, que nous présenterons plus tard, travaillent soit en mode absolu, soit en mode relatif. Il est donc souhaitable de pouvoir passer d'un système de codage à l'autre.

Les deux types de codage étant relativement simples, il ne me semble pas nécessaire d'exposer le détail de ces transformations.

2.3 Interpolation de la géométrie d'un *gstack*

2.3.1 Construction par modélisation séparée de chaque niveau

Tous les niveaux d'un gstack peuvent être choisis comme niveau de référence de cet objet, et héritent donc à ce titre de l'ensemble des fonctionnalités de modification de la géométrie des surfaces triangulées. Ces fonctionnalités, essentiellement liées à l'interpolateur D.S.I, sont mentionnées dans le chapitre 2.2.1 de la page 18.

2.3.2 Construction par interpolation des niveaux

L'intérêt de l'objet *gstack* est qu'il permet de stocker au sein d'un même objet la géométrie de plusieurs surfaces. Il apparaît donc possible de déduire la géométrie de certains niveaux à partir de géométries connues d'autre niveaux.

Pour cela, nous avons introduit la notion de niveaux de contrôle dans le *gstack*. Ces niveaux se caractérisent donc par le fait que leur géométrie est supposée connue et fixée.

Nous avons ensuite mis en place une routine d'interpolation de niveaux. Cette routine est basée sur la méthode d'interpolation D.S.I et est illustrée dans la Figure 16. Elle fonctionne en mode de codage absolu.

Le principe de cette méthode est très simple: tout se passe comme si chaque pile du gstack était une ligne polygonale, dont certains nœuds, qui correspondent aux niveaux de contrôle, seraient eux mêmes des nœuds de contrôle, et dont on modéliserait la géométrie par la méthode D.S.I.

La Figure 16 illustre un exemple de cette application:

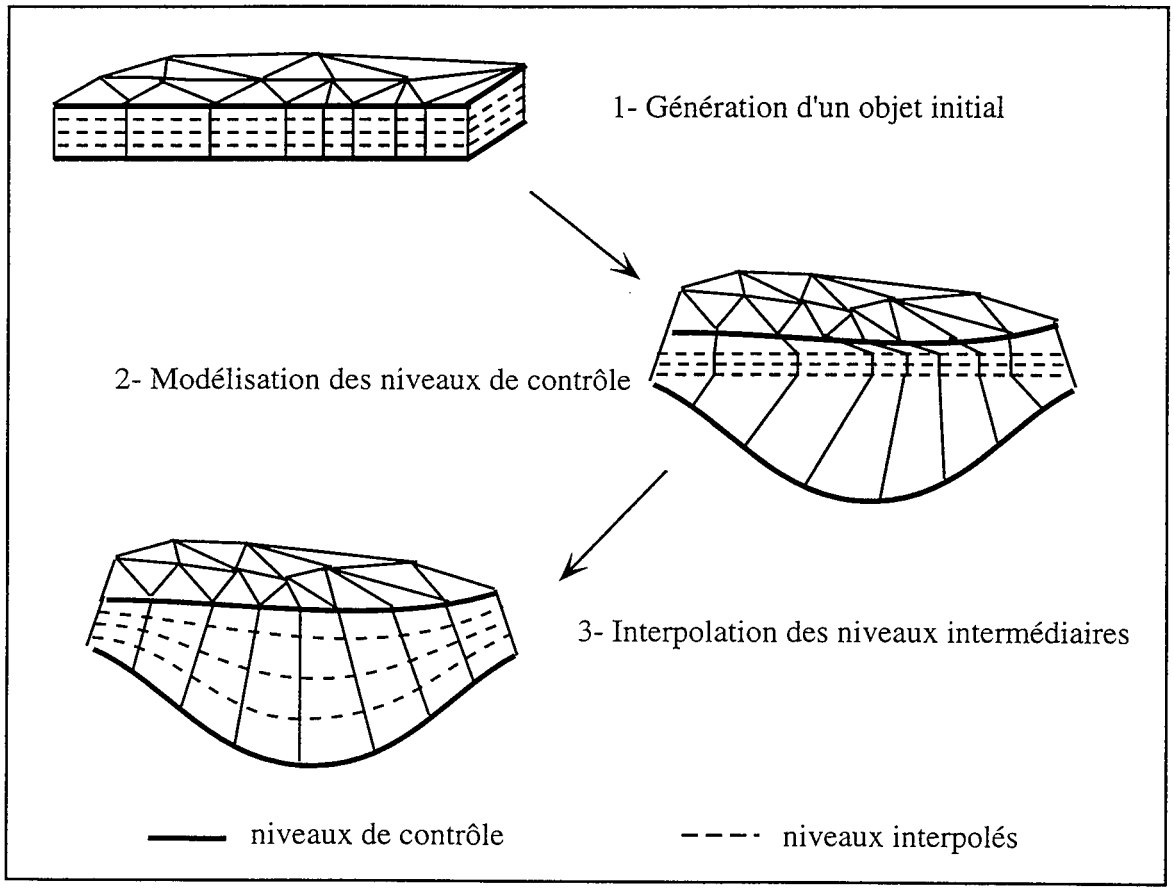


FIGURE 16. Modélisation de l'objet gstack par interpolation de niveaux.

2.3.3 Construction par interpolation des piles

Ce type de construction fonctionne en mode relatif. Il s'agit d'une interpolation classique par D.S.I du contenu des piles de l'objet. Ceci suppose l'introduction de la notion de pile de contrôle, dont la géométrie est connue et fixée avant l'interpolation.

En mode relatif, les vecteurs sont stockés sur quatre composantes, les trois premières correspondant aux coordonnées du vecteur normé, la quatrième étant la norme de ce vecteur.

L'interpolation des piles va donc estimer séparément des vecteurs normés et des normes, ce qui n'est pas équivalent à interpoler des vecteurs de norme quelconque. En effet, d'un point de vue géologique, ce type de calcul revient à estimer séparément le pendage d'une couche et son épaisseur, qui sont deux quantités indépendantes.

La Figure 17 fournit un exemple de ce type d'interpolation:

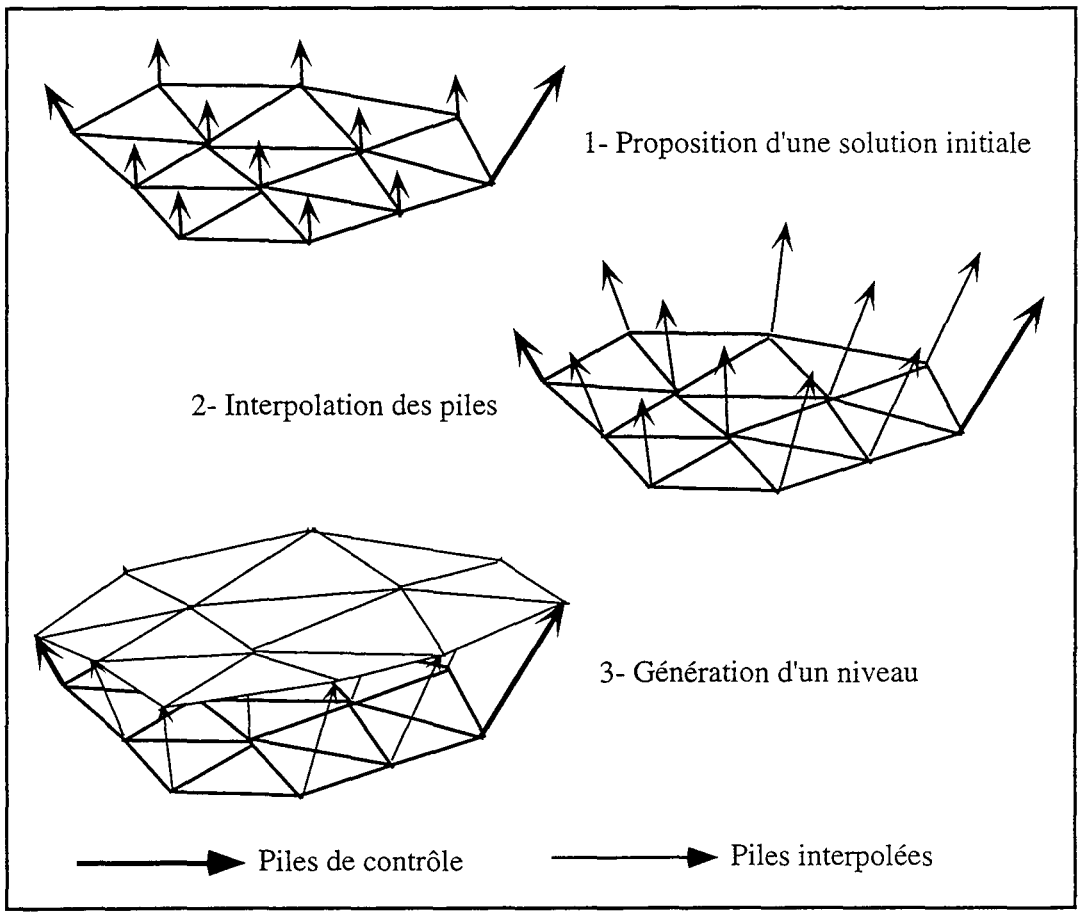


FIGURE 17. Modélisation de l'objet gstack par interpolation des piles.

2.4 Dimension volumique de l'objet *gstack*

2.4.1 Introduction

Comme nous venons de le voir dans le chapitre précédent, le *gstack* est un objet qui permet de déterminer la géométrie de surfaces triangulées par interpolation de géométries de surfaces connues, le résultat final se présentant sous la forme d'une succession de surfaces isomorphes.

Chacun des niveaux de cet objet est un objet surfacique à part entière qui peut être exporté sous la forme d'une surface triangulée Gocad classique, possédant un ensemble de nœuds et un ensemble de triangles propres.

La vocation première du *gstack* apparaît donc comme la possibilité de générer des objets surfaciques. Cependant, le fait que les différents niveaux de cet objet partagent la même topologie et que les informations géométriques de ces niveaux soient stockées au sein d'un seul et même objet confèrent au gstack un caractère d'objet volumique.

Ainsi, si l'on considère deux niveaux successifs d'un *gstack*, ceux-ci délimitent un volume donné de l'espace 3D. Comme l'illustre la Figure 18, le volume total d'un gstack peut donc être vu comme la réunion des volumes compris entre deux niveaux successifs de cet objet, qui eux mêmes peuvent être décomposés en volumes élémentaires à six sommets.

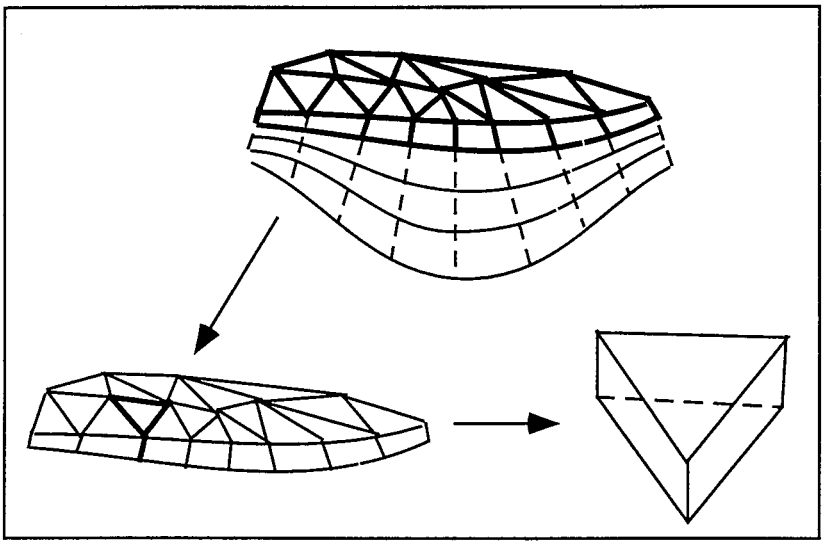


FIGURE 18. Décomposition du volume d'un *gstack* en prismes élémentaires.

Ces volumes élémentaires sont délimités par deux types de faces:

- deux face triangulaires inférieure et supérieure.
- trois faces latérales à quatre sommets.

Si l'on considère que les surfaces latérales de ces volumes peuvent être décomposées en deux triangles, ce volume pourrait être vu comme un octaèdre. Cependant, nous avons préféré le désigner sous le terme de «prisme à base triangulaire», bien que celui-ci ne présente pas toutes les caractéristiques d'un prisme.

La caractérisation mathématique d'un tel volume est relativement complexe. La première partie de ce chapitre sera consacrée à la façon de le décomposer en sous-volumes élémentaires tétraédriques, qui, eux, peuvent être caractérisés mathématiquement de manière simple.

Par la suite, nous verrons que la décomposition automatique en tétraèdres du volume compris entre deux niveaux successifs d'un *gstack* pose des problèmes de consistance de maillage au niveau des faces latérales communes à deux prismes. Nous tenterons d'apporter une solution à ce problème.

En conclusion, nous envisagerons les perspectives offertes par cette décomposition volumi-

que de l'objet *gstack*.

2.4.2 Remplissage de prismes à base triangle par des tètraèdres

Introduction

Le volume délimité par deux niveaux consécutifs d'un gstack peut être décomposé en un ensembles de prismes élémentaires à base triangle (voir Figure 18). La caractérisation analytique de tels volumes est relativement complexe à réaliser. Ceci nous a donc amené à considérer un niveau de décomposition supérieur, dans lequel le volume de chaque prisme est considéré comme l'union de volumes élémentaires tétraédriques. Une telle décomposition des prismes en tétraèdres soulève un certain nombre de difficultés, auxquelles nous allons tenter d'apporter des réponses.

Formalisme du problème

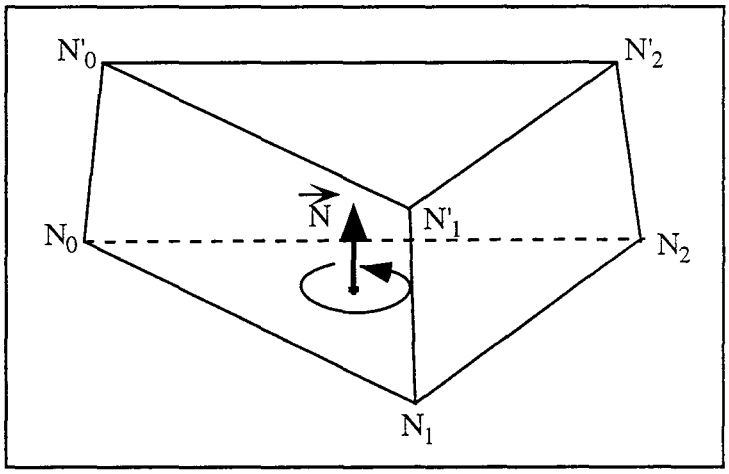


FIGURE 19. Notion de prisme à base triangulaire.

Un prisme à base triangulaire peut être vu de la manière suivante:

- Un élément de base triangulaire T composé de trois nœuds N0, N1 et N2 numérotés de 0 à 2. Cette numérotation définit un sens de parcours des sommets du triangle commun à tous les triangles d'un même objet (cette idée est illustrée par la Figure 31 de la page 63). Elle permet également de définir le sens du vecteur normal au triangle (N0N1N2) comme le produit vectoriel des vecteurs $\overrightarrow{N_0N_1}$ et $\overrightarrow{N_0N_2}$. N0, N1 et N2 définissent ainsi la face triangulaire inférieure du prisme.

- A chaque nœud Ni ($0 \leq i \leq 2$) de T est associé un nœud N'i. N'0, N'1 et N'2 définissent alors un nouvel élément triangulaire du prisme, qui constitue sa face triangulaire supérieure.

On dispose alors d'un objet volumique composé de six sommets N0, N1, N2, N'0, N'1 et N'2. Chacun de ces nœuds possède trois voisins:

- Les deux sommets avec lesquels il forme une face triangulaire du prisme.

- Le sommet auquel il est associé dans la face triangulaire opposée à celle à laquelle il

appartient.

Cette description naturelle du prisme, si elle rend bien compte de la dimension volumique et de la nature de l'objet, est en revanche très pénalisante du point de vue de la caractérisation mathématique de son volume.

Par exemple, le test d'appartenance d'un point P(x,y,z) à un tel élément apparaît comme relativement complexe. Il s'agit là pourtant d'une opération de base qui peut être amenée à être utilisée très fréquemment dans tout type d'algorithme mettant en œuvre un objet gstack avec un objet extérieur (nuage de points, grille régulière, etc...).

Résoudre ce problème revêt donc une très grande importance pour l'utilisation de l'objet *gstack* dans de telles conditions. La solution que nous proposons d'y apporter consiste à décomposer le volume du prisme en tétraèdres. En effet, si une telle décomposition est possible, le problème de l'appartenance d'un point à un prisme se limitera à des calculs d'appartenance de ce point à un tétraèdre. Ce type de calcul est à la fois très simple et très rapide.

Résolution du problème

La décomposition en tétraèdres d'un prisme doit être consistante. En particulier, il faut que les deux conditions suivantes soient respectées:

- La somme des volumes des tétraèdres est égale au volume du prisme.
- Quels que soients T_1 et T_2 deux tétraèdres de cette décomposition, l'intersection entre T_1 et T_2 est nulle.

Ces deux conditions imposent que les tétraèdres soient adjacents les uns aux autres par leurs faces et partagent ensemble trois sommets. Par ailleurs, un prisme à base triangle possède six sommets et un tétraèdre en possède quatre.

En toute logique, décomposer un prisme à base triangle en tétraèdres, sans ajouter de nœuds autres que les sommets du prisme, doit pouvoir se faire en trois tétraèdres. Le problème étant relativement simple, nous avons procédé à une étude systématique de toutes les décompositions consistantes possible. Les résultats obtenus sont présentés dans la Figure 20.

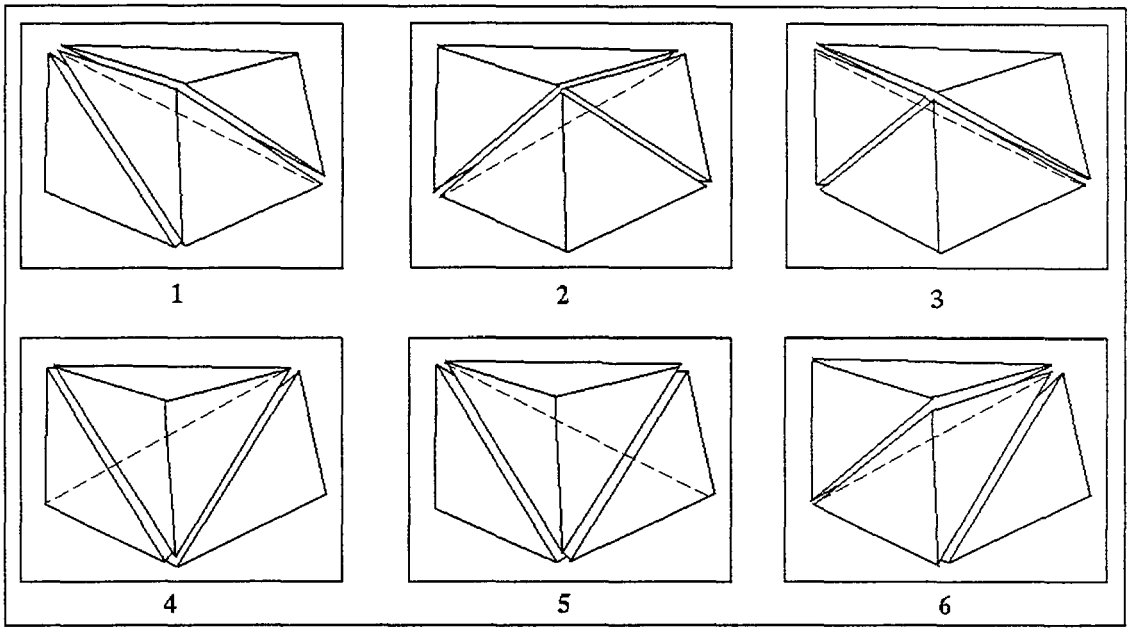


FIGURE 20. Décomposition d'un prisme à base triangle en tétraèdres.

Ces résultats amènent les commentaires suivants:

- Il existe six décompositions consistantes d'un prisme en tétraèdres.

- Parmi les six décompositions possibles, chacune peut être obtenue par symétrie planaire d'une autre. Ainsi les configurations 2 et 3 sont symétriques, ainsi que 4 et 5, et 1 et 6.

- Toutes ces configurations diffèrent les unes des autres par les connexions entre les sommets du prisme. Chaque configuration correspond à un découpage spécifique des faces latérales du prisme.

Cette spécificité nous permet de caractériser ces configurations d'une façon plus appropriée que la dénomination par indice totalement arbitraire que nous avons employée dans la Figure 20.

Considérons la face composée des nœuds N_0, N_1, N'_0, N'_1. Après la décomposition, deux

cas de figure peuvent se présenter:

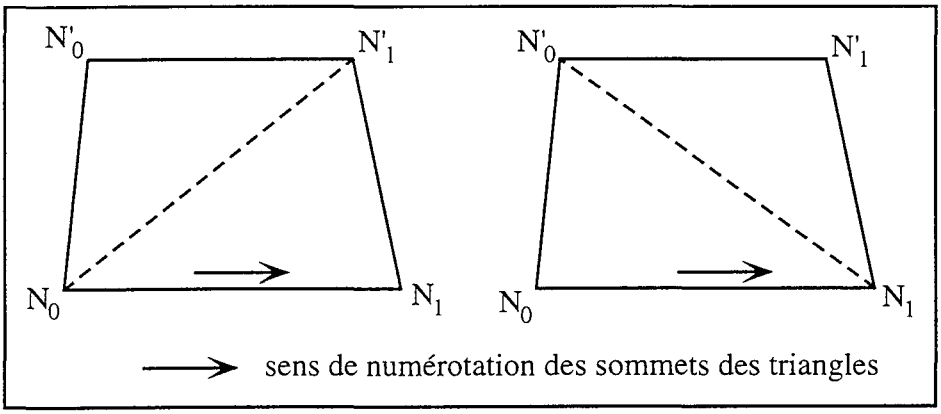


FIGURE 21. Représentation des deux découpages de faces possibles.

Considérons le segment induit au niveau de chaque face latérale (représenté en pointillés sur la figure). Ce segment est constitué d'un sommet de la face triangulaire inférieure et d'un sommet de la face triangulaire supérieure. Il peut donc s'exprimer sous la forme [NiN'j] ([N0N'1] dans le premier cas de la figure, [N1N'0] dans le second).

Deux cas sont donc à considérer:

1. les indices i et j des nœuds du segment se suivent dans l'ordre de numérotation des nœuds des triangle, à savoir l'ordre 0->1->2->0->1... C'est le cas dans l'exemple de gauche de la figure où i et j prennent les valeurs respectives 0 et 1. Une telle face sera alors dite de type 0.
2. les indices i et j des nœuds du segment se suivent dans l'ordre inverse de l'ordre de numérotation des nœuds des triangle, à savoir l'ordre 2->1->0->2->1... C'est le cas dans l'exemple de droite de la figure où i et j prennent les valeurs respectives 1 et 0. Une telle face sera alors dite de type 1.

Chacune des configurations présentées dans la Figure 20 peut ainsi être caractérisée en fonctions des trois types des trois faces latérales qui composent le prisme.

Prenons l'exemple de la configuration 1 de cette figure. En reprenant les notations de la Figure 19 de la page 25, nous allons déterminer le type de chacune des faces latérales du prisme:

- Pour la face (N_0, N_1, N'_1, N'_0), la décomposition crée le segment [$N_1N'_0$], la face est donc de type 1.

- Pour la face (N_1, N_2, N'_2, N'_1), la décomposition crée le segment [$N_2N'_1$], la face est donc de type 1.

- Pour la face (N_2, N_0, N'_0, N'_2), la décomposition crée le segment [$N_2N'_0$], la face est donc de type 0.

Cette disposition des tétraèdres peut donc être caractérisée par le triplet de booléens (110).

De la même manière, à chacune des cinq autres configurations possibles peut être associé un

triplet de booléens. Le tableau ci-dessous détaille les résultats obtenus.

Tableau .1 : Caractérisation des différentes décompositions par un triplet de booléens.

configuration 1	(110)
configuration 2	(011)
configuration 3	(010)
configuration 4	(101)
configuration 5	(100)
configuration 6	(001)

Proposition: *Les triplets de booléens (000) et (111) ne correspondent pas à une décomposition consistante du prisme en trois tétraèdres.*

Démonstration:

Cette proposition se démontre par l'absurde. Supposons que l'on ait pu remplir le prisme (N_0, N_1, N_2 , N'_0, N'_1,N'_2) de telle sorte que ses trois faces latérales soient de type 0. Si tel est le cas, alors dans aucun des tétraèdres de la décomposition l'on ne rencontrera les arètes (N_0, N'_2), (N_1, N'_0) et (N_2, N'_1).

Dans toute décomposition, la face triangulaire inférieure (N_0, N_1, N_2) est une face de tétraèdre. Trois cas de figure sont alors à considérer:

- N'_0 est le quatrième sommet de ce tétraèdre. Or dans un tétraèdre, les quatre sommets sont tous reliés entre eux par des arètes. Ceci suppose donc que N_1 et N'_0 forment une arète, ce qui est impossible.
- N'_1 est le quatrième sommet de ce tétraèdre. Alors N_2 et N'_1 forment une arète, ce qui est impossible.
- N'_2 est le quatrième sommet de ce tétraèdre. Alors N_0 et N'_2 forment une arète, ce qui est impossible.

Conclusion: le triplet (0,0,0) ne correspond pas à une décomposition consistante du prisme en tétraèdres. Une démonstration analogue peut être effectuée avec le trplet (1,1,1) qui conduit à la même conclusion.

Le même type de raisonnement permet également de démontrer qu'à chaque triplet de booléen ne correspond qu'une décomposition en tétraèdres possible.

Conclusion

Il existe six manières différentes de décomposer le volume d'un prisme à base triangle en trois tétraèdres. Celles-ci diffèrent les unes des autres par la répartition des différents sommets entre les tétraèdres qui se traduit au niveau des faces latérales du prisme par un découpage spécifique.

Nous avons introduit un formalisme qui permet de caractériser de manière unique chacune des six décompositions mises en évidence par un triplet de booléens.

Il faut noter que ces décompositions, si elles sont toutes consistantes, ne sont pas équivalentes du point de vue de la nature du volume qu'elles délimitent. En effet, si l'on considère une face latérale d'un prisme, les quatre sommets qui la composent ne sont pas à priori coplanaires. Il en résulte que le fait qu'une face soit de type 0 ou 1 influe non seulement sur la connectivité des tétraèdres mais aussi sur le volume du prisme.

Dans le but que nous nous sommes fixés de caractériser de manière simple le volume compris entre deux niveaux d'un *gstack* par une décomposition de ce volume en tétraèdres, ce problème ne posera de difficulté qu'au niveau du bord de l'objet, ce qui n'est pas très génant. En revanche, un autre problème est celui de l'agencement correct des prismes les uns aux autres par leurs faces latérales. Ce dernier point est fondamental.

Ce problème va être étudié dans le chapitre qui suit.

2.4.3 Agencement des prismes les uns aux autres

La Figure 18 de la page 24 montre que le volume délimité par deux niveaux successifs d'un *gstack* peut être vu comme la réunion de volumes élémentaires prismatiques.

Dans le chapitre précédent, nous avons montré que la décomposition en tétraèdres de ces volumes élémentaires n'était pas unique et nous avons vu qu'il existait six possibilités. Celles-ci diffèrent les unes des autres par la nature des faces latérales des prismes, dont nous avons pu définir deux types opposés.

Ceci nous amène à envisager un nouvel aspect de cette décomposition. Si l'on veut définir un maillage de manière correcte par tétraédrisation d'un volume compris entre deux niveaux d'un *gstack*, il va falloir résoudre les problèmes que soulève l'agencement des prismes à base triangulaire les uns aux autres par leurs faces latérales.

Nous allons d'abord étudier le problème de deux prismes adjacents pour ensuite l'étendre au niveau de l'objet *gstack*.

Agencement de deux prismes adjacents

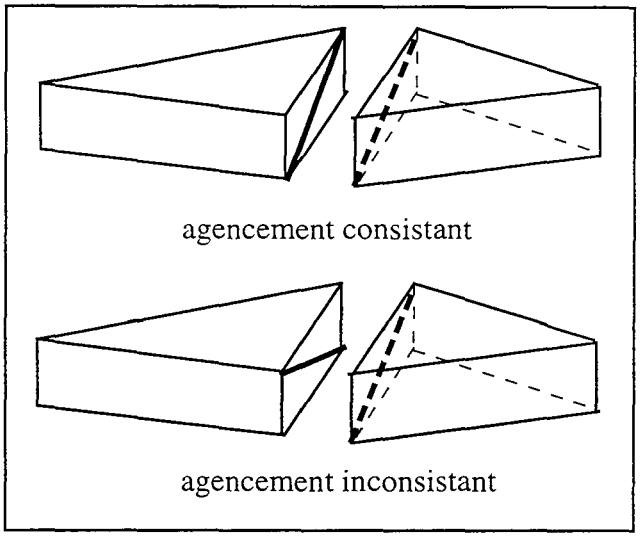
Le formalisme employé ici est le même que celui qui a été utilisé dans le chapitre précédent.

Nous disposons de deux prismes P_1 et P_2 accolés l'un à l'autre par une face latérale commune F et partageant ainsi les quatre nœuds qui la composent. Ceux-ci partagent également le même sens de numérotation des sommets des faces triangulaires, sens qui est d'ailleurs commun à tous les triangles de chaque niveau du *gstack*.

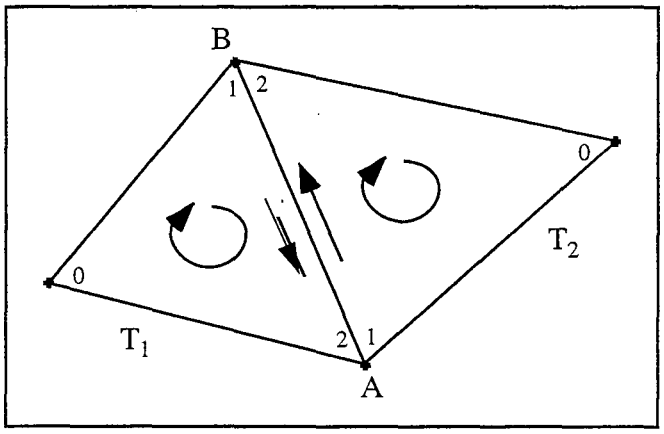
Chacun des prismes P_1 et P_2 a pu être décomposé en tétraèdres de telle sorte que chacune des faces latérales de ces deux prismes est découpée extérieurement en deux triangles et a pu être affecté d'un type 0 ou 1 (. Notons F_{P_1} et F_{P_2} le type de la face F dans les prismes P_1 et P_2.

Le problème est alors le suivant: quelle est la relation entre F_{P_1} et F_{P_2} qui assure un agencement correct des faces des tétraèdres de part et d'autre de chaque prisme au niveau de la face F?

Ce problème d'agencement est illustré par la figure suivante:



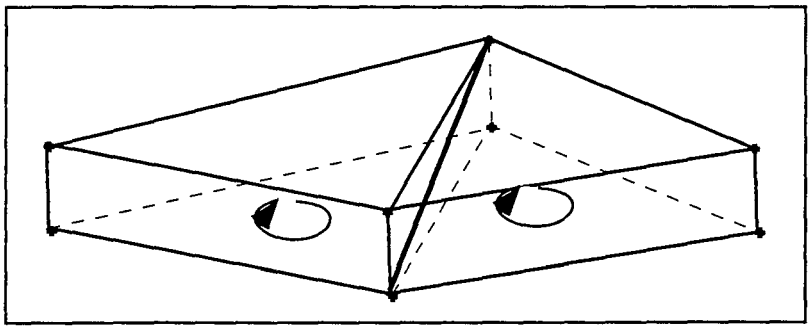
La figure ci-dessous représente les deux triangles T1 et T2, qui sont les faces triangulaires inférieures des prismes P1 et P2. Ces deux triangles partagent une arête commune [AB] ainsi qu'un sens de numérotation commun des sommets matérialisé par les flèches circulaires.



Considérons maintenant les nœuds A et B partagés par T1 et T2. Si l'on parcourt l'arête [AB] en partant de A pour aller vers B, on peut observer que le parcours se fait dans le même sens que le sens de numérotation des sommets au niveau d'un triangle (T2 sur la figure) alors qu'il se fait dans le sens opposé au niveau de l'autre triangle (T1 sur la figure).

En d'autres termes, si le parcours d'une arête partagée par deux triangles se fait dans le sens de la numérotation des sommets des triangles au niveau du triangle adjacent gauche, il se fera

dans le sens inverse au niveau du triangle adjacent droit.



La figure ci-dessus représente deux prismes adjacents par une face latérale. La décomposition des prismes en tétraèdres produit un découpage de cette face que nous avons représenté en gras.

A la lumière de cette représentation, et compte tenu de la définition des types de faces de prismes introduite dans le chapitre précédent (cf. Figure 21 de la page 28), la condition garantissant un ajustement correct des faces est la suivante:

soit P_1 un prisme à base triangulaire partageant une face latérale commune F avec un prisme P_2. P_1 a été décomposé en trois tétraèdres de telle sorte que la face F est de type T. La condition pour que la tétraédrisation de P_2 soit consistante avec celle de P_1 au niveau de F est que la face F soit de type $\neg T$ au niveau de P_2.

Tétraédrisation automatique de l'objet *gstack*

Nous allons maintenant pouvoir aborder le problème de la tétraédrisation au niveau de l'objet *gstack*. Après avoir résolu le problème de l'agencement de deux prismes par une face latérale, il ne nous reste plus qu'à résoudre le problème de l'agencement des prismes les uns aux autres à l'échelle d'un maillage triangulé.

Ce problème peut s'exprimer de la manière suivante: considérons la triangulation commune à tous les niveaux du *gstack*. Le problème du remplissage automatique par des tétraèdres revient à affecter à chaque triangle T du maillage un indice compris entre 1 et 6 correspondant à une des six décompositions possibles représentées sur la Figure 20 de la page 27.

En reprenant le formalisme à base de triplets de booléens introduit précédemment, le problème se résume à affecter à chaque arête de chaque triangle une valeur booléenne , et ce en respectant les deux conditions suivantes:

- Au niveau d'un triangle, les trois valeurs booléennes des trois arêtes qui le composent ne doivent pas être identiques. En effet, nous avons vu dans le chapitre précédent que les configurations (000) et (111) sont inconsistantes au niveau de la tétraédrisation.

- Au niveau d'une arête A commune à deux triangles T_1 et T_2, si A a été affectée du type 0 (respectivement 1) au niveau de T_1, alors celle-ci sera affectée du type 1 (respectivement 0) au niveau de T_2.

La Figure 22 présente un exemple d'affectation de booléens vérifiant ces deux conditions:

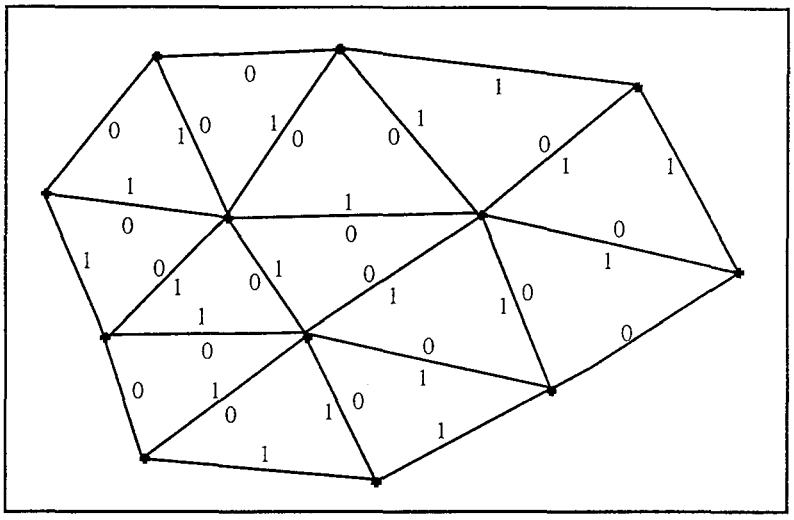


FIGURE 22. Exemple d'affectation de booléens à une triangulation.

Pour effectuer ces affectations de manière automatique, nous avons mis en place un algorithme récursif de type recherche en profondeur d'abord avec retour arrière. Cet algorithme se compose de trois parties: une procédure de premier appel, un test de compatibilité des types de triangle et la procédure récursive proprement dite.

La procédure de premier appel

```
Boolean GSTACK_Initialise_classes( gstack G )
    Soit S l'ensemble des triangle de G;
    Tant que (S n'est pas vide) faire
        Soit T un triangle de S;
        Bool = Triangle_Affecte_Classe(T); /*procédure récursive*/
        Si (bool == FAUX) retourne(FAUX) ;
        Pour (chaque T de S) faire
            Si (T a été affecté d'une classe) Supprimer T de S;
        fin Pour
    fin Tant que
retourne (VRAI);
```

ALGORITHME 1. Assignation de classes: la procédure de premier appel.

La procédure de premier appel constitue l'entrée dans la procédure récursive.

La boucle Tant que() vise à traiter les cas où le maillage de l'objet gstack se décompose en plusieurs parties non connectées.

Nous envisagerons à la fin de ce chapitre le cas où la valeur FAUX est retournée par l'algorithme.

Le test de compatibilité

```
Boolean Triangle_Type_OK( TRGL T, Type t)
    Pour (chaque arête Ai de T partagée avec le triangle Ti) faire
        Si ( (Ti == NULL) ou (Ai n'a pas encore de type dans Ti) ) continue ;
        Si (le type de l'arête Ai est le même dans T et Ti) retourne (FAUX) ;
    fin Pour
    retourne (VRAI) ;
```

ALGORITHME 2. Assignation de classes: test de compatibilité.

Le test de compatibilité porte sur un triangle T auquel on tente d'affecter un type t. Pour chaque arête Ai de ce triangle T, on essaie de valider le type qui lui est affecté à partir des informations du triangle Ti qui partage l'arête Ai avec T.

La procédure récursive

La procédure récursive effectue le travail d'assignation de classes au triangle, en leur affectant un indice C compris entre 1 et 6.

Le cœur de la procédure consiste à considérer un triangle du maillage et à lui affecter temporairement un indice, après avoir étudié la compatibilité de cette affectation par rapport aux affectations des triangles voisins.

Si cette compatibilité existe, un appel récursif est alors effectué sur les triangles voisins du triangle affecté.

Dans le cas où cette fonction récursive retourne la valeur VRAI pour les trois voisins de T, alors l'affectation de la classe C au triangle T est confirmée et la fonction retourne la valeur VRAI à son tour.

Comme toute fonction récursive, la fonction Triangle_Affecte_Classe possède un cas d'arrêt. En effet, chaque triangle parcouru dans les niveaux inférieurs de la pile est marqué temporairement. Une condition nécessaire à la terminaison de l'algorithme est qu'un triangle ne doit être pris en considération qu'une seule fois. C'est le sens du cas d'arrêt qui a été mis en place.

Pour un triangle donné, il peut arriver que, au nivau n+1 de la pile des appels de fonctions et après avoir envisagé les six configurations possibles, aucune d'entre elles n'ait pu être validée. La fonction récursive renvoie alors la valeur FALSE, ce qui a pour conséquence le rejet de la configuration courante envisagée sur le triangle considéré au niveau n de la pile. C'est la procédure de retour arrière qui garantit que toutes les combinaisons possibles d'affectation de classes

sont envisagées.

```
Boolean Triangle_Affecte_Classe( TRGL T )
  Si ( T a déjà été affecté ) retourne(VRAI) ; cas d'arrêt
  Pour (chacune des six configurations C possibles) faire
    Si ( Triangle_Type_OK(T, C) == TRUE) /* test de compatibilité */
      Marquer temporairement T ;
      Affecter temporairement la classe C à T ;
      Pour ( chaque voisin Ti de T ) faire
        Si ( Triangle_Affecte_Classe(Ti) == FAUX) /* appel récursif */
          break ;
      fin Pour
      Si ( tous les voisins Ti ont pu être affectés) break ;
    fin Si
  fin Pour
  Si (aucune des six configurations n'est compatible) faire
    Supprimer le marquage de T.
    retourne (FAUX); /* retour arrière */
  fin Si
  Sinon
    Confirmer l'affectation de la classe C au triangle T;
    retourne (VRAI) ;
  fin Sinon
```

ALGORITHME 3. Assignation de classes: la procédure rcursive

Conclusion

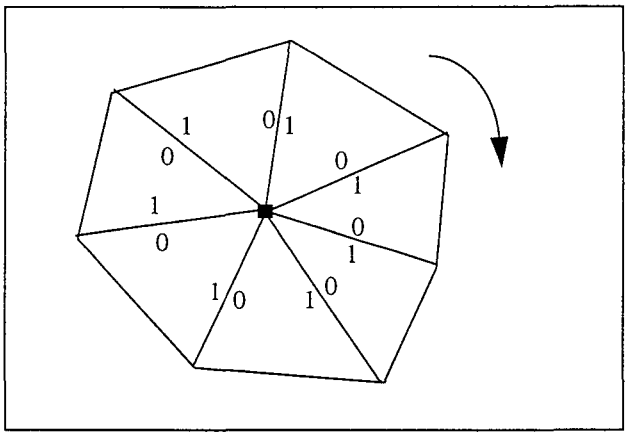
L'algorithme que nous venons de présenter procède de manière récursive à la recherche d'une solution dans laquelle chaque triangle d'un maillage donné est affecté d'un indice compris entre 1 et 6 qui correspond à une des 6 décompositions en tétraèdres possibles.

Dès qu'une solution au problème posé est trouvée, l'algorithme sort de la procédure récursive. Cependant, il est clair que cette solution n'est pas la seule possible. En effet, si l'on considère la Figure 22 de la page 33 où est représentée une solution de ce problème pour une triangulation donnée, il est clair qu'il existe une seconde solution à ce problème qui découle de

la précédente, et que nous pouvons appeler la «solution complémentaire». Celle ci est obtenue en affectant le type 0 aux arètes de type 1 et le type 1 aux arètes de type 0.

La présence du retour arrière dans l'algorithme garantit que, si une solution existe pour une triangulation donnée, alors celle ci sera détectée.

Il existe un autre algorithme permettant d'affecter des booléens aux arêtes des triangle. Celui-ci est itératif. Il procède, pour chaque noeud de la surface à un parcours systématique des triangles dont se noeud est un sommet, comme illustré par la figure ci-dessous..



Le parcours se fait dans le sens de rotation déterminé à partir de la normale à la surface au noeud en question. Si le parcours d'un ensemble de triangles passe en revue une arète déjà affectée, les anciennes valeurs booléenes sont écrasées par les nouvelles.

La condition «aucun triangle n'est affecté de trois booléens identiques» est remplacée par «deux arètes de chaque triangle ont toujours deux booléens opposés» grace au sens de rotation commun. Ces deux conditions sont équivalentes.

Cet algorithme apporte également la preuve de l'existence d'une solution à la décomposition en tétraèdres, quelle que soit la triangulation considérée.

La planche 1 représente un exemple de tétraédrisation automatique du volume compris

entre deux surfaces isomorphes.

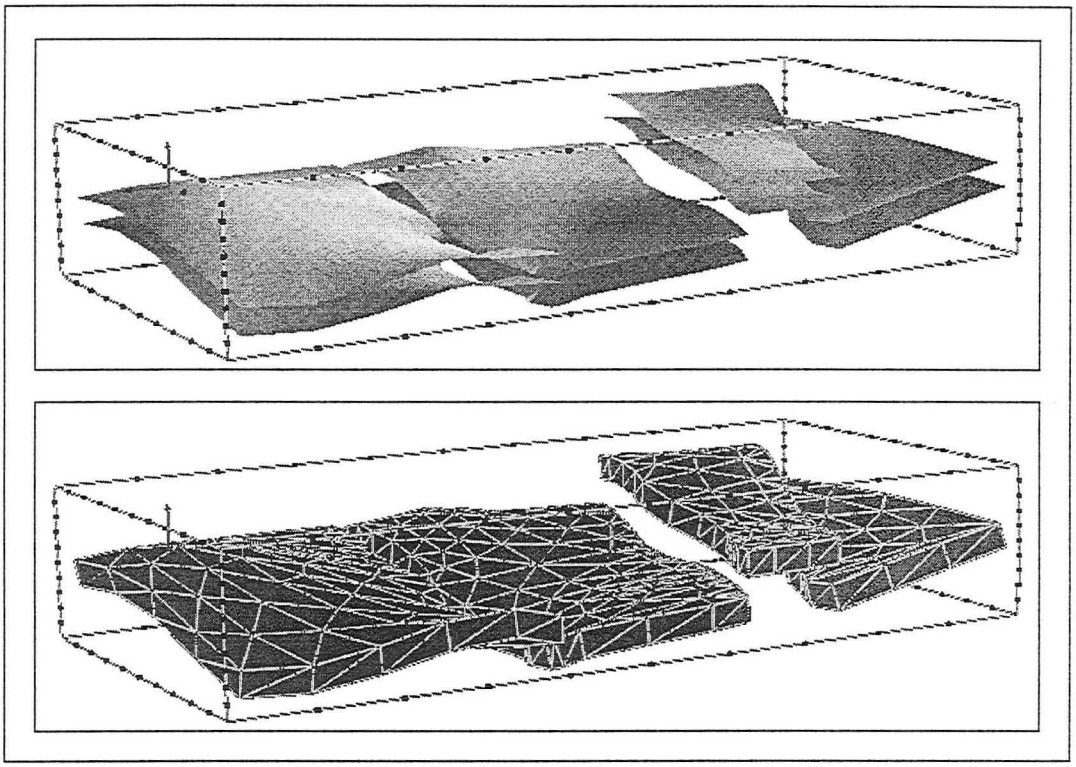


PLANCHE 1. Tétraédrisation automatique du volume compris entre deux surfaces isomorphes.

2.4.4 Conclusion et perspectives

Tout au long de ce chapitre, nous avons vu que l'objet *gstack* permettait de stocker au sein d'un même objet une succession de surfaces triangulées isomorphes.

Les instruments de modélisation géométrique de cet objet ont été ensuite décrits, et notamment la manière avec laquelle il est possible de générer des surfaces par interpolations de géométries préexistantes.

Enfin, nous avons mis en évidence le caractère volumique de cet objet, en proposant une méthode de décomposition du volume délimité par celui-ci sous forme de tétraèdres élémentaires.

Cette décomposition est déjà intéressante, dans le sens où elle permet de caractériser un volume compris entre deux surfaces isomorphes, ce qui constitue une manière d'introduire la notion de couche géologique.

Le principal intérêt de cet objet se situe au nivaeu de l'estimation de propriétés physiques dans le domaine de la modélisation du sous-sol.

Ces perspectives sont liées à l'introduction d'un nouveau type d'objet de type grille régulière dans la base de données géométrique du logiciel Gocad: le *voxet*. Cet objet sera présenté dans le chapitre suivant et les fonctionnalités que je viens de mentionner seront détaillées dans le chapitre 3.6 de la page 71.

Chapitre 3

Grilles régulières et maillages irréguliers: l'objet *voxet*

3.1 Introduction

L'objet «grille régulière», qu'il s'agisse d'une grille à deux ou trois dimensions, est un outil d'usage fréquent dans de nombreuses disciplines, telles que les mathématiques appliquées, les statistiques, les méthodes de traitement d'images numériques, etc...

Jusqu'à une date récente, il constituait le support de calcul exclusif des méthodes utilisées pour modéliser la géométrie des objets naturels (cartographie automatique, géostatistiques, ...).

Le projet de recherche Gocad a permis de mettre en évidence l'intérêt d'utiliser des maillages irréguliers dans ce domaine. En particulier, l'utilisation des surfaces triangulées a apporté une solution cohérente au problème de la modélisation géométrique des surfaces discontinues ou multi-valuées (dômes de sel, surfaces faillées, etc...).

Ceci nous a amené à émettre un certain nombre de critiques sur l'usage des grilles régulières dans ce domaine. La principale de ces critiques concerne le fait que celles-ci étaient implicitement considérées comme la structure de base définissant la topologie des objets à modéliser. Or il est bien clair que ceci pose des problèmes dans certains exemples de géométries complexes tels que ceux cités plus haut.

Tout au long de ce chapitre, nous allons introduire la notion de *voxet* (qui est l'abbréviation de voxel set). Il s'agit là d'un outil de construction de modèles à base de grilles régulières. Ces modèles peuvent être soit des modèles géométriques, soit des modèles de répartition de propriétés physiques. Cet outil diffère des autres outils de type «grille régulière» par la finalité qui lui a été assignée. En effet, dès sa création, cet objet a été voulu comme le moyen pour le logiciel Gocad de communiquer avec d'autres applications utilisant les grilles régulières comme support d'information. Dans le cas présent, communiquer consistera essentiellement à traduire les modèles produits par Gocad sous forme de maillages réguliers. Les destinataires de ces maillages pourront être très divers, mais les premiers auxquels nous avions pensé au départ étaient les outils de traitement de données sismiques ([9]), de modélisation géostatistique et de

simulation d'écoulements.

La première partie de ce chapitre présente les fonctionnalités de base du *voxet* ainsi que les techniques qui ont été employées pour introduire la notion de discontinuité au sein de maillages réguliers.

La seconde partie porte sur des aspects de conception informatique. Elle est consacrée à une description du schéma de fonctionnement interne du *voxet* et nous verrons que celui ci lui assure une modularité et une évolutivité tout à fait intéressantes.

La suite du chapitre sera consacrée à une description des principaux outils de conversion des différents objets utilisés dans le cadre du logiciel Gocad, sous forme de grille régulière. Pour chacun d'entre eux, l'algorithme de conversion sera présenté en détail et les perspectives offertes seront discutées et illustrées à l'aide d'exemples.

Dans le but de proposer une implémentation cohérente pour l'ensemble des fonctionnalités présentées, une architecture orientée objet sera proposée qui intègre le schéma de fonctionnement décrit dans la deuxième partie.

3.2 Présentation générale de l'outil *voxet*

3.2.1 Définition de l'objet *voxet*

Un *voxet* se compose d'une grille régulière, qui se définit d'abord par des paramètres géométriques:

1. Un triplet $(O'x,O'y,O'z)$ de coordonnées du noeud origine O' de la grille.
2. Trois vecteurs $\vec{U}(Ux,Uy,Uz)$, $\vec{V}(Vx,Vy,Vz)$ et $\vec{W}(Wx,Wy,Wz)$ définissant l'espacement entre deux noeuds adjacents dans les trois directions principales de la grille. La seule contrainte existant sur ces trois vecteurs est qu'ils doivent former une base. Dans la majorité des cas, ceux ci seront orthogonaux entre eux et disposés parallèlement au axes du référentiel dans lequel leurs coordonnées sont exprimées. Cependant, l'objet *voxet* a été conçu de telle façon que des cas beaucoup plus généraux puissent être également pris en compte.
3. Trois entiers nu, nv et nw définissant le nombre de nœuds de la grille le long de ses trois axes $\vec{U}$,$\vec{V}$ et $\vec{W}$.

Tout ceci définit la géométrie d'une grille régulière. Son contenu est constitué par un tableau unidimensionnel de taille nu.nv.nw. Les informations stockées dans les entrées de ce tableau sont des réels à virgule flottante. Ainsi, une grille de taille 100*100*100 sera stockée sur 4 Mo

en mémoire.

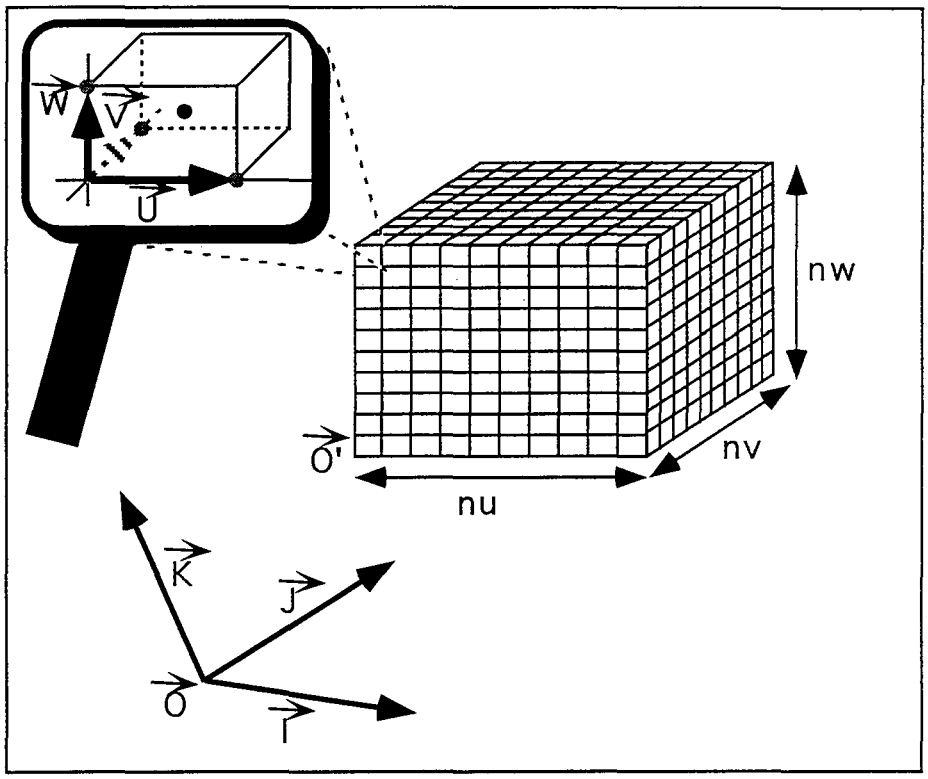


FIGURE 23. Géométrie de l'objet *voxet*.

Un certain nombre de notions et de conventions régissent le fonctionnement de l'objet *voxet*. Nous allons les présenter dans la suite de ce chapitre.

Notion de *Voxel*

A chaque nœud de la grille est associé un élément de volume parallélépipédique, également appelé *voxel*, centré autour de ce nœud. Ce *voxel* V est défini par trois indices entiers u, v et w vérifiant:

$$0 \leq u < nu \qquad 0 \leq v < nv \qquad 0 \leq w < nw$$

La position P du nœud situé au centre du *voxel* V(u, v, w) est donnée par:

$$\vec{P}(u, v, w) = \vec{O'} + u \cdot \vec{U} + v \cdot \vec{V} + w \cdot \vec{W}$$

Les huit sommets du *voxel* V(u, v, w) sont donc les suivants:

$$\vec{P1}\left(u-\frac{1}{2},v-\frac{1}{2},w-\frac{1}{2}\right) \qquad \vec{P2}\left(u-\frac{1}{2},v+\frac{1}{2},w-\frac{1}{2}\right)$$

$$\vec{P3}\left(u+\frac{1}{2},v-\frac{1}{2},w-\frac{1}{2}\right) \qquad \vec{P4}\left(u+\frac{1}{2},v+\frac{1}{2},w-\frac{1}{2}\right)$$

$$\vec{P5}\left(u-\frac{1}{2},v-\frac{1}{2},w+\frac{1}{2}\right) \qquad \vec{P6}\left(u-\frac{1}{2},v+\frac{1}{2},w+\frac{1}{2}\right)$$

$$\vec{P7}\left(u+\frac{1}{2},v-\frac{1}{2},w+\frac{1}{2}\right) \qquad \vec{P8}\left(u+\frac{1}{2},v+\frac{1}{2},w+\frac{1}{2}\right)$$

La figure ci-dessous illustre cette notion de *voxel:*

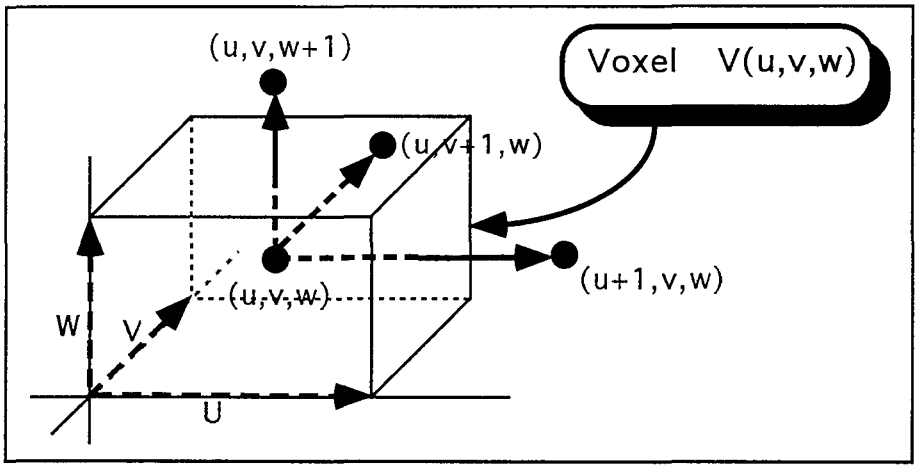


FIGURE 24. Notion de *voxel* centré autour d'un nœud.

Notion de grille régulière

Une grille régulière peut alors être vue de différentes manières:

- soit comme un ensemble de nœuds disposés de manière régulière le long de ses axes.

- soit comme un ensemble d'éléments de volume centrés autour de ces nœuds et accolés les uns aux autres par leurs faces. Cette vision est celle utilisée dans le domaine du traitement des images numériques où interviennent la notion de pixel (picture element, élément d'image) pour les images 2D et la notion de *voxel* (volume element, élément de volume) pour les images 3D.

- soit enfin comme un ensemble d'éléments de volume parallélépipédiques dont les huit sommets sont des nœuds de la grille. Par analogie avec la définition précédente, les huit

sommets d'un *voxel* V(u, v, w) sont alors les suivants:

$$\vec{P1}(u,v,w) \qquad \vec{P2}(u+1,v,w)$$
$$\vec{P3}(u,v+1,w) \qquad \vec{P4}(u+1,v+1,w)$$
$$\vec{P5}(u,v,w+1) \qquad \vec{P6}(u+1,v,w+1)$$
$$\vec{P7}(u,v+1,w+1) \qquad \vec{P8}(u+1,v+1,w+1)$$

La figure ci-dessous illustre cette notion de *voxel:*

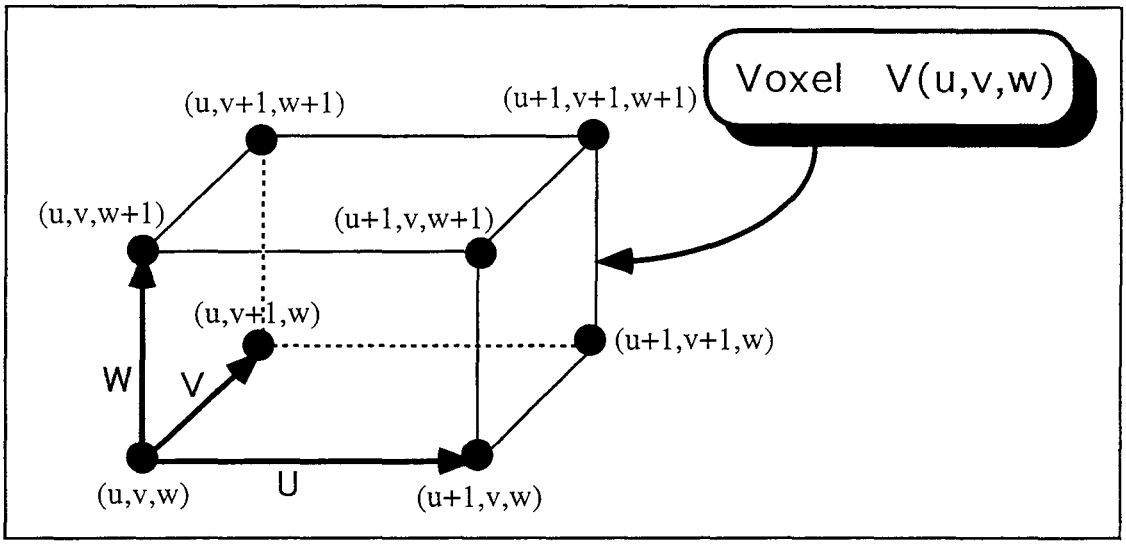


FIGURE 25. Notion de *voxel* composé de huits nœuds.

Cette description de la grille régulière est celle employée dans les méthodes de calcul par éléments finis, où chaque élément est un voxel composé de huits nœuds de grille.

Toutes ces manières de voir une grille régulière sont équivalentes et n'influent en aucune manière sur le contenu de l'objet. Le fait d'adopter l'une ou l'autre d'entre elles dépend uniquement du contexte dans lequel on les utilise.

Dans le cas de l'objet *voxet*, la convention utilisée est celle des *voxels* centrés autour des nœuds de la grille.

Informations attachées aux nœuds d'un *voxet*

En plus d'une valeur numérique, il est possible d'associer à chaque nœud de la grille un certain nombre d'informations supplémentaires. Pour des raisons évidentes de coût en mémoire, nous nous sommes limités à stocker des informations booléennes, qui peuvent être stockées sur un bit. Les deux principales informations que nous avons jugé nécessaires de stocker sont les suivantes:

- une information intitulée *painted* sera utilisé pour déterminer si la valeur numérique d'un nœud lui a été assignée directement par l'utilisateur, auquel cas elle prendra la valeur 1, ou si elle a été déterminée par un autre moyen (valeur nulle par défaut, valeur calculée par inter-

polation, etc...) et elle prendra la valeur 0. Par défaut, à la construction d'une grille, tous les bits *painted* prennent la valeur zéro.

- une information intitulée *region* sera également utilisée pour introduire la notion de sous-ensemble dans une grille. Il fonctionne de la manière suivante: si un nœud appartient à ce sous-ensemble, le bit correspondant à l'information *region* sera mis à 1, sinon il sera mis à 0. Par défaut, à la construction d'une grille, tous les bits *region* sont mis à 1.

Il sera possible par la suite de stocker des informations supplémentaires dans le cadre d'algorithmes spécifiques pour stocker des informations temporaires.

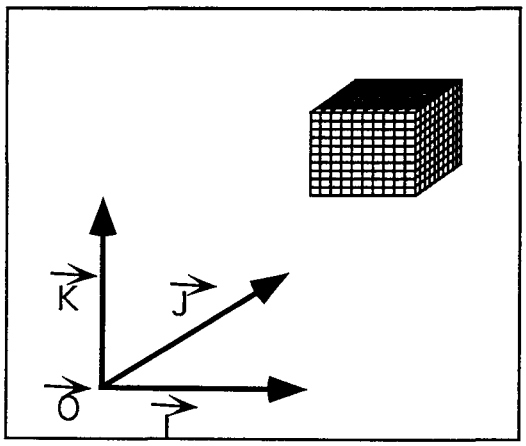
3.2.2 Manipulation des différents systèmes de coordonnées

Nous avons vu que la définition de la géométrie d'une grille faisait intervenir tout un ensemble de nouveaux paramètres. Ainsi, la position d'un nœud dans l'espace peut être définie de trois manières différentes:

1. Dans le réferentiel universel $(\vec{O},\vec{I},\vec{J},\vec{K})$ dans lequel les coordonnées des vecteurs $\vec{O}',\vec{U},\vec{V}$ et $\vec{W}$ de la grille sont exprimées, la position du *voxel* V(x, y, z) est donnée par:

$$V(x,y,z)= \vec{O} + x\vec{I} + y\vec{J} + z\vec{K}$$

Celle-ci est alors exprimée dans un repère de l'espace 3D sous la forme d'un triplet de trois réels.

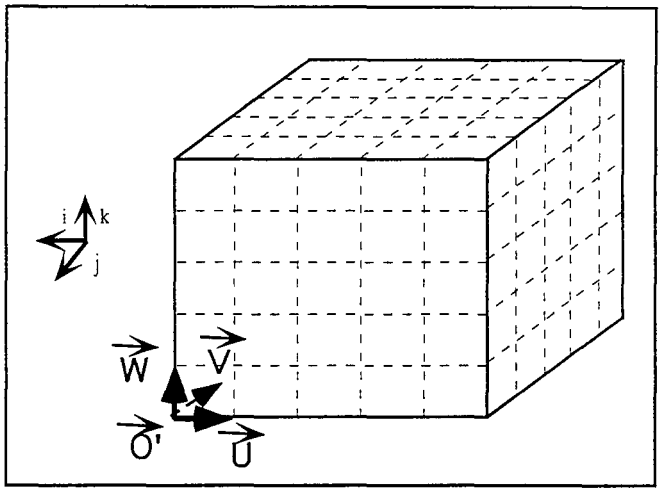


2. Dans le référentiel $(\vec{O}',\vec{U},\vec{V},\vec{W})$ de la grille ou référentiel local. La position de V(u, v, w) peut être obtenue par:

$$V(u,v,w)= \vec{O}' + u\vec{U} + v\vec{V} + w\vec{W}$$

Celle-ci est exprimée sous la forme d'un triplet de valeurs entières dans un repère de

l'espace 3D entier .



3. Dans l'espace entier à une dimension lié à l'emplacement mémoire alloué pour stocker le contenu de la grille, à chaque *voxel* est associé un indice uvw tel que:

$$0 \leq uvw < nu \cdot nv \cdot nw$$

La position du *voxel* est donc ici décrite par un indice entier uvw.

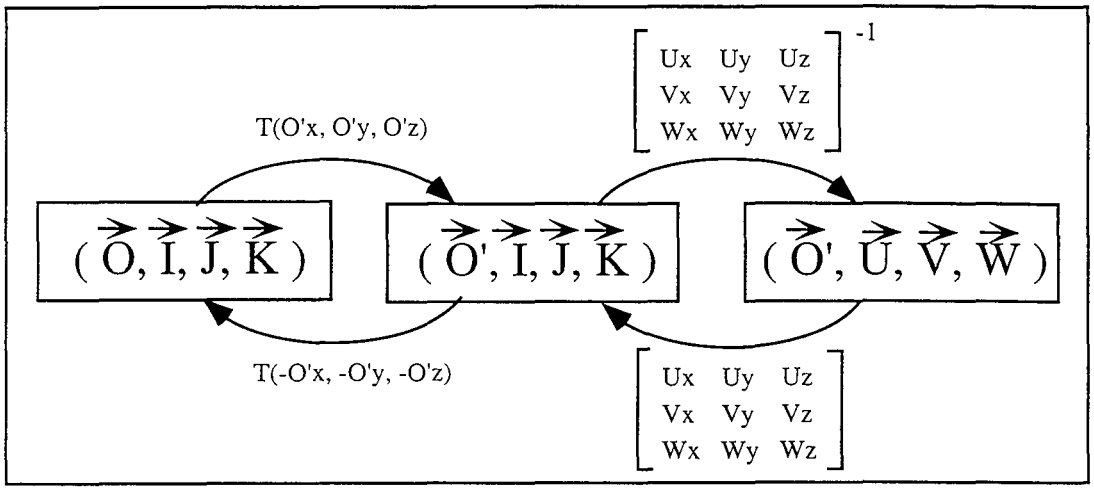
Chacune de ces représentations présente un intérêt. Ainsi, dans toute opération sur une grille faisant intervenir un objet extérieur exprimé dans le repère universel, le premier formalisme sera sans doute très utile.

De même, dans tous les algorithmes où devront être prises en compte des relations entre nœuds au sein d'une même grille (des relations de voisinage par exemple), on aura tendance à utiliser en priorité la représentation sous forme d'un triplet d'entiers.

Enfin, dans toutes les applications où un parcours systématique de tous les nœuds d'une grille doit être effectué, la représentation entière à une dimension apparaîtra comme la plus simple à utiliser et la plus efficace.

Quoi qu'il en soit, il apparaît indispensable d'étudier les moyens de passer d'un système de coordonnées à un autre.

Passage du système universel réel au système local entier



Ce passage s'effectue en deux étapes:

1. La première est un changement de base du repère $(\vec{O},\vec{I},\vec{J},\vec{K})$ au repère $(\vec{O'},\vec{U},\vec{V},\vec{W})$. L'expression des vecteurs $\vec{O'},\vec{U},\vec{V}$ et $\vec{W}$ dans le repère $(\vec{O},\vec{I},\vec{J},\vec{K})$ étant connue, le changement de base se résume à la combinaison d'une matrice de translation $T(-O'x, -O'y, -O'z)$ avec la matrice inverse de $\begin{bmatrix} Ux & Uy & Uz \\ Vx & Vy & Vz \\ Wx & Wy & Wz \end{bmatrix}$ (voir schéma ci-dessus).

   Le résultat de cette combinaison est stocké sous la forme d'une matrice 4*4 car exprimée en coordonnées homogènes, ce qui permettra de combiner à nouveau cette transformation avec d'autres.
2. La transformation des coordonnées x, y et z d'un point P(x,y,z) par cette matrice produit trois réels uréel, vréel et wréel, qu'il convient d'arrondir à la valeur entière la plus proche. On obtient ainsi les trois entiers u, v et w qui correspondent au *voxel* contenant le point P(x,y,z).

Le problème inverse est nettement plus simple. La position du point P correspondant au triplet d'entiers (u,v,w) est donnée par:

$$\vec{P} = \vec{O'} + u\vec{U} + v\vec{V} + w\vec{W}$$

Passage du système local entier au système unidimensionnel

Soit V un *voxel* défini dans le repère local de la grille par les trois entiers u, v et w. Compte tenu du choix que nous avons fait de stocker en mémoire la grille dans le sens u d'abord, puis v, puis w, l'indice uvw de V dans le repère lié au tableau de cette grille de taille nu*nv*nw est

donné par:

$$uvw = u + v \cdot nu + w \cdot nu \cdot nv$$

Inversement, si l'on connait l'indice uvw d'un *voxel* d'une grille, ses trois composantes u, v et w sont données par:

$$u = ((uvw \,\%\, (nu \cdot nv)) \,\%\, nu)$$

$$v = \frac{((uvw \,\%\, (nu \cdot nv)) - (uvw \,\%\, nu))}{nu}$$

$$w = \frac{(uvw - (uvw \,\%\, (nu \cdot nv)))}{nu \cdot nv}$$

où l'expression A%B désigne le reste de la division entière de A par B.

Passage du système universel au système unidimensionnel entier

Le passage direct du système universel au système à une dimension se fait par l'intermédiaire du système local entier. Il s'effectue donc en combinant les deux transformations décrites plus haut.

3.2.3 Gestion de discontinuités dans une grille régulière

La notion de discontinuité est intrinsèquement liée aux problèmes de modélisation des objets naturels. Ainsi, en tectonique, on distingue classiquement deux types de déformation: la déformation souple (ou continue) qui se manifeste par le plissement des couches géologiques et la déformation cassante (ou discontinue) à l'origine des failles. De même, en sismique, la notion de réflecteur est liée au caractère discontinu des lois de vitesse en fonction de la profondeur.

Un outil destiné à être utilisé dans le but de modéliser le sous-sol se doit donc d'intégrer la notion de discontinuité.

Dans la suite de ce chapitre, nous allons présenter deux approches différentes, qui ont toutes deux été testées, qui visent à introduire la notion de discontinuité à l'intérieur d'une grille régulière.

Notion de voisinage dans une grille

Dans une grille régulière, les *voxels* sont répartis régulièrement selon trois directions. Si l'on considère l'un d'entre eux, noté V(u,v,w), on peut alors définir une notion de voisinage (ou de connectivité) autour de ce voxel. Ce voisinage sera constitué d'un ensemble de nœuds dont les indices entiers sont liés à u, v et w par une relation.

Dans le cas d'un voisinage de six *voxels* (c'est à dire d'une 6-connectivité), les voisins de

V(u,v,w) sont:

V1(u – 1,v,w)	V2(u + 1, v, w)
V3(u, v–1, w)	V4(u, v + 1, w)
V5(u, v, w – 1)	V6(u, v, w + 1)

Ceci n'est qu'un exemple de voisinage. Les liaisons entre nœuds n'étant pas explicitement codées dans une grille régulière, comme c'est le cas dans un graphe, il est tout à fait possible de définir un autre type de connectivité en ajoutant des voisins à ce premier sous ensemble.

Dans la suite de ce chapitre, c'est néanmoins la 6-connectivité qui sera employée.

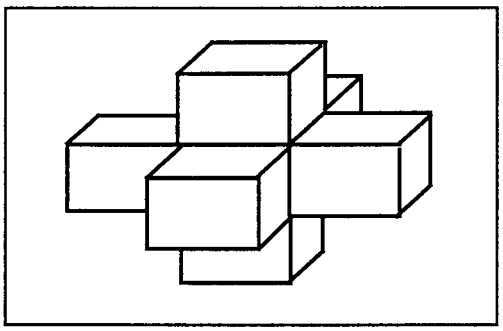


FIGURE 26. Définition de la 6-connectivité.

Gestion de discontinuités par cellules dégénérées

La notion de voisinage étant définie, il apparait clairement qu'une grille régulière est constituée de deux types de cellules:

- Les cellules du bord dont le voisinage est incomplet.
- Les autres cellules qui possèdent six voisins.

Nous allons introduire ici la notion de cellule dégénérée. Celle-ci suppose l'emploi d'une unité binaire supplémentaire associée à chaque nœud que nous appelerons unité *black.*

Par défaut, l'unité *black* des nœuds du bord sera mise à 1, alors qu'elle sera à 0 pour les autres nœuds.

Créer des discontinuités dans une grille reviendra donc à affecter la valeur *black*=1 à certains nœuds. Une cellule ainsi affectée sera dite dégénérée. Tous les voisins de cette cellule verront alors leur voisinage amputé de celle-ci, d'où la notion de discontinuité (voir Figure 27). Par ailleurs, ces cellules dégénérées ne sont plus concernées par les différents algorithmes travaillant sur les valeurs numériques de la grille. Leur rôle se limite à coder les discontinuités

dans le modèle.

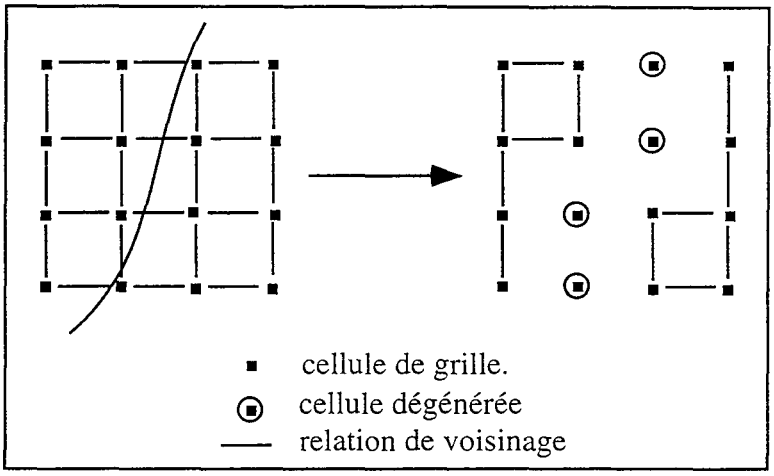


FIGURE 27. Prise en compte de discontinuités par cellules dégénérées.

Cette approche a été testée et fonctionne correctement. Son coût en mémoire est minimal (un bit par *voxel*) et la recherche des voisins d'une cellule est très simple.

Cependant, le fait que les cellules dégénérées soient exclues des opérations numériques sur la grille impose que ces cellules (et notamment les cellules du bord) soient affectées de la valeur nulle par défaut. Or, l'objet *voxet* est utilisé pour transférer le contenu numérique des grilles vers d'autres applications. Le fait que la notion de discontinuité influe sur celui-ci nous a amené à développer une autre approche de codage des discontinuités.

Gestion des discontinuités par codage explicite des voisinages

Comme cela a été dit précédemment, une grande différence entre les maillages de type irréguliers, tels que les maillages triangulés, et les grilles régulières, est que les voisinages entre nœuds sont explicitement décrits dans les premiers, alors que ceux-ci sont implicitement connus dans le cas des seconds.

Dans cette approche, la mise en place de discontinuités à l'intérieur d'une grille régulière nous a amené à coder les voisinages entre cellules de manière explicite. Pour ce faire, dans le cas de la 6-connectivité, il a fallu associer à chaque *voxel* trois bits, nommés *next_u*, *next_v* et *next_w*. Ceux-ci fonctionnent de la manière suivante:

Si un *voxel* V est voisin avec le *voxel* qui lui est immédiatement adjacent dans la direction U (resp. V, W) dans le sens des indices croissants, alors le bit *next_u* (resp. *next_v*, *next_w*) du *voxel* V est mis à 1, sinon il est mis à 0.

A la création d'un objet de type *voxet*, il convient d'initialiser ces bits de manière à expliciter le bord de la grille en terme de voisinage entre nœuds. Pour cela, je propose l'algorithme sui-

vant:

```
uvw = 0 ;
Pour (w compris entre 0 et nw-1) faire
   Pour (v compris entre 0 et nv-1) faire
      Pour (u compris entre 0 et nu-1) faire
         Mettre les bits next_u, next_v et next_w du voxel V(u,v,w) à 1 ;
         Si (u == nu-1) Mettre le bit next_u du voxel V(u,v,w) à 0 ;
         Si (v == nv-1) Mettre le bit next_v du voxel V(u,v,w) à 0 ;
         Si (w == nw-1) Mettre le bit next_w du voxel V(u,v,w) à 0 ;
      fin Pour
   fin Pour
fin Pour
```

ALGORITHME 4. Initialisation des voisinages dans un *voxet*.

Considérons maintenant le *voxel* V(u,v,w), ses six voisins sont obtenus de la manière suivante:

- V(u-1,v,w) est voisin de V(u,v,w) si le bit *next_u* de V(u-1,v,w) a la valeur 1.
- V(u+1,v,w) est voisin de V(u,v,w) si le bit *next_u* de V(u,v,w) a la valeur 1.
- V(u,v-1,w) est voisin de V(u,v,w) si le bit *next_v* de V(u,v-1,w) a la valeur 1.
- V(u,v+1,w) est voisin de V(u,v,w) si le bit *next_v* de V(u,v,w) a la valeur 1.
- V(u,v,w-1) est voisin de V(u,v,w) si le bit *next_w* de V(u,v,w-1) a la valeur 1.
- V(u,v,w+1) est voisin de V(u,v,w) si le bit *next_w* de V(u,v,w) a la valeur 1.

La création de discontinuités dans une grille revient alors à mettre à 0 certains de ces bits.

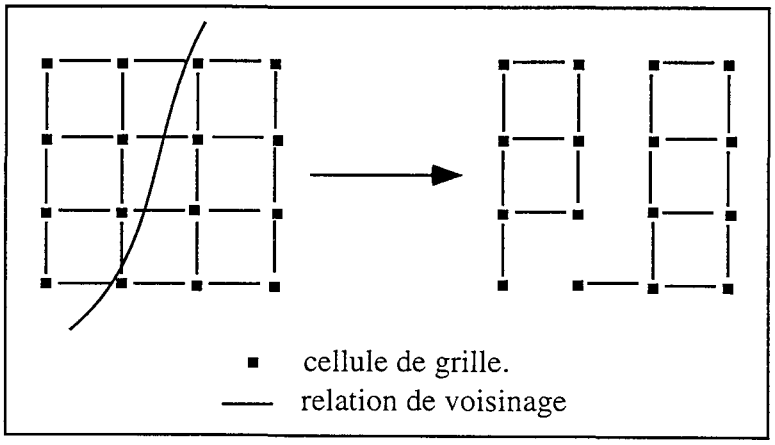


FIGURE 28. Prise en compte de discontinuités par codage explicite des voisinages.

Le coût en mémoire de ce type de codage est supérieur au précédent (trois bits par *voxel* au lieu d'un seul) et la recherche des voisins d'une cellule est plus complexe (les informations sont localisées sur quatre emplacements mémoire distincts).

Cependant, ce mode de codage est très intéressant car il présente l'avantage de dissocier les informations topologiques (de connectivité) des informations numériques. Il est en ce sens en accord parfait avec la philosophie de construction des objets géométriques de Gocad. C'est pourquoi ce codage a été choisi.

3.3 Communication avec les objets Gocad: le modèle de fonctionnement *sélecteur/affecteur*

Comme cela a été mentionné dans l'introduction du chapitre, le *voxet* est un outil permettant de convertir des modèles crées par Gocad à partir d'objets de type maillage irrégulier (surfaces triangulées, solides à base de tétraèdres, etc...), sous forme de grilles régulières. Les fonctionnalités de base du *voxet* mettent donc en relation une grille régulière avec un objet Gocad et visent à calculer l'intersection de ce dernier avec la grille.

Le chapitre présent est consacré à la description du schéma de fonctionnement du *voxet*, que nous avons baptisé «modèle *sélecteur/affecteur*».

Ce schéma sera illustré par un exemple trivial où l'on souhaite assigner un valeur constante C aux nœuds d'une sous-partie S de la grille définie par six paramètres u_{min}, u_{max}, v_{min}, v_{max}, w_{min} et w_{max}, de la manière suivante:

$$V(u,v,w) \in S \Leftrightarrow \begin{cases} u_{min} \leq u \leq u_{max} \\ v_{min} \leq v \leq v_{max} \\ w_{min} \leq w \leq w_{max} \end{cases}$$

Cet exemple nous permettra d'illustrer les différents éléments intervenant dans le modèle *sélecteur/affecteur*. Ces éléments sont au nombre de trois, l'*affecteur*, le *sélecteur et* l'*application* et communiquent entre eux par le biais d'une *structure de passage d'information* commune. Nous établirons ensuite les relations qui existent entre ces éléments.

3.3.1 L' *affecteur*

L'*affecteur* représente la modification apportée au contenu de la grille. Cette modification peut prendre différentes formes, par exemple:

- L'assignation d'une valeur numérique donnée à un nœud.
- La modification de la valeur d'une des unités binaires associées au nœud et décrites en page 43 (l'unité *painted*, *region*, etc...).
- La modification du voisinage d'un nœud.

Ces actions peuvent être combinées les unes aux autres. Ainsi, l'assignation d'une valeur donnée à un nœud de la grille ira souvent de pair avec la mise à 1 de son bit *painted*.

Dans l'exemple de la sous-grille, l'*affecteur* réalisera l'action consistant à «*affecter la valeur constante C à une entrée du tableau de la grille*».

3.3.2 Le *sélecteur*

Le *sélecteur* est chargé d'effectuer le calcul d'intersection d'une grille avec un objet externe. Pour cela, il sélectionne un sous-ensemble des nœuds de la grille qui sont concernés par l'intersection.

Dans l'exemple de la sous-grille, le *sélecteur* prend la forme suivante:

Pour (chaque *voxel* V(u,v,w) de la grille) **faire**

  Si ((umin ≤ u ≤ umax) et (vmin ≤ v ≤ vmax) et (wmin ≤ w ≤ wmax)) **faire**

    Selectionner le *voxel* V ;

  fin Si

fin Pour

ALGORITHME 5. Exemple type d'un *sélecteur*.

Le critère d'appartenance consiste donc ici à effectuer un test sur les coordonnées entières de chaque nœud afin de sélectionner ceux qui sont contenus dans le sous ensemble défini par (u_{min}, u_{max}, v_{min}, v_{max}, w_{min}, w_{max}).

On voit donc que le *sélecteur*, contrairement à l'*affecteur*, est spécifique de l'objet mis en relation avec le *voxet* (ici une sous-partie de celui-ci). Chaque objet aura donc un *sélecteur* associé.

3.3.3 L'*application*

L'*application* est l'élément contenant l'ensemble des informations nécessaires à toute intervention d'un objet sur un *voxet*. En particulier, elle contient:

- L'objet intervenant.
- L'ensemble des informations nécessaires à la modification de la grille.
- Le type de modification à effectuer (le type d'*affecteur* à employer).

Ainsi dans l'exemple de la sous-grille, c'est au niveau *application* que sont connues et stockées les coordonnées (u_{min}, u_{max}, v_{min}, v_{max}, w_{min}, w_{max}) qui la définissent. De même, si l'action consiste à assigner une valeur constante aux nœuds de cette sous-grille, c'est à ce niveau que sera connue cette valeur.

Il en résulte que, pour un objet donné, c'est à dire pour un *sélecteur* donné, la mise en œuvre d'un nouveau type de modification (d'un nouvel *affecteur*) passe par la création d'une nouvelle *application*, à même de stocker les nouvelles informations relatives à cette nouvelle action.

3.3.4 La structure de passage d'informations

Nous avons vu que le schéma de fonctionnement *sélecteur/affecteur* faisait intervenir trois composantes. Il va de soi que celles-ci vont devoir communiquer entre elles. En particulier, l'*affecteur* doit avoir accès aux informations issues de l'*application* (nature de l'action à entreprendre) et du *sélecteur* (quelles cellules sont concernées par cette action?).

Ceci nous amène à introduire le quatrième intervenant du modèle: la structure de passage d'informations ou structure *vx_info*. Cette structure est chargée de stocker les informations relatives à l'action à entreprendre au fur et à mesure que celles-ci sont disponibles. On distingue deux parties:

1. Les informations accessibles au niveau *application*: Dans le cas de la sous grille, il s'agit
   - du *voxet* sur lequel on travaille.
   - des paramètres (umin, umax, vmin, vmax, wmin, wmax) définissant l'objet extérieur.
   - de la valeur C à assigner aux nœuds selectionnés par le *sélecteur*.
2. Les informations accessibles au niveau *sélecteur*: Dans le cas de la sous-grille, il s'agira des coordonnées u, v et w des cellules selectionnées par celui-ci.

3.3.5 Les relations entre les quatre intervenants

Le schéma suivant résume le fonctionnement du modèle *sélecteur/affecteur*.

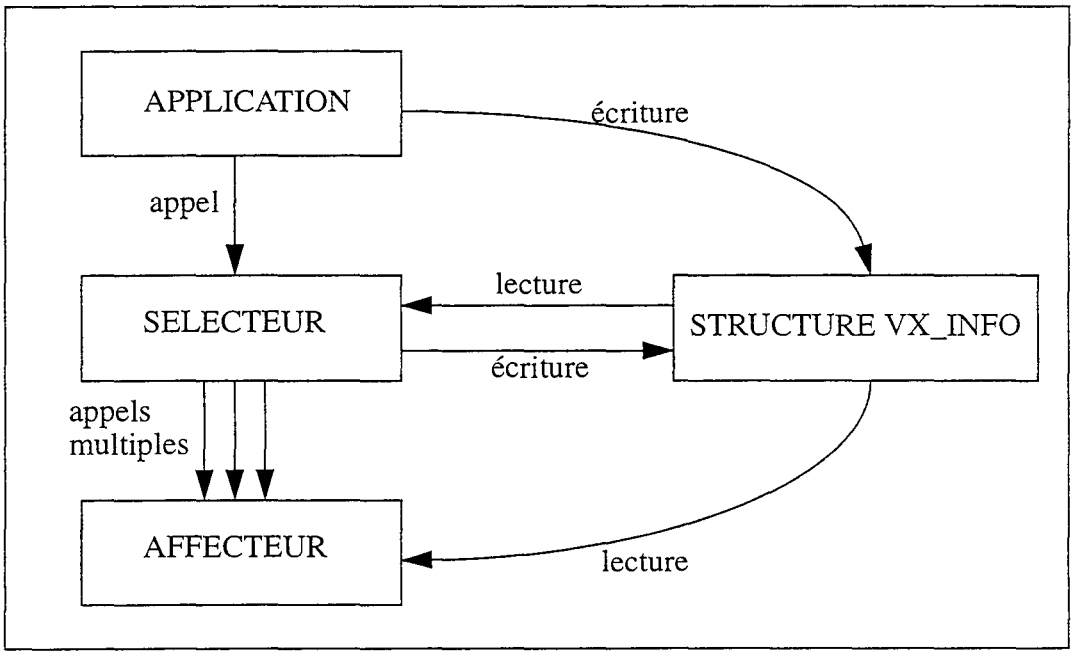


FIGURE 29. Le modèle de fonctionnement *sélecteur/affecteur*.

1. L'*application* écrit dans la structure *vx_info* les informations concernant le *voxet*, l'objet qui lui est associé, ainsi que l'*affecteur* à utiliser pour modifier la grille. Il effectue ensuite un appel au *sélecteur*.
2. Le *sélecteur*, à partir des informations fournies par l'*application* via la structure *vx_info*, sélectionne un sous-ensemble de cellules de la grille en fonction de critères relatifs à la géométrie de l'objet extérieur. Pour chacune de ces cellules, il transmet des informations à

la structure *vx_info*, (notamment ses coordonnées u, v et w) et effectue un appel à l'*affecteur* défini par l'*application*.

3. L'*affecteur* intervient au sommet de la chaîne et effectue la modification du contenu d'une cellule, à partir des informations disponibles dans la structure *vx_info* et qui ont été chargées par le *sélecteur* et l'*affecteur*.

3.3.6 Discussion

Le modèle de fonctionnement *sélecteur/affecteur* présente essentiellement deux avantages:

- Les actions de parcours et de recherche de cellules en fonction d'un objet donné sont séparées des interventions effectuées sur le contenu d'une grille. L'architecture de ce dispositif est très modulaire.

- A une fonction de parcours (un *sélecteur*) peuvent être associés plusieurs *affecteurs* sans que le *sélecteur* n'ait à être modifié. Ce modèle de fonctionnement est donc très évolutif. La mise en œuvre d'une nouvelle opération sur une grille par un objet dont le *sélecteur* a déjà été conçu ne suppose que de réecrire l'*application* et l'*affecteur*. Ces deux entités étant de très courts programmes le plus souvent, créer une nouvelle opération devient alors très simple.

3.4 Intersection d'un *voxet* avec un solide à base de tétraèdres

3.4.1 Caractéristiques d'un solide à base de tétraèdres

L'objet *tsolid*, ou solide à base de tétraèdres, est un objet volumique qui, par opposition au *voxet*, se présente sous la forme d'un maillage irrégulier composé:

- d'un ensemble Ω de nœuds N(x,y,z). Chacun de ces nœuds possède un voisinage, c'est à dire un sous-ensemble de Ω, dont les éléments sont alors «voisins de N».

- d'un ensemble de tétraèdres, dont les quatre sommets sont des nœuds du maillage. Ces tétraèdres s'agencent les uns aux autres par leurs faces, de telle sorte que chacun d'entre eux peut avoir au plus quatre tétraèdres voisins.

Tous les nœuds N de Ω appartiennent à un tétraèdre. De fait, chacun d'entre eux possède au moins trois voisins.

D'un point de vue informatique, il existe une convention qui régit la numérotation des quatre nœuds d'un tétraèdre. Soient N_0, N_1, N_2 et N_3 ces quatre nœuds numérotés de 0 à 3, le maillage d'un *tsolid* sera consistant si, pour chacun des tétraèdres, le produit mixte des vecteurs $\overrightarrow{N_0N_1}$,$\overrightarrow{N_0N_2}$et $\overrightarrow{N_0N_3}$ est de signe positif.

3.4.2 Routine de *sélection* associée à l'objet *tsolid*

Le problème consiste, étant donné un solide formé de tétraèdres et une grille régulière, à

sélectionner parmi les nœuds de la grille ceux qui sont contenus dans le solide, c'est à dire ceux qui sont contenus dans l'espace délimité par un de ses tétraèdres.

Pour cela, nous proposons l'algorithme suivant:

Pour (chaque tétraèdre T du solide S) **faire**

Passage des coordonnées des nœuds Ni(xi,yi,zi) de T dans le repère $(O',\vec{U},\vec{V},\vec{W})$

-> Ni(ui ,vi ,wi)

Calcul du sous-domaine S(umin, umax, vmin, vmax, wmin, wmax) de la grille contenant le tétraédre T.

Calcul des vecteurs $\overrightarrow{N_0N_1}$, $\overrightarrow{N_0N_2}$ et $\overrightarrow{N_0N_3}$ exprimés dans le repère $(O',\vec{U},\vec{V},\vec{W})$.

Calcul de la matrice de passage du repère $(O',\vec{U},\vec{V},\vec{W})$ au repère $(N_0, \overrightarrow{N_0N_1}, \overrightarrow{N_0N_2}, \overrightarrow{N_0N_3})$.

Pour chaque nœud N(u,v,w) tel que:

$((umin \leq u \leq umax)$ et $(vmin \leq v \leq vmax)$ et $(wmin \leq w \leq wmax))$ **faire**

Passage des coordonnées (u, v, w) de N dans le repère $(N_0, \overrightarrow{N_0N_1}, \overrightarrow{N_0N_2}, \overrightarrow{N_0N_3})$.

-> Ni(a, b, c)

Si $((0 \leq a \leq 1)$ et $(0 \leq b \leq 1)$ et $(0 \leq c \leq 1)$ et $(0 \leq a+b+c \leq 1))$ **alors**

Selectionner le nœud N.

fin Si

fin Pour

fin Pour

ALGORITHME 6. Calcul de l'intersection d'une grille avec un objet *tsolid*.

Le schéma de cet algorithme amène plusieurs commentaires:

La notion de repère de l'espace est essentielle et apparaît ici à trois niveaux:

- au niveau du repère universel $(\vec{O},\vec{I},\vec{J},\vec{K})$ dans lequel sont exprimées les coordonnées de la grille et des nœuds du *tsolid*.

- au niveau de la grille dont les coordonnées $(O',\vec{U},\vec{V},\vec{W})$ forment également un repère de l'espace.

- au niveau de chaque tétraèdre T où l'association du nœud N_0 et des vecteurs $(\overrightarrow{N_0N_1}, \overrightarrow{N_0N_2}, \overrightarrow{N_0N_3})$ forme une base.

Les changements de repère sont utilisés très fréquemment. En particulier:

- Le passage des coordonnées des nœuds des tétraèdres du repère universel vers le repère de la grille permet de déterminer la boite englobante entière de ce tétraèdre, et ainsi d'isoler un

sous ensemble de la grille S(umin, umax, vmin, vmax, wmin, wmax) dont les paramètres sont définis de la manière suivante:

$$u_{min} = Int(\underset{i}{Min}\{u_{N_i}\} + 1) \qquad u_{max} = Int(\underset{i}{Max}\{u_{N_i}\})$$

$$v_{min} = Int(\underset{i}{Min}\{v_{N_i}\} + 1) \qquad v_{max} = Int(\underset{i}{Max}\{v_{N_i}\})$$

$$w_{min} = Int(\underset{i}{Min}\{w_{N_i}\} + 1) \qquad w_{max} = Int(\underset{i}{Max}\{w_{N_i}\})$$

Ceci permet par la suite de n'effectuer le test d'appartenance d'un nœud à un tétraèdre que sur les nœuds N(u, v, w) vérifiant:

$$((umin \leq u \leq umax) \text{ et } (vmin \leq v \leq vmax) \text{ et } (wmin \leq w \leq wmax))$$

En effet, les nœuds ne vérifiant pas cette condition sont forcément à l'extérieur du tétraèdre puisqu'ils sont par définition à l'extérieur d'un volume parallélépipédique contenant ce tétraèdre.

- Par ailleurs, le test d'appartenance d'un nœud à un tétraèdre passe lui aussi par un changement de base, celui entre le repère de la grille et le repère du tétraèdre. Pour chaque nœud N(u, v, w), on effectue ce changement de repère et on obtient le triplet (a,b,c) des coordonnées barycentriques du nœud dans le tétraèdre. La condition d'appartenance est alors: $((0 \leq a \leq 1) \text{ et } (0 \leq b \leq 1) \text{ et } (0 \leq c \leq 1) \text{ et } (0 \leq a + b + c \leq 1))$

- Dans le but de s'affranchir des problèmes de précision numérique inhérents à tout calcul géométrique ([40],[41]), nous conseillons d'effectuer en cours de calcul une transformation sur les coordonnées des sommets des tétraèdres. Cette transformation est un affinité de centre le barycentre des quatre sommets du tétraèdre, et de facteur 1.01 (déterminé empiriquement).
Il en résulte une légère modification de la géométrie des tétraèdres mais ceci permet de résoudre des problèmes d'erreurs d'arrondi, tels que le cas de figure décrit ci-dessous, que nous avons rencontré fréquemment:
considérons le cas d'un point de l'espace appartenant, au sens mathématique, à une face commune à deux tétraèdres. Le test d'appartenance de ce point aux tétraèdres peut, du fait des erreurs d'arrondi, déboucher sur un résultat négatif dans les deux cas. On voit bien alors que ceci introduirait une erreur majeure dans le résultat final.

3.4.3 Algorithmes d'*affectation* associés

Conversion de l'objet *tsolid* sous forme de maillage régulier

La première application qui vient à l'esprit est de traduire le volume contenu dans un *tsolid* sous la forme d'un ensemble de voxels. Le travail consiste ici simplement à marquer avec une valeur d'indice donnée les cellules sélectionnées par le *sélecteur*. L'*affecteur* en question pren-

dra donc la forme suivante:

Pour (chaque voxel V(uvw)) **faire**

Affecter une valeur constante C dans l'entrée d'indice uvw du tableau de la grille.

et/ou

Mettre le bit *painted* correpondant à V à la valeur 1.

et/ou

Mettre le bit *region* correpondant à V à la valeur 1.

fin Pour

ALGORITHME 7. Conversion d'un *tsolid* sous la forme d'un ensemble de voxels.

L'application essentielle de cet *affecteur* est la définition d'un sous-ensemble des cellules d'une grille régulière à partir d'une géométrie existante. Ceci peut permettre de définir une sous région de la grille au sein de laquelle sera limitée l'étendue de toutes les opérations de calcul effectuées à postériori.

Echantillonage d'une propriété physique connue aux nœuds d'un *tsolid*

Les maillages à base de tétraèdres sont utilisés fréquemment comme support de calcul par éléments finis. Il en résulte que, au niveau des sommets de ces maillages, on dispose souvent, en plus des informations géométriques (coordonnées x,y,z), de paramètres physiques (conductivités thermiques, tenseurs de contraintes, température, etc...) fournis ou estimés par les méthodes de calcul citées ci-dessus.

Il apparaît ici intéressant de soulever le problème de l'échantillonage d'un paramètre connu aux nœuds d'un *tsolid* sur un maillage de type grille régulière. Celui-ci s'exprime de la manière suivante: étant donné un *tsolid* aux nœuds duquel est connu un paramètre φ, et une grille régulière de géométrie connue, à quels nœuds de la grille doit on affecter une valeur de φ et quelle valeur doit on leur affecter?

La détermination des nœuds concernés à déjà été décrite et s'effectue au niveau *sélecteur*. C'est au niveau *affecteur* que se fait l'estimation de la valeur de φ à affecter à ces nœuds. Pour

cela, nous proposons la stratégie suivante:

Pour (chaque nœud N(uvw) contenu dans le tétraèdre T) **faire**

Récupérer le tableau des valeurs de φ aux sommets de T.

$$\rightarrow \quad \varphi_0(T),\varphi_1(T),\varphi_2(T),\varphi_3(T)$$

Récupérer les coordonnées barycentriques (a,b,c) du nœud N dans T.

Assigner au nœud d'indice uvw la valeur:

$$\varphi(N) = a \cdot \varphi_1(T) + b \cdot \varphi_2(T) + c \cdot \varphi_3(T) + (1 - a - b - c) \cdot \varphi_0(T)$$

Mettre le bit *painted* du nœud N à 1.

fin Pour

ALGORITHME 8. Echantillonage d'un paramètre physique connu aux nœuds d'un tétraèdre sur une grille régulière

Dans ce schéma, le *sélecteur* transmet à l'*affecteur* les coordonnées barycentriques de chaque nœud par rapport aux quatre sommets du tétraèdre qui le contient, ainsi que les quatre valeurs de φ correspondant à ces quatre sommets. La quantité φ assignée au nœud est la somme des quatre valeurs $\varphi_i(T)$ pondérées par la coordonnée barycentrique qui leur correspond.

Ce mode de calcul présente l'avantage d'être très simple. En effet, le calcul des coordonnées barycentriques n'est pas en lui même très coûteux et il a déjà été effectué au niveau *sélecteur* pour tester l'appartenance des nœuds de grille aux tétraèdres.

Evaluation des performances

Des tests de performances ont été effectués sur un exemple synthétique mais néanmoins réaliste en terme de taille des objets. Nous disposons ici d'un *tsolid* constitué de 3158 sommets et de 10512 tétraèdres (voir la planche 2 de la page 60). En chacun de ses sommets est connue une valeur d'un paramètre φ que l'on va échantillonner sur une grille contenant le TSOLID (voir la planche 3 de la page 60). On veut étudier le temps que prend cette opération quand on augmente le nombre de nœuds de la grille.

Les résultats obtenus sont les suivants:

Tableau .2 : Tests d'échantillonage à partir de l'objet *tsolid*.

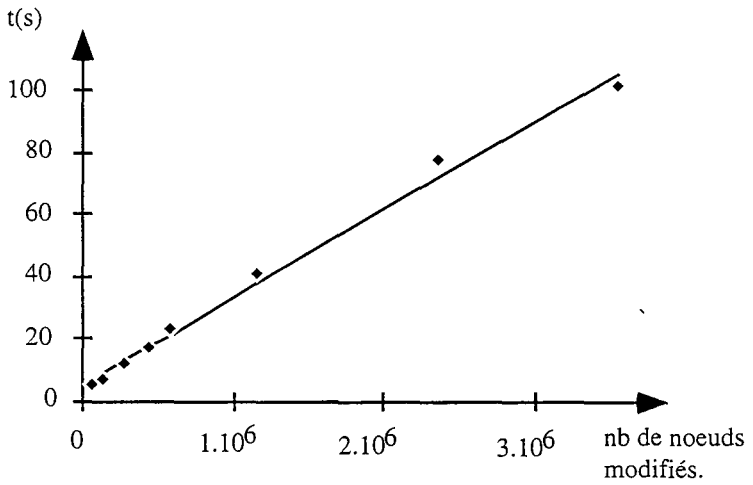
test n°	1	2	3	4
taille de la grille	50*50*50	100*50*50	100*100*50	91*91*91
nombre de nœuds	125000	250000	500000	753571
temps de calcul (s)	5	7	12	17
nb de nœuds modifiés	71175	143814	290675	438462

Tableau .2 : Tests d'échantillonage à partir de l'objet *tsolid*.

test n°	1	2	3	4
% de nœuds modifiés	56.94	57.52	58.135	58.18

test n°	5	6	7	8
taille de la grille	100*100*100	126*126*126	159*159*159	182*182*182
nombre de nœuds	1000000	2000376	4019679	6028568
temps de calcul (s)	23	41	77	101
nb de nœuds modifiés	581678	1169375	2357727	3540380
% de nœuds modifiés	58.16	58.45	58.65	58.72

L'évolution du temps de calcul en fonction de la taille de la grille est la suivante:



Cette figure montre clairement que le temps de calcul évolue linéairement en fonction du nombre de nœuds modifiés. De plus, compte tenu que l'objet tsolid utilisé est de taille réaliste, le temps de calcul absolu de 28.4 secondes par million de nœuds, obtenu avec le matériel «extreme» (voir page 149), représente une valeur tout a fait convaincante, et permet d'envisager l'utilisation de cet algorithme sur des grilles de taille plus conséquente.

Cette rapidité d'exécution est tout à fait intéressante. En effet, dans certaines applications telles que les méthodes de traitement sismique, on travaille sur des grilles régulières de très grande taille (de l'ordre de la centaine de nœuds dans les trois dimensions). Cet aspect des cho-

ses prend alors une importance prépondérante.

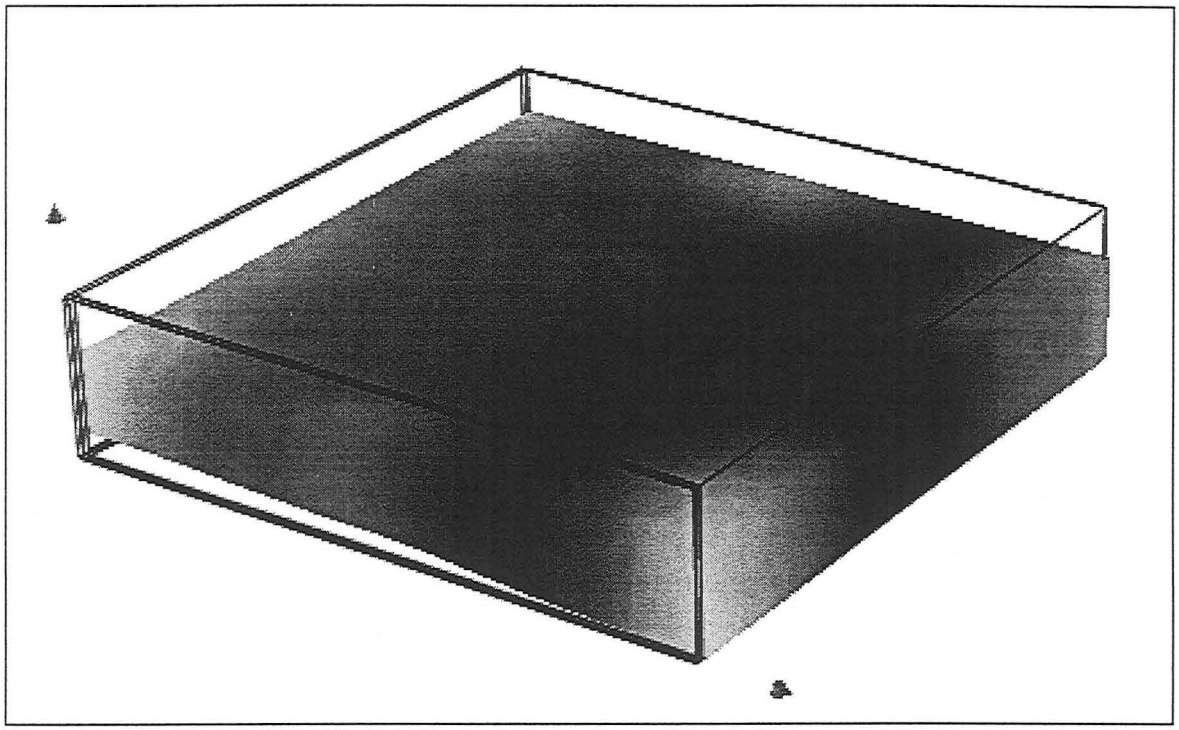


PLANCHE 2. Un exemple d'objet *tsolid* avec propriété.

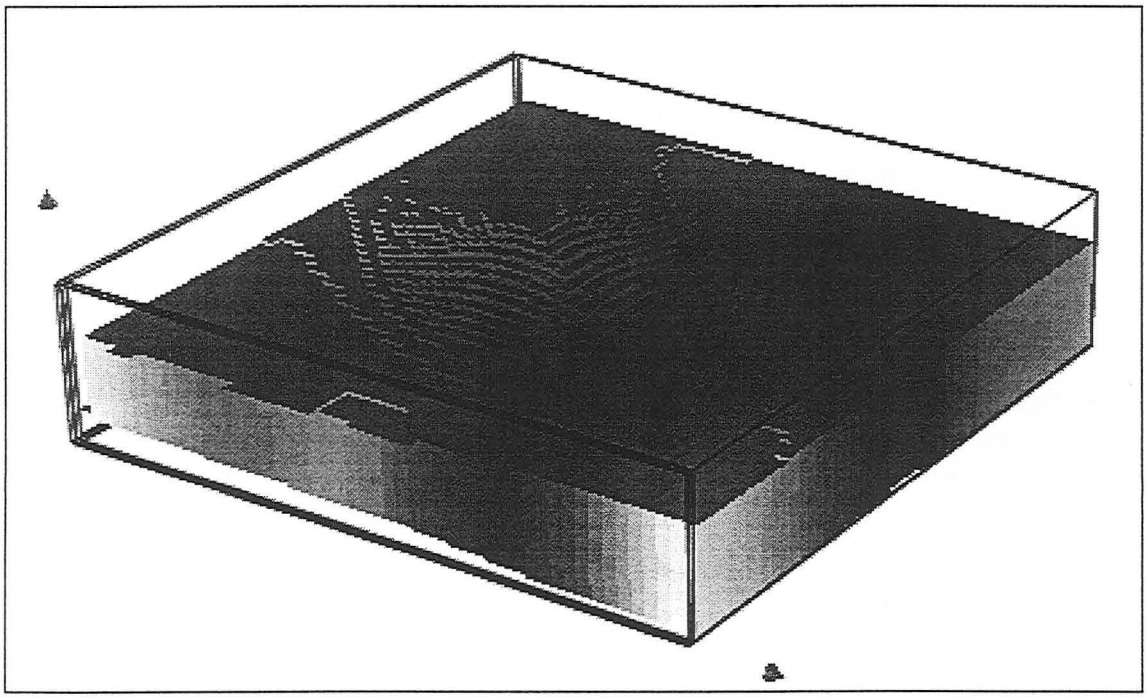


PLANCHE 3. Représentation de l'ensemble des voxels d'une grille de taille 50*50*50, après échantillonage de la propriété du *tsolid* sur celle-ci.

Etude de la continuité de φ

La fonction φ(x,y,z) utilisée ci-dessus est définie de manière locale pour chaque tétraèdre T. Son expression est une combinaison linéaire de x,y et z dont les facteurs s'expriment en fonction des quatre valeurs $\varphi_0(T),\varphi_1(T),\varphi_2(T)$ et $\varphi_3(T)$. φ(x,y,z) est donc linéaire par morceaux.

Il semble alors important d'étudier la continuité de cette fonction φ de part et d'autre d'une face commune à deux tétraèdres.

Soient T_1 et T_2 deux tétraèdres adjacents par une face. Pour simplifier la démonstration, on adoptera les conventions de numérotation des sommets des tétraèdres suivantes:

- les nœuds communs à T_1 et T_2 seront numérotés de 1 à 3, chaque nœud possédant le même indice dans T_1 et T_2 .
- Les nœuds non partagés par T_1 et T_2 sont numérotés avec l'indice 0.

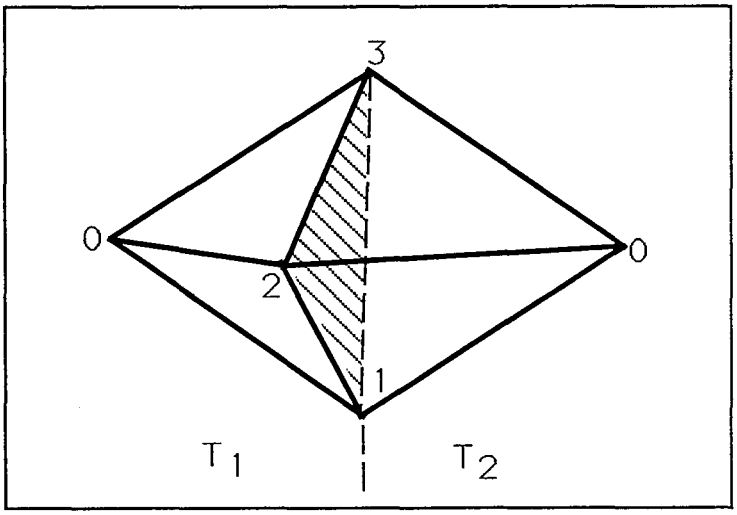


FIGURE 30. Convention de numérotation des sommets de T_1 et T_2.

Les expresions de φ dans T_1 et T_2, exprimées à partir des coordonnées barycentriques sont les suivantes:

$$\varphi(T_1)= a_{T1}\cdot\varphi_1(T_1) + b_{T1}\cdot\varphi_2(T_1) + c_{T1}\cdot\varphi_3(T_1) + (1 - a_{T1} - b_{T1} - c_{T1})\cdot\varphi_0(T_1)$$

$$\varphi(T_2)= a_{T2}\cdot\varphi_1(T_2) + b_{T2}\cdot\varphi_2(T_2) + c_{T2}\cdot\varphi_3(T_2) + (1 - a_{T2} - b_{T2} - c_{T2})\cdot\varphi_0(T_2)$$

Or, d'après les conventions de numérotation:

$$\varphi_1(T_1)= \varphi_1(T_2)= \varphi_1 \qquad \varphi_2(T_1)= \varphi_2(T_2)= \varphi_2 \qquad \varphi_3(T_1)= \varphi_3(T_2)= \varphi_3$$

Au voisinage de la face commune à T_1 et T_2 , on a:

$$a_{T1} + b_{T1} + c_{T1} \to 1 \quad \Rightarrow \quad 1 - a_{T1} - b_{T1} - c_{T1} \to 0$$
$$a_{T2} + b_{T2} + c_{T2} \to 1 \quad \Rightarrow \quad 1 - a_{T2} - b_{T2} - c_{T2} \to 0$$

Du fait des conventions de numérotation, on a également:

$$a_{T1} \to a_{T2} \qquad b_{T1} \to b_{T2} \qquad c_{T1} \to c_{T2}$$

Soient a, b et c les valeurs communes vers lesquelles tendent ces trois expressions. Au voisinage d'un point de la face commune aux deux tétraèdres, on a donc:

$$\varphi(T_1) \to a \cdot \varphi_1 + b \cdot \varphi_2 + c \cdot \varphi_3$$
$$\varphi(T_2) \to a \cdot \varphi_1 + b \cdot \varphi_2 + c \cdot \varphi_3$$

La fonction φ définie localement dans chaque tétraèdre est donc continue de part et d'autre des faces communes à deux tétraèdres. Elle est donc continue sur l'ensemble du volume délimité par un objet *tsolid*.

Etude de la continuité de φ'

Nous reprendrons ici le même cas d'étude que précédemment, ainsi que les mêmes conventions de numérotation.

L'expression de $\frac{\partial \varphi}{\partial x}$ dans les tétraèdres T_1 et T_2 est:

$$\left(\frac{\partial \varphi}{\partial x}\right)_{T_1} = \frac{\partial \varphi}{\partial a_{T_1}} \cdot \frac{\partial a_{T_1}}{\partial x} = (\varphi_1(T_1) - \varphi_0(T_1)) \cdot \frac{\partial a_{T_1}}{\partial x}$$
$$\left(\frac{\partial \varphi}{\partial x}\right)_{T_2} = \frac{\partial \varphi}{\partial a_{T_2}} \cdot \frac{\partial a_{T_2}}{\partial x} = (\varphi_1(T_2) - \varphi_0(T_2)) \cdot \frac{\partial a_{T_2}}{\partial x}$$

$\frac{\partial a_{T_1}}{\partial x}$ et $\frac{\partial a_{T_2}}{\partial x}$ sont des constantes dont la valeur dépend des positions des quatre sommets du tétraèdre considéré, c'est à dire ici de la position du nœud «libre» N_0 (les trois autres nœuds étant communs aux deux tétraèdres).

Les dérivées partielles de φ par rapport à x, y et z sont donc constantes à l'intérieur de chaque tétradre. Leur valeur dépend à la fois de la position des nœuds «libres» de chaque tétraèdre et de la valeur de φ qui leur est affectée.

Ces deux quantités étant totalement indépendantes, nous aurons tout lieu de considérer que les dérivées partielles de φ sont discontinues de part et d'autre d'une face commune à deux tétraèdres au sein d'un même objet *tsolid*.

3.5 Intersection d'un *voxet* avec une surface triangulée

3.5.1 Caractéristiques d'une surface triangulée

Une surface triangulée, ou *tsurf*, est, par opposition au *voxet*, un objet surfacique se présentant sous la forme d'un maillage irrégulier composé:

- d'un ensemble de nœuds N(x,y,z). Chacun de ces nœuds possède un voisinage, qui est un sous-ensemble des nœuds du maillage . Chacun des éléments de ce sous-ensemble est alors un «voisin de N» ou encore un «satellite de N».

- d'un ensemble de triangles dont les trois sommets sont des nœuds du maillage. Ceux-ci s'agencent les uns aux autres par leurs arêtes. Ainsi chaque triangle peut avoir au plus trois triangles adjacents. Chaque nœud N du maillage appartient à un triangle et possède donc au minimum deux satellites.

D'un point de vue informatique, il existe une convention de numérotation des trois sommets d'un triangle. Soient N_0,N_1 et N_2 ces trois sommets, la convention impose que ceux-ci soient numérotés de telle manière que les normales des triangles, définies comme le produit vectoriel $\overrightarrow{N_0N_1} \wedge \overrightarrow{N_0N_2}$, soient toutes orientées du même coté que la surface.

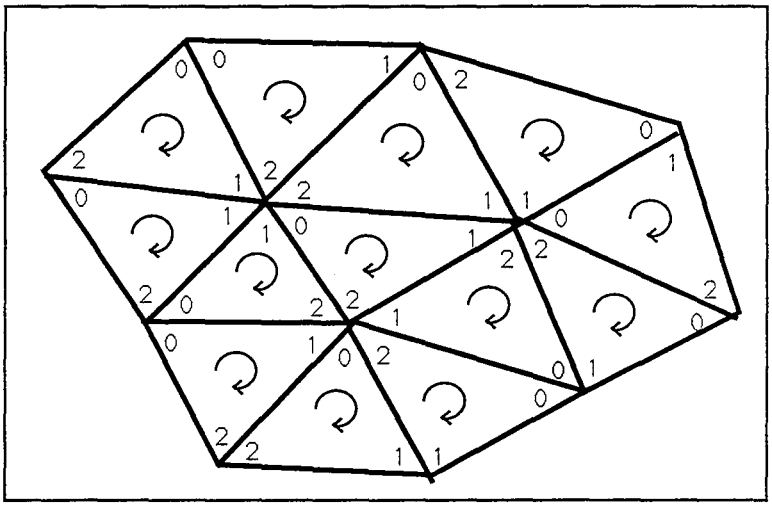


FIGURE 31. Convention de numérotation des sommets des triangles.

3.5.2 Routine de *sélection* associée à l'objet *tsurf*

Le problème diffère de celui du *tsolid* vu précédemment en ce sens que l'on cherche ici à calculer l'intersection entre un objet surfacique (la surface triangulée ou *tsurf*) et un objet volumique (la grille régulière ou *voxet*). Il ne s'agit donc pas ici d'un problème d'appartenance de points à un sous-domaine de l'espace.

La notion de grille régulière assimilable à un écran 3D prend ici tout son sens. En effet, si l'on considère un écran d'ordinateur, celui-ci peut être vu comme une grille régulière à deux

dimensions dont chaque nœud serait l'équivalent d'un pixel. Représenter un segment, c'est à dire un objet à une dimension, sur cet écran à deux dimensions revient donc à allumer certains pixels en fonction des coordonnées de ce segment de façon à produire une image du segment la plus fidèle possible.

De la même manière, représenter une surface triangulée (c'est à dire un objet à deux dimensions) sur une grille régulière à trois dimensions reviendra à sélectionner un sous-ensemble de voxels de la grille dont la réunion produira une image de la surface dans la grille la plus fidèle possible.

De même que l'on dispose de plusieurs algorithmes pour représenter un segment sur un écran d'ordinateur, il est bien clair que l'algorithme que nous allons proposer pour donner une image d'une *tsurf* dans une grille n'est pas le seul envisageable. Il représente cependant à nos yeux un bon compromis entre efficacité et simplicité de conception.

Le schéma de cet algorithme, présenté en page 65, amène les commentaires suivants:

- on retrouve l'idée de convertir les coordonnées des sommets des triangles dans le repère $(O',\vec{U},\vec{V},\vec{W})$ de la grille. Ceci permet d'isoler un sous-ensemble S(umin, umax, vmin, vmax, wmin, wmax) contenant le triangle T.

- Une fois ce sous ensemble déterminé, on procède, pour chacune des trois directions principales de la grille à des calculs d'intersection de droites avec le triangle courant. Dans le cas de la direction $\vec{U}$ (resp. $\vec{V},\vec{W}$), ces droites sont définies par:

• un point origine dont les deux coordonnées (v,w) (resp. (u,w),(u,v)) varient dans la limite du sous-domaine S, la troisième coordonnée étant fixée à 0.

• un vecteur: ici $\vec{U}$ (resp. $\vec{V},\vec{W}$).

- Dans le cas où l'une de ces droites intersecte le triangle courant et ce dans les limites géométriques de la grille, on récupère l'abscisse curviligne du point d'intersection sur la droite de tir. Ceci donne la position du voxel à sélectionner et on définit ainsi le critère de proximité des nœuds par rapport aux triangles de la surface.

Dans le but de s'affranchir des problèmes de précision numérique inhérents à tout calcul géométrique, nous conseillons d'effectuer en cours de calcul une transformation sur les coordonnées des sommets des triangles. Cette transformation est un affinité de centre le barycentre des trois sommets du triangle et de facteur 1.01 (déterminé empiriquement). Pour plus de précisions sur ce sujet, on se reportera au chapitre correspondant dans le cas de l'objet *tsolid*, où le type de problème que l'on peut rencontrer a été décrit en détail .

Pour (chaque triangle $T(N_0, N_1, N_2)$ de la surface S) **faire**

Passage des coordonnées des nœuds Ni(xi,yi,zi) de T dans le repère $(O',\vec{U},\vec{V},\vec{W})$

-> Ni(ui ,vi ,wi)

Calcul du sous-domaine S(umin, umax, vmin, vmax, wmin, wmax) de la grille contenant le triangle T.

Mettre la direction courante de tir à $\vec{U}$.

Pour (chaque nœud N(u,v,w) tel que $(vmin \leq v \leq vmax)$ et $(wmin \leq w \leq wmax)$) **faire**

Si ((la droite ((0, v, w), $\vec{U}$) intersecte T exprimé dans la base $(O',\vec{U},\vec{V},\vec{W})$) et

(l'abscisse curviligne s du point d'intersection est comprise entre 0 et nu-1))

Selectionner le nœud $N(Int(s + 0 \cdot 5), v, w)$

fin Si.

fin Pour

Mettre la direction courante de tir à $\vec{V}$.

Pour (chaque nœud N(u,v,w) tel que $(umin \leq u \leq umax)$ et $(wmin \leq w \leq wmax)$) **alors**

Si ((la droite ((u, 0, w), $\vec{V}$) intersecte T exprimé dans la base $(O',\vec{U},\vec{V},\vec{W})$) et

(l'abscisse curviligne s du point d'intersection est comprise entre 0 et nv-1))

Selectionner le nœud $N(u, Int(s + 0 \cdot 5), w)$

fin Si

fin Pour

Mettre la direction courante de tir à $\vec{W}$.

Pour (chaque nœud N(u,v,w) tel que $(umin \leq u \leq umax)$ et $(vmin \leq v \leq vmax)$) **faire**

Si ((la droite ((u, v, 0), $\vec{W}$) intersecte T exprimé dans la base $(O',\vec{U},\vec{V},\vec{W})$) et

(l'abscisse curviligne s du point d'intersection est comprise entre 0 et nw-1))

Selectionner le nœud $N(u, v, Int(s + 0 \cdot 5))$

fin Si

fin Pour

fin Pour

ALGORITHME 9. Calcul de l'intersection d'une grille régulière avec un objet *tsurf*.

-

3.5.3 Algorithmes d'*affectation* associés

Coupure d'une grille régulière par une surface

Le problème est relativement simple à exprimer. En utilisant le mécanisme mis en place pour gérer la notion de discontinuité entre cellules, on souhaite supprimer des voisinages dans une grille de telle manière que les cellules situées d'un coté d'une surface triangulée S donnée soient isolées de celles situées de l'autre coté.

D'un point de vue informatique, le problème revient à:

- Affecter la valeur 0 au bit *next_u* de chaque nœud V(u,v,w) de la grille si le segment $[V(u,v,w),V(u+1,v,w)]$ intersecte un triangle de la surface S.
- Affecter la valeur 0 au bit *next_v* de chaque nœud V(u,v,w) de la grille si le segment $[V(u,v,w),V(u,v+1,w)]$ intersecte un triangle de la surface S.
- Affecter la valeur 0 au bit *next_w* de chaque nœud V(u,v,w) de la grille si le segment $[V(u,v,w),V(u,v,w+1)]$ intersecte un triangle de la surface S.

La routine mise en place est la suivante:

Récupérer l'abscisse curviligne s et les coordonnées entières du point d'intersection.

Suivant (la direction courante de tir du *sélecteur*)

Dans le cas $(\vec{U})$

$uvw = Int(s) + n_u \cdot v + n_u \cdot n_v \cdot w$

Mettre le bit *next_u* du voxel V(uvw) à 0;

break ;

Dans le cas $(\vec{V})$

$uvw = u + n_u \cdot Int(s) + n_u \cdot n_v \cdot w$

Mettre le bit *next_v* du voxel V(uvw) à 0;

break;

Dans le cas $(\vec{W})$

$uvw = u + n_u \cdot v + n_u \cdot n_v \cdot Int(s)$

Mettre le bit *next_w* du voxel V(uvw) à 0;

break;

fin Suivant

ALGORITHME 10. Coupure d'une grille régulière par une surface triangulée.

Le *sélecteur* transmet à l'*affecteur* la direction courante de la droite d'intersection avec le triangle. Dans le cas de la direction $\vec{U}$ (resp. $\vec{V}$,$\vec{W}$), il transmet également les coordonnées entières connues de la cellule à considérer, c'est à dire ici le couple (v,w) (resp. (u,w),(u,v)). Enfin, la troisième coordonnée entière u (resp. v,w) est obtenue en prenant la partie entière de l'abscisse curviligne s du point d'intersection. Toujours dans le cas de la direction $\vec{U}$ (resp. $\vec{V}$,$\vec{W}$), le bit NEXT_U (resp NEXT_V, NEXT_W) du voxel v(u,v,w) est mis à la valeur 0, traduisant ainsi la perte pour cette cellule de son voisin immédiat dans la direction considérée.

Evaluation des performances

Des tests de performances ont été effectués sur un exemple réaliste en terme de nombre et de tailles d'objets. Cet exemple correspond au modèle surfacique de chevauchement, qui a été utilisé dans le cadre du projet *SEG/EAEG modelling Project* ([3]). Nous disposons ici d'un ensemble de surfaces (13 horizons et 3 failles) représentant un modèle géologique complet. L'ensemble des surfaces représente un total de 31000 triangles que l'on va utiliser pour compartimenter une grille dont on fait varier le nombre de nœuds sans en changer la position. On veut étudier le temps que prend cette opération quand on augmente le nombre de nœuds de la grille.

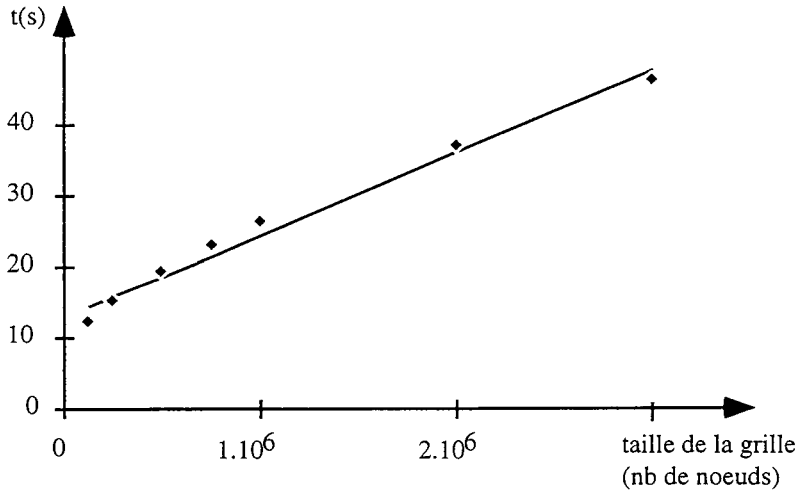
Les résultats obtenus sont les suivants:

Tableau .3 : Tests de coupure d'une grille par des surfaces.

test n°	1	2	3	4	5
taille de la grille	50*50*50	63*63*63	79*79*79	91*91*91	100*100*100
nombre de nœuds	125000	250047	493039	753571	1000000
temps de calcul sur *extreme* (s)					6
temps de calcul sur *indigo* (s)	12	15	19	23	26

test n°	6	7	8	9
taille de la grille	126*126*126	144*144*144	159*159*159	182*182*182
nombre de nœuds	2000376	2985984	4019679	6028568
temps de calcul sur *extreme* (s)	9	10	12	15
temps de calcul sur *indigo* (s)	37	46		

L'évolution du temps de calcul en fonction de la taille de la grille est la suivante:



Commentaires

Les tests on été effectués sur deux types de matériel (voir page 149). Ceux effectués sur la machine «extreme» rendent compte du temps nécessaire en valeur absolue à l'heure actuelle pour effectuer l'opération décrite plus haut. Ceux effectués sur la machine «indigo» l'ont été pour obtenir une gamme de temps de calcul plus large pour rendre mieux compte de la complexité de l'algorithme.

Le graphe de performances montre clairement que le temps de calcul évolue linéairement en fonction de la taille de la grille.

Il est à noter à ce sujet que nous avons fait évoluer le nombre de nœuds de la grille de manière constante dans les trois dimmensions. En effet, nous avons affaire ici à des objets surfaciques et le nombre de connections rompues par l'algorithme n'est pas le même dans les trois directions. Si l'on veut exprimer le temps de calcul en fonction du nombre de nœuds total de la grille, il convient donc d'augmenter proportionnellement les trois nombres de nœuds dans les trois dimmensions de façon à ne pas privilégier tel ou tel type de connection dans une même série de tests.

Les valeurs de temps de calcul absolues obtenues sur la machine «extreme», c'est à dire sur du matériel récent (année 1995), montrent que cet algorithme est très rapide compte tenu de la taille du modèle surfacique employé, et permettent d'envisager son utilisation sur des grilles de taille beaucoup plus importante.

Les applications de cette routine sont très nombreuses est très intéressantes. D'une manière générale, celle-ci permet d'utiliser un modèle géométrique à base de surfaces triangulées pour compartimenter l'espace 3D, décrit au moyen d'une grille régulière.

Cette idée de compartimenter l'espace en sous-éléments n'est pas sans rappeler la théorie de Weiler ([30],[46],[47]). Les finalités sont en effet identiques, à ceci près que le modèle de Weiler conserve les maillages irréguliers comme support alors que l'approche présentée ici traduit

le résultat final sous forme de maillage régulier.

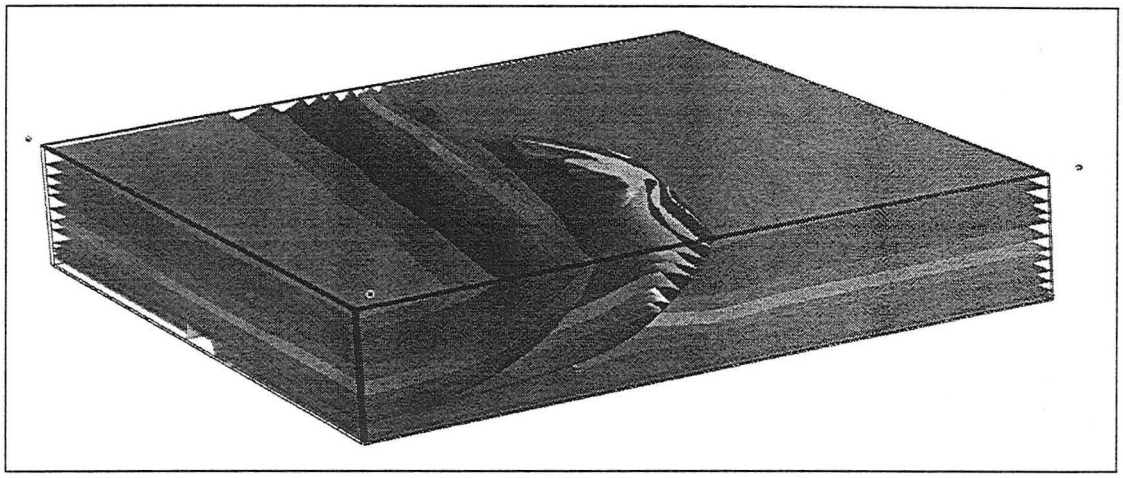


PLANCHE 4. Modèle surfacique utilisé pour la série de tests.

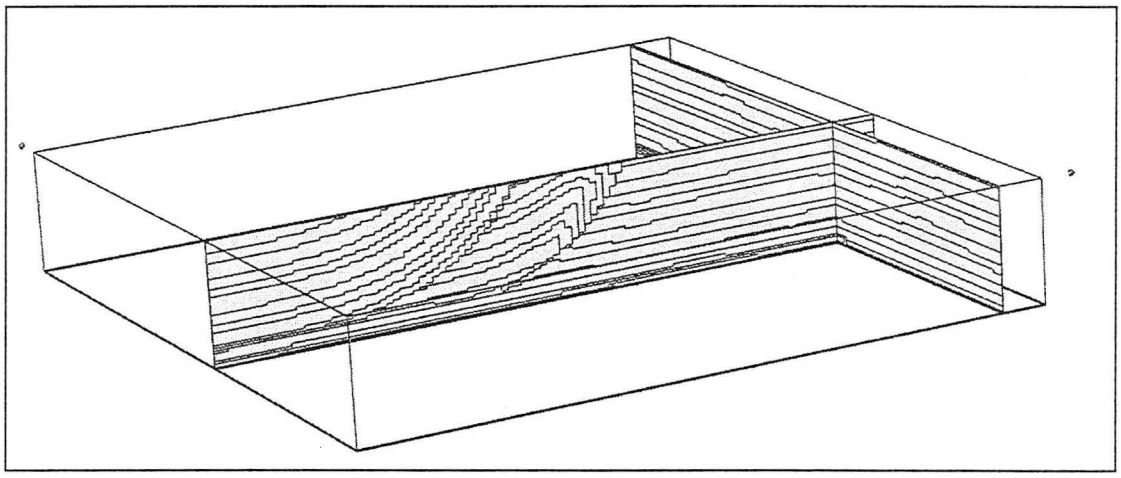


PLANCHE 5. Résultat de la coupure d'une grille par le modèle surfacique. Les ruptures de connexions sont représentées sur deux plans de cette grille.

Conversion de l'objet *tsurf* sous forme de grille régulière

Le problème consiste ici à convertir une surface triangulée sous forme de grille régulière. On veut donc représenter une image de la surface dans la grille, la plus fidèle possible de l'objet initial.

Le *sélecteur* présenté dans le chapitre précédent a permis de sélectionner les cellules de la grille les plus proches de la surface. L'*affecteur* sera chargé de les marquer. Pour cela, nous

proposons la routine suivante.

Récupérer l'abscisse s et les coordonnées entières connues du point d'intersection.

Suivant (la direction courante de tir du *sélecteur*)

Dans le cas $(\vec{U})$

$u = \mathrm{Int}(s + 0 \cdot 5)$

break ;

Dans le cas $(\vec{V})$

$v = \mathrm{Int}(s + 0 \cdot 5)$

break;

Dans le cas $(\vec{W})$

$w = \mathrm{Int}(s + 0 \cdot 5)$

break;

fin Suivant

$uvw = u + v \cdot nu + w \cdot nu \cdot nv$

Affecter une valeur constante C dans l'entrée d'indice uvw du tableau de la grille.

et/ou

Mettre le bit *painted* correpondant à V à la valeur 1.

et/ou

Mettre le bit *region* correpondant à V à la valeur 1.

ALGORITHME 11. Conversion d'une surface triangulée en grille régulière

On cherche à marquer les cellules les plus proches de la surface. L'abscisse s du point d'intersection est donc arrondie à la valeur entière la plus proche, contrairement à l'application précédente où l'on prenait sa partie entière pour marquer la cellule contenant une information de voisinage.

L'application essentielle de cet *affecteur* est la définition d'un sous-ensemble des cellules d'une grille régulière, à partir d'une géométrie existante. Ceci peut permettre de définir une sous région de la grille pour limiter par la suite l'étendue de toutes les opérations de calcul à ce domaine.

Echantillonage d'une propriété physique connue aux nœuds d'une *tsurf*

Les surfaces triangulées sont utilisées fréquemment comme support de calculs par éléments finis. Il en résulte que, au niveau des sommets de ces maillages, on dispose souvent, en plus d' informations géométriques (coordonnées x,y,z), de paramètres physiques. Il apparaît donc inté-

ressant, comme cela a été fait dans le cas de l'objet *tsolid*, de soulever le problème de l'échantillonage de ces paramètres sur une grille régulière».

Le problème va ici consister à affecter aux cellules sélectionnées une valeur de ce paramè tre. Pour cela, nous proposons la routine suivante:

Récupérer les trois coordonnées barycentriques bc[3] du point d'intersection dans le triangle courant.

Récupérer les trois valeurs $\varphi[3]$ du paramètre φ aux trois sommets du triangles.

Affecter au voxel V(u,v,w) la quantité:

$$bc[0] \cdot \varphi[0] + bc[1] \cdot \varphi[1] + bc[2] \cdot \varphi[2]$$

ALGORITHME 12. Echantillonage d'un paramètre physique connu aux nœuds d'une surface triangulée sur une grille régulière

Le *sélecteur* transmet à l'*affecteur* les coordonnées barycentriques du point d'intersection courant dans le triangle, ainsi que les trois valeurs du paramètre φ aux sommets de ce triangle. La valeur affectée est une somme de ces trois valeurs pondérées par les coordonnées barycentriques.

Cette façon de faire n'est pas celle qui produit la représentation de φ sur la grille la plus proche de celle de la surface. En effet, elle ne garantit pas qu'à chaque nœud est affectée la valeur de φ correspondant au point de la surface qui lui est le plus proche. Cependant, nous avons privilgié cette approche qui présente l'avantage de ne nécessiter qu'un surcoût en calcul minime par rapport au traitements déjà effectués au niveau *sélecteur*.

La définition de la fonction φ utilisée pour l'échantillonage est locale à chaque triangle car dépendante des valeurs de φ à ses trois sommets. Il est donc naturel de s'assurer de sa continuité de part et d'autre d'une arète partagée par deux triangles.

La démonstration de cette continuité ne sera pas développée ici car elle est tout à fait similaire à celle qui a été effectuée dans le cas de l'objet *tsolid*. Pour plus de précisions sur le sujet, on se reportera donc a la section correspondante dans le chapitre précédent.

3.6 Intersection d'un *voxet* avec un objet de type *gstack*

3.6.1 Caractéristiques de l'objet *gstack*

L'objet *gstack* a été abondament décrit dans le chapitre 2 de la page 17 de ce rapport qui lui est entièrement consacré. Il est constitué par une surface triangulée aux nœuds de laquelle sont stockés sous forme de tableaux les coordonnées des points composant une ligne polygonale ouverte.

Ceci permet de définir le *gstack* comme un empilement (stack) de surfaces triangulées partageant le même nombre de nœuds et les mêmes connections entre nœuds.

3.6.2 Algorithme de *sélection* associé

Dans le chapitre 2.4.2 de la page 25, nous avons étudié le problème du remplissage par des tétraèdres du volume délimité par les différents niveaux d'un *gstack* et ainsi établi la dimension volumique de cet objet. Rappelons que le problème avait alors consisté à assurer l'agencement correct de prismes à base triangulaires décomposés sous la frorme d'une triplet de tétraèdres par leurs faces latérales. La figure Figure 20 de la page 27 montre les six différentes manières de remplir ces prismes.

Nous allons donc partir du principe qu'à chaque triangle du *gstack* a été associé un indice compris entre 1 et 6 correspondant à une de ces six familles de tétraèdres présentées ci-dessus. Le problème de calculer l'intersection entre une grille régulière et un *gstack* revient donc à tester l'appartenance des nœuds de la grille à un volume décomposé en éléments tétraédriques.

Ce problème a déjà été résolu dans le chapitre consacré à l'objet *tsolid* et ne sera donc pas repris ici.

L'algorithme du calcul de l'intersection entre un objet *gstack* et une grille régulière pourra donc prendre la forme suivante:

Soit G un objet de type *gstack* et V un objet de type *voxet.*

Pour (chaque niveau n de G à l'exception du dernier) **faire**

Pour (chaque triangle T de la triangulation) **faire**

Récupérer les coordonnées des sommets de T_n (T au niveau i) et T_{n+1}.

Récupérer l'indice i associé à T.

Pour (chaque tétraèdre correspondant au type de décomposition i) **faire**

Calculer l'intersection entre ce tétraèdre et le voxet V.

fin Pour

fin Pour

fin Pour

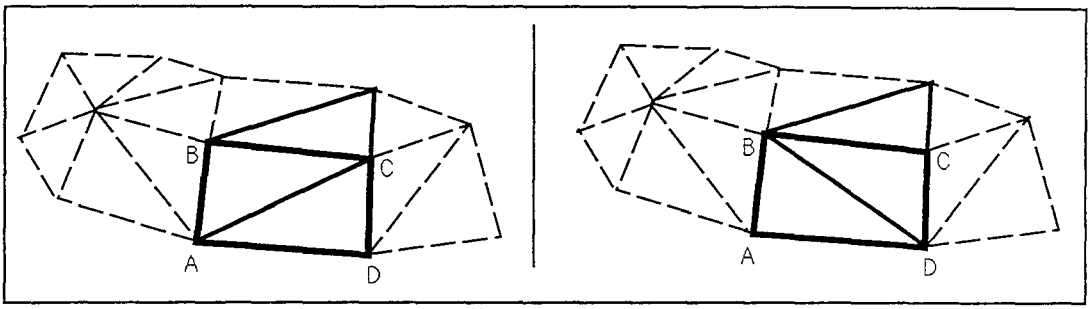
ALGORITHME 13. Calcul de l'intersection d'une grille régulière avec un objet *gstack*

Ceci amène un certain nombre de commentaires:

- Le fait que l'on ait pu remplir par des tétraèdres le volume interne du *gstack* en assurant la consistance de ce maillage au niveau des faces de chaque élément tétraédrique prend ici tout son intérêt. En effet, on a pu ici réutiliser les procédures de calcul d'intersection entre un tétraèdre et une grille régulière qui avaient été développées dans le cadre de l'objet *tsolid.* Cette routine s'était alors avérée très performante, tant au point de vue de sa simplicité de conception que de ses performances en calcul. On pourra donc attendre les mêmes qualités de l'outil qui vient d'être présenté.

- Comme nous l'avons vu dans le chapitre 2.4.3 de la page 30, l'affectation des types de décomposition en tétraèdres des triangles n'est pas unique. Ce problème peut par consé-

quence affecter le résultat du calcul d'intersection d'un *voxet* avec un *gstack* aux niveaux des bords de ce dernier.
Dans la figure qui suit, nous avons représenté un élément de volume situé au niveau du bord d'un GSTACK.



Considérons la face extérieure ABCD de cet élément de volume. Rien n'impose à priori que les quatre nœuds qui la composent soient coplanaires. Suivant que l'on considère l'un ou l'autre des types de dcomposition représentés, il est clair que le volume du prisme n'est pas le même, et que donc l'intersection d'une grille régulière avec cet objet sera elle même différente.
Cependant, l'approche tétraèdre, de par les avantages qu'elle présente notamment aux niveaux des performances en calcul, a été jugée acceptable malgré ce problème d'effet de bord. Une description analytique du volume de chaque prisme aurait été beaucoup trop complexe et pénalisante au niveau du temps de calcul. L'approche développée ici présente en outre l'avantage d'être en harmonie avec l'idée force du projet Gocad de représenter les objets naturels de manière discrète (lignes polygonales, surfaces triangulées, etc...).

3.6.3 Algorithmes d'*affectation* associés

Conversion de l'objet *gstack* sous forme de maillage régulier

La première application qui vient à l'esprit consiste à décrire le volume délimité par deux niveaux consécutifs d'un *gstack* sous la forme d'un ensemble de voxels. Le travail consiste ici simplement à marquer les cellules isolées par le *sélecteur*. L'*affecteur* en question aura donc la forme suivante:

Pour (chaque voxel V(uvw) selectionné) **faire**

Affecter une valeur constante C dans l'entrée d'indice uvw du tableau de la grille.

et/ou

Mettre le bit *painted* correpondant à V à la valeur 1.

et/ou

Mettre le bit *region* correpondant à V à la valeur 1.

fin Pour

ALGORITHME 14. Conversion d'un *gstack* sous la forme d'un ensemble de voxels.

Ceci permet de caractériser un sous-ensemble d'une grille régulière délimité par deux surfaces triangulées isomorphes de manière automatique.

Echantillonage de propriétés physiques connues aux nœuds d'un *gstack*

Nous supposerons ici que l'on dispose d'un objet *gstack* dont la géométrie a été construite en utilisant les outils classiques de modélisation des surfaces triangulées ou les routines d'interpolation de surfaces décrites dans le chapitre CHAP. La géométrie de l'objet est donc connue. Nous supposerons également que, pour chaque niveau du *gstack*, un paramètre φ a pu être estimé de telle manière que pour chaque nœud, on dispose d'une valeur numérique correspondant à ce paramètre.

Le problème est alors le suivant: on souhaite échantilloner ce paramètre φ aux nœuds d'une grille régulière c'est à dire calculer en chaque nœud une valeur de φ en fonction de celles connues aux nœuds du *gstack*.

Le *gstack* est un objet relativement «neutre» du point de vue informatique. Il a cependant été conçu dans un but bien précis: décrire des formes de sédimentation se caractérisant par l'agencement de couches géologiques isomorphes les unes par rapport aux autres (voir chapitre CHAP). Dans ce contexte, il est clair que la répartition du paramètre φ n'est pas quelconque et qu'elle est influencée par la disposition des couches géologiques. On parlera ici de «répartition surfacique de φ».

C'est cette répartition surfacique qu'il s'agit de préserver lors de l'échantillonage aux nœuds de la grille régulière et c'est en cela que l'*affecteur* de l'objet *gstack* diffère de celui de l'objet *tsolid* que nous avons vu précédemment. Nous procèderons à une analyse comparée des deux techniques par la suite.

Le premier problème qu'il convient de résoudre est le suivant: considérons un point N(x,y,z) contenu dans un prisme à base triangle de géométrie donnée. Comment peut on caractériser la position de ce point par rapport aux deux plans définis par les faces triangulaires inférieure et supérieure du prisme.

Formalisation du problème

Le prisme est constitué de six nœuds $(P_0,P_1,P_2,P'_0,P'_1,P'_2)$, (P_0,P_1,P_2) constitue sa face triangulaire inférieure, (P'_0,P'_1,P'_2) sa face supérieure.

Soit $\vec{N_0}$ le vecteur normal au plan $\Pi_0 = (P_0,P_1,P_2)$ et $\vec{N_1}$ le vecteur normal à la face

$\Pi_1 = (P'_0,P'_1,P'_2)$.

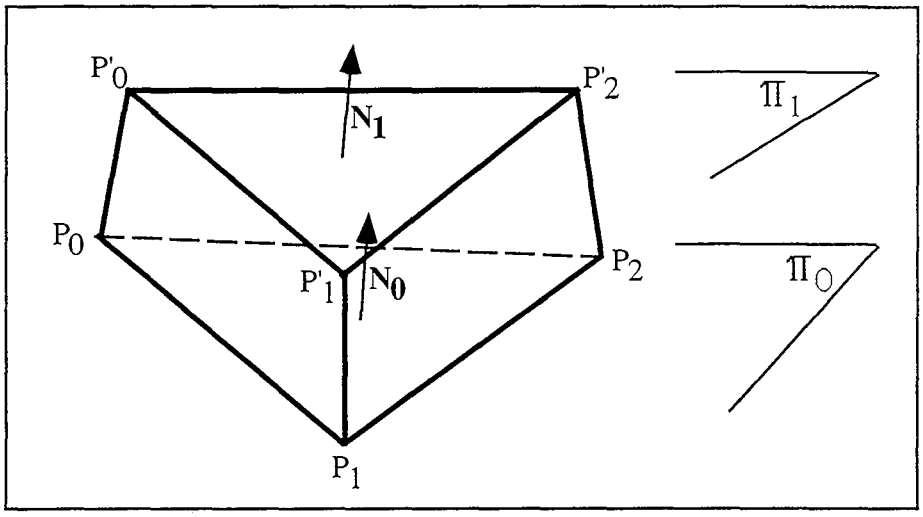


FIGURE 32. Formalisme du problème.

Solution du problème

Le problème consiste alors à positionner le nœud N(x,y,z) par rapport aux deux plans Π_0 et Π_1 . Pour cela, nous introduisons un paramètre α que nous pourrons appeler «coefficient d'éloignement de N par rapport au plan Π_0 ».

Par définition, le nœud N appartient au prisme $(P_0,P_1,P_2,P'_0,P'_1,P'_2)$, on peut donc écrire:

$$\exists\ \alpha \text{ tel que } N(x,y,z) \in (P''_0,P''_1,P''_2) \qquad \text{avec} \qquad \begin{cases} P''_0 = P_0 + \alpha \cdot (P'_0 - P_0) \\ P''_1 = P_1 + \alpha \cdot (P'_1 - P_1) \\ P''_2 = P_2 + \alpha \cdot (P'_2 - P_2) \end{cases}$$

Les six points $(P_0,P_1,P_2,P_0',P_1',P_2')$ étant donnés, α caractérise le plan $\Pi_\alpha = (P''_0,P''_1,P''_2)$, les trois points P''_0,P''_1 et P''_2 appartenant respectivement aux arêtes (P_0,P'_0) , (P_1,P'_1) et (P_2,P'_2) .

Positionner le nœud N par rapport à Π_0 et Π_1 revient donc à déterminer la valeur de α comprise entre 0 et 1 qui caractérise Π_α. La définition de α permet d'écrire:

$$\overrightarrow{N_\alpha} = \overrightarrow{N_0} + \alpha \cdot (\overrightarrow{N_1} - \overrightarrow{N_0})$$

Le plan Π_α qui contient N(x,y,z) est défini par la relation:

$$\overrightarrow{P''_0P_0} \cdot \overrightarrow{N_\alpha} = 0 \qquad \text{avec} \qquad \begin{cases} \overrightarrow{P''_0P_0} & \begin{cases} [x - P_{ox}] - \alpha \cdot (P'_{ox} - P_{ox}) \\ (y - P_{oy}) - \alpha \cdot (P'_{oy} - P_{oy}) \\ (z - P_{oz}) - \alpha \cdot (P'_{oz} - P_{oz}) \end{cases} \\ \overrightarrow{N_\alpha} & \begin{cases} N_{0x} + \alpha \cdot (N_{1x} - N_{0x}) \\ N_{oy} + \alpha \cdot (N_{1y} - N_{0y}) \\ N_{oz} + \alpha \cdot (N_{1z} - N_{0z}) \end{cases} \end{cases}$$

Ceci s'exprime sous la forme d'une équation du second degré en α, dont la solution comprise entre 0 et 1 caractérise Π_α.

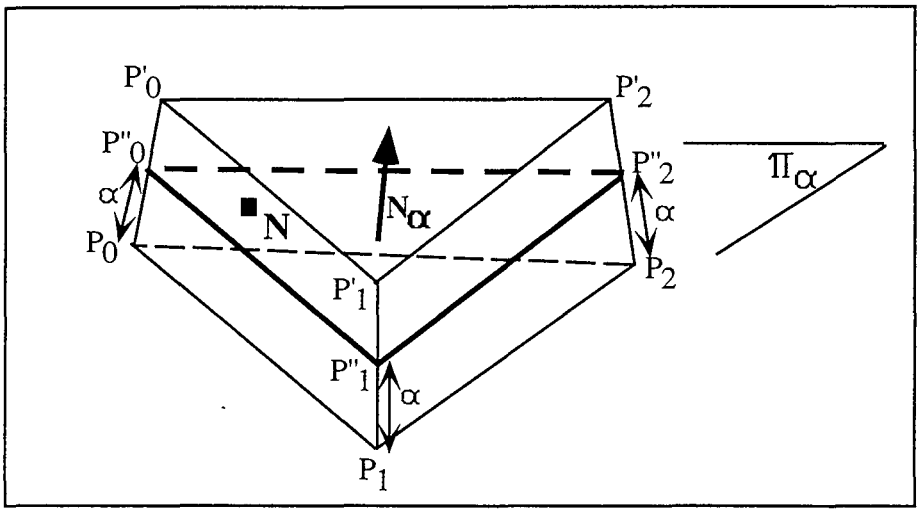


FIGURE 33. Positionnement du nœud N dans un prisme.

Assignation d'une valeur de φ au nœud N

La position du nœud N a été caractérisée de manière relative aux plans Π_0 et Π_1. Il convient maintenant de lui affecter une valeur numérique correspondant à la variable φ. Or, comme cela a été dit dans l'introduction, on souhaite conserver à φ une répartition surfacique lors de

l'échantillonage. Pour cela, je propose d'opérer de la manière suivante:

Pour (chaque nœud N sélectionné par le *sélecteur*) **faire**

α étant déterminé, on calcule les coordonnées barycentriques (u,v) de N dans le triangle $\Pi_\alpha = (P''_0, P''_1, P''_2)$.

On calcule alors les valeurs φ''_0, φ''_1 et φ''_2 de φ aux points P''_0, P''_1 et P''_2 avec:

$$\begin{cases} \varphi''_0 = \varphi_0 + \alpha \cdot (\varphi'_0 - \varphi_0) \\ \varphi''_1 = \varphi_1 + \alpha \cdot (\varphi'_1 - \varphi_1) \\ \varphi''_2 = \varphi_2 + \alpha \cdot (\varphi'_2 - \varphi_2) \end{cases}$$

Affecter au nœud N la quantité:

$$\varphi_N = u \cdot \varphi''_1 + v \cdot \varphi''_2 + (1 - u - v) \cdot \varphi''_0$$

fin Pour

ALGORITHME 15. Echantillonage d'un paramètre physique connu aux nœuds d'un objet de type *gstack* sur une grille régulière

Etude de la continuité de la fonction φ ainsi définie

La définition de φ que nous venons de fournir est une définition locale à chaque prisme. Par ailleurs, le positionnement d'un nœud N par rapport aux faces Π_0 et Π_1 a permis de définir un nouveau système de coordonnées. Ainsi la position de N a pu être caractérisée par un triplet de valeurs (α, u, v), α caractérisant le plan Π_α compris entre Π_0 et Π_1 contenant N, et (u, v) décrivant la position de N dans Π_α.

Ce nouveau système de coordonnées permet de définir un nouvel élément volumique: le prisme à faces latérales curvilinéaires $P_{CL}(P_0, P_1, P_2, P'_0, P'_1, P'_2)$. Ce volume est défini par ses deux faces (P_0, P_1, P_2) et (P'_0, P'_1, P'_2) triangulaires et par ses trois faces latérales curvilinéaires à quatre cotés obtenues par interpolation linéaires des arètes latérales du prisme. Dans le nouveau système de coordonnées mis en place, P_{CL} se définit comme l'ensemble des nœuds N(a, u, v) tels que:

$$\alpha \in [0,1] \qquad u \in [0,1] \qquad v \in [0,1] \qquad (1 - u - v) \in [0,1]$$

Considérons maintenant une face latérale ABCD commune à deux prismes de ce type notés P_{1CL} et P_{2CL}. Si l'on reprend le même type de dmonstration de la continuité de φ de part et d'autre de ABCD que celle qui a été effectuée dans le cas de l'objet *tsolid*, on arrive aisément à démonter que, au voisinage de la face ABCD, $\varphi(P_{1CL})$ et $\varphi(P_{1CL})$ tendent vers une expression commune qui dépend uniquement des quatres valeurs φ_a, φ_b, φ_c et φ_d de φ aux nœuds A, B, C et D. La continuité de φ de part et d'autre d'une face latérale commune à deux prismes à faces curvilinéaires est donc assurée.

Cependant, il faut bien remarquer que les éléments de volume qui ont été considérés au niveau *sélecteur* ne sont pas du même type que ceux que nous venons de décrire. En effet, il s'agit de prismes à faces latérales triangulées issus de la décomposition en tétraèdres. En conséquence, il apparait que la continuité de φ, de part et d'autre d'une face latérale triangulée ABCD commune à deux prismes, n'est pas établie à l'exception du cas où les quatre sommets ABCD sont coplansires (dans ce cas, face curvilinéaire et face coplanaire sont confondues).

Pour s'affranchir de ce problème, il convient donc, au niveau *affecteur*, de tester si la conditions d'appartenance $(\alpha \in [0,1] \wedge u \in [0,1] \wedge v \in [0,1] \wedge (1-u-v) \in [0,1])$ est vérifiée pour le nœud courant $N(\alpha, u, v)$. Dans le cas où celle ci ne le serait pas, on effectuera ce test d'appartenance pour les prismes voisins du prisme courant. Une fois le volume V aux faces curvilinéaires contenant N déterminé, la valeur de φ assignée à N sera calculée à partir de l'expression de φ locale à V. La continuité de φ sera alors assurée.

Analyse comparée des méthodes d'echantillonage de l'objet *tsolid* et de l'objet *gstack*

Nous avons vu que la méthode d'échantillonage de l'objet *gstack* mettait en œuvre une décomposition de son volume en éléments tétraédriques. La raison en est que ceci permet d'isoler de manière très efficace l'ensemble des nœuds concernés par l'échantillonage.

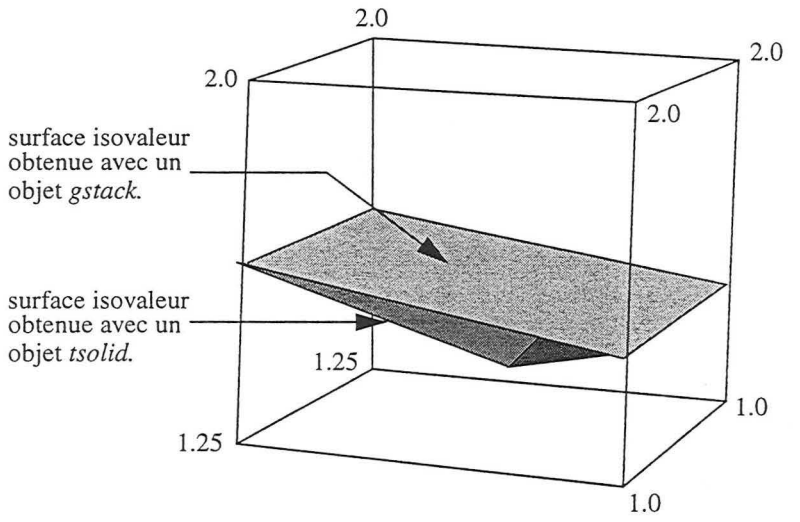
Cependant, l'expression de φ qui a été employée ici est différente de celle utilisée pour l'objet *tsolid*. Elle dépend de la position et des valeurs affectées à six sommets qui forment un prisme, contrairement à celle utilisée pour l'objet tsolid où la valeur de chaque nœud ne dépend que de la position et des valeurs de φ des quatre sommets du tétraèdre qui le contient.

Il aurait été possible de reprendre cette dernière expression pour le compte de l'objet *gstack*. Nous avons cependant privilégié une expression plus complexe et plus coûteuse en terme d'opérations de calcul, et ce pour que l'échantillonage préserve la «répartition surfacique de φ» liée à la nature même de l'objet *gstack*.

Mieux qu'un long discours, l'exemple présenté dans la Figure 34 illustre cette idée. Nous disposons ici d'un objet de forme cubique dont les huit sommets sont affectés d'une valeur numérique donnée pour φ. Cet objet est décrit de deux manières différentes:

- sous la forme d'un objet de type *tsolid* T composé de six tétraèdres
- sous la forme d'un objet *gstack* G à deux triangles et deux niveaux parallèles.

Nous avons échantilloné le paramètre φ sur une grille régulière à partir de ces deux objets. L'echantillonage façon *gstack* passe par la décomposition du volume de G sous la forme de l'ensemble des tétraèdres de T. Seules les expressions de φ diffèrent. Sur les deux grilles obtenues, nous avons construit une surface d'isovaleur pour mettre en évidence les différences entre les deux méthodes. La propriété représentée sur le plan du fond de la grille est le résultat de l'échantillonnage par l'objet *tsolid*.



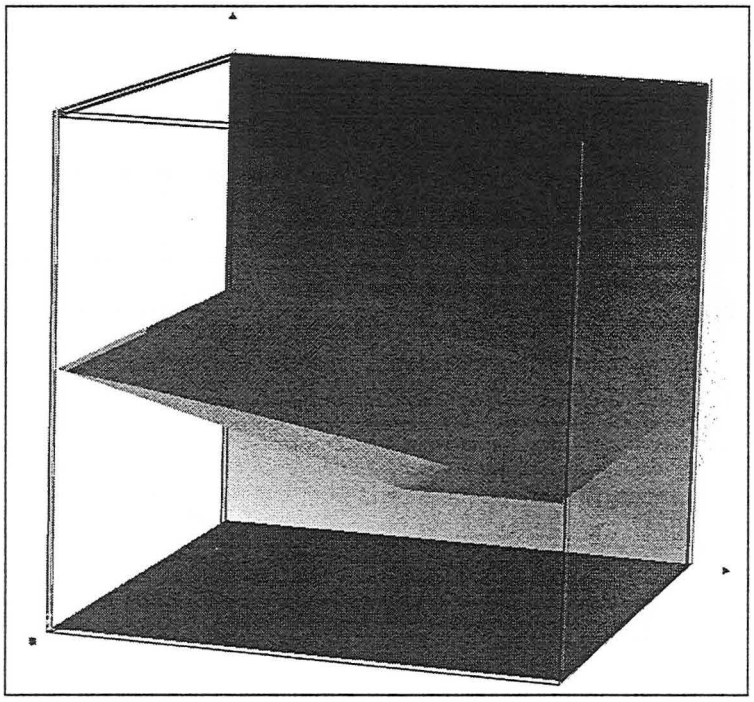


FIGURE 34. Différences entre l'échantillonage à partir d'un objet *tsolid* et à partir d'un objet *gstack*.

Ceci montre clairement que l'échantillonage façon *tsolid* introduit une rupture de pente dans la répartition de φ, que l'on ne retrouve pas avec l'objet *gstack*. Cette rupture de pente est due au caractère discontinu de φ' d'un tétraèdre à l'autre de l'objet *tsolid* (la démonstration est établie en page 62).

Dans le cas de l'objet *gstack*, la surface d'isovaleur est un plan, ce qui correspond au résultat que nous espérions compte tenu de la géométrie et des valeurs imposées aux huit sommets du cube de départ.

Evaluation des performances

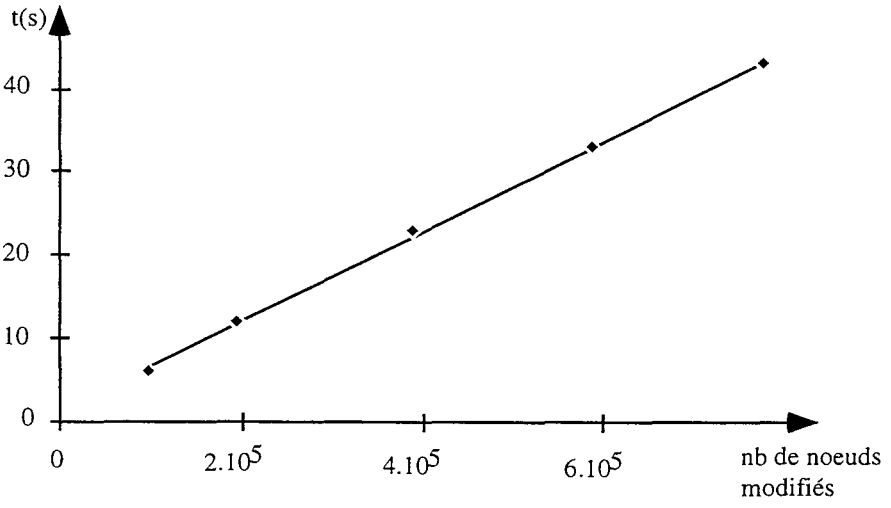
Des tests de performances ont été effectués sur un exemple synthétique. Nous disposons ici d'un modèle surfacique composé de deux failles et d'une discordance, ces objets délimitant une sous-région de l'espace 3D (voir la planche 6 de la page 81). Un objet *gstack,* composé de 64 triangles sur 8 niveaux, a été construit pour modéliser l'agencement des couches géologiques dans cette sous-région. En chacun des sommets du *gstack* est connue une valeur numérique que l'on souhaite échantillonner sur une grille régulière (voir la planche 7 de la page 82). La taille de la grille est variable et on veut étudier le temps que prend cette opération quand on augmente le nombre de nœuds de la grille.

Les résultats obtenus sont les suivants:

Tableau .4 : Tests d'échantillonage à partir de l'objet *gstack*.

test n°	1	2	3	4	5
taille de la grille	50*50*50	100*50*50	100*100*50	91*91*91	100*100*100
nombre de nœuds	125000	250000	500000	753571	1000000
temps de calcul (s)	6	12	23	33	43
Nombre de cellule modifiées	97380	194503	389990	586717	776131
% de cellules modifiées	77.9	77.8	78	77.86	77.6

L'évolution du temps de calcul en fonction de la taille de la grille est la suivante:



Commentaires

Le graphe de performances montre que le temps de calcul évolue linéairement en fonction du nombre de cellules modifiées. La pente de la droite de regression calculée est de 54.1 secondes par million de cellules et permet d'envisager avec des machines actuelles une utilisation de

cet algorithme sur des grilles de taille beaucoup plus importante que celles utilisées dans la série de tests.

La méthode d'échantillonage qui vient d'être présentée ici est une méthode de calcul qui permet d'effectuer les interpolations linéaires entre niveaux successifs d'un objet *gstack*. Il va de soi que ce n'est pas la seule méthode possible. Il est possible d'effectuer n'importe quel autre type d'échantillonage. Par exemple:

- on pourra également échantilloner un paramètre connu aux nœuds d'une surface sur un volume d'épaisseur donnée autour de cette surface.
- on pourra affecter aux nœuds de la grille une valeur calculée à partir d'une loi donnée. Cette loi pourra par exemple tenir compte de la distance d'éloignement des nœuds par rapport à un niveau du *gstack*, par exemple.

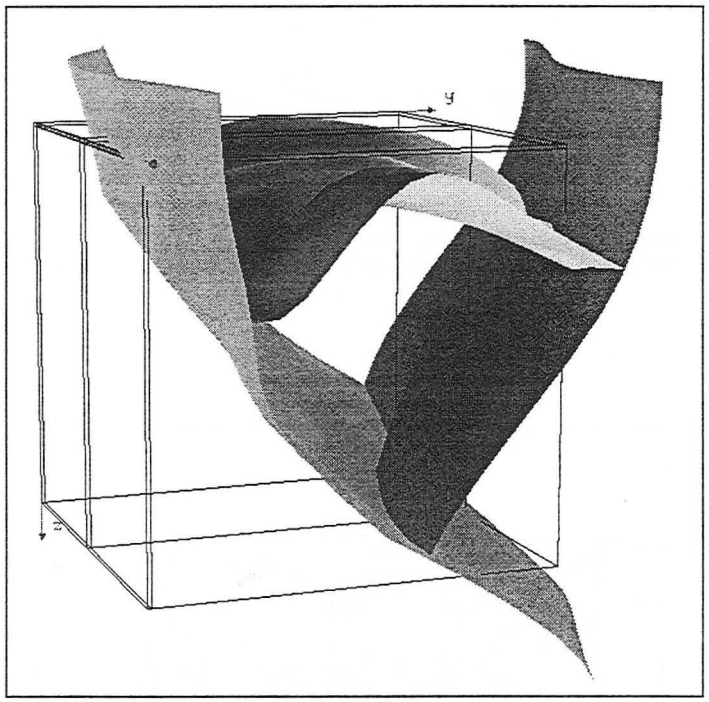


PLANCHE 6. Modèle surfacique délimitant une sous région de l'espace.

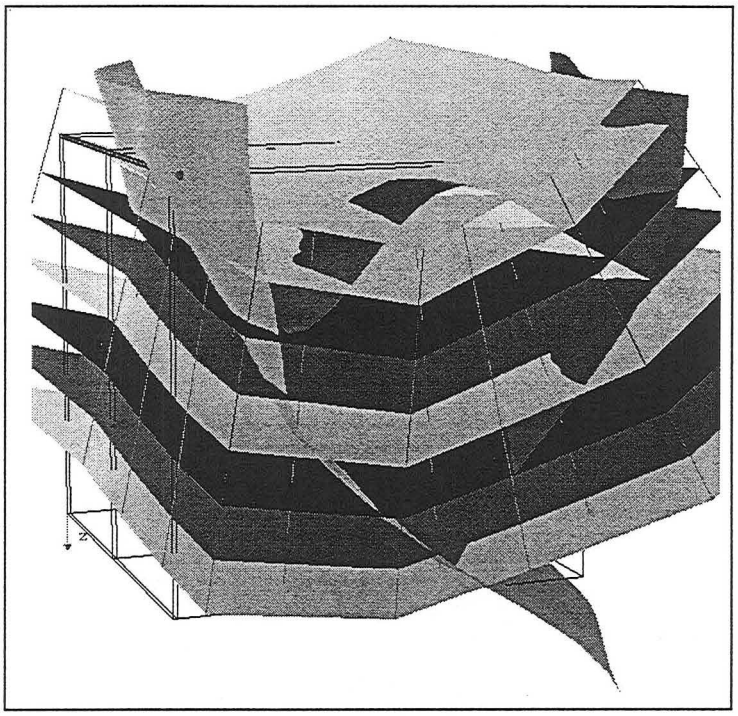


PLANCHE 7. Représentation de l'objet gstack décrivant la stratigraphie.

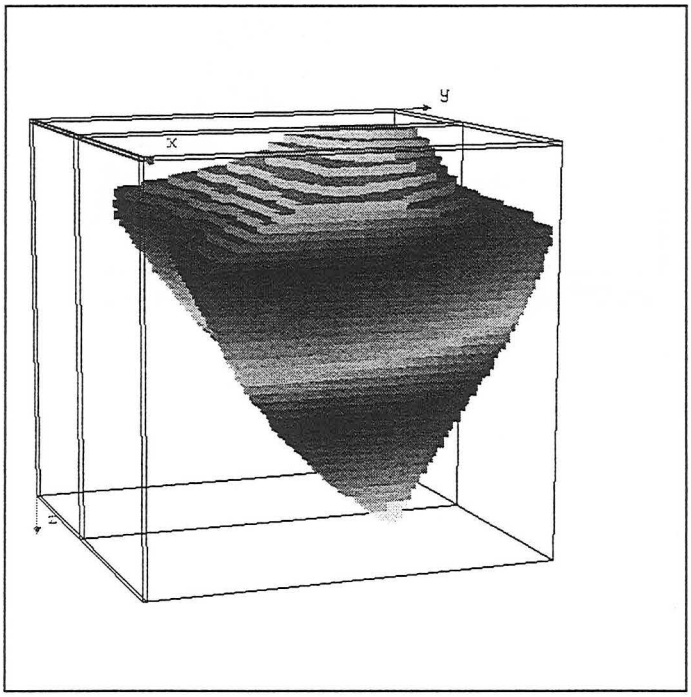


PLANCHE 8. Résultat de l'échantillonage d'une propriété connue aux nœuds de l'objet gstack sur un sous-domaine d'une grille régulière.

3.7 Intersection d'un *voxet* avec un objet de type *gshape*

3.7.1 Caractéristiques de l'objet *gshape*

Le *gshape* (ou geological shape) ([48],[49],[50],[51]) est un objet volumique composé:

- d'un axe polygonal, aussi appelé «backbone», qui décrit l'allure générale de l'objet.

- d'un ensemble de «sections» planaires associées à chacun des sommets du «backbone». Chacune de ces sections est décrite par un ensemble de vecteurs, dont le nombre est commun à tous les nœuds du «backbone».

- d'une enveloppe, surface qui lie les sections entre elles et délimite ainsi un volume fermé caractéristique de l'objet en question.

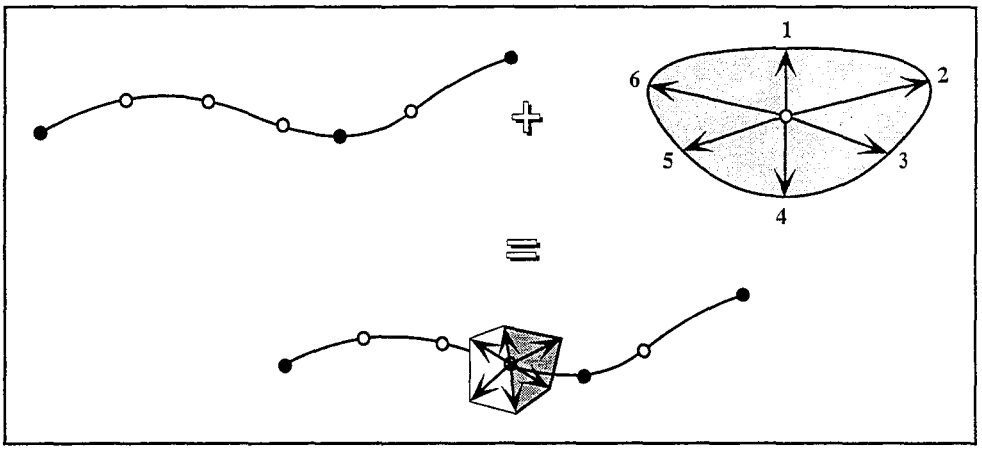


FIGURE 35. Notion de *gshape*.

L'objet *gshape* est le plus souvent utilisé comme un outil de modélisation d'objets naturels qui, par nature, peuvent être vus comme le résultat de l'interpolation de sections planaires le long d'un axe polygonal. C'est le cas par exemple de formes de sédimentation particulières, telles que les chenaux, les lentilles, etc...

L'objet *gshape* a ainsi été utilisé dans le cadre de la simulation stochastique de la distribution d'hétérogénéités au sein des réservoirs pétroliers.

Dans ce contexte, il nous est apparu essentiel de pouvoir convertir les modèles construits à base d'objets de type *gshape*, sous forme de maillages réguliers, pour pouvoir alimenter avec ces maillages les logiciels de simulation d'écoulements.

3.7.2 Routine de *sélection* associée à l'objet *gshape*

Le respect de l'ordre de numérotation des vecteurs de chaque section du *gshape* permet de définir la notion d'enveloppe. Comme le montre la Figure 36, cette enveloppe peut elle même

être convertie sous la forme d'une surface triangulée fermée délimitant le volume de l'objet.

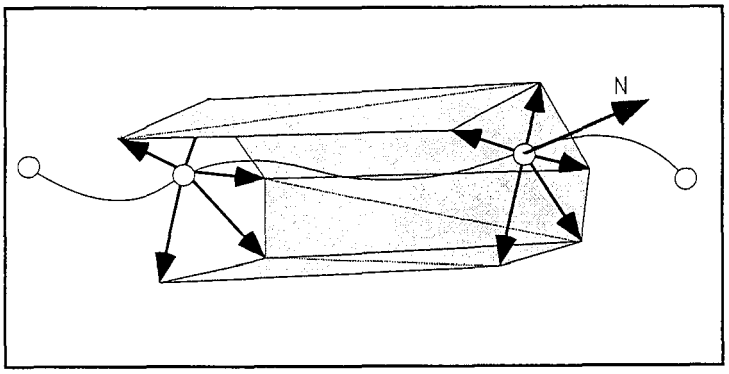


FIGURE 36. Définition de l'enveloppe d'un *gshape*.

Etant donnés une grille régulière et un objet *gshape*, le problème consiste à sélectionner les nœuds de la grille qui appartiennent au volume délimité par cette enveloppe.

L'idée consiste à convertir l'enveloppe du *gshape*, qui est un objet surfacique, en un objet volumique. Pour une surface triangulée, la façon de faire la plus naturelle est de remplir le volume qu'elle délimite par des tétraèdres. La littérature fournir des solutions à ce problème, essentiellement basées sur les travaux des mathématiciens Delaunay et Voronoï ([13]). Une stratégie possible pourrait donc être la suivante:

Soient G un objet de type *gshape* et V une grille régulière.

Extraire la surface Su construite à partir de l'enveloppe de G.

Construire le solide So en remplissant avec des tétraèdres le volume délimité par Su.

Calculer l'intersection de So avec V.

ALGORITHME 16. Calcul de l'intersection d'une grille régulière avec un objet *gshape*(cas général)

Cette méthode est tout à fait valable et fournit des résultats de bonne qualité quelle que soit la géométrie de l'objet *gshape* de départ.

Cependant, comme cela a été dit précédemment, l'objet *gshape* est utilisé pour simuler des distributions d'hétérogénéités de réservoir. Ces simulations peuvent mettre en jeu de nombreux objets et, du fait même que ce sont des simulations, elles sont également déstinées à être produites en de nombreux exemplaires équiprobables.

Dans ce contexte, la vitesse d'exécution de l'algorithme présenté ci-dessus représente donc un critère de choix essentiel. Or il s'est avéré que cette méthode n'était pas suffisament performante de ce point de vue, du fait notammnent qu'elle induit la création de deux objets temporaires (une surface triangulée et un solide à base de tétraèdres) dont la création, le stockage et la destruction sont coûteuses, aussi bien en terme de mémoire que de temps de calcul.

Il convient donc de proposer une alternative à cette approche. Celle que nous allons présenter ici conserve la même idée de remplir le volume du *gshape* en tétraèdres mais, pour que ceci soit moins coûteux, le remplissage tient compte de la nature de l'objet en question. Nous verrons également que cette routine ne fonctionne que pour certaines géométries, qui seront défi-

nies par la suite.

Comme l'illustre la Figure 36 de la page 84, le volume du *gshape* est la réunion des volume délimités par deux sections adjacentes. Compte tenu des conventions de numérotations des vecteurs de chaque section, chacun de ces volumes peut lui-même être décomposé en prismes à base triangulaire. Le problème du remplissage de ces volumes par des tétraèdres et de leur agencement les uns aux autres a été longuement décrit dans le chapitre 2.4 de la page 23 consacré à l'objet *gstack.*

Le formalisme de ce problème a lui aussi été décrit en détail dans ce chapitre. Son énoncé nous avait amené à la conclusion que, compte tenu d'une triangulation donnée, celui-ci pouvait être résolu si l'on pouvait affecter à chaque arête de chaque triangle un type 0 ou1 sous les conditions suivantes:

- une arête commune à deux triangles ne peut pas être du même type dans ces deux triangles.
- une triangle ne peut pas avoir trois arêtes du même type.

Le problème d'affecter un type à chaque arête est beaucoup plus simple que pour l'objet *gstack.* En effet, les triangulations des sections du *gshape* sont très simples (voir Figure 37). En outre, chacun des triangles présente la particularité de posséder une arête libre et respectivement chaque prisme possède une face latérale libre. L'affectation à chaque arête de chaque triangle d'un type 0 ou 1 est donc toujours possible et consécutivement le remplissage cohérent par des tétraèdres du *gshape* est lui aussi toujours possible.

En reprenant le formalisme utilisé dans le chapitre 2.4.2 de la page 25, l'affectation des types 0 ou 1 aux arêtes des triangles peut se faire de la manière suivante:

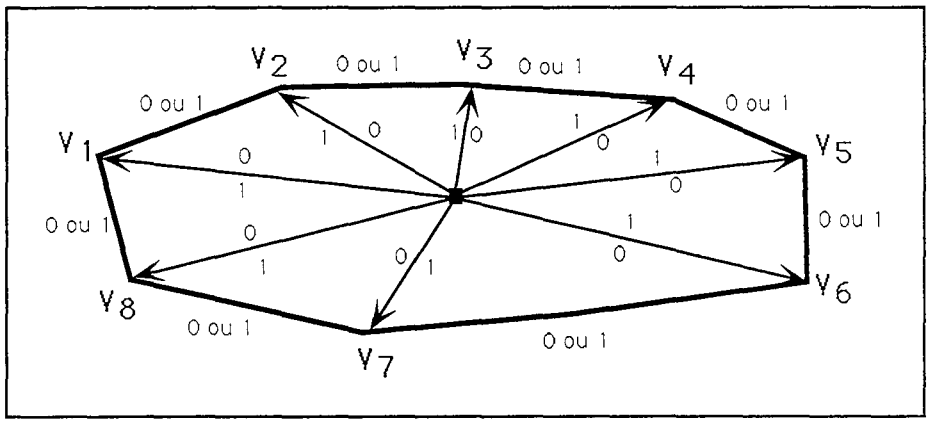


FIGURE 37. Décomposition de l'enveloppe d'un gshape en tétraèdres.

Nous supposerons donc qu'à chaque triangle a été affecté un type compris entre 1 et 6 décrivant la façon de décomposer en tétraèdres le prisme qui lui correspond . L'algorithme d'inter-

section entre un *gshape* et une grille régulière devient alors:

Soit G un objet de type *gshape* et V un objet de type *voxet.*

Pour (chaque segment S du backbone de G) **faire**

Pour (chaque arête A_n de la section de G) **faire**

Récupérer les coordonnées des 6 sommets du prisme correspondant à S et A_n.

Récupérer l'indice i associé à T.

Pour (chaque tétraèdre correspondant au type de décomposition i) **faire**

Calculer l'intersection entre ce tétraèdre et le *voxet* V.

fin Pour

fin Pour

fin Pour

ALGORITHME 17. Calcul de l'intersection d'une grille régulière avec un objet *gshape* (cas simplifié)

Cet algorithme n'est pas valable quelle que soit la géométrie de l'objet *gshape*. En effet, comme le montre la Figure 38, la position du nœud du backbone par rapport à sa section et la forme de cette section peuvent être incompatibles avec une telle décomposition systématique du volume du *gshape*. On voit en effet ici que le volume balayé par l'ensemble des tétraèdres déborde de l'enveloppe du gshape au niveau de la concavité de la section.

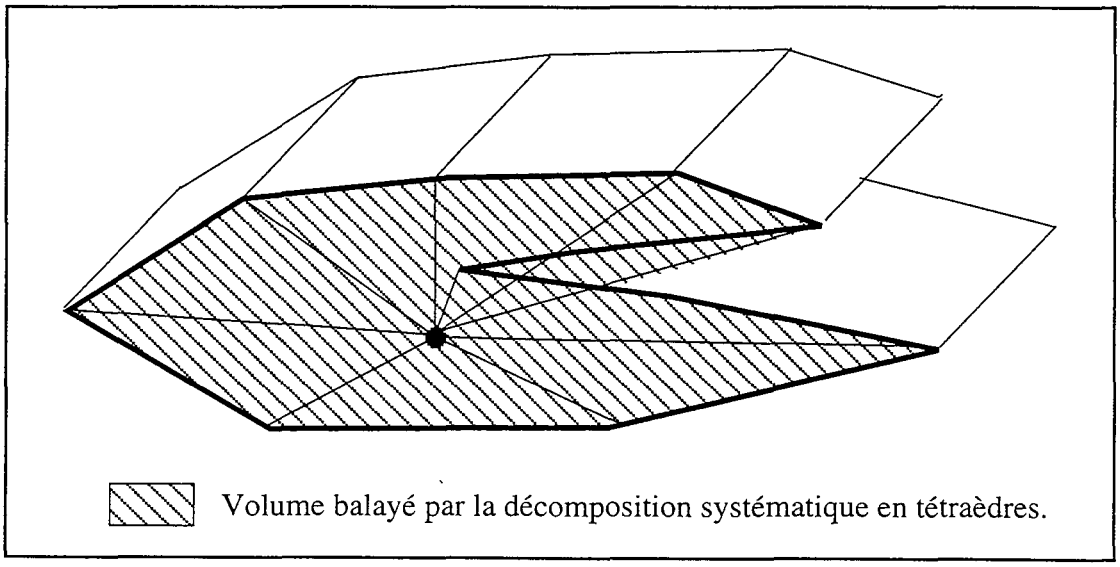


FIGURE 38. Problème lié à une décomposition automatique de l'enveloppe d'un *gshape* en tétraèdres.

Considérons un *gshape* G orienté de manière consistante par ses tangentes aux nœuds du backbone et ses normales aux sections (voir chapitre 36 de la page 84). La décomposition auto-

matique en tétraèdres est possible, si le parcours de chaque section le long de son périmètre se fait, d'un point de vue angulaire, dans un sens de rotation constant et commun à toutes les sections.

D'un point de vue mathématique, le problème se formule de manière plus simple. Considérons une section S de G orientée par un vecteur normal $\vec{N}$ et deux vecteurs $\vec{V_i}$ et $\overrightarrow{V_{i+1}}$ consécutifs le long de cette section. G est décomposable en tétraèdres si quels que soient S, $\vec{N}$ et $(\vec{V_i}, \overrightarrow{V_{i+1}})$) le produit mixte $(\vec{V_i} \wedge \overrightarrow{V_{i+1}}) \cdot \vec{N}$ est de signe constant.

Ce test sera effectué sur chaque objet gshape avant toute opération sur une grille. Si celui-ci est positif, la décomposition automatique en tétraèdres sera utilisée. Si cela n'est pas le cas, il faudra passer par une surface triangulée correspondant à l'enveloppe, puis par un objet tsolid, comme décrit dans l'algorithme 16 de la page 84.

3.7.3 Algorithmes d'*affectation* associés

Génération de modèles de faciès

La modification qui sera la plus souvent effectuée sur les cellules isolées par le *sélecteur* va consister à affecter une valeur d'indice aux cellules contenues dans un même *gshape*.

Dans la plupart des cas, les simulations à base d'objets *gshape* sont constituées de plusieurs familles de ces mêmes objets. A chacune de ces familles sera associée une valeur d'indice spécifique.

L'utilisateur aura également la possibilité d'intervenir sur l'ordre dans lequel il «dessinera» ses objets dans la grille régulière, et ainsi il pourra définir des relations d'antériorité ou de postériorité de dépôt entre objets ou familles d'objets.

Evaluation des performances

Des tests de performances ont été effectués sur un modèle de réservoir réaliste en terme de nombre et de tailles d'objets (voir la planche 9 de la page 89). Nous disposons ici d'un ensemble de 120 objets *gshape* de deux types:

- 114 objets de type chenal: le backbone est constitué de 29 nœuds, chaque section comporte 8 sommets.
- 6 objets de type lobe: le backbone est fait de 17 nœuds, chaque section comporte 8 sommets.

Pour chacun de ces types, on affecte une valeur d'indice spécifique aux nœuds de grilles contenus dans chaque objet (voir la planche 10 de la page 89). Tous les objets sont décomposables automatiquement en tétraèdres. La taille de la grille est variable et on veut étudier le temps que prend cette opération quand on augmente le nombre de nœuds de la grille.

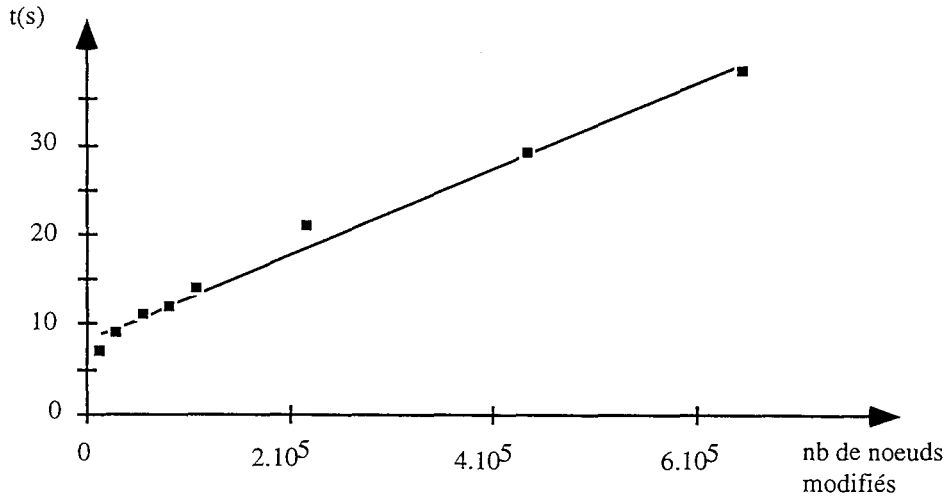
Les résultats obtenus sont les suivants:

Tableau .5 : Tests de conversion de modèles de réservoir sous forme de grille régulière

test n°	1	2	3	4
taille de la grille	50*50*50	100*50*50	100*100*50	91*91*91
nombre de nœuds	125000	250000	500000	753571
temps de calcul (s)	7	9	11	12
Nombre de nœuds modifiés	13756	27540	54946	81217

test n°	5	6	7	8
taille de la grille	100*100*100	126*126*126	159*159*159	182*182*182
nombre de nœuds	1000000	2000376	4019679	6028568
temps de calcul (s)	14	21	29	38
Nombre de nœuds modifiés	108156	215212	435347	644810

L'évolution du temps de calcul en fonction de la taille de la grille est la suivante:



Cette figure montre clairement que le temps de calcul évolue linéairement en fonction du nombre de nœuds modifiés. Par ailleurs, les temps absolus obtenus sur la machine «extreme» (voir page 149) permettent d'affirmer que cet algorithme ne pose pas de problèmes de performances, compte tenu du fait que le modèle utilisé est un modèle de taille conséquente et que ce type d'algorithme n'aura pas à être utilisé sur des grilles de taille supérieure à celles utilisées dans cette série de tests. En effet, les modèles de réservoirs sont souvent construits à partir de jeux de données peu denses et il n'est pas nécessaire de produire des modèles de grande définition.

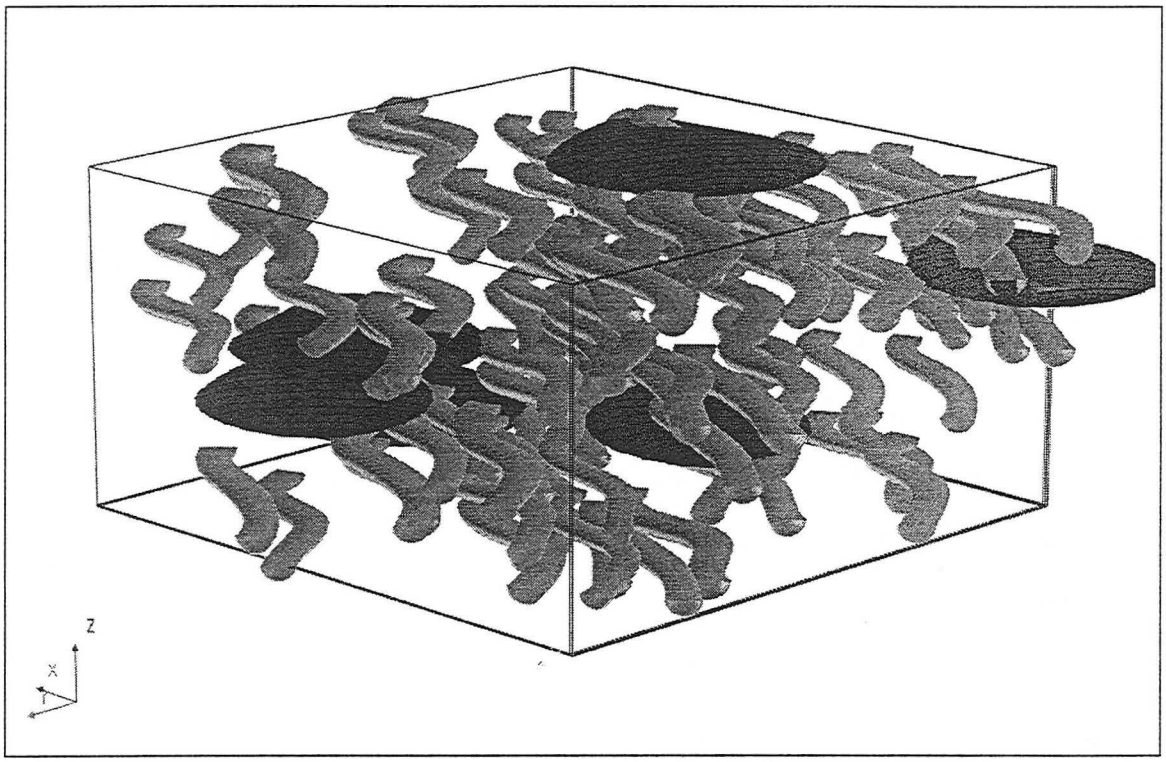


PLANCHE 9. Modèle de réservoir à base d'objets *gshape*.

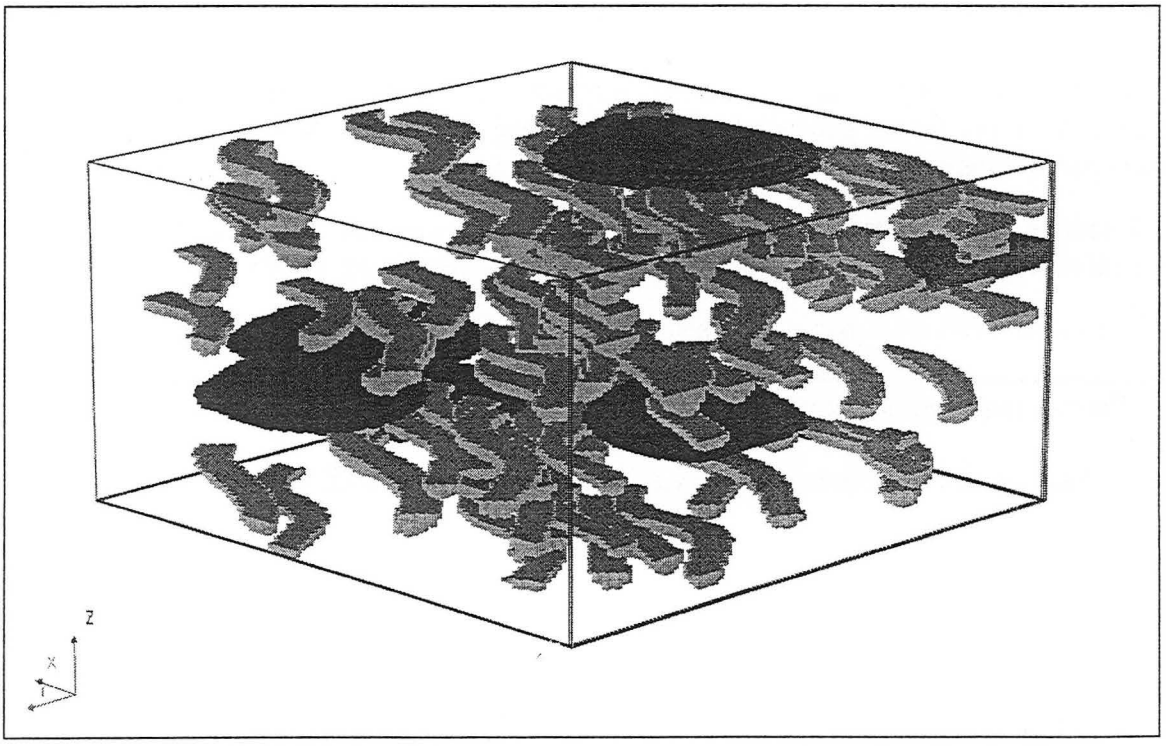


PLANCHE 10. Résultat de la conversion de ce modèle sous forme de grille régulière.

Génération de champs de direction

Cette application est liée au nouvel algorithme d'interpolation contrainte par un champ de directions non uniforme qui sera présenté dans le chapitre suivant. Dans ce contexte, on a besoin de connaître en chaque nœud trois valeurs correspondant à une direction donnée.

Il apparaît intéressant d'utiliser l'objet *gshape* pour initialiser de tels «champs de direction». A chaque cellule isolée par le *sélecteur* pourront être affectées les trois valeurs correspondant à la direction du segment courant du backbone.

Pour (chaque voxel V isolé par le *sélecteur*) **faire**

Récupérer les coordonnées des sommets du segment courant du gshape.

Affecter au voxel V le tableau des trois coordonnées de ce segment.

Mettre le bit *painted* du voxel Và 1.

fin Pour

ALGORITHME 18. Génération de champs de direction à partir de l'objet *gshape*.

3.8 Intersection avec d'autres objets

3.8.1 Intersection avec un nuage de points

De très nombreux jeux de données se présentent sous la forme de nuages de points P(x,y,z). A chacun de ces points peut également être attaché un tableau de valeurs correspondant à divers paramètres physiques.

Il semble donc intéressant, pour utiliser ces données sur une grille régulière, de pouvoir déterminer à quelle cellule doivent être affectées les valeurs de ces points.

La routine d'échantillonage est très simple:

Pour (chaque point P(x,y,z)) **faire**

Passage des coordonnées de (x,y,z) de P dans le repère $(O',\vec{U},\vec{V},\vec{W})$

-> P(uréel, vréel, wréel)

Arrondir uréel ,vréel ,wréel à leur valeur entière la plus proche

-> P(u, v, w)

Affecter le tableau des valeurs de P au voxel V(u, v, w).

fin Pour

ALGORITHME 19. Echantillonage d'un ensemble de points sur une grille régulière.

Ceci revient, pour chaque point P, à déterminer quelle cellule le contient et à affecter à cette

cellule l'ensemble des valeurs qui lui sont rattachées.

Cette routine ne nécessite qu'un très petit nombre d'opérations mathématiques et ne pose par conséquent aucun problème de performances.

3.8.2 Intersection avec une ligne polygonale

Pour (chaque segment $[P_n,P_{n+1}]$ de la ligne polygonale) **faire**

Passer les coordonnées de P_n et P_{n+1} dans le repère $(O',\vec{U},\vec{V},\vec{W})$.

-> $P_n(u_n, v_n, w_n)$, $P_{n+1}(u_{n+1}, v_{n+1}, w_{n+1})$

Calculer la sous-grille (u_{min}, u_{max}, v_{min}, v_{max}, w_{min}, w_{max}) contenant $[P_n,P_{n+1}]$.

Pour (u entier compris entre u_{min} et u_{max}) **faire**

Calculer l'abscisse curviligne t du point de coordonnée u dans $[P_n,P_{n+1}]_{(O',\vec{U},\vec{V},\vec{W})}$

Déduire de t les coordonnées v et w de la cellule la plus proche de ce point.

Selectionner le voxel V(u,v,w).

fin Pour

Pour (v entier compris entre v_{min} et v_{max}) **faire**

Calculer l'abscisse curviligne t du point de coordonnée v dans $[P_n,P_{n+1}]_{(O',\vec{U},\vec{V},\vec{W})}$

Déduire de t les coordonnées u et w de la cellule la plus proche de ce point.

Selectionner le voxel V(u,v,w).

fin Pour

Pour (w entier compris entre w_{min} et w_{max}) **faire**

Calculer l'abscisse curviligne t du point de coordonnée w dans $[P_n,P_{n+1}]_{(O',\vec{U},\vec{V},\vec{W})}$

Déduire de t les coordonnées u et v de la cellule la plus proche de ce point.

Selectionner le voxel V(u,v,w).

fin Pour

fin Pour

ALGORITHME 20. Intersection entre une ligne polygonale et une grille régulière.

Il peut arriver, dans certains cas, que des données se présentent sous la forme de lignes polygonales. C'est le cas par exemple des trajets de puits, dont rien ne laisse supposer qu'íls soient verticaux ou rectilignes. Echantilloner les données de ces puits sur une grille régulière suppose donc de poser le problème de représenter une ligne polygonale dans une grille régulière.

La notion d'intersection entre une ligne polygonale et une grille régulière est à rapprocher de celle que nous avions définie dans le cas de la surface triangulée, où il s'agissait de sélectionner un sous-ensemble de voxels de la grille dont la réunion produira une image de l'objet en question dans la grille (voir page 63 pour de plus amples explications)

Une fois les cellules selectionnées, il est possible de leur affecter un indice entier ou une valeur numérique correspondant à une donnée de puits mesurée le long de l'axe polygonal du trajet de ce puits.

Cette routine ne pose pas de problèmes de performances particuliers car elle ne concerne généralement qu'un sous-ensemble restreint des cellules de la grille concernée, étant donné que la ligne polygonale est un objet de dimension 1.

3.9 Proposition d'une architecture orientée objet

Dans les chapitres qui ont précédé, nous avons présenté toute une famille d'algorithmes. Il s'agit bien d'une famille car tous ces algorithmes présentent des caractéristiques communes. Ainsi, il s'agit dans tous les cas d'effectuer des modifications sur une grille régulière à partir d'informations relatives à un objet extérieur.

De plus, nous avons vu que plusieurs algorithmes faisant intervenir des objets très différents pouvaient nécessiter l'intervention d'une même technique. L'exemple le plus frappant est celui des trois algorithmes relatifs aux objets *tsolid*, *gstack* et *gshape* qui font tous trois intervenir un calcul d'intersection entre une grille et un tétraèdre.

Toutes ces considérations nous ont permis de mettre en évidence l'intérêt d'utiliser les techniques de la programmation orientée objet ([5],[6],[16],[22]) pour implémenter toutes ces fonctionnalités.

Dans ce chapitre, nous allons présenter une hiérarchie possible de classes correspondant à tous les éléments décrits dans ce chapitre.

Nous expliciterons ensuite comment se situent chacunes des composantes du modèle *sélecteur/affecteur* décrites dans le chapitre 3.3 de la page 51.

3.9.1 Présentation générale de la hiérarchie de classes

Dans ce chapitre, nous proposons une hiérarchie d'objets relatives aux fonctions de l'objet *voxet*. Celle-ci se décompose en quatre niveaux de classes. La raison d'être de chacuns de ces niveaux sera expliquée par la suite. Notons cependant que seules les classes de niveau 4 sont instanciables.

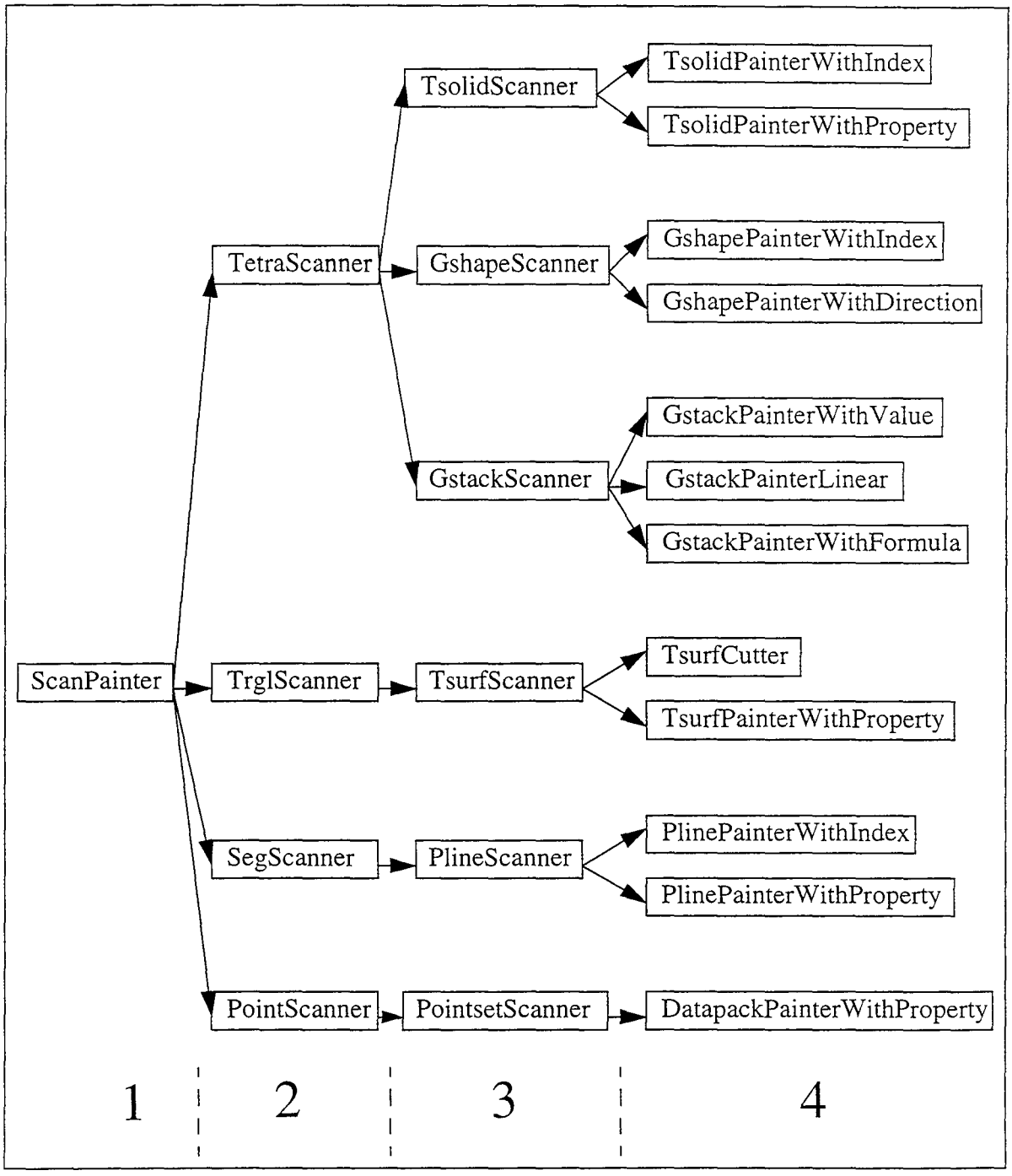


FIGURE 39. Présentation d'une hiérarchie de classes possible.

3.9.2 La classe de niveau 1

Cette classe, dénommée `ScanPainter`, se situe en sommet de hiérarchie et regroupe les

caractéristiques communes à toutes les fonctionnalités que nous avons présenté.

```
class ScanPainter {

protected :
        ScanPainter() {}
        int get_u() { return _u; }
        int get_v() { return _v; }
        int get_w() { return _w; }
        void set_u( int value ) { _u = value; }
        void set_v( int value ) { _v = value; }
        void set_w( int value ) { _w = value; }

        virtual void execute(Voxet *voxet) = 0 ;

private :
        int _u, _v, _w ;
};
```

Cette classe contient donc:.

- *_u, _v, _w*: trois entiers pour stocker les coordonnées d'une cellule de la grille.
- *execute(Voxet *voxet)* : fonction virtuelle pure dans laquelle s'effectue la modification à apporter à la cellule de coordonnées *(_u, _v, _w)*.

En effet, toutes les actions détaillées dans ce chapitre partagent le point commun qu'elles consistent à effectuer une modification sur des cellules d'une grille régulière.

3.9.3 Les classes de niveau 2

Ces classes contiennent chacunes une fonction membre *scan()*, où s'effectue le calcul d'intersection entre une grille régulière et un objet géométrique élémentaire (un point, un segment, un triangle ou un tétraèdre). C'est également dans ces classes que seront stockées les informations qui ne sont, par définition, qu'accessibles au cours de cette opération.

Prenons l'exemple de la classe TetraScanner:

```
class TetraScanner : public ScanPainter {

protected :
        TetraScanner() {}
        float * get_bc() ;
        void set_bc(float bc0, float bc1, float bc2) ;
        void scan(Voxet* voxet, Tetra* tetra);

private :
        float bc[3] ;

}
```

bc[3] est un tableau de trois réels servant à stocker les coordonnées barycentriques d'un nœud de la grille dans un tétraèdre du solide. *get_bc()* et *set_bc()* sont les fonctions virtuelles de lecture et d'affectation des valeurs de ces coordonnées barycentriques.

void scan(Voxet voxet, Tetra* tetra)* est la fonction membre dans

laquelle s'effectue le calcul d'intersection entre un tétraèdre et une grille régulière.

3.9.4 Les classes de niveau 3

Ces classes contiennent une fonction membre *scan()*, où s'effectue la décomposition géométriques d'objets de haut niveau (nuages de points, lignes polygonales, surfaces triangulées, etc...) sous forme d'entités élémentaires (points, segments, triangles, tétraèdres). Ces classes contiennent également les informations qui sont accessibles au cours de cette étape.

Prenons l'exemple de la classe TsolidScanner:

```
class TsolidScanner : public TetraScanner {

protected :
        TsolidScanner() {}
        Tetra* get_tetra() ;
        void set_tetra(Tetra* tetra) ;
        void scan(Voxet* voxet, Tsolid* solid) ;

private :
        Tetra *tetra ;

}
```

scan(Voxet* voxet, Tsolid* solid) est une fonction membre de la classe *TsolidScanner* qui parcours l'ensemble des tétraèdres d'un objet *tsolid* et qui, pour chacun d'entre eux, effectue un appel à la fonction membre *scan()* de la classe *TetraScanner* dont elle dérive.

tetra est un pointeur sur le tétraèdre courant, à partir duquel on peut accéder à la géométrie et aux propriétés de ses quatre sommets.

3.9.5 Les classes de niveau 4

Ces classes sont les seules à pouvoir être instanciées. Elles contiennent les informations définissant la modification à apporter aux nœuds de la grille régulière.

Ces classes contiennent une fonction membre *paint()* qui est l'appel de l'utilisateur pour effectuer la modification à proprement parler.

C'est à ce niveau qu'est explicitée la fonction virtuelle *execute()* qui modifie le contenu de chaque cellule de la grille. Cette fonction a accès aux informations fournies par les fonctions membres *scan()* et stockées dans les classes de niveau 2 et 3.

Chacune des classes de niveau 4 correspond à un type de modification.

Prenons l'exemple de la classe TsolidPainterWithProperty:

```
class TsolidPainterWithProperty : public TsolidScanner{

public
      TsolidPainterWithProperty() {};

      boolean paint( Voxet *voxet,
                     const Tsolid &tsolid,
                     const char *property) ;

protected :
      virtual void execute(Voxet *voxet) ;

}
```

Cette classe correspond à l'echantillonage d'une propriété connue aux nœuds d'un *tsolid* sur une grille régulière, que nous avons décrit en page 57.

La fonction membre `paint()` est la fonction qui permet à l'utilisateur d'accéder à cette fonctionnalité.

La fonction virtuelle `void execute(Voxet *voxet)` est appelée par la fonction membre `scan()` de la classe TetraScanner, et effectue la modification d'une cellule de la grille en fonction des informations stockées au niveau de la classe `ScanPainter` (les coordonnées u,v,w de la cellule), de la classe `TetraScanner` (les coordonnées barycentriques de cette cellule dans le tétraèdre courant), de la classe `TsolidPainter` (les informations relatives au tétraèdre courant) et de la classe `TsolidPainterWithProperty` (les instructions de la fonction virtuelle `execute()`).

3.9.6 Relation avec les intervenants du modèle *sélecteur/affecteur*

Nous avions décrit dans le chapitre 3.3 de la page 51 le modèle général de fonctionnement de l'objet *voxet*. Nous avions alors introduit les quatre intervenants de ce modèle: l'*application*, le *sélecteur*, l'*affecteur* et la structure *vx_info*.

Nous allons ici établir les correspondances qui existent entre ce modèle de fonctionnement et la hiérarchie de classes qui vient d'être décrite.

En premier lieu, nous pouvons dire que chaque classe de niveau 4 est l'équivalent d'une *application*. C'est en effet à ce niveau qu'est définie la nature de la modification à apporter à la grille régulière et c'est aussi ici que se fait l'appel au *sélecteur*.

Le *sélecteur* correspond, pour une *application* donnée représentée par une classe de niveau 4 donnée, aux classes de niveau 2 et 3 dont celle ci dérive. Nous avons vu en effet que le calcul d'intersection à été décomposé en deux niveaux selon la nature des objets géométriques considérés.

En ce qui concerne l'*affecteur*, celui-ci est très clairement représenté par la fonction virtuelle *execute()*, introduite au niveau de la classe de base de la hiérarchie, et explicitée dans les classes de niveau 4.

L'équivalent de la structure *vx_info* est plus flou. En effet, nous avons vu que les informa-

tions requises pour une action à effectuer sur une grille étaient disséminées à tous les niveaux de la hiérarchie, étant donné que chacune de ces informations n'est accessible qu'à certaines étapes bien précises de la procédure de calcul. Nous pourrons cependant considérer que la structure *vx_info* est, pour une action donnée, la réunion des variables de la classe qui lui correspond, et des variables des classes qui en dérivent.

3.9.7 Evolutivité de cette hiérarchie de classes

L'intérêt de mettre en place une telle hiérarchie de classes est double.

Il permet tout d'abord de profiter de tous les apports des techniques de programmation orientée objet. A ce propos, la technique la plus courament employée ici est sans doute la «réification» ([42]). Celle ci consiste à créer des objets (des classes) à partir d'entités qui n'ont pas de correspondant dans le monde des objets réels. C'est le cas de la totalité des classes de cette hiérarchie, qui sont chacunes l'équivalent d'une action, d'un traitement particulier sur un objet donné. Cette technique est avant tout un choix de conception, qui offre au programmeur un cadre de travail plus convivial qu'une programmation classique par fonction et passage d'arguments.

Le second intérêt que nous voyons à la mise en place de cette hiérarchie est que celle ci présente tout un ensemble de garanties pour assurer les évolutions futures de ce système. Cette évolutivité repose sur deux principes:

- La séparation claire entre les classes de niveau 3 et les classes de niveau 4: Cette séparation n'était pas apparue comme évidente au début de la conception de ce système. Pour chaque objet (chaque *sélecteur*), on ne pensait au départ qu'implémenter un type de modification (un *affecteur*). Une seule classe aurait pu suffire pour cela, mais, dès qu'un second type de modification a été nécessaire, la séparation entre niveau 3 et niveau 4 est apparue comme évidente.
C'est la raison pour laquelle nous avons introduit deux classes (`PointSetScanner` et `DatapackPainterWithProperty`) pour l'échantillonage d'un nuage de points sur une grille, alors qu'un seul type de modification n'est disponible jusqu'à présent à partir de cet objet.

- Les modifications ultérieures à apporter aux classes de niveau 2 et 3 ne peuvent être que des ajouts de variables (si une information supplémentaire est nécessaire), et des ajouts d'instructions d'affectation de valeurs à ces variables (dans les fonctions membres `scan()` de ces classes). En effet, il nous est apparu que le rôle des classes de niveau 2 et 3 devait impérativement se limiter aux calculs d'intersection (qui sont la raison d'être de ces classes) et aux stockage des informations accessibles au cours de ces calculs.

Cette dernière idée nous amène à soulever un problème que pose cette hiérarchie. Prenons l'exemple des classes `TsurfCutter` et `TsurfPaintWithProperty`. Ces deux classes dérivent toutes les deux de la classe `TrglScanner` et leurs fonctions `execute()` utilisent les informations accessibles à ce niveau. Cependant, ces informations ne sont pas nécessairement les mêmes dans les deux cas. Or la hiérarchie mise en place impose qu'une même fonction est chargée de collecter ces informations pour les deux actions. Ceci représente bien évidemment un surcoût (le calcul de coupure d'une grille suppose la collecte des informations pour échantilloner une propriété connue aux nœuds d'une surface!).

Ce problème est l'inconvénient majeur de la mise en place de la hiérarchie. Cependant,

celui-ci nous a semblé mineur par rapport aux avantages présentés ci-dessus. De plus, le surcoût dont il est question peut être minimisé si l'on suit le principe que les fonction `scan()` servent à stocker les informations brutes et que les fonctions `execute()` effectuent l'ensemble des instructions de traitement de ces informations. A cette condition, le surcoût se limite à quelques opérations d'affectation, qui, par définition, sont peu coûteuses.

3.10 Conclusion

Tout au long de ce chapitre, nous avons présenté un nouvel objet de la base de données géométrique du logiciel Gocad: l'objet *voxet*. Cet objet est une grille régulière à laquelle nous avons associé un système de codage d'informations par bits, ainsi que tout un ensemble de fonctionnalités propres.

Cet ensemble a été conçu dans un but bien précis: créer des modèles à base de grille régulière à partir d'un ensemble d'objets de type maillage irrégulier préexistant. Pour illustrer cette idée, nous pourrions dire que nous avons développé l'équivalent 3D d'un logiciel de D.A.O (dessin assisté par ordinateur) dans lequel:

- la grille régulière remplace l'écran de l'ordinateur.
- les objets Gocad sont les outils de dessin.
- les différentes façons de modifier une grille avec un même objet sont les couleurs de cet objet.

Pour tous les algorithmes que nous avons mis en place, nous avons fait en sorte que ceux-ci s'exécutent suffisamment rapidement pour pouvoir être utilisés sur des grilles de grande taille (de l'ordre de la centaine de nœuds dans les trois dimensions). Les tests de performances que nous avons effectués ont mis en évidence une complexité linéaire de chacuns de ces algorithmes et les temps de calcul absolus constatés permettent d'hors et déjà de créer des modèles de taille conséquente avec ces outils.

A la fin de ce chapitre, nous avons proposé une hiérarchie de classes possible pour implémenter ces outils et nous avons montré que celle-ci permettait d'envisager de manière sereine l'ajout de nouveaux outils à cet ensemble.

Chapitre 4

Interpolations de propriétés contraintes par un champ de directions

4.1 Introduction

Dans le second chapitre de cette thèse où nous avons décrit brièvement l'ensemble des données dont nous disposions pour la construction d'un modèle du réservoir de Gaz-de-France, nous avons mentionné le fait qu'une étude sédimentologique avait été effectuée sur les données de puits. Cette étude avait permis d'effectuer des corrélations entre les puits et de décomposer la séquence en unités sédimentologiques.

Pour chacune de ces unités, le sédimentologiste avait pu caractériser les formes de sédimentation en présence de plusieurs manières:

- par leur type de dépôt (réseaux en tresse, ceinture de méandre, etc...).
- par des attributs pétrophysiques (rapport sable/argile, porosité).
- par des propriétés géométriques (direction d'alongement principal, sinuosité, etc...).

Dans le cadre que nous nous sommes fixé de reconstituer la structure interne des réservoirs pétroliers, il est bien évident que ces informations géométriques doivent être exploitées en priorité.

C'est dans ce but que nous allons présenter un nouveau procédé d'interpolation, dans lequel il est possible d'injecter de telles données. Nous avons baptisé ce procédé de «méthode D.S.I anisotrope» ou encore de «méthode D.S.I contrainte par un champ de directions».

Il s'agit là d'un cas particulier d'utilisation de la méthode générale d'interpolation D.S.I (Discrete Smooth Interpolation). Dans la première partie de ce chapitre, nous présenterons un rapide descriptif de cette méthode. Une implémentation classique de celle-ci a été mise en place sur l'objet *voxet*. Nous présenterons les spécificités de l'utilisation de D.S.I sur des maillages de type grille régulière.

Le nouveau procédé d'interpolation en question sera détaillé dans la seconde partie de ce chapitre. En particulier, nous verrons les limitations qui se sont imposées quant au choix du support de calcul et à la stratégie pour optimiser le temps de calcul par rapport au coût en mémoire.

Nous envisagerons ensuite le problème de la génération des champs de directions qui contraignent les interpolations en question.

Enfin, une présentation détaillée des résultats obtenus sera proposée, dans laquelle nous considérerons une à une l'influence des différents facteurs intervenant dans la méthode de calcul.

A la fin de ce chapitre, nous envisagerons bien entendu les différents domaines d'utilisation de cette méthode. Une attention toute particulière sera portée à la reconstitution des formes chenalisées au sein des réservoirs pétroliers.

4.2 Présentation générale de la méthode DSI et implémentation sur l'objet *vozet*

Les techniques de C.A.O classiques permettent de construire des objets géométriques à l'aide de paramétrisations relativement simples. Dans le cas des surfaces, les représentations les plus couramment utilisées sont les paramétrisations de type *Bézier, Bsplines* ou *Nurbs*. Ces méthodes supposent la donnée d'un maillage dont les nœuds sont les points de contrôle de la surface et produisent une représentation mathématique polynomiale de cette dernière.

L'application de ces méthodes à la modélisation des surfaces géologiques rencontre un certain nombre de difficultés:

- la construction du maillage des points de contrôle se fait le plus souvent par l'ajustement de la géométrie de ce maillage à des points de données extérieurs et non par intervention de l'utilisateur comme c'est le cas dans les applications industrielles de la C.A.O classique.
- La répartition des informations géométriques extérieures est souvent très hétérogène, aussi bien par leur densité que par leur nature (points de données, mesures de pendagemétrie, mesures de rejets de failles, etc...).
- La construction des surfaces géologiques est un processus par étapes dont certaines imposent une remise en cause importante de la topologie du maillage de départ, et par la même des nœuds de contrôle de la surface.

La méthode D.S.I tend à s'imposer face aux méthode de C.A.O classique dans le domaine de la modélisation des surfaces géologiques car elle a su apporter des réponses cohérentes à tous ces problémes.

4.2.1 Présentation de la méthode D.S.I

La méthode D.S.I constitue le moteur du logiciel de modélisation d'objets géologiques Gocad. Il s'agit d'une méthode d'interpolation travaillant sur des maillages irréguliers, au sens large, qui permet l'estimation d'un paramètre physique φ stocké au niveau de chaque sommet de ces maillages. Cette estimation passe par:

- la minimisation d'un critère global de «rugosité» de φ.

- la minimisation au sens des moindres carrés d'expressions de φ, qui permet de tenir compte de contraintes de type linéaire.

Dans tout ce qui va suivre, nous n'envisagerons pas l'aspect contraintes linéaires de D.S.I. Seul l'aspect minimisation du critère global de rugosité nous intéressera.

Définition du support de calcul

Comme nous l'avons dit précédemment, D.S.I fonctionne sur des maillages irréguliers, sans restriction en théorie. Dans la pratique les courbes sont modélisés par des lignes polygonales, les surfaces par des surfaces triangulées et les volumes par des solides tétraédrisés. Ces derniers peuvent également être décrits par des grilles.

Nous supposerons donc que l'on dispose d'un maillage constitué par un ensemble Ω de nœuds k. Cet ensemble Ω est décomposé en deux sous-ensembles:

- un sous-ensemble L de nœuds sur lesquels un paramètre φ est connu.

- un sous-ensemble I de nœuds où φ est inconnu. $I = \Omega - L$.

Chacun de ces nœuds k possède un ensemble de voisins noté $\Lambda(k)$. Le voisinage N(k) est défini par:

$$N(k) = \Lambda(k) \cup \{k\}$$

Définition d'un critère de rugosité

En chaque nœud k de Ω, on définit la rugosité locale de φ, notée $R(\varphi|k)$ par:

$$R(\varphi|k) = \left| \sum_{\alpha \in N(k)} v^{\alpha}(k) \cdot \varphi(\alpha) \right|^2$$

où les coefficients $v^{\alpha}(k)$ sont des coefficients pondérateurs vérifiant:

$$\begin{cases} \forall \alpha \in \Lambda(k), \quad v^{\alpha}(k) > 0 \\ v^{k}(k) = - \sum_{\alpha \in \Lambda(k)} v^{\alpha}(k) \end{cases}$$

A partir de ces rugosités locales, on définit la rugosité globale de φ sur l'ensemble du maillage, notée $R(\varphi)$, par:

$$R(\varphi) = \sum_{k \in \Omega} \mu(k) \cdot R(\varphi|k)$$

où les coefficients $\mu(k)$ sont les pondérateurs définis positifs de chaque sommet k du maillage.

La rugosité de φ est donc une quantité positive que D.S.I tend à minimiser. Au minimun de

$R(\varphi)$, on a donc:

$$\frac{\partial R(\varphi)}{\partial \varphi} = 0$$

Ce qui équivaut à écrire:

$$\forall \alpha \in \Omega, \frac{\partial R(\varphi)}{\partial \varphi(\alpha)} = 0$$

En développant cette expression, on obtient:

$$\frac{1}{2} \cdot \frac{\partial R(\varphi)}{\partial \varphi(\alpha)} = \varphi(\alpha) \cdot \left\{ \sum_{k \in N(\alpha)} \mu(k) \cdot (v^{\alpha}(k))^2 \right\} + \sum_{k \in N(\alpha)} \left\{ \mu(k) v^{\alpha}(k) \sum_{\beta \in N(k), \beta \neq \alpha} v^{\beta}(k) \varphi(\beta) \right\}$$

L'algorithme mis en place pour parvenir à la solution qui minimise $R(\varphi)$ est un algorithme itératif de type Gauss-Seidel dans lequel on procède à chaque itération au remplacement de la valeur φ de chaque sommet α de I par:

$$\varphi(\alpha) = -\frac{1}{\sum_{k \in N(\alpha)} \mu(k) \cdot (v^{\alpha}(k))^2} \cdot \sum_{k \in N(\alpha)} \left\{ \mu(k) v^{\alpha}(k) \sum_{\beta \in N(k), \beta \neq \alpha} v^{\beta}(k) \varphi(\beta) \right\}$$

La convergence de cette méthode vers une solution finale a été établie, ainsi que l'unicité de cette solution finale. Il a également été démontré que cette solution de dépendait pas de la solution initiale employée.

4.2.2 Implémentation de la méthode D.S.I sur l'objet *voxet*

L'objet *voxet* a été décrit en détail dans le chapitre 3.2 de la page 40. Il s'agit d'une grille régulière à laquelle a été greffé un système de codage d'informations propre et un certains nombre de fonctions liées à son utilisation dans le cadre de la modélisation des objets naturels.

Nous allons montrer ici comment ces propriétés du *voxet* ont été employées pour mettre en œuvre l'implémentation de D.S.I sur cet objet.

Le support de calcul

Dans le cas de l'objet *voxet*, Ω est constitué dans le cas général par l'ensemble des nœuds de la grille. Cependant, il faut noter que l'introduction du bit *region* pour chaque nœud de la grille nous a permis de définir la notion de sous-ensemble d'une grille. Il nous a semblé utile de permettre à l'utilisateur de restreindre l'étendue de certains algorithmes, et en particulier de celui de D.S.I, à ce sous-ensemble. Ω sera donc par la suite l'ensemble des nœuds de la grille dont le bit *region* est à 1.

De même l'ensemble L sera constitué par les nœuds dont le bit *region* et le bit *painted* sont à 1 et l'ensemble I sera l'ensemble des nœuds dont le bit *region* est à 1 et le bit *painted* est à 0. On vérifie instantanément que la relation $\Omega=I+L$ est vérifiée.

Notion de voisinage dans une grille régulière

Nous avons vu que la mise en place de D.S.I sur un maillage Ω supposait en chaque sommet de Ω la définition d'un voisinage. Dans le cas d'un maillage irrégulier, les voisinages sont connus car ils font partie intégrante de la définition de l'objet. Dans le cas d'une grille régulière, il faut faire un choix sur le type de voisinage à utiliser.

La choix qui a été fait est celui de la 6-connectivité (voir la Figure 26 de la page 48). En effet, celle ci correspond à un voisinage symétrique autour du nœud central où le nombre de voisins est minimal (ce qui entraîne automatiquement un nombre d'opérations de calcul minimal).

Chaque nœud de la grille à l'exception des nœuds du bord possède donc au plus six voisins. Cependant, il nous a paru important que l'algorithme de D.S.I ait accès aux informations de codage explicite des voisins présentées dans le chapitre 3.2.3 de la page 47. Le voisinage d'un nœud pourra donc être amputé de un ou plusieurs voisins en fonction de ces informations.

La routine de recherche des voisins effectuera donc pour chacun des six voisins potentiels d'un nœud N un double test:

- un test d'appartenance à la région courante de la grille.
- un test de connection à partir des informations de voisinage.

Modalités du calcul proprement dit

Comme cela a été décrit dans le chapitre 4.2.1 de la page 102, l'algorithme mis en place est un algorithme itératif. A chaque itération, on procède à un remplacement des valeurs de φ des nœuds de L par la valeur issue de l'équation de D.S.I développée plus haut.

Les paramètres $\mu(k)$ sont positionnés à la valeur 1, ainsi que les coefficients $v^{\alpha}(k)$.

Le nombre de nœuds d'une grille régulière étant souvent très supérieur à celui d'un maillage irrégulier, il n'a pas été jugé souhaitable de stocker en mémoire les voisinages de chaque nœud. En conséquence, la routine de recherche de voisins que nous venons de décrire est appelée en cours de calcul pour chacun des nœuds parcourus. Ceci induit un surcoût en calcul par rapport aux implémentations déjà mises en place sur des objets de type maillage irrégulier.

Les premiers essais effectués ont permis de s'assurer de la convergence de l'algorithme.

Nous avons également envisagé le problème de fournir une solution initiale la plus proche possible de la solution finale de D.S.I. Pour cela, nous avons testé deux approches:

- Un algorithme de type «données explosives» ([10]) qui consiste, pour chaque nœud de L, à lui affecter la valeur du nœud de I qui lui est le plus proche. Cet algorithme, inspiré des techniques de morphologie mathématique , est très simple mais fournit une solution encore éloignée de la solution de D.S.I.
- Un algorithme d'initialisation de type multigrille ([10]). Cet algorithme initialise une grille

en effectuant un calcul récursif de la solution D.S.I sur une sous-grille. Cette approche est plus coûteuse que la précédente mais fournit des résultats très proches de la solution finale.

Evaluation des performances

Des tests de performances ont été effectués afin de donner un ordre de grandeur des temps de calcul. Pour effectuer ces tests, on dispose:

- d'un modèle surfacique synthétique composé de quatre horizons et d'une surface correspondant à une goutte de sel.
- d'un ensemble de points de données auxquels sont affectés des valeurs de vitesse sismique.

On se propose ici de fournir une estimation du paramètre vitesse par l'algorithme D.S.I sur l'ensemble du domaine décrit au moyen d'une grille régulière. Cette estimation respecte les points de données et prend en compte les surfaces triangulées du modèle comme des discontinuités de la loi de vitesse.

Le calcul complet se fait en plusieurs étapes:

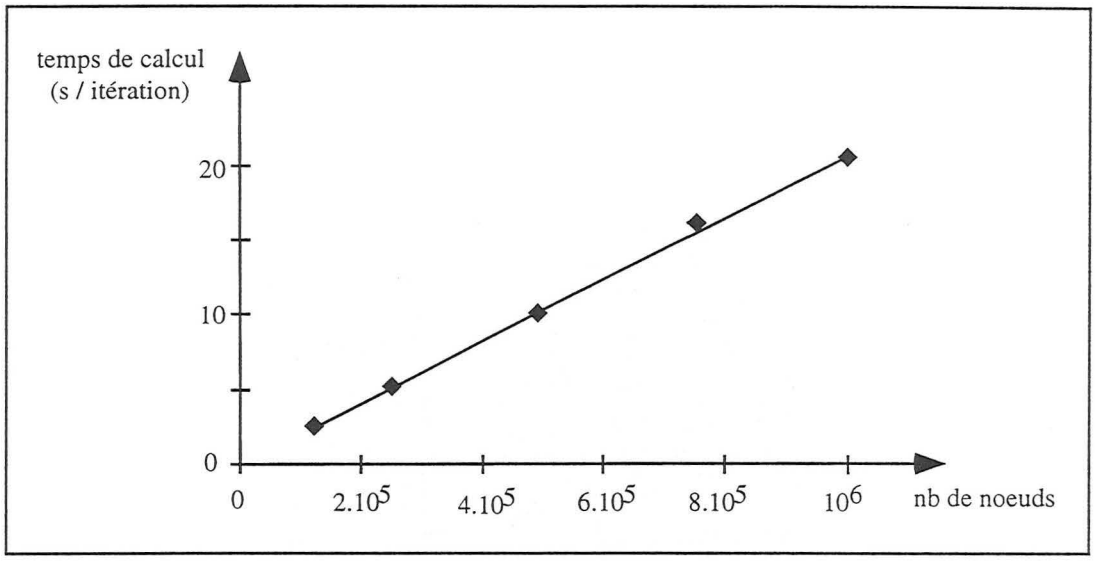
- Echantillonnage des valeurs de vitesse des points de données sur les nœuds de la grille.
- Coupure de la grille par les surfaces correspondant aux horizons et à la goutte de sel.
- Initialisation de la grille par l'algorithme de type «multigrille» décrit plus haut.
- Estimation de la vitesses sismique sur l'ensemble du modèle par la méthode D.S.I. Le nombre d'itérations de l'algorithme est de 30, ce qui garantit la convergence de l'algorithme vers sa solution, compte tenu de la méthode d'initialisation employée.

Dans ce qui suit, les résultats présentés ne concernent que la dernière étape de ce calcul et ont été obtenus sur la machine «extreme» (voir page 149).

Tableau .6 : Evaluation des performances de l'algorithme D.S.I sur une grille régulière

test n°	1	2	3	4	5
taille de la grille	50*50*50	63*63*63	79*79*79	91*91*91	100*100*100
nombre de nœuds	125000	250047	493039	753571	1000000
temps de calcul total (s)	75	153	301	481	615
temps de calcul par itération (s)	2.5	5.1	10	16	20.5

L'évolution du temps de calcul en fonction de la taille de la grille est la suivante:



Cette figure montre clairement que le temps de calcul évolue linéairement en fonction de la taille de la grille. Ceci ne nous surprend pas, étant donné que l'algorithme D.S.I consiste à effectuer un parcours systématique des nœuds de la grille sur lesquels il effectue une opération locale, donc indépendante de la taille globale de l'objet sur lequel on travaille.

Les temps de calcul absolus obtenus sur la machine extreme ne permettent pas d'envisager une utilisation en temps réel de cet algorithme. Nous pensons néanmoins que celui-ci présente un grand intérêt, notamment quand la distribution du paramètre estimé doit tenir compte de la présence de discontinuités, comme c'est la cas dans l'exemple présenté ci-dessous.

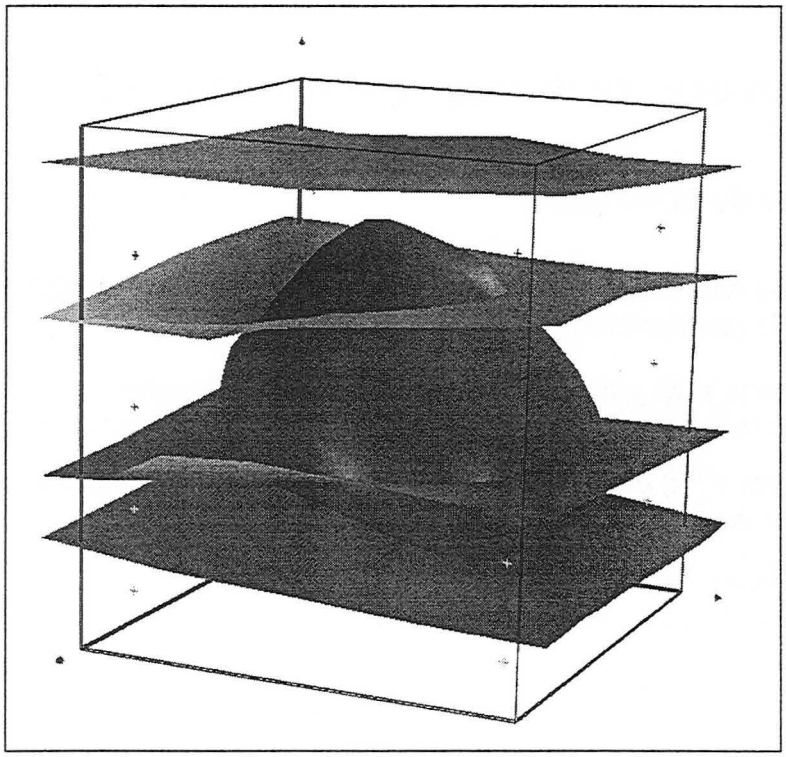


FIGURE 40. Modèle surfacique et points de données de vitesse.

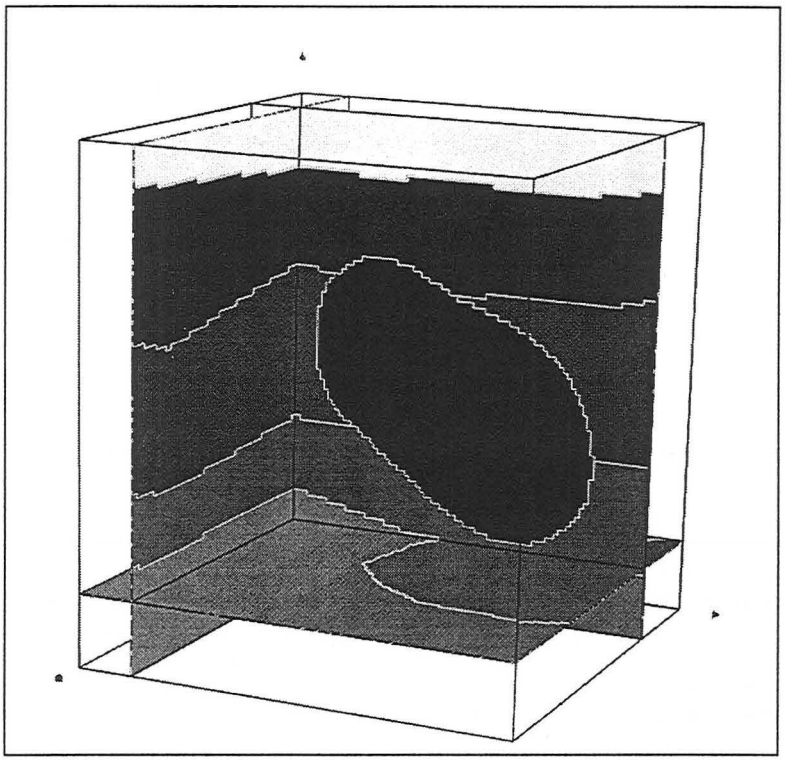


FIGURE 41. Modèle de vitesse obtenu.

Le modèle obtenu est représenté sur trois plans de la grille, les discontinuités sont représentées en blanc sur ces plans.

4.3 Mise en oeuvre de l'algorithme de D.S.I anisotrope

4.3.1 Principe de la méthode

Nous avons vu dans le chapitre qui précède que le formalisme de D.S.I fait intervenir un certain nombre de pondérateurs:

- les coefficients $\mu(k)$ qui pondèrent l'influence de la rugosité de chaque sommet k du maillage dans l'expression de la rugosité globale de φ sur l'ensemble du maillage.

- les coefficients $v^{\alpha}(k)$ qui pondèrent l'importance de chaque satellite α d'un nœud k dans la formule locale de la rugosité de φ en ce nœud.

Dans l'implémentation classique de D.S.I que nous venons de présenter, tous ces pondérateurs sont positionnés à une valeur constante égale à 1.

La solution de D.S.I correspond alors à une configuration dans laquelle la valeur de φ de chaque nœud de Ω est la plus proche possible de la moyenne des valeurs de φ de ses satellites.

De même, dans le cas d'une modélisation géométrique, la solution D.S.I correspond à un maillage dont chaque nœud est le plus proche possible du barycentre de ses satellites.

Dans le but que nous nous sommes fixé de contraindre le résultat des interpolations de D.S.I par des données de direction, moduler les valeurs des coefficients $v^{\alpha}(k)$ nous a apparu comme une voie intéressante à explorer.

Dans ce cas, la solution de D.S.I correspond à une distribution de φ dans laquelle la valeur de φ de chaque nœud de Ω est la plus proche possible de la moyenne des valeurs de φ de ses satellites, ponérée par les coefficients $v^{\alpha}(k)$.

La démonstration théorique de la convergence de la méthode D.S.I n'impose que le signe (positif) de ces coefficients, et permet donc ce type de modification de l'algorithme initial.

4.3.2 Méthodes de calcul des pondérateurs

Le problème posé est alors le suivant: on dispose d'un nœud k muni d'un ensemble de voisins α_i, on connait également en ce nœud k les trois composantes Dx, Dy et Dz d'un vecteur $\vec{D}$. Comment peut on traduire la donnée de $\vec{D}$ sous la forme d'un contraste entre les pondérateurs des satellites de k?

Deux méthodes ont été envisagées et mises en place.

La méthode des ellipsoïdes

Dans cette méthode, le contraste entre pondérateurs peut être matérialisé au moyen d'un ellipsoïde centré autour du nœud k, de grand axe la direction D. L'alongement principal λ_1 de cet ellipsoïde est fourni par l'utilisateur et correspond au degré d'anisotropie des pondérateurs.

Pour ne pas rendre le problème trop complexe inutilement, les alongements moyens et mineurs λ_2 et λ_3 de l'ellipsoïde sont égaux et prennent la valeur 1.

La valeur du pondérateur $v^{\alpha_i}(k)$ du satellite α_i de k est donnée, comme l'illustre la Figure 42, par la longueur du segment de la demi-droite $[k,\alpha_i)$ avec cet ellipsoïde.

De la sorte, plus la direction du segment $[k,\alpha_i]$ est proche de la direction D, plus le poids du

satellite α_i est élevé.

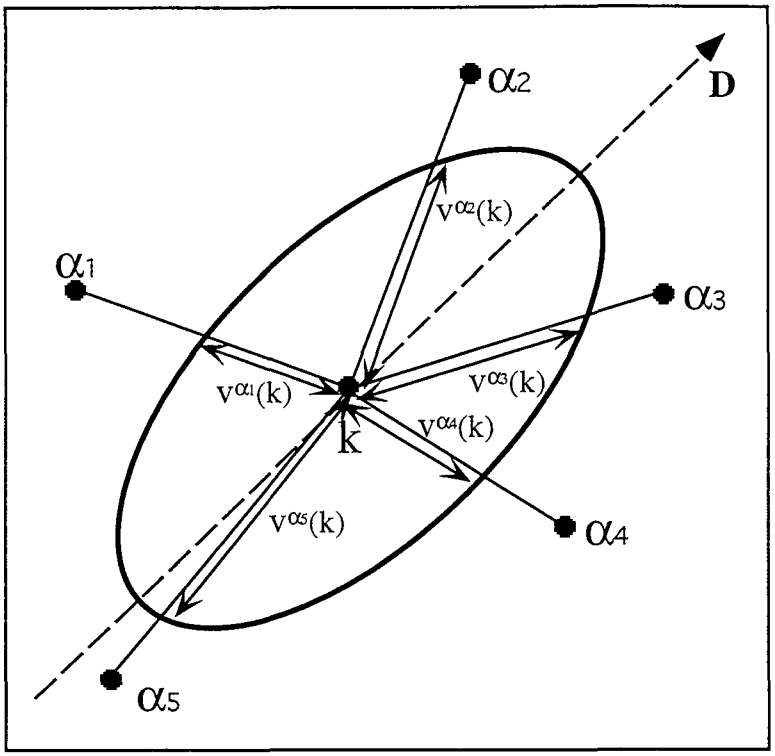


FIGURE 42. Calcul des pondérateurs par la méthode des ellipsoïdes.

La méthode de l'antialiasing

L'antialiasing ([21],[25]) est une technique utilisée en infographie pour améliorer le dessin des segments de droites sur un écran d'ordinateur.

Les algorithmes classiques de dessin tels que l'algorithme de Bresenham fournissent des rendus de lignes en marches d'escaliers, du fait de la décomposition de l'écran en pixels élémentaires.

Des routines d'antialiasing ont été mises en place pour améliorer le rendu de ces lignes. Ces routines allument plus de pixels que les précedentes pour représentrer les lignes et modulent l'intensité de l'allumage des pixels de l'écran de façon à obtenir un résultat plus réaliste.

La méthode de calcul des pondérateurs que nous allons présenter est directement inspirée de ces techniques d'antialiasing. Il existe plusieurs algorithmes d'antialiasing. Celui que nous avons sélectionné présente l'avantage d'être relativement simple et peu coûteux en opérations de calcul.

La méthode consiste, pour chaque satellite, à calculer la distance de ce point à la droite $(k,\vec{D})$. Si celle-ci est supérieure à une distance seuil d_{max} à définir (voir la Figure 43), le pondérateur prend pour valeur 1 , sinon il est obtenu par la formule:

$$v^{\alpha_i}(k) = 1 + \frac{d_{max} - d}{d_{max}}(D_a - 1)$$

où d désigne la distance mesurée entre le point α_i et la droite $(k,\vec{D})$, et $D_a \geq 1.0$ le degré d'anisotropie défini par l'utilisateur.

Ce type de calcul n'a de sens que dans le cas des maillages de type grille régulière. En effet, la notion de distance ne peut pas être prise en compte dans le cas des maillages irréguliers, où l'orientation et la longueur des arêtes des maillages sont par définition quelconques.

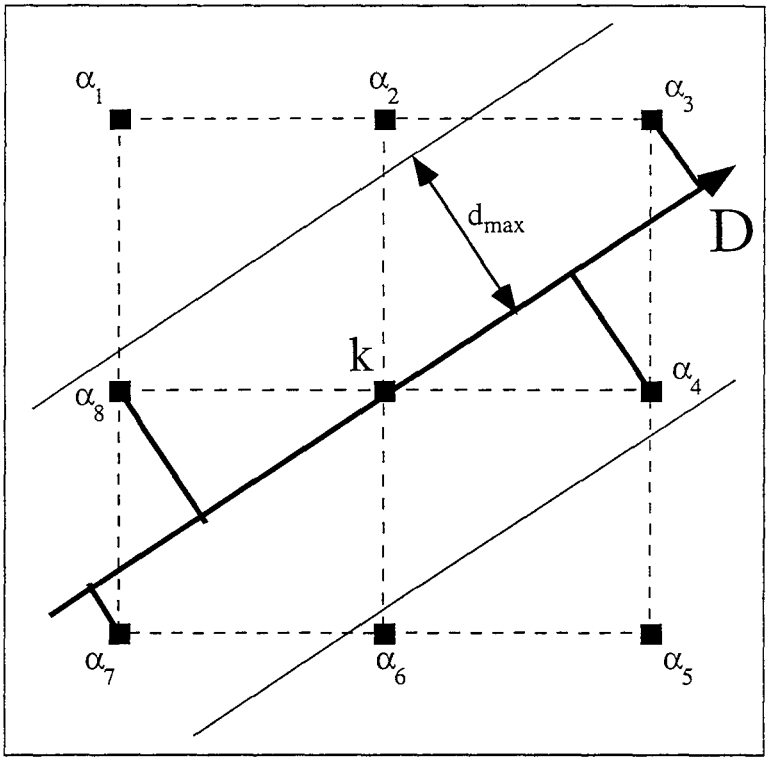


FIGURE 43. Calcul des pondérateurs par la méthode de l'antialiasing.

4.3.3 Choix du support de calcul, quelques limitations

Nous avons vu comment il était possible de traduire la donnée d'un vecteur au nœud k d'un maillage sous la forme d'un contraste entre les pondérateurs de ses satellites. Nous allons maintenant poser le problème de déterminer le type de maillage sur lequel nous allons faire fonctionner l'algorithme de D.S.I anisotrope avec ces nouveaux pondérateurs.

D.S.I fonctionne par définition sur n'importe quel type de maillage. Cependant, il est bien clair que nous n'obtiendrons de bons résultats dans le cas qui nous intéresse, que si l'agencement des nœuds avec leurs satellites au sein du maillage permet de traduire les données de direction qui y sont stockées, par de forts contrastes de valeurs entre les pondérateurs.

Les premiers essais de notre nouvel algorithme ont été effectués sur les surfaces triangulées. Ils ont tout d'abord permis de vérifier par l'exemple que, à la condition que les conventions de signe des pondérateurs soient respectées, celui-ci convergeait effectivement vers une solution finale unique et indépendante de la solution de départ proposée.

Par ailleurs, il est apparu très tôt que le choix d'un tel type de maillage ne pouvait pas convenir. Le type de problème que nous avons rencontré est illustré dans la Figure 44. Considérons deux cas de figure :

1. La direction d'anisotropie qui contraint l'interpolation est verticale:
   Dans ce cas, quelle que soit la méthode de calcul des pondérateurs, en reprenant les notations de la figure, on a:

$$v^{\alpha_2}(k) = v^{\alpha_5}(k)$$

$$v^{\alpha_2}(k), v^{\alpha_5}(k) \quad \gg \quad v^{\alpha_1}(k), v^{\alpha_3}(k), v^{\alpha_4}(k), v^{\alpha_6}(k)$$

   on obtient donc un fort contraste entre pondérateurs, qui traduit bien la direction verticale de contrainte de l'interpolation.

2. La direction d'anisotropie qui contraint l'interpolation est horizontale (cas représenté sur la figure):
   Dans ce cas, quelle que soit la méthode de calcul des pondérateurs, en reprenant les notations de la figure, on obtient:

$$v^{\alpha_3}(k) = v^{\alpha_4}(k) = v^{\alpha_6}(k) = v^{\alpha_1}(k)$$

   Par ailleurs, aucune des arêtes n'étant proche de la direction D, on n'obtient pas un fort contraste entre les pondérateurs, qui ne traduit pas la direction de contrainte d'interpolation.

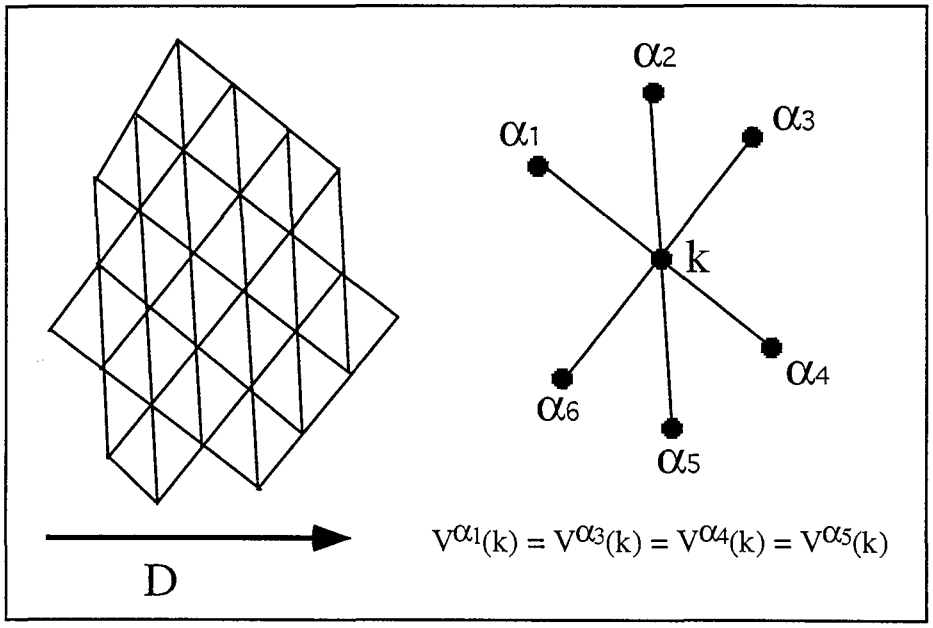


FIGURE 44. Problèmes liés à l'utilisation des surfaces triangulées.

L'importance du contraste entre les pondérateurs varie donc fortement en fonction de l'orientation de la direction d'anisotropie considérée.

Les conséquences de ce problème ont pu être constatées de visu. Sur l'exemple ci-dessus, l'influence de la direction d'anisotropie est plus ou moins marquée selon que celle-ci coïncide avec les trois directions des arêtes du maillage. Dans le cas le plus défavorable où D est hori-

zontale, son influence est quasiment nulle et le résultat obtenu est très proche de celui obtenu avec le D.S.I classique (ou isotrope).

Par ailleurs, la triangulation présentée ici est très régulière. Dans le cas d'une triangulation quelconque, la réponse de l'interpolateur à une direction de contrainte est à la fois sensible à l'orientation de cette direction et aux variations locales du voisinage des nœuds.

En conséquence, nous avons abandonné l'idée d'utiliser les surfaces triangulées comme support de l'algorithme de D.S.I anisotrope.

Cet échec nous a cependant permis de définir les critères de qualité d'un maillage dans le cadre d'une utilisation par l'algorithme D.S.I anisotrope. Ces critères sont au nombre de trois:

1. Les calculs des pondérateurs doivent donner le même résultat en tous les nœuds du maillage pour une direction donnée.
2. La configuration des voisinages de chaque nœud doit permettre de matérialiser toutes les directions de l'espace par de forts contrastes entre pondérateurs.
3. La configuration des voisinages de chaque nœud doit permettre de traduire de la manière la plus égale possible toutes les directions de l'espace.

A la lumière des échecs essuyés avec les surfaces triangulées, le respect du premier critère nous a amené à abandonner l'idée d'utiliser tout type de maillage irrégulier. En effet, les variations locales de voisinage entre nœuds, qui portent non seulement sur la disposition des satelittes par rapport au nœud central mais aussi sur leur nombre, font que les conversions des directions en pondérateurs sont très variables d'un nœud du maillage à un autre.

Nous avons donc été amenés à envisager l'utilisation des grilles régulières. Dans ce cas, le premier critère de qualité est automatiquement respecté.

Le problème qui se pose alors est de trouver un type de voisinage pour les cellules de la grille, qui permette de respecter les deux autres critères. En effet, il est bien évident que la 6-connectivité des grilles 3D (et son équivalent la 4-connectivité en 2D) que nous avons utilisée jusqu'à présent ne nous permet pas d'atteindre ces deux objectifs.

Il va donc falloir introduire un nouveau type de voisinage. Pour répondre au critère 2, ce nouveau voisinage devra comporter plus de voisins, ceux-ci étant toujours répartis de manière symétrique par rapport au nœud central. Cependant, il est bien clair qu'introduire plus de nœuds dans les voisinages se traduira par une augmentation du nombre d'opérations de calcul pendant l'interpolation.

Nous avons donc opté pour une solution intermédiaire qui permet de respecter convenablement les critères de qualité sans imposer un surcoût en calcul trop important. Cette solution consiste à passer en 2D de la 4-connectivité à la 8-connectivité, et en 3D de la 6-connectivité la

26-connectivité, comme illustré sur la Figure 45.

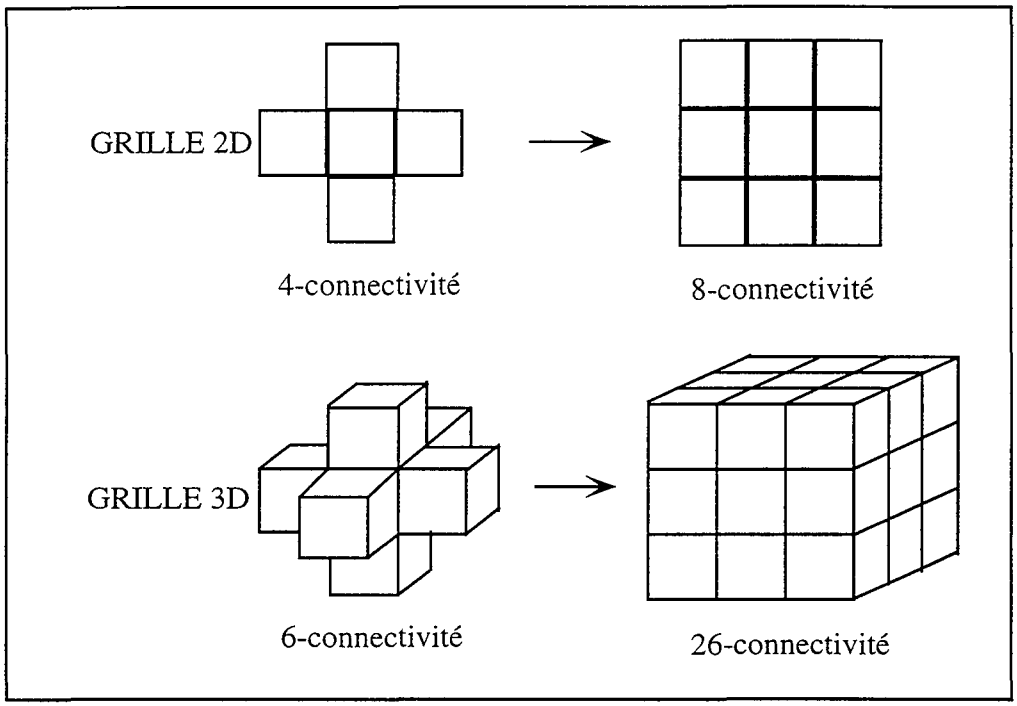


FIGURE 45. Modification des voisinages dans les grilles régulières.

4.3.4 Méthodes de recherche des satellites

Nous avons vu précédemment que l'algorithme D.S.I ne pouvait fonctionner convenablement sur des grilles régulières qu'en modifiant le voisinage des cellules. La mise en place de ce nouveau type de voisinage suppose la mise au point d'une nouvelle routine de recherche de voisins.

Cette recherche des voisins se fait suivant les mêmes modalités que dans le cas de l'algorithme D.S.I classique. Les tests de voisinages portent à la fois sur les données de codage de voisinage (les bit *next_u*, *next_v* et *next_w*) et sur l'appartenance des cellules à la région courante de la grille.

La routine mise en place pour effectuer la recherche des voisins dans le cas de la 26-connectivité utilise en sous-main la routine de recherche relative à la 6-connectivité, et prend la forme

suivante:

```
Voxel_get_26_neighbours(long uvw, long uvw_26[ ]) {
    Voxel_get_6_neighbours(uvw, uvw_6[ ]) ;
    Pour (chacun des 6 voisins uvw_6[i] de uvw) faire
        Stocker l'indice uvw_6[i] dans le tableau uvw_26[ ] ;
        Voxel_get_6_neighbours(uvw_6[i], uvw_sats_6[ ]) ;
        Pour (chacun des six voisins uvw_sats_6[j] de uvw_6[i]) faire
            Si (le nœud uvw_sats_6[j] appartient à l'auréole de uvw) alors
                Stocker l'indice uvw_sats_6[i] dans le tableau uvw_26[ ] ;
            fin Si
            Voxel_get_6_neighbours(uvw_sats_6[j], uvw_sats_sats_6[ ]) ;
            Pour (chacun des six voisins de uvw_sats_sats_6[k] de uvw_sats_6[j]) faire
                Si (le nœud uvw_sats_sats_6[k] appartient à l'auréole de uvw) alors
                    Stocker l'indice uvw_sats_sats_6[k] dans le tableau uvw_26[ ] ;
                fin Si
            fin Pour
        fin Pour
    fin Pour
}
```

ALGORITHME 21. Algorithme de recherche des voisins dans le cas de la 26-connectivité.

On procède donc ici à une recherche des voisins par l'appel sur trois niveaux de la routine de recherche des voisins dans le cas de la 6-connectivité. A chaque voisin potentiellement isolé, on effectue un test d'appartenance de ce voisin à l'auréole du nœud central. Si ce test est positif, son indice est stocké dans le tableau des 26 voisins potentiels.

Est considéré comme voisin d'un nœud N, tout nœud de la grille appartenant à l'auréole à 26 nœuds de N, qui peut être atteint en au plus trois déplacements élémentaires en partant de N.

4.3.5 Implantation finale de l'algorithme

gestion du coût en mémoire par rapport au temps de calcul

L'implantation de l'algorithme de D.S.I anisotrope introduit de nouvelles opérations de calcul ou de stockage d'informations, telles que:

- le stockage des données de direction pour contraindre l'interpolation.
- le calcul et le stockage des pondérateurs des satellites de chaque nœud.
- une fonction de recherche des satellites sur une auréole de 26 voisins potentiels.

- un accroissement du nombre d'opérations de calcul dû à l'augmentation du nombre des satellites de chaque nœud.

Toutes ces opérations introduisent bien entendu des surcoûts en terme de temps de calcul et en terme d'espace mémoire utilisé.

Pour certaines de ces opérations, des choix peuvent être opérés pour privilégier l'aspect temps de calcul par rapport à l'aspect espace mémoire.

Ainsi, les pondérateurs des satellites peuvent être calculés en amont de l'interpolation et stockés dans une mémoire à laquelle accède l'interpolateur, ou alors être recalculés au fur et à mesure des besoins en fonction des données de direction disponibles.

Dans ce cas là, les tests de performances que nous avons effectués montrent clairement que le stockage en mémoire des pondérateurs constitue l'option la plus économique.

Le même type de problème s'est posé pour la recherche des satellites. Est t'il préférable de déterminer le voisinage de chacun des nœuds de la grille et de le stocker en mémoire en amont de l'interpolateur, ou alors de recalculer celui-ci au cours des itérations de l'algorithme?

Là encore, les tests de performances que nous avons effectués pour répondre à cette question nous ont amenés à privilégier la recherche en amont et le stockage en mémoire des voisinages. Plusieurs méthodes ont été envisagées pour minimiser la quantité de mémoire nécessaire à ce stockage. Cependant, tous les mécanismes de compression d'information que nous avons mis en place pour minimiser l'espace utilisé ont introduit un surcoût en calcul lié à la décompression de ces informations et nous avons là encore du adopter l'option la plus coûteuse en mémoire, au profit néanmoins du temps de calcul global de l'algorithme.

Ce constat est certes décevant, mais il n'est en revanche pas totalement surprenant. En effet, l'algorithme de minimisation de la rugosité d'un maillage par D.S.I est un algorithme de calcul itératif par nature extrêmement répétitif qui met en œuvre relativement peu d'opérations élémentaires de calcul auquelles il fait appel un très grand nombre de fois.

En conséquence, il est très préférable de stocker en mémoire le résultat de ces opérations si celui ci est invariable d'une itération à l'autre. C'est le cas des deux exemples qui ont été présentés plus haut.

En conclusion, nous pouvons dire que l'implantation de l'algorithme de D.S.I anisotrope que nous avons effectuée privilégie le temps de calcul aux dépends du coût en mémoire. Il est alors clair que ceci va introduire des limitations sur la taille des grilles que nous allons pouvoir utiliser.

Evaluation des performances

Des tests de performances ont été effectués pour donner un ordre de grandeur de ces temps de calcul. Les calculs portent sur une grille régulière dont on fait varier la taille en nombre de nœuds, sans en modifier la géométrie. Sur chacun de ces objets, on effectue 30 itérations de D.S.I, de façon à étudier l'évolution du temps de calcul en fonction de la taille de la grille. A titre de comparaison, nous avons également fait tourner l'algorithme de D.S.I classique sur ces objets dans les mêmes conditions, de façon à estimer le surcoût en temps de calcul induit par le

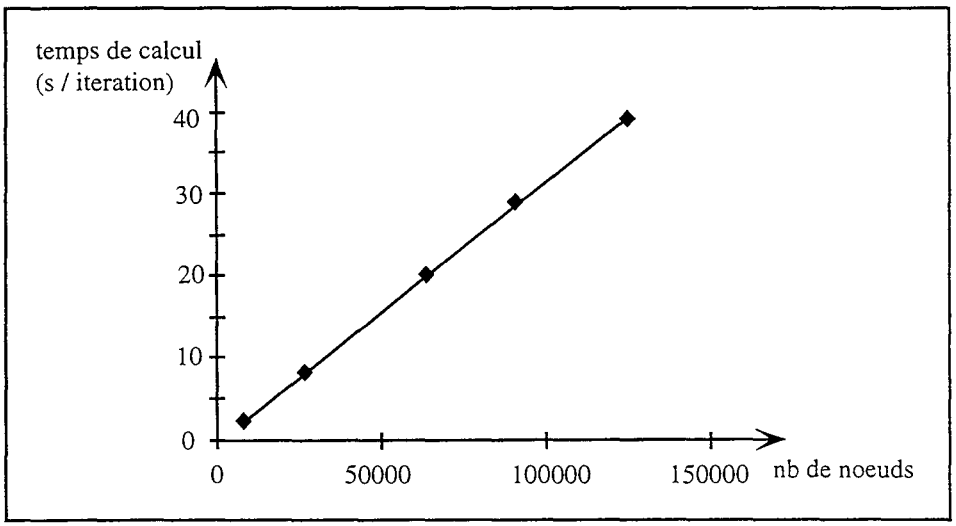
nouvel algorithme.

Tableau .7 : Evaluation des performances de l'algorithme de D.S.I anisotrope

test n°	1	2	3	4	5
taille de la grille	20*20*20	30*30*30	40*40*40	45*45*45	50*50*50
nombre de nœuds	8000	27000	64000	91125	125000
tps de calcul (s) D.S.I anisotrope	68	242	600	863	1170
tps /itération (s) D.S.I anisotrope	2.25	8.1	20	28.8	39
tps de calcul (s) D.S.I classique	5	17	40	57	78
rapport $\frac{t_{anisotrope}}{t_{classique}}$	13.6	14.23	15	15.14	15

Les résultats présentés ci-dessus ont été obtenus avec la machine «extreme» (voir page 149).

L'évolution du temps de calcul en fonction de la taille de la grille est la suivante:



Cette figure montre clairement que le temps de calcul évolue linéairement en fonction de la taille de la grille. Ceci tient à la nature de l'algorithme D.S.I, qui effectue un parcours systématique de tous les nœuds de la grille, aux niveaux desquels il opère une modification locale. L'étendue de cette modification est limité à deux auréoles autour des nœuds, quelle que soit la taille globale de la grille en question, d'où la linéarité de l'algorithme.

Les analyses comparées des temps de calcul entre l'algorithme de D.S.I classique et l'algorithme de D.S.I anisotrope montrent que ce dernier est de 14 à 15 fois plus lent que le précédent. Ceci s'explique de deux façons:

- l'introduction des pondérateurs des satellites a engendré un accroissement du nombre des

opérations de calcul au cours de la mise à jour itérative du paramètre estimé en chaque nœud.

- La modification du voisinage des cellules de la grille dans le sens d'un accroissement du nombre des satellites a elle aussi engendré une augmentation du nombre des opérations de calcul.

Compte tenu de la stratégie de minimisation du temps de calcul que nous avons adoptée, la quantité de mémoire nécessaire est 26 fois plus importante que précédemment dans le cas où le champ de directions est uniforme (une même direction pour l'ensemble de la grille), et 52 fois plus importante dans le cas d'un champ de directions non uniforme. Dans ce dernier cas, il faut en effet stocker pour chaque nœud le tableau des pondérateurs de ces satellites.

Ces chiffres ne sont importants que parce que nous avons choisi de minimiser le temps de calcul. D'autres stratégies conduisent à un coût en mémoire plus faible, mais c'est le temps de calcul qui est alors pénalisé.Dans l'optique coût en mémoire minimal, l'algorithme utilise le même espace mémoire que celui de D.S.I classique pour un champ de directions uniforme, et 4 fois plus d'espace pour un champ de directions non uniforme (pour le stockage de ce dernier).

En conclusion, il faut bien admettre que le nouvel algorithme de D.S.I anisotrope est beaucoup plus coûteux, aussi bien en terme de temps de calcul que de coût en mémoire, que l'algorithme de D.S.I classique. Cependant, les résultats qu'il produit et qui vont être présentés dans le chapitre suivant, nous ont incités à poursuivre dans cette voie.

4.4 Présentation des résultats

Nous allons dans ce chapitre étudier le comportement de l'algorithme. Pour cela, nous allons envisager l'influence des différents paramètres qui interviennent dans le processus de calcul.

4.4.1 Influence de la direction

Comme cela a été dit dans le chapitre 4.3.3 de la page 111, la méthode de calcul des pondérateurs et le support de calcul ont été choisis pour que toutes les directions de contraintes puissent être prises en compte. Nous allons illustrer cette idée par l'exemple de la Figure 46 où nous avons représenté le résultat de plusieurs calculs effectués à partir du même jeu de données en faisant varier la direction de contrainte d'interpolation.

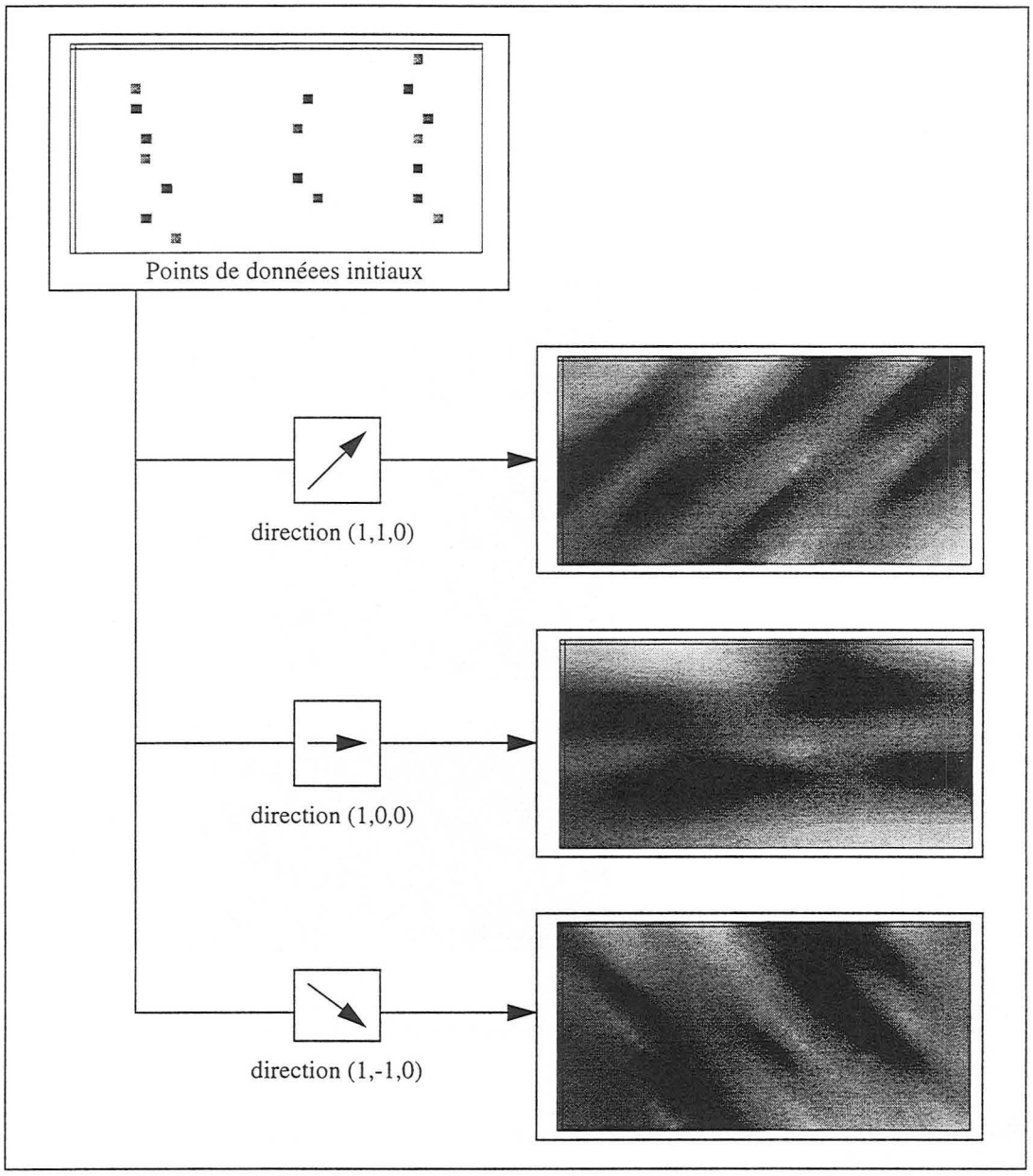


FIGURE 46. Influence de la direction d'anisotropie.

Comme nous pouvons le voir, il est possible de tenir compte de l'ensemble de la gamme des directions possibles. Les points de données initiaux conditionnent le fait que la répartition du paramètre interpolé suive plus ou moins bien telle ou telle direction de contrainte. Dans la Figure 46, il est clair que la direction d'anisotropie (1,1,0) est celle qui est la mieux exprimée parmi les exemples qui sont proposés.

4.4.2 Influence du degré d'anisotropie

Une autre manière de paramétrer ce type d'interpolation consiste à faire varier le degré d'anisotropie, noté D_a. Rappelons que D_a représente dans le calcul des pondérateurs des satellites la valeur maximale de contraste possible entre ces pondérateurs. Plus cette valeur est

grande, plus l'anisotropie est marquée.

Pour illuster ceci, nous allons reprendre l'exemple de la Figure 46 dans le cas de la direction (1,1,0) et nous allons faire varier la valeur du degré d'anisotropie. La Figure 47 présente les résultats obtenus:

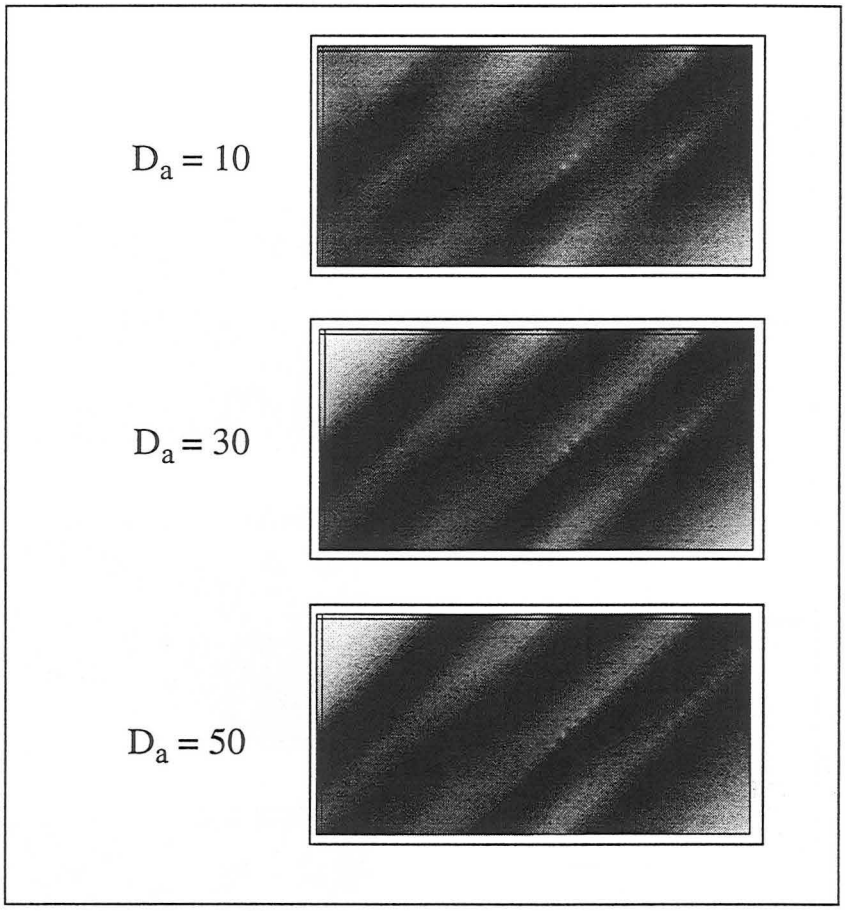


FIGURE 47. Influence du degré d'anisotropie.

Comme nous pouvons le constater, l'influence de la direction d'anisotropie se fait d'autant plus sentir que le degré d'anisotropie est élevé, ce qui correspond au résultat que nous attendions.

4.4.3 Prise en compte de champs de directions non uniformes

Jusqu'à présent, nous n'avons envisagé que des cas où l'interpolation est contrainte par un champ de direction uniforme, c'est à dire que la direction de contrainte est la même en chaque nœud de la grille.

Cependant, la méthode D.S.I permet en principe la prise en compte de champs de directions

non uniformes. La Figure 48 montre le résultat d'un tel calcul:

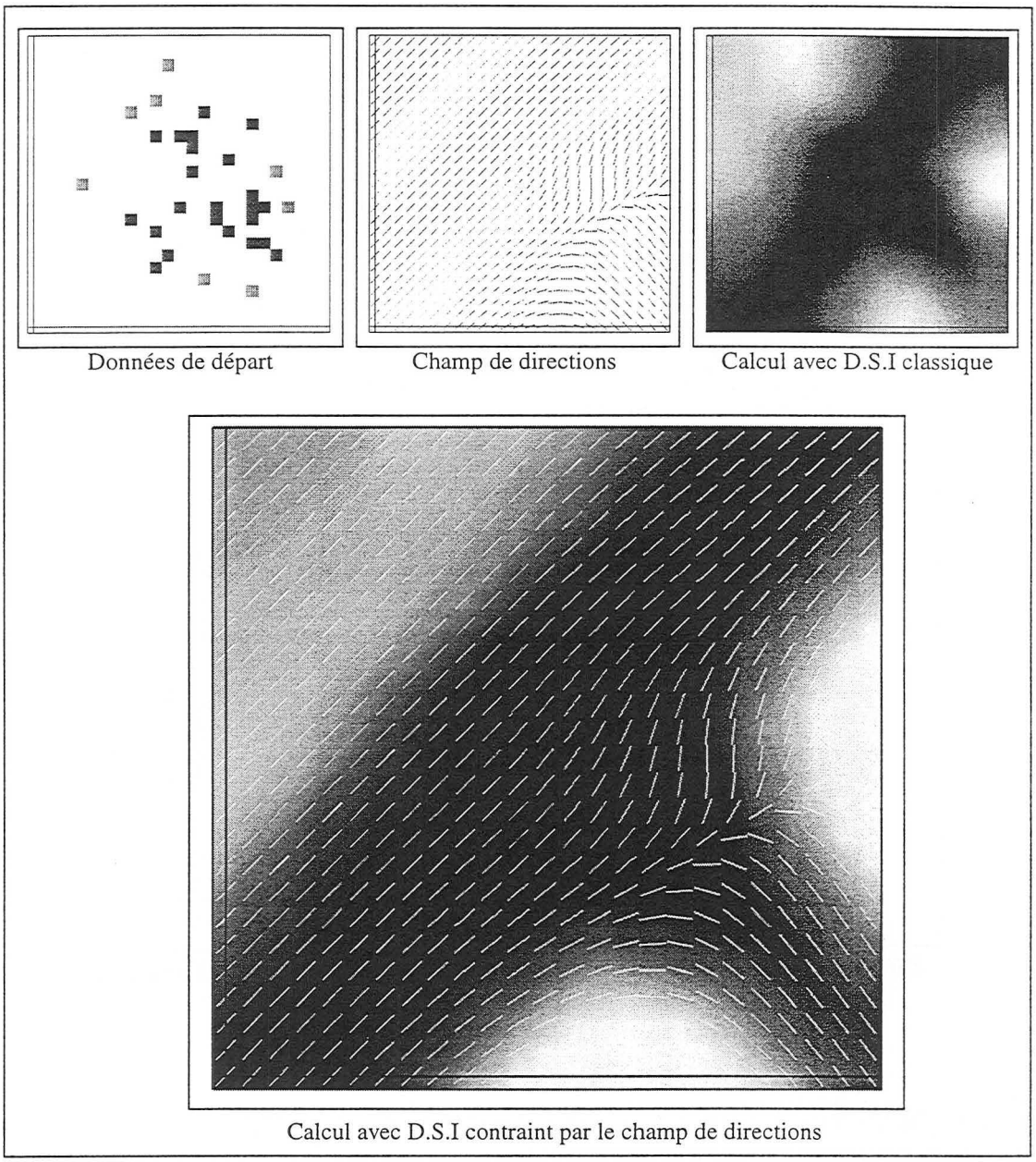


FIGURE 48. Prise en compte de champs de directions non uniformes.

Par ailleurs, cet exemple illustre mieux que tout autre le fait que toutes les directions peuvent être prises en compte dans ce type d'interpolation.

Il montre également la fidélité avec laquelle l'interpolation est contrainte par le champ de directions. On peut en effet remarquer qu'en de très nombreux points de la grille, les direction de contraintes sont très souvent tangentes aux courbes d'isovaleurs du paramètre interpolé.

4.4.4 Prise en compte de discontinuités

De plus, la possibilité de tenir compte de discontinuités dans le processus d'interpolation est conservée. Cette idée est illustrée dans la Figure 49, où nous avons repris le même calcul que

précédemment, en introduisant deux discontinuités dans la grille, en «coupant» cette dernière par deux surfaces triangulées (voir page 66 pour de plus amples explications).

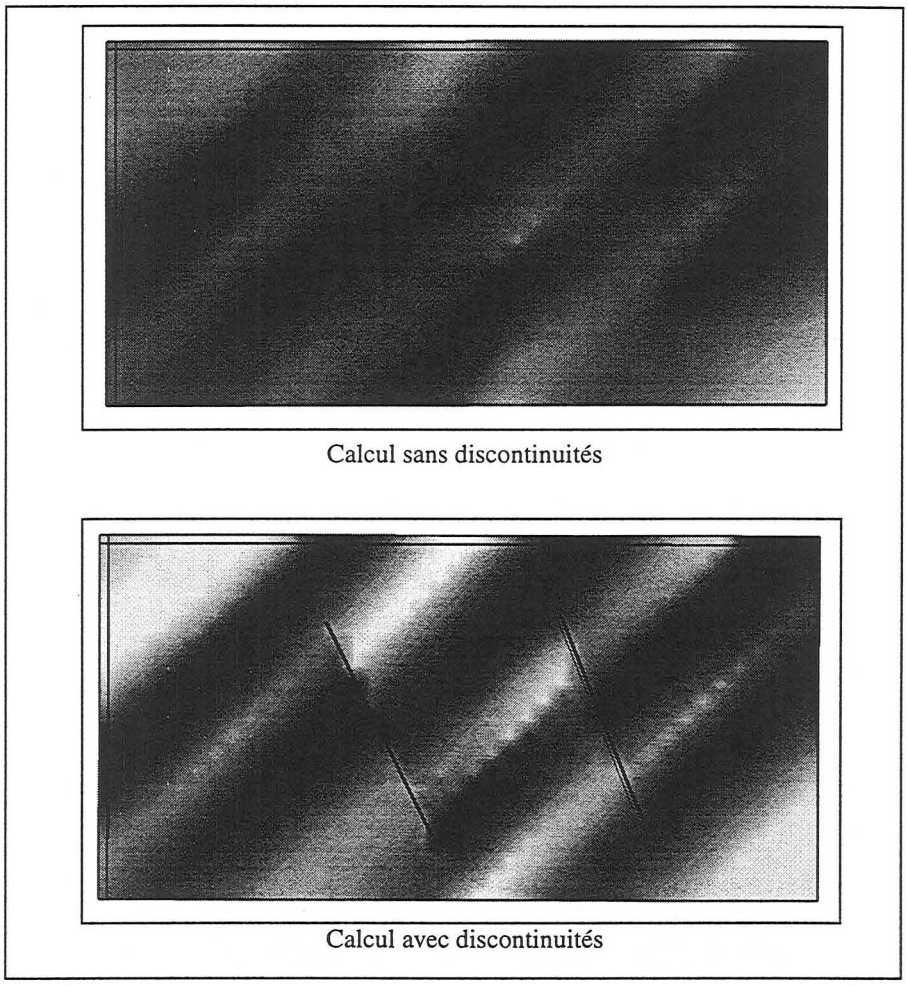


FIGURE 49. Prise en compte de discontinuités dans la méthode D.S.I anisotrope.

4.5 Le problème de la génération des champs de directions

Nous avons vu que l'algorithme de D.S.I anisotrope permettait d'interpoler un paramètre physique sur une grille régulière sous la contrainte d'un champ de directions. Nous allons envisager maintenant les différentes façons de créer ces champs de directions.

4.5.1 Prise en compte de fonctions analytiques

La première manière de procéder consiste à construire le champs de directions à partir d'une loi définie par l'utilisateur. Dans le contexte où nous souhaitons utiliser l'algorithme, c'est à dire dans la modélisation de formes chenalisées, deux types de loi peuvent être envisagés:

- la loi de direction uniforme: à chaque nœud de la grille est affectée une direction commune. La répartition du paramètre physique est contrainte de la même manière en tout point de la grille.

- la loi de direction sinusoïdale: cette loi suppose la donnée d'une direction d'alongement principal, de l'amplitude et de la longueur d'onde des sinusoïdes pour déterminer en chaque nœud la direction qui lui correspond. Cette loi paraît intéressante pour modéliser la répartition de paramètres quand celle-ci est influencée par une dynamique d'ensemble exprimable sous la forme d'une sinusoïde.

4.5.2 Construction par interpolation de données

Lorsqu'il n'est pas possible d'approximer le champ de directions par une fonction analytique, nous avons envisagé la possibilité d'interpoler ces directions sur des grilles régulières.

Ces interpolations de directions sont en fait des interpolations de vecteurs, où les trois composantes de ces vecteurs sont estimées indépendamment les unes des autres par l'algorithme de D.S.I classique présenté au début de ce chapitre.

Ceci n'est d'ailleurs pas sans poser de problème, notamment au niveau de l'orientation des vecteurs. En effet, une direction peut être décrite par deux vecteurs opposés. Le résultat de l'interpolation de ces vecteurs varie fortement selon que chacune de ses directions de contrôle est représentée par un vecteur ou par son opposé. La Figure 50 illustre cette idée:

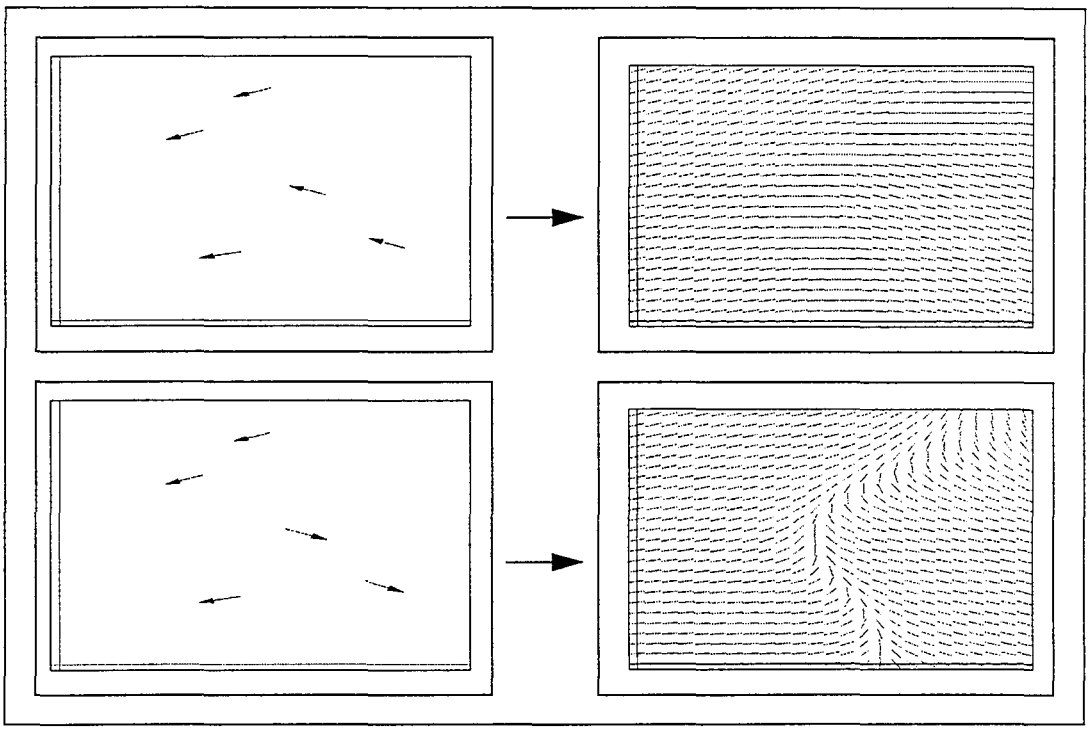


FIGURE 50. Influence de l'orientation des vecteurs de contrôle sur l'interpolation d'un champ de directions.

Nous avons entrepris des recherches pour nous affranchir du problème de l'orientation des vecteurs de contrôle de l'interpolation. Nous avons notamment envisagé de faire porter les interpolations non plus sur les trois composantes des vecteurs de contrôle mais sur des objets mathématiques (formes complexes, formes matricielles, etc...) dérivées de ces vecteurs. Ces recherches n'ont malheureusement donné aucun résultat significatif jusqu'à présent.

Le résultat des interpolations de directions est donc toujours conditionné par l'orientation des vecteurs de contrôle.

4.5.3 Construction à l'aide de l'objet *gshape*

L'objet gshape est brièvement décrit dans le chapitre 3.7.1 de la page 83. Les détails de la méthode de construction de champs de directions par l'objet gshape sont fournis dans la suite de ce chapitre, et notamment en page 89. La Figure 51 montre un exemple de génération de champs de direction par cette méthode:

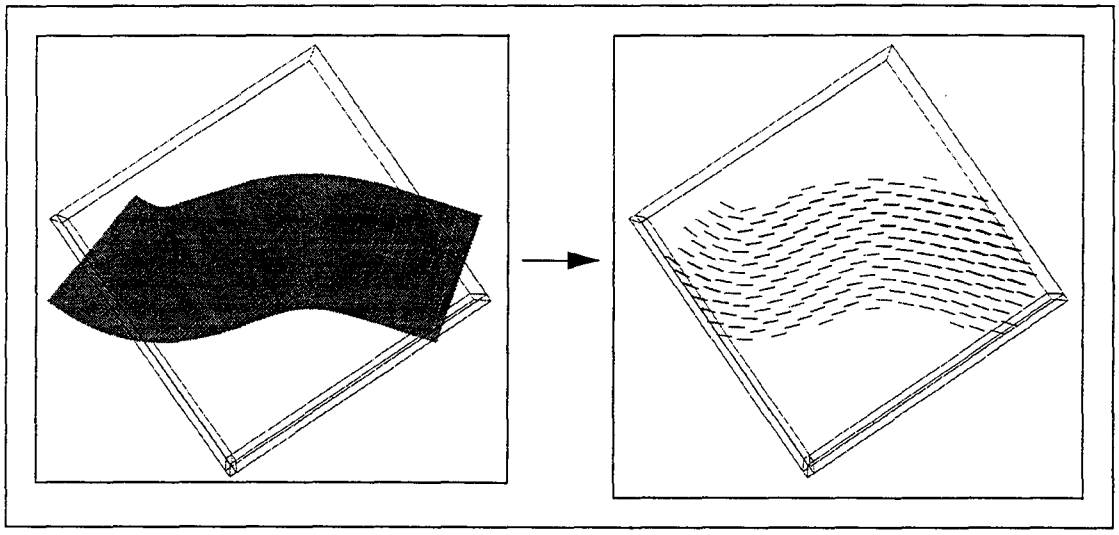


FIGURE 51. Génération de champs de directions avec l'objet *gshape*.

Cet algorithme offre plusieurs possiblités, telles que:

- construire des modèles de directions à partir de lignes polygonales.

- Interpolation d'un paramètre physique dans le volume délimité par un *gshape*, sous la contrainte d'un champ de direction parallèle à l'axe de cet objet.

4.6 Propriétés générales de l'algorithme D.S.I anisotrope

Nous venons d'envisager tous les problèmes technique qui se sont posés au cours de la mise au point de l'algorithme d'interpolation de D.S.I anisotrope. Il convient maintenant d'envisager les différents usages que l'on peut attendre de cette méthode de calcul.

4.6.1 Idée de base

L'algorithme D.S.I anisotrope est et reste un interpolateur. Si l'on dispose d'un maillage Ω (ici une grille régulière) et si l'on connait les valeurs d'un paramètre physique φ en certains nœuds de ce maillage, celui-ci fournit une estimation de φ sur l'ensemble des nœuds du maillage. Les valeurs de φ aux nœuds où celles-ci sont connues restent inchangées (l'interpolateur respecte les points de données). Les valeurs de φ aux nœuds où celles-ci sont inconnues

sont calculées en fonction des valeurs connues et de la nature de l'interpolateur.

Dans le cas qui nous concerne, l'objectif de l'interpolateur est, comme dans toute méthode dérivée de D.S.I, de minimiser un critère de rugosité global du maillage, dont l'expression locale à chaque nœud est pondérée par la donnée d'une direction de contrainte. Il en résulte que l'estimation de φ fournie par l'interpolateur est contrainte par un champ de directions.

En pratique, cette contrainte se matérialise de la manière suivante: en chaque nœud où la valeur de φ est connue, la variabilité locale de φ est d'autant plus faible que la direction de calcul du variogramme est proche de la direction de contrainte stockée au niveau de ce nœud. En clair, l'influence de ces valeurs des nœuds de données est plus grande le long de la direction de contrainte qui leur est associée, qu'à la perpendiculaire de cette direction.

Cette influence se fait sentir d'autant plus loin dans le maillage que le degré d'anisotropie déterminé par l'utilisateur est grand.

Par ailleurs, D.S.I autorise qu'à chaque nœud du maillage soit associée une direction de contrainte spécifique. Il est donc possible de contraindre les interpolations par un champ de directions non uniforme.

Le rappel des principales caractéristiques de l'algorithme étant fait, nous pouvons dire que le champ d'application de cette méthode est l'estimation de paramètres dont la distribution suit préférentiellement un direction donnée ou un champ de direction donné, pour telle ou telle raison liée à la nature même de ce paramètre.

La donnée du champ de contrainte de l'interpolateur est donc une information supplémentaire par rapport au points et aux valeurs de donnée dont on dispose. Ce type d'informations peut provenir de différentes sources:

- de mesures géométriques effectuées sur le terrain ou par l'étude de diagraphies.
- d'études portant sur l'ensemble des données d'un cas d'étude particulier et permettant de définir des propriétés géométrique globales pour les objets que l'on souhaite modéliser.

4.6.2 Modélisation de formes chenalisées

Dans le second chapitre de ce rapport, nous avons vu qu'une étude sédimentologique portant sur l'ensemble des données du site de stockage de Gaz-de-France avait permis de définir les caractéristiques géométriques globales des corps chenalisés de chaque unité sédimentologique de la séquence.

La méthode d'interpolation que nous venons de présenter a été conçue dans le but de reconstruire la géométrie de ces corps chenalisés en tenant compte de ces propriété géométriques.

Pour cela, nous avons imaginé la stratégie suivante:

1. On définit le support de calcul, c'est à dire de l'ensemble des cellules de la grille régulière représentant la géométrie d'une unité sédimentologique donnée.
2. Au niveau des cellules interceptant le trajets des puits, on affecte une valeur de «probabilité d'intercepter un corps chenalisé». Cette valeur est égale à 1 lorsque les faciès aux puits sont des faciès de corps chenalisés, et 0 lorsqu'il s'agit de faciès de plaine d'innondation.

3. L'utilisateur construit un champ de directions à l'aide des outils décrits dans le chapitre 4.5 de la page 122, en tenant compte des informations géométriques dont il dispose.
4. On fait tourner l'interpolateur, la variable est estimée sur l'ensemble du support de calcul. Les corps chenalisés sont alors représentés par l'ensemble des cellules dont la valeur de «probabilité d'intercepter un corps chenalisé» est supérieure à 0.5. A l'inverse, les dépôts de plaine d'innondation sont représentés par les cellules dont la valeur est inférieure à 0.5.

De la sorte, on obtient un modèle de réservoir qui honore les valeurs aux puits et dont la géométrie des objets est contrainte par les données de directions exprimées au moyen du modèle de directions.

Ce type de calcul permet d'exprimer l'idée que, si la direction générale des corps chenalisés est connue, et si l'on intercepte un de ces objets au niveau d'un puits, la probabilité de rencontrer le même objet est plus forte le long de la direction générale de cet objet qu'à la perpendiculaire de cette direction.

Un exemple synthétique 2D illustrant cette démarche est présenté dans la Figure 48 de la page 121. Une application complète de cette méthode sera présentés dans le chapitre suivant consacré à la construction du modèle du réservoir de Gaz-de-France.

4.6.3 Modélisation de formes minéralisées

Une autre application de cet algorithme que nous avons envisagée concerne le modélisation de formes minéralisées. En effet, dans le cas où la formation des minéralisations est le produit de circulations de fluides, la géométrie des corps résultants présente des orientations géométriques préférentielles très marquées, que les géostatisticiens identifient au travers des études de variogrammes.

Dans ces conditions, il semble intéressant d'utiliser l'algorithme de D.S.I anisotrope pour modéliser la répartition des concentrations en minerais. La possibilité de tenir compte dans le même calcul des données de puits, des directions principales des objets et d'éventuelles discontinuités liées à la fracturation du gisement semble en effet très intéressante pour ce type de problème.

4.7 Conclusion

Tout au long de ce chapitre, nous avons présenté un nouvel algorithme d'interpolation de données. Cet algorithme est une extension de la méthode D.S.I utilisée jusqu'à présent dans le projet Gocad. L'extension de la méthode a consisté à considérer la possibilité qu'offre D.S.I d'introduire des pondérations entre les nœuds du graphe sur lequel elle travaille.

Ceci nous a permis de contraindre l'estimation d'un paramètre physique φ sur une grille régulière par un champ de directions, c'est dire par la donnée en tout nœud de la grille d'un vecteur défini par ses trois composantes.

La mise au point de cet algorithme a introduit un surcoût tout fait conséquent au niveau du temps de calcul et de l'espace mémoire utilisé par rapport à l'algorithme classique de D.S.I sur les grilles régulières. Cependant, les résultats obtenus sont à la hauteur de nos espérances.

En effet, cet algorithme permet:

- d'effectuer des estimations de paramètres physiques en respectant les points de données initiaux.
- de contraindre ces estimations par un champ de directions qui traduit une anisotropie de la distribution du paramètre interpolé.
- de tenir compte dans l'interpolation de discontinuités liées à la présence de failles dans le modèle.

Cette polyvalence constitue sans aucun doute l'argument majeur qui nous a incité à persévérer dans cette voie.

Le champ d'application de cet algorithme est l'interpolation sur des grilles régulières de paramètres physiques ([32]), en imposant des contraintes géométriques globales sur la distribution de ce paramètre.

Chapitre 5

Proposition d'une méthodologie pour la construction d'un modèle du site de stockage en aquifère

5.1 Introduction

Au cours des trois chapitres précédents, nous avons présenté tout un ensemble de nouveaux outils de modélisation qui ont chacun un intérêt propre. Ainsi:

- l'objet *gstack* a permis d'introduire la notion de couche ou d'unité sédimentaire, en tenant compte à la fois du caractère surfacique de cette dernière (une couche est constituée d'un toit et d'un mur) et de son caractère volumique (le mur et le toit délimitent un volume qu'il convient de matérialiser géométriquement, soit par un ensemble de tétraèdres, soit sous la forme d'un ensemble de voxels).

- l'objet *voxet* est un outil très utile et très efficace qui permet, dans le cadre d'un projet de modélisation géologique complexe où plusieurs types de maillages différents sont utilisés à différentes étapes de ce projet, d'intégrer l'ensemble des résultats obtenus sur un support de travail commun.

- La méthode D.S.I anisotrope, bien que relativement coûteuse à l'usage, a permis de montrer comment il était possible de contraindre un calcul d'interpolation par un champ de directions non uniforme et a par la même montré le caractère général de la méthode D.S.I.

Dans ce chapitre, nous allons montrer de quelle manière il est possible de combiner ces outils ainsi que les fonctionnalités de Gocad préexistantes, afin de modéliser la structure du réservoir de stockage de gaz. Cette opération s'est effectuée en deux étapes:

- la construction d'un modèle surfacique qui comprend l'ensemble des surfaces qui délimitent les différentes unités de la séquence de ce réservoir, ainsi que ses failles.

- la construction d'un modèle volumique dans lequel les unités sont décrites sous la forme d'un ensemble de cellules de grille régulière, dans lequel nous avons cherché à reconstituer

la géométrie des corps sableux.

Pour chacune des ces étapes, nous nous attacherons à décrire les outils qui ont été employés et à justifier le choix de ces outils par rapport au problème posé.

5.2 Construction du modèle surfacique

5.2.1 Construction de la surface interprétée

La première étape de ce travail consiste à modéliser le seul horizon appartenant à la séquence du réservoir, qui a pu être interprété à partir des données sismiques 2D et 3D.

Le point de départ de cette modélisation est donc un ensemble de points appartenant à l'horizon issu de cette interprétation. La surface modélisée est construite au moyen d'une surface triangulée car ce type de maillage est le plus flexible et le plus efficace pour modéliser les surfaces géologiques.

La digitalisation de la carte nous a également fourni la trace des failles qui découpent cet horizon.

Le résultat de cette modélisation est représenté dans la planche 11.

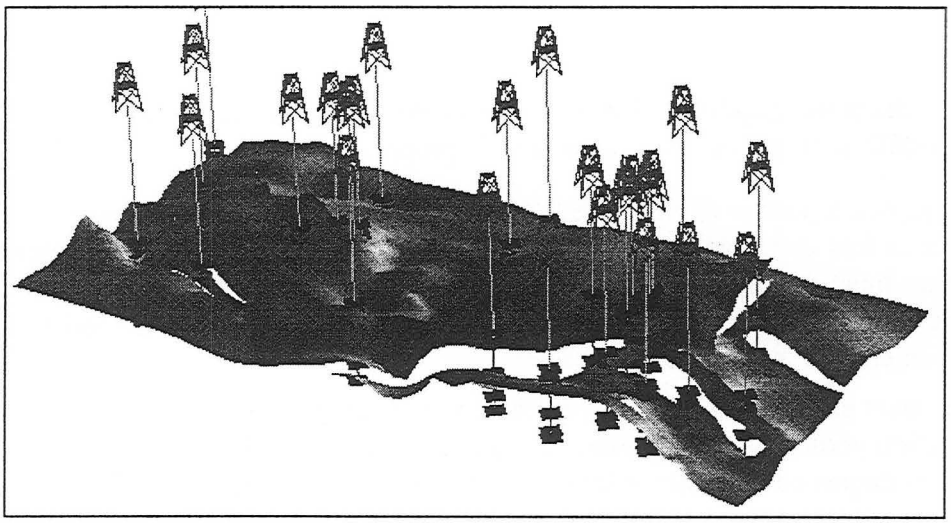


PLANCHE 11. Modélisation de la surface issue de la sismique.

Pour parvenir à ce premier résultat, nous avons effectué les opérations suivantes.

Création d'un maillage triangulé initial régulier de forme rectangulaire couvrant l'ensemble de la zone d'étude

La seule difficulté de cette opération consiste à adapter la taille des triangles à la densité des données dont on dispose, de façon à ne pas obtenir un maillage inutilement coûteux, ou bien un maillage trop simple ne permettant pas de rendre compte de la complexité de la surface modélisée

Découpage de ce maillage initial par l'ensemble des failles

Cette opération, désignée sous le terme de «cut» dans le vocabulaire de Gocad, permet d'effectuer un découpage du maillage initial de façon à introduire des discontinuités dans ce dernier. D'un point de vue pratique, seules les failles dont le rejet maximal est supérieure à 5 mètres ont été sélectionnées pour ne pas compliquer inutilement le modèle.

Utilisation de l'interpolateur D.S.I

La troisième et dernière étape consiste à utiliser l'interpolateur D.S.I pour adapter la géométrie du maillage aux points de données issus de la sismique. Cette opération nécessite l'utilisation de deux contraintes de l'interpolateur D.S.I:

- La contrainte FCP («fuzzy control points») qui permet d'attirer le maillage par les points de données.

- la contrainte OSL («on straight line») qui s'applique ici aux bords du maillage et impose que ceux-ci ne se déplacent que sans la direction verticale. Ceci revient à dire que nous faisons l'approximation que les rejets des failles qui découpent la suface sont verticaux. Cette approximation est acceptable compte tenu des faibles variations en z de la surface (de l'ordre de la dizaine de mètres) par rapport à ses dimensions latérales (de l'ordre du kilomètre).

Cette stratégie de modélisation produit la surface représentée sur la planche 11, à partir de laquelle il est déjà possible d'effectuer des calculs d'aire, d'analyser les rejets et la disposition des failles et de visualiser la structure du réservoir.

5.2.2 Reconstitution de l'ensemble des surfaces de la séquence

La seconde étape de la modélisation surfacique du réservoir de stockage consiste à déduire la géométrie de l'ensemble des surfaces de la séquence de ce réservoir, à partir de la surface qui vient d'être construite.

La séquence du réservoir, représentée schématiquement sur la Figure 52 de la page 132, se présente de la manière suivante:

- Cinq niveaux d'érosions numérotés E_0 à E_4, correspondant à des épisodes d'interruption de sédimentation sont reconnaissables sur l'ensemble des puits de la zone d'étude. Le niveau S3 correspond à la surface interprétée.

- Une étude sédimentologique a permis de décomposer les intervalles compris entre deux niveaux d'érosion en un ensemble d'unités chenalisées indépendantes. Ainsi, par exemple, entre les niveaux E_2 et E_3, le sédimentologue a pu identifier 7 unités indépendantes numérotées U_7 à U_{13}.

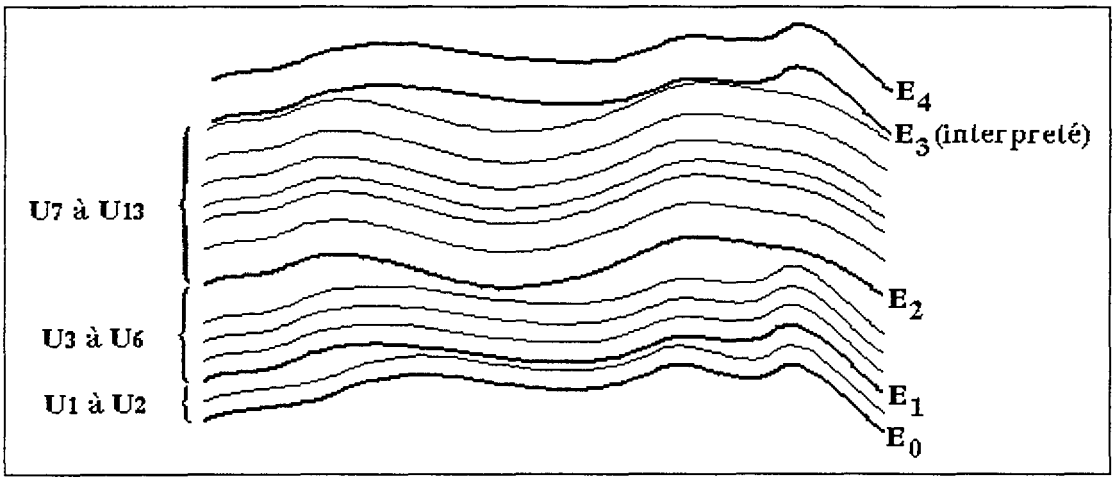


FIGURE 52. Représentation schématique de la séquence du réservoir.

Dans la suite de la modélisation, nous prendrons comme hypothèse que toutes les surfaces de la séquence peuvent être modélisées avec une triangulation commune, chaque surface différant de l'autre seulement par les coordonnées des nœuds de cette triangulation. Cette hypothèse n'est pas contraignante compte tenu des dimensions de la séquence (très grandes en x et y, très limitées en z). Il n'est donc pas déraisonnable de considérer que les failles découpent uniformément l'ensemble des unités de la séquence.

La construction de l'ensemble des surfaces de la séquence se fait surface par surface et fait appel à une procédure de construction commune. Cette procédure permet de générer une surface E_f à partir d'une surface initiale E_i, et se décompose en deux étapes:

- L'estimation de l'épaisseur entre E_i et E_f à partir des mesures effectuées aux puits. Cette estimation se fait par l'interpolateur D.S.I sur une grille régulière 2D couvrant l'ensemble de la zone d'étude, et non sur le maillage triangulé de E_i. En effet, ce dernier est discontinu et la propriété d'épaisseur que nous interpolons est à priori continue, car liée au dépôt qui est à priori antérieur au faillage.
- Une fois l'épaisseur estimée, E_f est obtenue à partir de E_i par translation verticale de chaque nœud du maillage de E_i, de l'épaisseur estimée à la position de ce nœud. Le fait que cette translation soit verticale a pour conséquence que les failles de la séquence sont verticales. Cette approximation n'est cependant pas génante, toujours compte tenu du fait que la séquence est limitée en épaisseur (environ 30 mètres) est que, à ce titre, les éventuels rejets horizontaux sont de toute manière négligeables.

La Figure 53 décrit l'ordre dans lequel les surfaces sont construites. Le point de départ est la

surface S3, construite à partir de la sismique.

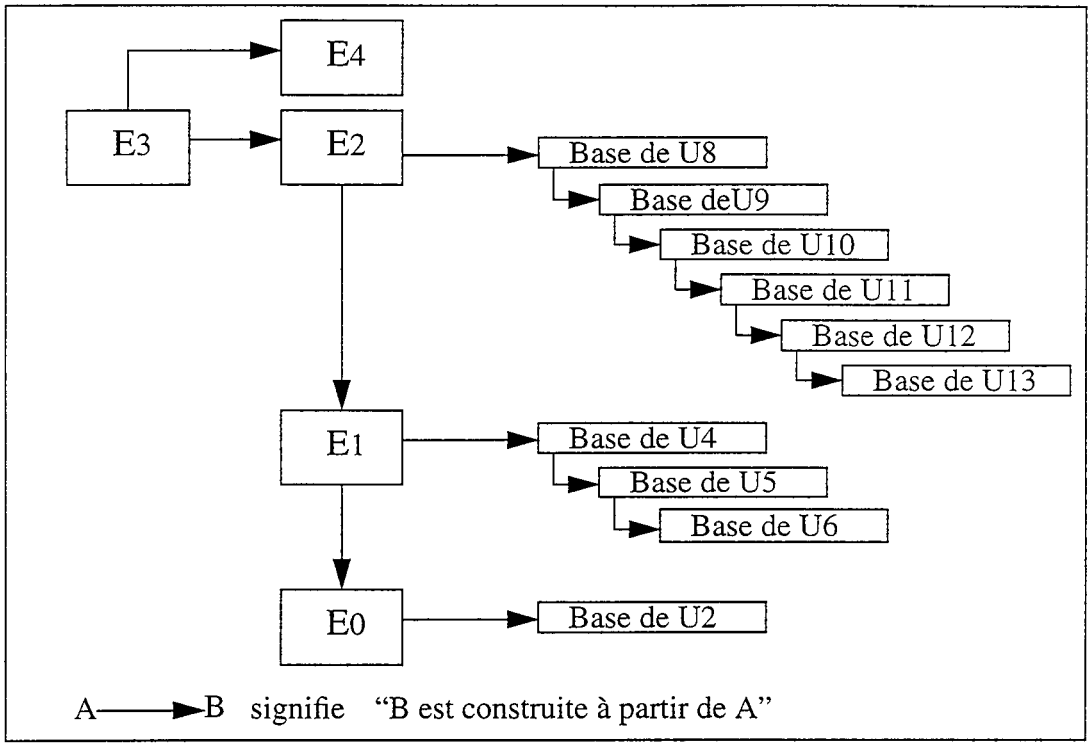


FIGURE 53. Ordre de construction des surfaces de la séquence du réservoir.

Ce schéma amène les commentaires suivants:

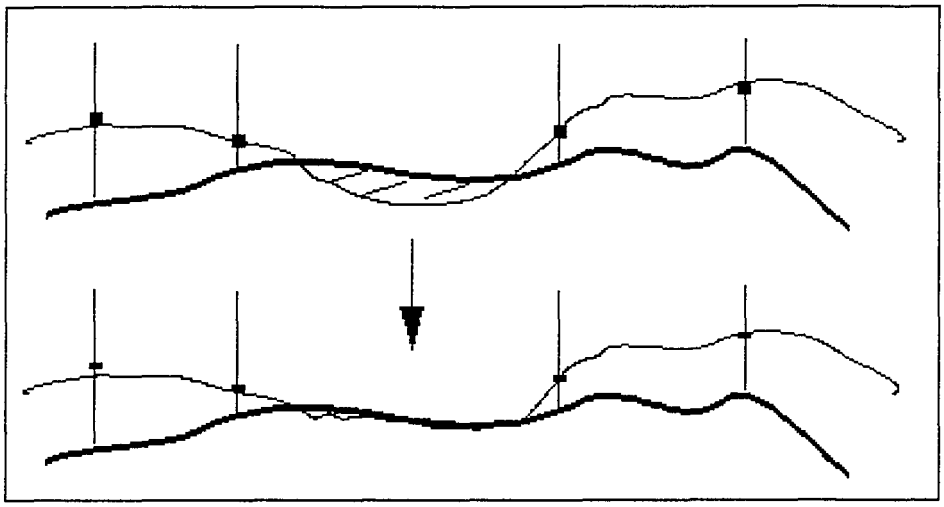
La première étape consiste à reconstituer les quatre niveaux d'érosion à partir de E3. Cette opération ne pose pas de problèmes particuliers, si l'on prend soin de bien construire chaque surface à partir de la surface existante qui lui est la plus proche dans la séquence. Ainsi, E2 et E4 sont construites à partir de E3, puis E1 à partir de E2, et E0 à partir de E1.

En ce qui concerne les horizons correspondant aux surfaces de base des unités sédimentologiques, la même règle s'applique mais il convient de bien démarrer la chaîne de construction à partir de la surface d'érosion par rapport à laquelle les unités en question sont les plus «concordantes» (au sens géologique du terme). Ainsi, le point de départ des surfaces «base de U4», «base de U5» et «Base de U6» est le niveau d'érosion E1 par rapport auquel elles sont le plus concordantes (voir Figure 52).

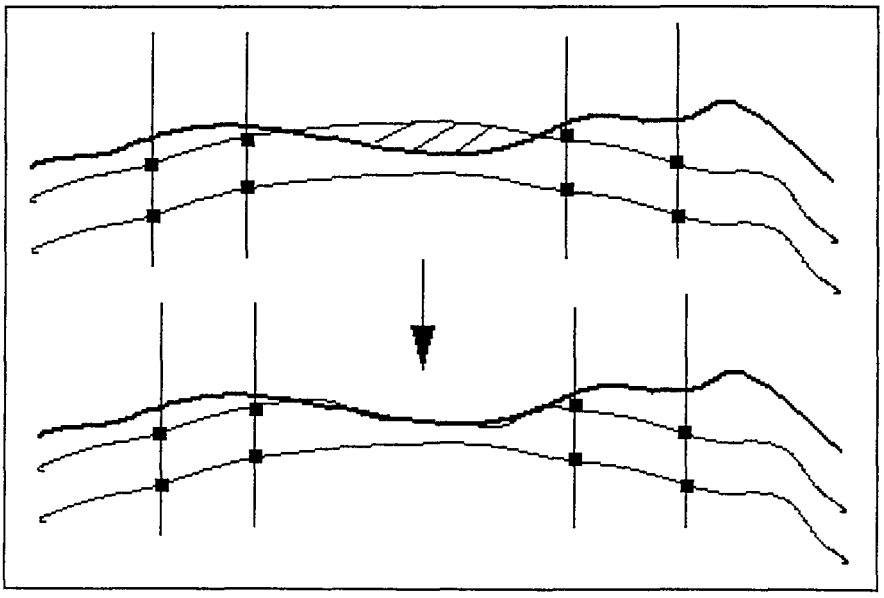
La construction de ces surfaces n'est pas d'ailleurs pas sans poser quelques problèmes, notamment du fait que celles ci ne sont généralement pas identifiables sur l'ensemble des puits. Deux types de problèmes peuvent alors se poser:

1. L'interpolation des épaisseurs produit en certaines zones des valeurs négatives, qu'il con-

vient de remettre à zéro après interpolation.



2. Du fait de l'absence d'information au niveau de certains puits, une surface de type «Base de Ux» peut être amenée à intersecter le niveau d'érosion qui lui est supérieur dans la séquence. Il convient donc de corriger ces parties de surface, de façon à assurer la consistance géologique du modèle.



A cette étape de la modélisation, le réservoir est décrit par un ensemble de surfaces isomorphes que l'on stocke dans un objet *gstack*. Ces surfaces honorent les données de puits, sont faillées, et les relations qu'elles entretiennent entre elles sont respectées (érosion d'une surface par une autre, pas d'intersections entre surfaces, etc...).

5.3 Reconstruction de la géométrie des corps chenalisés

5.3.1 Passage d'un modèle surfacique à un modèle volumique

Le fait que les surfaces du réservoir soient stockées au sein d'un objet *gstack* permet de passer automatiquement d'un modèle à base de surfaces triangulées à un modèle volumique décrit sous la forme d'une grille régulière (en utilisant la technologie décrite dans le chapitre 2.4 de la page 23).

D'un point de vue pratique, chaque unité sédimentologique qui était jusqu'à présent délimitée par une surface de base et une surface sommet est maintenant représentée sous la forme d'un ensemble de cellules d'une grille régulière. La planche 12 de la page 135 montre deux unités de la séquence décrites de cette manière.

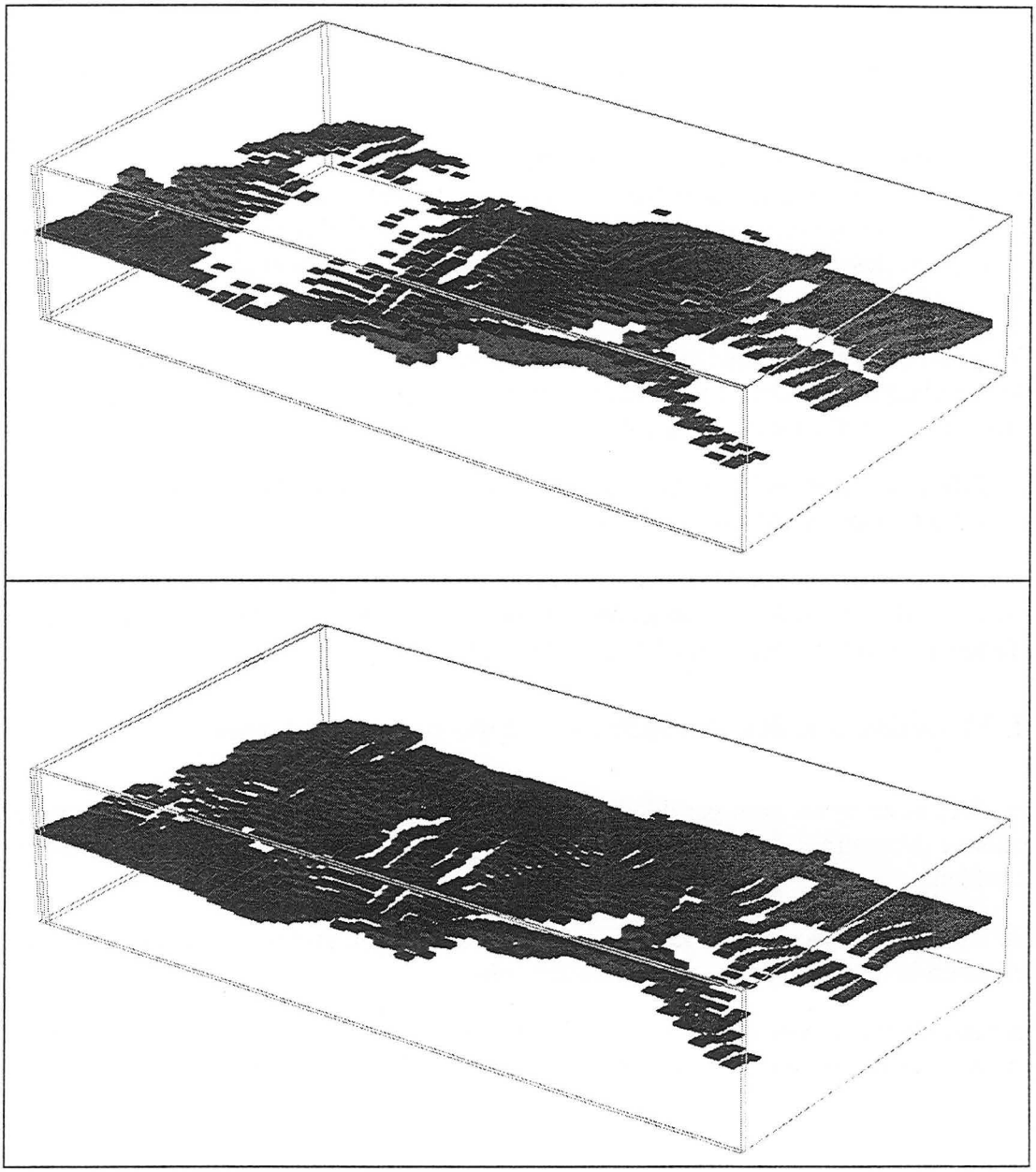


PLANCHE 12. Représentation des unités sédimentologiques sous la forme d'un ensemble de voxels.

Comme on peut le constater sur cette figure, cette description conserve le caractère faillé des unités (voir planche 11 de la page 130). On peut également remarquer sur la planche du haut que certaines parties de l'unité sont manquantes, car l'unité en question est absente de cer-

tains puits (par absence de dépôt, érosion, etc...).

Le terme unité sédimentologique que nous avons employé jusqu'à présent désigne un volume contenant des matériaux partageant une histoire géologique commune. Il s'agit donc ici à la fois de corps sableux et de dépôts argileux de toute nature.

Une fois le passage à un modèle volumique effectué, notre objectif est maintenant de modéliser la géométrie des corps sableux dans ces unités sédimentologiques. Pour cela, nous avons immaginé deux stratégies correspondant aux deux modes de dépôt rencontrés sur notre site de stockage: les réseaux en tresse et les méandres.

Ces deux stratégies font néanmoins appel à une procédure commune. Cette procédure est basée sur la méthode d'interpolation D.S.I anisotrope, que nous avons présentée dans le chapitre 4 de la page 101.

Le paramètre que nous allons interpoler par cette méthode est la «probabilité d'intercepter un corps réservoir». Ce paramètre prend la valeur 1 au niveau des cellules où les puits interceptent des faciès de type réservoir (grès propres, grès plus ou moins argileux) et la valeur 0 là où ils interceptent des faciès non réservoirs (argiles de plaine d'innondation, bouchons argileux, etc...).

D'un point de vue technique, ces interpolations se font après une remise à plat par le bas des cellules de chaque unité, et ce pour tenir compte du fait que la structure des corps sableux est continue de part et d'autre des failles du réservoir (le dépôt est antérieur au faillage).

Les différentes modalités de l'interpolation varient selon que l'on modélise des réseaux en tresse ou des bouchons argileux.

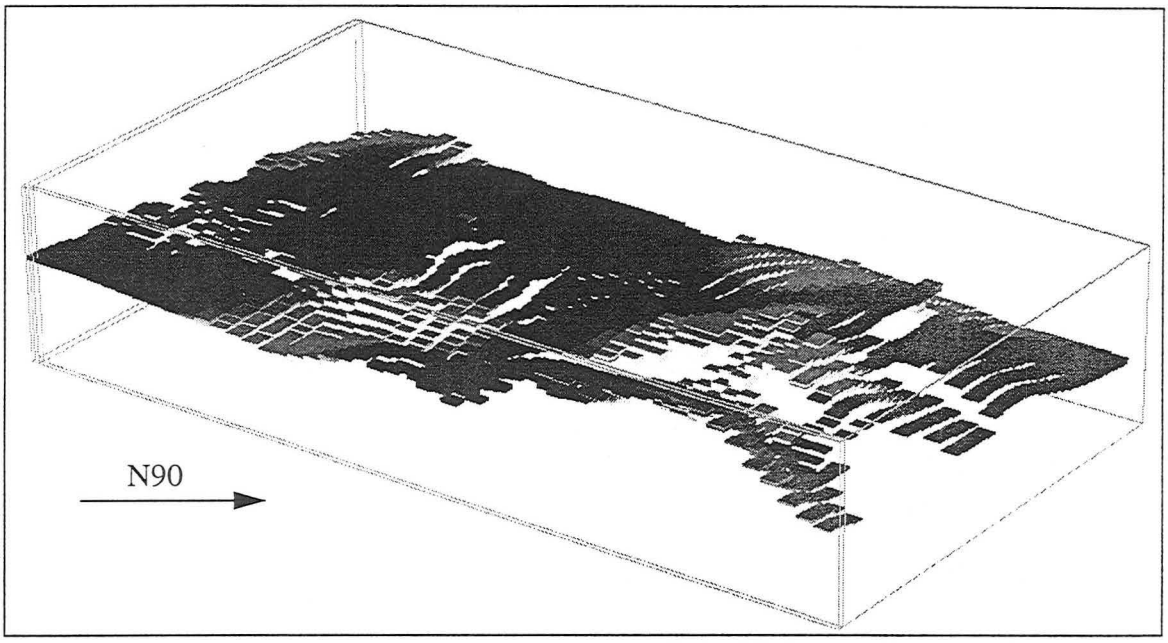
Après interpolation, les cellules de la grille dont la «probabilité d'intercepter un corps réservoir» est supérieure à 0.5 sont rattachées à la famille des faciès gréseux, celles dont la valeur est inférieure à 0.5 sont rattachées à la famille des faciès argileux.

5.3.2 Modélisation des formations de type réseau en tresse

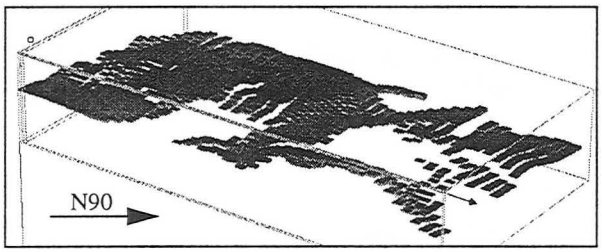
Les réseaux en tresse sont par définition relativement rectilignes. L'étude sédimentologique a permis de déterminer pour chaque unité de ce type la direction d'alongement principal des corps sableux.

L'interpolation anisotrope se fait donc sous la contrainte d'un champ de directions constant orienté selon la direction d'alongement déterminée.

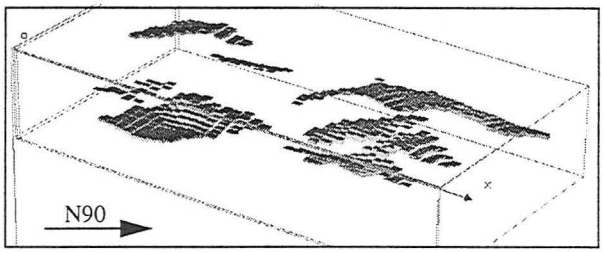
Un exemple de ce type de calcul est présenté dans la planche 13. Les cellules sombres correspondent aux faciès sableux et les cellules claires aux faciès de type argileux.



Résultat de l'interpolation anisotrope.



Géométrie des corps sableux.



Géométrie des intercalations argileuses.

PLANCHE 13. Reconstitution de la géométrie des corps sableux dans les réseaux en tresse.

Comme on peut le constater dans l'exemple présenté ici, la géométrie des corps sableux est préférentiellement orientée suivant la direction de contrainte de l'interpolation, ce qui correspond à ce que nous espérions.

En effet, la méthode employée ici permet d'exprimer une idée très simple, qui est la suivante: si un puits intercepte un faciès réservoir appartenant à un objet dirigé suivant une direction donnée, la probabilité de rencontrer ce même faciès est plus grande si l'on se déplace à

partir de ce puits le long de cette direction que si l'on se déplace à la perpendiculaire de cette direction.

Nous avons également vu que cette façon de faire impose de regrouper les faciès présents en deux familles. Cette approximation ne nous semble pas irréaliste dans le cas des réseaux en tresse compte tenu que les mtériaux observés dans ce type de dépôt sont relativement homogènes.

5.3.3 Modélisation des formations de type méandre

Les objets de type méandre sont par nature plus sinueux que les réseaux en tresse. Par conséquent, l'utilisation d'un champ de directions non uniforme s'impose.

Par ailleurs, les dépôts de type sableux sont ici limités en étendue par ce que l'on appelle une ceinture de méandre. Dans le cas d'étude qui nous concerne, les géologues ont établi des cartes structurales qui fournissent une géométrie probable de ces ceintures de méandre.

Cette géométrie va nous permettre de construire un champ de directions et servira également de discontinuité pour l'interpolateur. Dans la planche 14, nous avons représenté un exemple de ce type de calcul.

Commentaires

Nous pouvons constater que la géométrie des corps sableux est là aussi influencée par le champ de directions, ainsi que par la géométrie de la ceinture de méandre qui est fournie à l'interpolateur.

Cette méthode impose également de définir trois types de dépôts: les corps sableux, les bouchons argileux et les dépôts de plaine d'innondation. Cette approximation est relativement limitative dans le cas des méandres où la diversité des faciès, notamment pour les corps sableux, que dans le cas des réseaux en tresse.

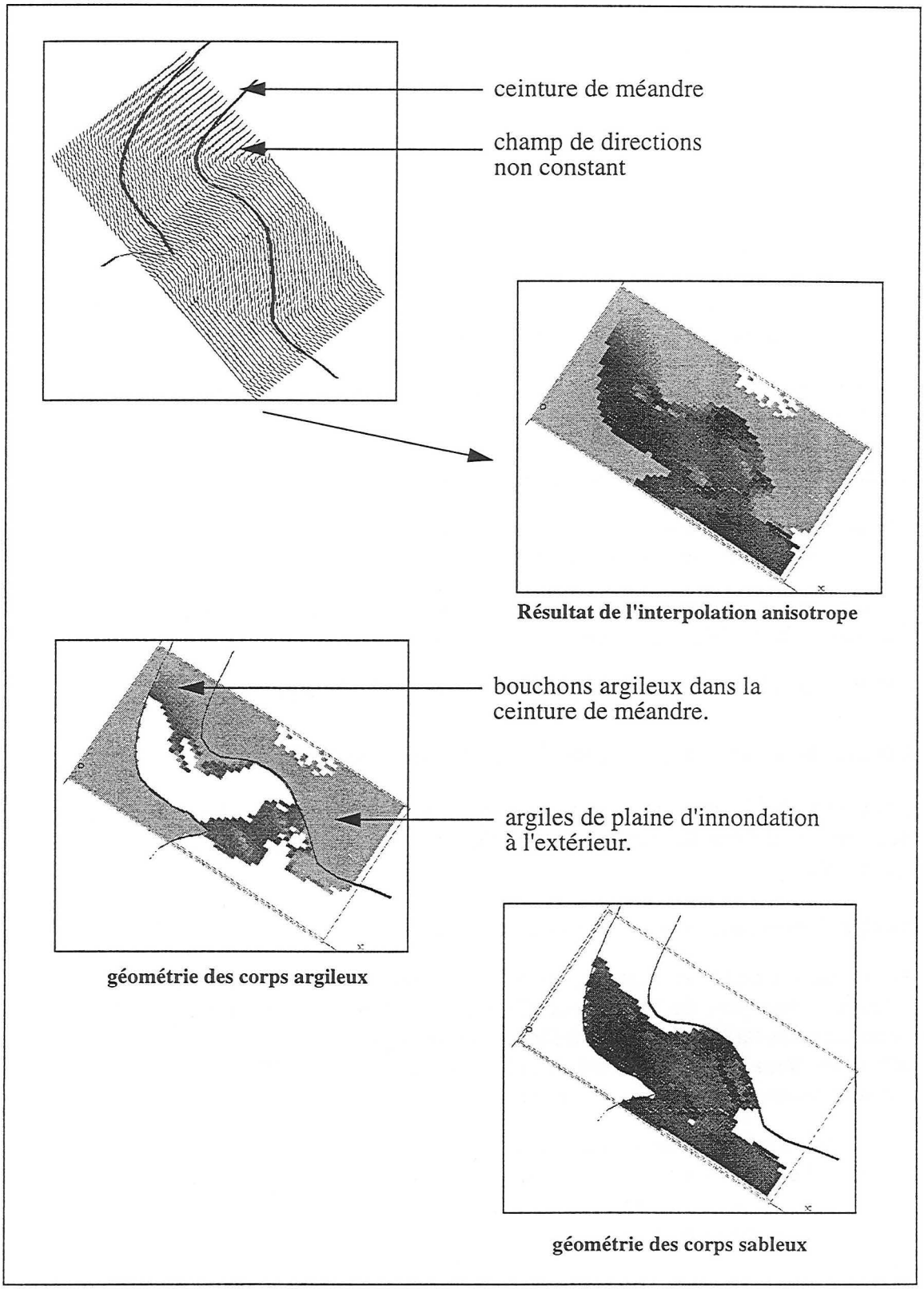


PLANCHE 14. Reconstitution de la géométrie des corps sableux de type méandriforme.

5.4 Conclusion: perspectives futures et possibilités d'enrichir le modèle

5.4.1 Validation du modèle

Une fois le modèle construit, il convient de le valider et ainsi d'apprécier les qualités et les défauts de la méthodologie que nous avons employée. Pour cela, nous avions extrait de la base de données initiale qui a servi à construire le modèle deux puits, que nous désignerons par le terme de «puits tests», et qui n'ont donc pas été utilisés pour construire le modèle.

L'un de ces deux puits est situé au centre de la zone d'étude dans une zone où la densité de puits est importante. Nous le désignerons sous l'appellation «puits test favorable». L'autre puits témoin est situé en bordure de la zone d'étude, et est très éloigné des puits qui lui sont les plus proches. Nous l'appellerons le «puits test défavorable».

Le modèle que nous venons de créer nous permet de générer un «puits test favorable synthétique» et un «puits test défavorable synthétique».

La comparaison de chaque puits avec son équivalent synthétique va nous permettre d'apprécier la qualité du modèle et la gamme d'erreur que celui ci commet par rapport à la réalité.

Les résultats obtenus sont regroupés dans la Figure 54 de la page 141.

Commentaires des résultats des tests de validation

Deux types de résultats sont à examiner séparément: les erreurs sur les cotes auxquelles sont interceptées les différentes surfaces du modèle, et le type de dépôt que l'on trouve au sein de chaque unité.

Analyse des erreurs sur les cotes des surfaces

Au niveau du puits «favorable», les résultats sont relativement honorables car les erreurs n'excèdent jamais plus de 3-4 mètres. Ceci est acceptable compte tenu que les surfaces en question sont très faillées et que les failles les moins significatives n'ont pas été utilisées dans la modélisation. Par ailleurs, sur ce même puits, les épaisseurs des différentes unités sédimentologiques sont relativement fidèles, à l'exception des unités U6, U12 et U13.

Au niveau du puits «défavorable» où toutes les conditions sont remplies pour que le résultat soit entaché d'erreurs , on observe un décalage de la séquence synthétique d'environ 10 mètres vers le bas par rapport à la séquence réelle. L'épaisseur des différentes unités sédimentologiques est pardoxalement assez fidèlement respectée (les erreurs maximales n'excèdent pas 2-3 mètres).

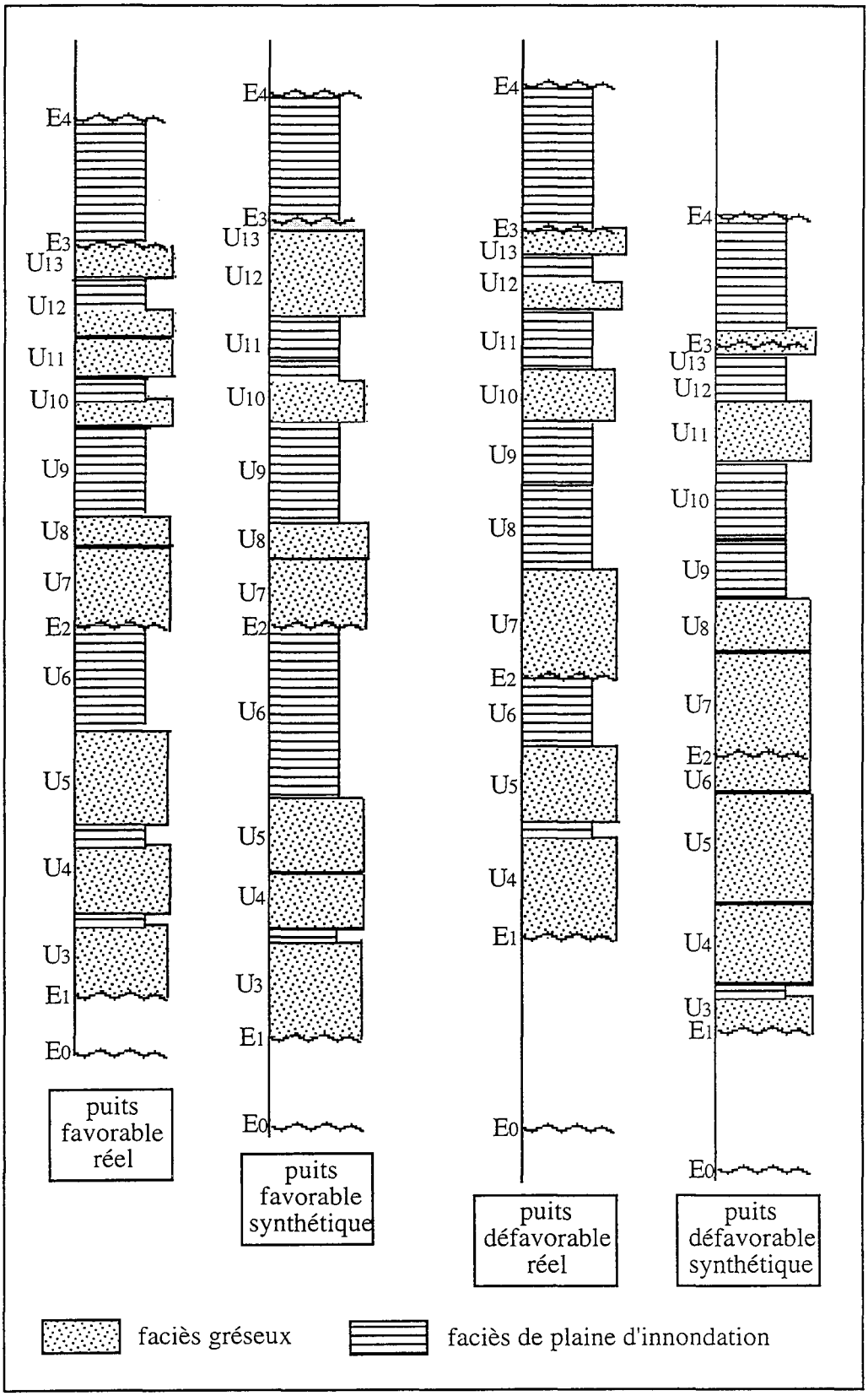


FIGURE 54. Présentation des résultats des tests de validation du modèle de réservoir.

Analyse des erreurs sur le type de dépôt

Un premier commentaire s'impose en ce qui concerne ce problème: nous avons opéré une simplification importante en ne retenant que deux familles de faciès:

- les faciès de type gréseux: grès propre, grès arkosique, grès silteux, grès argileux.
- les faciès de plaine d'innondation (argiles) et les faciès marins (argiles, calcrètes, dolocrètes).

Comme nous l'avons dit plus haut, cette simplification n'est pas très pénalisante en ce qui concerne les dépôts de type réseau en tresse qui sont relativement homogènes. Elle l'est plus pour les dépôts de type méandre qui sont plus complexes aussi bien dans leur structure que dans la diversité des faciès rencontrés.

D'une manière génèrale, nous pouvons dire que les faciès sont reproduits avec une grande fidélité dans les cas des unités correspondant à des réseaux en tresse, qui sont les unités U_3, U_4, U_5, U_7 et U_8, alors que les résultats sont nettement moins bons dans le cas des méandres, qui correspondent aux unités U_6, et U_9 à U_{13}. On retrouve là le même problème du à la grande variabilité structurale et pétrographique des formations de type méandre.

Cette règle ne souffre que de deux exceptions, l'unité U_9 correspondant à un méandre et qui est bien rendue dans les deux puits, et l'unité U_8 de type réseau en tresse qui est érronée sur le puits défavorable.

5.4.2 Exploitation du modèle

Une fois ce modèle construit et validé, tout du moins en ce qui concerne les unités correspondant à des réseaux en tresse, qui sont les unités réservoir, nous pouvons passer à la phase d'exploitation de ce modèle. Les résultats de cette exploitation ne seront bien entendu pas présentés ici pour des raisons évidentes de confidentialité.

Notre modèle va donc nous permettre:

- de visualiser en trois dimensions la structure du réservoir (distribution des failles, géométrie de sa structure).
- d'effectuer des estimations de volume du réservoir en fonction de la profondeur de son plancher. Dans l'état actuel des choses, nous ne pouvons qu'estimer des volumes de roche réservoir. Une stratégie d'affectation de valeurs de porosités aux cellules du modèle reste à définir, pour pouvoir estimer des volumes de gaz stockables.
- ce modèle géologique peut être utilisé, après affectation de valeurs de porosités et de perméabilités aux cellules de la grille, comme point de départ pour effectuer des simulations de mise en gaz du réservoir.

Conclusion générale

Les travaux présentés tout au long de cette thèse peuvent se regrouper en trois parties relativement indépendantes:

1. La première partie concerne la modélisation des séquences stratigraphiques, qui nous a amené à créer un nouvel objet: le *gstack*. L'intérêt de cet objet est double. Il permet d'une part de générer des structures sédimentaires avec une relative facilité. D'autre part, il permet d'introduire la notion de couche géologique, qui peut être vue soit comme un objet constitué de deux surfaces isomorphes (le toit et le mûr de la couche), soit comme un objet volumique par décomposition de l'espace entre ces deux surfaces par des tétraèdres.
2. En second lieu, les fonctions de conversion de maillage irréguliers en grille régulière que nous avons conçues sur la base de l'objet *voxet*, sont d'un grand intérêt.

   - Dans le cas des surfaces triangulées, elles permettent en effet de compartimenter une grille.

   - Dans le cas d'un solide à base de tétraèdres, l'échantillonage d'une propriété connue aux nœuds de cet objet sur une grille s'est avéré être une opération très efficace et peu coûteuse en temps de calcul.

   - La conversion en grille d'un modèle à base de *gshapes* est une condition importante pour utiliser ces objets dans le cas de simulations stochastiques booléenne. En effet, elle permet de s'affranchir du problème des intersections entre objets lors de l'estimetion du rapport chenaux/matrice. dans le modèle

   - La conversion de l'objet gstack en grille régulière quant à elle permet d'effectuer des interpolation 3D de propriétés contraintes par une structure géologique.
3. La méthode D.S.I anisotrope, outre l'intérêt théorique qu'elle présente en montrant comment il est possible de contraindre une interpolation par un champ de direction non uniforme, constitue un moyen de tenir compte du caractère fortement anisotrope des réservoirs fluviatiles.

Chacun de ces éléments pris un à un présente donc un intérêt certain. Cependant, et c'est ce que j'ai voulu démontrer dans le dernier chapitre de cette thèse, c'est la manière de combiner tous ces outils les uns aux autres qui permet, lors de la construction d'un modèle géologique, d'intégrer un maximum d'informations provenant souvent de différentes sources (données sismique, données de puits, analyses sédimentologiques, etc..).

Cette intégration de plusieurs sources de données représente un investissement en temps et en énergie tout à fait important. Il impose en effet souvent tout un ensemble d'opérations intermédiaires (de conversions de formats, de changement de maillage en cours de travail, etc...) que les anglo-saxons désigent sous le terme de «data gymnastics» (gymnastique des données). Cependant, il me semble que c'est le prix à payer pour construire des modèles géologiques de qualité.

Liste des figures

Liste des tableaux

Liste des algorithmes

Liste des planches

Matériel informatique

Deux types de matériels ont été utilisés dans le but d'effectuer des tests de performances des algorithmes présentés dans cette thèse. Nous présentons ici leurs caractéristiques techniques:

1. La machine désignée sous l'appellation *extrème* est une machine Silicon Graphics, modèle Indigo 2 Extreme équipée:
   - d'un processeur IP22 cadencé à 200 Mhz.
   - d'une unité CPU R4400.
   - d'une unité FPR R$010.

2. La machine désignée sou l'appelleation indigo est une machine Silicon Graphics, modèle indigo XS24 équipée:
   - d'un processeur IP12 cadencé à 33 MHz.
   - d'une unité CPU R2000A/R3000.
   - d'une unité FPU R2010A/R3010.

Bibliographie

[1] T. Aït Ettajer, J.L Mallet. *Modelling the Fault-Horizon relationship*. Expanded abstracts of papers, 7th EAPG conference and technical exhibition, Vienne 1994.

[2] T. Aït Ettajer. *Modélisation de Surfaces Géologiques Complexes sous Contraintes Géométriques*. Thèse de Doctorat I.N.P.L 1995.

[3] F. Aminzadeh, N. Burkhard, L. Nicoletis, F. Rocca, K. Wyatt, *SEG/EAEG 3-D modeling project, 2nd progress report*. First Break Vol. 12, No 10 October 1994.

[4] P. Bézier. *Mathématiques et C.A.O: Courbes et Surfaces*. Hermès.

[5] G.Booch (1992). *Conception orientée objets et applications*. Addison-Wesley Publishing Company.

[6] T.Budd (1991). *Introduction à la programmation par objets*. Addison Wesley Publishing Company.

[7] Y.Chipot, Y.Huang, P.Jacquemin, J.L.Mallet (Janvier 90). P*résentation du programme Gocad et premiers résultats*. Journées AFCET GROPLAN, BIGRE, num.67, pages 75-85.

[8] R.Clementsen, A.R Hurst, R.Knarud and H.Omre (Juin 89). *A Computer Program for Evaluation of Fluvial Reservoirs*. presented at the 1989 Intl Conference on North Sea Oil and Reservoirs, Trondheim, Norway.

[9] R. Cognot, P.Lavest, F.Bosquet, J.L Mallet, (1994). *Generating a 3D velocity model: the Gocad approach*. EAEG/SEG Summer Workshop on Construction of 3D Macro Velocity/Depth Models, Noordwijkerhout, The Netherlands.

[10] R. Cognot. *La méthode D.S.I.: Optimisation, Implémentation et Applications*. Thèse de Doctorat I.N.P.L (1996).

[11] J.D Collinson. *Vertical Sequence and sand body shape in alluvial sequences*. Fluvial Sedimentology, Miall A.D. ed, Can. Soc. of Petroleum Geologists - Memoir 5 - 1978.

[12] G.R. Cross, A.H Jain (Janvier 1983). *Markov Random Field Texture Models*. IEEE Transactions on pattern analysis and machine intelligence, vol. PAMI-5, No1.

[13] B.Delaunay. *Sur la sphère vide*. Bulletin de l'Académie des Sciences de l'U.R.S.S, Classe Sci. Mat. Nat., Vol. 7 pp. 793-800 (1934).

[14] O. Dubrule, (Mars 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. Einsele. *Sedimentary Basins*. Springer Verlag.

[16] A. Ellis, B. Stroustrup, (1991). *The Annotated C++ Reference Manual.* Addison-Wesley Publishing Company.

[17] F.G. Ethridge, S.A. Schumm. *Reconstructing paleochannel morphologic and flow characteristics: methodology, limitations and assessments.* Fluvial Sedimentology, Miall A.D ed. , Can. Soc. of Petroleum Geologists - Memoir 5 - 1978.

[18] G.Farin (1988). *Curves and Surfaces for Computed Aided Geometric Design. A practical guide*. Academic Press, Inc.

[19] G.Farin (1987). *Geometric modeling: algorithms and new trends.* Society for Industrial and Applied mathematics.

[20] G.Farin (1991). *Triangular Bernstein-Bezier patches*. Computer Aided Geometric Design.

[21] J.D Foley, A Van Dam, S.K. Feiner, J.F Hughes. *Computer graphics, principles and practice.* Addison-Wesley Publishing Company.

[22] E. Gamma, R.Helm, R. Johnson, J. Vlissides, (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Publishing Company.

[23] H.H. Haldorsen and E.Damsleth (Avril 1989). *Stochatic Modeling*. SPE Distinguished Author Series.

[24] A.Henriquez, K.J.Tyler, A.Hurst (Septembre 1990). *Characterization of Fluvial Sedimentology for Reservoir Simulation Modeling.* SPE Formation Evaluation, 1990.

[25] D.Hearn, M.P. Baker (1986). *Computer Graphics*. PRENTICE-HALL, Englewood Cliffs, New Jersey.

[26] T.A Hewett, R.A Behrens (Septembre 1990). *Conditional Simulation of Reservoir Heterogeneities with Fractals*. SPE formation Evaluation, 1990.

[27] Y. Huang (Juin 1990) *Modélisation et manipulation des surfaces triangulées*. Thèse de Doctorat de l'I.N.P.L.

[28] E.H Isaaks and R.M Srivastava (1989) *An introduction to applied geostatistics*. Oxford Press.

[29] A.G Journel (1989) *Fundamentals of geostatistics in Five Lessons*. American Geophysical Union.

[30] K.Lamboglia. *Modélisation Volumique de Surfaces Non-Manifold.* Thèse de doctorat I.N.P.L 1994.

[31] P.Lavest and Y. Chipot. *Building complex horizons for 3D seismic*. Technical Program and Expanded Abstracts, 1993 Annual SEG meeting, Washington.

[32] P. Lavest, R.Cognot, (1994). *3D anisotropic interpolation interpolation of geophysical parameters*. 6th EAPG Meeting and Technical Exhibition, Vienna.

[33] J.P. Le Roux. *Determining the channel sinuosity of ancient fluvial systems from paleocurrent data*. Journal of Sedimentary Petrology. Vol 63, No. 2, March 1993.

[34] J.L Mallet, R.Cognot, S.Cases (Juin 1993). *3D interpolation of geophysical parameters. The Gocad approach*. 55th EAEG meeting, Stavanger. Extended abstracts of papers.

[35] J.L Mallet, P.Le Melinaire (juin 1992). *Modeling complex faults. the Gocad approach.*

54th EAEG Meeting, Paris. Extended abstracts of papers.

[36] J.L Mallet (Avril 1992) *Discrete Smooth Interpolation in geometric modeling.* Computer Aided Design, Vol 24, No 4.

[37] J.L Mallet, (1989) *Discrete Smooth Interpolation.* ACM Transactions on graphics, Vol 8, No 2.

[38] M.E Mortenson, (1985) *Geometric Modelling.* John Wiley.

[39] L.Piegl, (Janvier 1992). *On NURBS: A survey.* IEEE Computer Graphics and Applications.

[40] F.P Preparata, M.I Shamos. *An introduction to computational geometry.* Springer Verlag, 1985.

[41] W.Press, B.Flannery, S.Teukolsky and W.Vetterling (1986). *Numerical Recipies.* Cambridge University Press, NY.

[42] J. Rumbaugh, (1992). *Let there be objects: ashort guide to reification.* Journal of Objetc Oriented Programming.

[43] B.R Rust. A classification of alluvial channel systems, Miall A.D ed. , Can. Soc. of Petroleum Geologists - Memoir 5 - 1978.

[44] S.A Schumm. *The fluvial system.* John Wiley & Sons Inc. 1977.

[45] Y. Touffait, H. Beucher, D. Guerillot. *A 3D integrated structure for computer-aided reservoir characterization.* 5th SPE Petroleum Computer Conference, Denver 1990.

[46] K. Weiler, *Edge Based Data Structures for Solid Modeling in Curved-Surface Environments.* IEEE computer Graphics and Applications.

[47] K. Weiler. *Topological Structures for Geometric Modeling.* Phd Thesis, Rensselaer Polytechnic Institute, August 1986.

[48] 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.

[49] L. 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.* Thèse de Doctorat I.N.P.L 1994.

[50] L.Wietzerbin, J.L Mallet (Septembre 1993). *Modeling complex 3D heterogeneities with the Discrete Smooth Interpolation Method.* Technical Program and Expanded Abstracts, 1993 Annual SEG meeting, New Orleans.

[51] L. Wietzerbin, J.L Mallet (Juin 1992). *A method for fitting three dimensional shapes to real reservoir data.* 54th EAEG meeting, Paris. Technical program and extended abstracts.

[52] K.D. 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.

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 CHEIMANOFF Nicola, Professeur, Ecole des Mines de Paris,
Monsieur PERRIN Michel, Professeur, Ecole des Mines de Paris,
Monsieur ROYER Jean-Jacques, Ingénieur de Recherche, CRPG/ENSG Vandoeuvre.

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

Monsieur LAVEST Pascal

à soutenir devant l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE, une thèse intitulée :
"Modélisation de la structure interne des réservoirs de type fluviatile. Application sur un site de stockage de gaz en aquifère".

en vue de l'obtention du titre de :

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Spécialité : **"GEOSCIENCES"**

Fait à Vandoeuvre le, **7 Juin1996**
Le Président de l'I.N.P.L.,
M. LUCIUS

TEL. 33/83.59.59.59
FAX. 33/83.59.59.55