A Isabelle, à mes parents.

Remerciements

==========

Rien n'aurait été possible sans la présence de mon directeur de thèse Jean-Laurent Mallet qui m'a épaulé tout au long et qui m'a poussé pour que je finisse d'écrire ce mémoire dans les temps. Je tiens aussi à remercier Yves Bertrand, mon co-directeur de thèse qui n'a jamais hésité à venir me voir pour faire le point sur l'avancement de mes travaux.

Evidemment, je remercie aussi bien les rapporteurs de ma thèse, Mariette Yvinec, Pascal Lienhardt et Jens Gustedt ainsi que le dernier membre du jury Charles H Sword.

Merci à Bruno Levy qui a partagé mon bureau et qui m'a beaucoup aidé. Je remercie aussi Jérome Massot qui partage mon bureau et aussi ma folie (au fait, il faut que tu arrêtes d'écouter Fun Radio à fond avec ton casque car cela risque de détériorer tes oreilles). Je remercie par la même occasion Guillaume Caumon qui partage notre bureau et qui doit subir nos démences (ne t'en fait pas, je reviendrai de temps en temps pour animer le bureau).

Et je remercie également Sophie Viseur qui, au moment où j'écris ces lignes doit être en train de finir sa thèse tout comme moi. Je remercie Nathalie Flammang qui partage le même bureau que Sophie et qui a aussi partagé le même bureau qu'Isabelle Duvinage, ma femme. Je continue dans les bureaux suivants et je remercie dans l'ordre alphabétique, Marcus Apel, Richard Cognot (y a encore des mars à la cafette ?), Matthieu Dazy (marchera pas :)), Olivier Grosse, Pierre Jacquemin (Mon Général), Thomas Jérome, Laurent Labat, Christien LeCarlier, David Ledez, Francois Lepage, César Mugerin, Laurent Souche et Jean-Charles Voillemont.

Relocalisé à deux kilomètres du laboratoire, je change de batiment et j'en profite pour remercier tous les t-surfers, Taoufik, Julien, Etienne le chanteur, Joel, Didier, Isabelle, Karine, Nicolas, Pascal, Isabelle M., Olivier, Philippe, Louis, Thierry Le Meulson, Bernadette.

C'est en changeant de pays que je remercie tous les houstoniens, Fabien, Amy, Chuck, Jean-Claude, Nathalie, Magali, David, Stanislas, et les autres.

Impossible de ne pas avoir une pensée pour ceux qui se trouvent ailleurs, Christophe Basire dit *Fofi*, Karen Payrazyan (ting ting), Wolfgang Velten. Par la même occasion, je remercie Arnaud Vauthier qui a partagé mon bureau pendant quelques mois (tu es où en ce moment ?!?).

Enfin, je remercie Jérome Vuittenez qui a fait l'IUT d'informatique de Belfort en même temps que moi et qui continue à m'envoyer régulièrement des mails pour me tenir au courant de ce qu'il fait (on arrivera bien à trouver du temps pour faire évoluer Fourmix).

Reste à remercier les gens de l'ASGA et notamment Elisabeth Mouillé et Nicole Galloy.

Résumé

La modélisation volumique a pour but de représenter un objet réel par des objets informatiques. En géologie, un modèle 3D peut être défini à partir d'un ensemble de surfaces partitionnant l'espace en *régions*. Par exemple, ces surfaces peuvent être des objets géologiques (horizons, failles, ...). Un objet volumique composé de 3-cellules (tétraèdres ou polyèdres) est aussi une représentation d'un modèle 3D. Avec ce deuxième type de modèle, il est possible d'attacher des propriétés physiques aux noeuds du maillage.

A l'aide d'un noyau topologique à base de G-Cartes, nous étudions les problèmes suivants :

- Définir des structures de données efficaces pour représenter la décomposition des objets en éléments finis,
- Générer et éditer des maillages d'objets surfaciques ou volumiques (destruction de cellules, éclatement de cellules ...),
- Utiliser un outil polyvalent appelé coraffinement qui consiste à combiner deux objets entre eux pour en former un qui correspond à leur union.

Nous présentons également des applications géologiques et notamment en utilisant l'opération de coraffinement : insérer un chenal (corps géologique) décomposé en cellules dans une grille régulière (les cellules intersectées de la grille ne sont pas préservées, le contact entre les cellules du bord du chenal et la grille sont parfaits), opérations booléennes sur des objets géologiques, ...

Abstract

Volumic modeling enables us to represent real objects by computer science objects. In geology, a 3D model may be defined by a set of surfacic objects partitioning the 3D space in *regions*. For instance, these surfaces can be horizons or faults (geological objects). A volumic object composed of 3-cells (tetrahedra or arbitrary polyhedra) could be a second way to represent a 3D model. With this kind of representation, it is possible to attach several properties on the nodes of the mesh.

Using a topological kernel based on G-Maps, we will study the following issues:

- Defining efficient data structures enabling the decomposition of objects into discrete elements to be represented,
- Generating and editing meshes for surfacic and volumic objects (removing cells, splitting cells, ...),
- Using a multi-purpose operation called corefinement.

We also present several geological applications using the corefinement operation: insertion of a gridded channel (geological object) in a regular grid (the intersected cells of the grid will not be preserved, the contact between the border cells of the channel and the grid are perfect), boolean operations between geological objects, ...

Table des matières

Introduction

Domaine d'étude

Cette thèse s'inscrit dans le cadre de la modélisation volumique d'objets géométriques 3D. Nous nous intéressons au partitionnement de l'espace par des frontières (surfaces) ou à des objets volumiques construits à partir de cellules volumiques (tetraèdres, polyèdres).

Cette étude s'inscrit dans le projet Gocad (**G**eological **O**bjects **C**omputer **A**ided **D**esign). Ce projet a vu le jour en 1988 à l'initiative du Professeur Jean-Laurent Mallet à l'Ecole Nationale Supérieure de Géologie de Nancy. Le but était de fournir un ensemble d'outils permettant de modéliser les objets naturels du sous-sol. Très vite, le développement du modeleur s'est étendu à la modélisation volumique, à la prise en compte de propriétés permettant d'ajouter un contexte géologique aux objets géométriques, à la modélisation d'objets plus particuliers tels que les chenaux ou les puits, à l'extension à des domaines autres que la géologie structurale ; par exemple la géophysique ou la géostatistique et enfin à la visualisation de tous ces modèles tridimensionnels.

La modélisation du sous-sol est un domaine en pleine expansion du fait des applications industrielles et des intérêts économiques en jeu mais aussi parce que les problèmes soulevés sont importants et difficiles. C'est dans cette optique que Gocad a donné d'abord naissance à un consortium regroupant différents partenaires de plusieurs domaines ; en particulier celui de compagnies pétrolières ou minières mais aussi instituts de recherche ou fabricants de logiciels pour le domaine pétrolier et minier. 30 industries et 30 universités adhèrent à ce consortium. Ces chiffres varient régulièrement lorsque de nouveaux sponsors arrivent ou lorsque des sponsors fusionnent. C'est en 1997 que des anciens élèves de l'école de géologie fondent une société nommée *T-Surf* pour répondre aux exigences des sponsors du consortium. Le logiciel est commercialisé et possède une équipe de maintenance pour le stabiliser. En l'espace de 4 ans, trois filiales (Houston, Nancy et Paris) sont créés pour permettre d'accueillir les 40 employés.

Domaine géologique

Le principal domaine d'application du modeleur Gocad est l'exploration pétrolière et minière. La présence de pétrole dans un bassin ne peut se faire que si un certain nombre de paramètres sont favorables : composition et structures des roches, température, pression, ... La recherche pétrolière est complexe et fait appel à de nombreuses branches des géosciences étudiant chacune un ou plusieurs de ces paramètres.

La géophysique étudie des propriétés physiques du sous-sol pour mettre en évidence sa structure et ses propriétés.

La géologie structurale consiste à représenter la structure du sous-sol, (c'est-à-dire la géométrie des couches délimitées par des horizons et failles) qui peut être complexe du fait de l'action de différents phénomènes physiques (extension, compression, ...).

Une autre branche des géosciences nécessite de représenter des propriétés physiques (porosité, champs de pression, température, ...). Pour répondre à cette demande, des supports tridimensionnels de différentes natures sont disponibles :

- subdivision de l'espace (représentation de la structure du sous-sol)
- grilles structurées ou non structurées (pour rattacher des propriétés physiques)

Enfin, la modélisation des hétérogénéités de réservoir est l'étude à plus petite échelle de zones plus précises du bassin (les *réservoirs*) où les conditions favorables à la présence de pétrole pourraient être réunies. Cette étude fait appel à des méthodes géostatistiques. Nous montrons dans le chapitre 2 un tel objet volumique.

Domaine topologique

Pour répondre à ces besoins, le coeur du logiciel Gocad se base sur la topologie de Weiler [**?**] amplement utilisée dans différents domaines et assez connue. Cependant, comme nous le verrons la structure est assez lourde et complexe et ceux malgré les formalismes instaurés par l'auteur. Nous proposons de changer de noyau topologique pour opter pour un noyau mathématique et robuste connu sous le nom de G-Carte [**?**]. Une G-Carte est constituée d'un ensemble d'éléments abstraits nommés brins qui, liés avec d'autres via des involutions, permettent de construire des cellules pour obtenir un objet de dimension quelconque avec une grande flexibilité. Nous developpons à la fois les opérations de base des G-Cartes ainsi que la généricité d'opérations légères (c'est-à-dire des opérations qui modifient le maillage d'un objet) et les différentes manières d'écrire certaines opérations lourdes telles que le calcul du dual, l'extrusion de surfaces, ...

Organisation du mémoire

Le premier chapitre introduit la modélisation 3D en géologie et étudie les différents modèles topologiques/géométriques déjà réalisés dans le domaine. Nous nous focalisons sur le modèle que nous avons choisi, développons le noyau topologique à base de G-Carte et expliquons comment construire un modèle géologique à partir de cette structure.

Le deuxième chapitre présente différents types d'opérations sur les G-Cartes :

- une description des opérations de base des G-Cartes est donnée ainsi qu'une implantation de ces dernières,
- des opérations dites *légères* sont expliquées en détail : fusion, scission, résorption, ... : ces opérations permettent de modifier des maillages de manière locale,
- le dual, l'extrusion ou insertions/subdivisions de maillages sont des opérations plus lourdes qui sont utilisées pour des applications géologiques.

Dans le dernier chapitre, nous étudions une opération qui peut être considérée comme une boîte à outil permettant de toucher à plusieurs applications : le coraffinement. Cette opération consiste à combiner deux objets entre eux de manière à obtenir un maillage qui correspond à leur union. Plusieurs méthodes sont étudiées et ont été implantées dans le modeleur pour des surfaces. Une extension aux volumes tetraédrisées est décrite par la suite. Nous donnons aussi des solutions pour réduire les erreurs numériques. Des résultats ainsi qu'une série de comparatifs sont donnés à la fin de ce chapitre juste avant les applications géologiques réalisées grâce au coraffinement. Finalement, la conclusion présente le bilan et les perspectives de recherche.

Chapitre 1

Présentation générale

Cette étude se situe dans le cadre de la modélisation géologique. Dans la section 1.1, nous nous familiarisons avec des termes et objets de géologie [**?**] et nous faisons le lien entre ces objets et l'informatique, puis en section 1.2 nous présentons les différents modeleurs existants. Nous expliquons dans la section 1.3 ce qu'est la modélisation géométrique à base topologique. Avant de conclure ce premier chapitre, nous présentons en section 1.4 la structure topologique que nous avons choisi d'utiliser ainsi que notre apport dans ce domaine.

1.1 Modélisation 3D en géologie

Dans cette section, nous ne cherchons pas à faire une étude complète du domaine géologique mais juste à introduire les bases et des notions qui pourront nous aider à expliquer l'intérêt de certaines applications géologiques.

1.1.1 Qu'est ce qu'un modèle géologique ?

Dans le domaine de la géologie, les objets modélisés représentent les empilements de couches du sous-sol (voir figure 1.1-A). Ces couches sont des régions volumiques ou *blocs* (voir figure 1.1-B) qui sont délimitées par leurs bords. Nous verrons dans les paragraphes suivants qu'un bord peut être de différentes natures et notamment un *horizon* ou une *faille*.

Un **horizon** (voir figure 1.1-C) correspond à l'interface qui sépare deux couches ou milieux de natures différentes. Au cours du temps, des sédiments de différentes natures (sable, carbonates, sel, argiles, ...) se sont superposés. Ces couches peuvent avoir été déformées sous l'action de forces tectoniques, ce qui peut rendre leur géométrie très compliquée.

Une **faille** (voir figure 1.1-D) correspond à une cassure d'un ou plusieurs blocs avec *déplacement relatif* des parties séparées [**?**, **?**]. Ces blocs ont été soumis à différentes contraintes (cisaillement, compression, extension) et ont été décalés

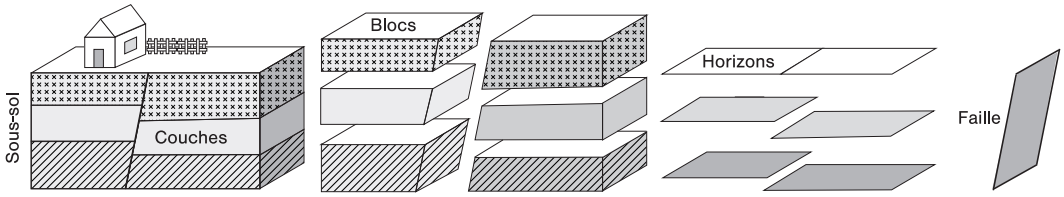


Figure 1.1 *Composition du sous-sol : empilements de couches. Les couches sont divisées en blocs délimités par des horizons et des failles.*

les uns par rapport aux autres (voir figure 1.2-A).

Pour obtenir un *modèle* du sous-sol, plusieurs méthodes sont utilisables : méthodes directes (puits de forage ou carottages), et indirectes (sismiques, gravimétrie, ...).

1.1.2 Comment acquérir des données du sous-sol ?

Historiquement, le premier problème rencontré par les géologues dans le domaine du pétrole était l'acquisition de données. En effet, le sous-sol est difficile d'accès et la taille du domaine d'étude se mesure en kilomètres voire en dizaines de kilomètres.

Avant l'apparition de techniques évoluées d'acquisition de données, les géologues n'avaient à leur disposition que les forages (*puits*) ou des données sismiques en deux dimensions. Les données de puits représentent la source d'information la plus précise (de l'ordre du centimètre). Leur localisation est tout à fait déterminée et la précision des données est plus fine. Le carottage consiste à extraire une colonne de roche. A partir des roches extraites, il est possible de faire des mesures directes de leurs propriétés physiques et chimiques. En général, les puits sont peu nombreux et dispersés car leur coût est exorbitant.

Puis sont intervenues les *campagnes sismiques*[1]. La sismique est une branche très importante de la géophysique puisqu'elle représente plus de 90% des sommes dépensées dans ce domaine. Elle est fondée sur l'étude de la propagation d'ondes élastiques dans le sous-sol. Deux types de méthodes sismiques sont à distinguer : la *sismique par réflexion* et la *sismique par réfraction*. Pour de plus amples descriptions, le lecteur pourra se référer à [**?**, **?**].

La sismique par réflexion, de loin la plus répandue, consiste à émettre des ondes élastiques à partir d'un point source puis à mesurer les caractéristiques des ondes réfléchies en différents points récepteurs, disposés régulièrement (voir figure 1.2-B). Ces ondes sont créées par explosion ou par vibration et se propagent élastiquement dans le sous-sol. A chaque interface entre deux couches, une partie

[1]Ces campagnes consistent à obtenir une échographie du sous-sol en envoyant des ondes dans le sol.

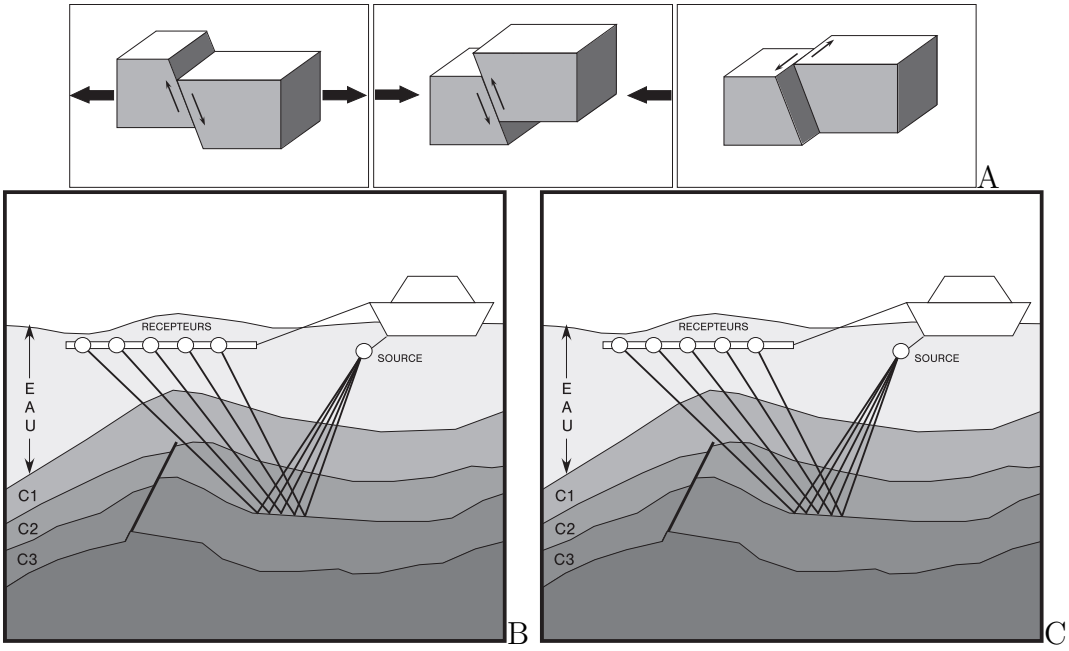


Figure 1.2 *A. Exemple de failles : faille normale (extension), faille inverse (compression) et faille transformante (décrochement). Il est à préciser que les deux premières failles peuvent avoir une composante décrochante, c'est-à-dire qu'en plus du rejet vertical (extensif ou compressif) peut s'ajouter un glissement. B. Principe simplifié de la sismique par réflexion. La source émet des ondes qui, après avoir rebondi sur une couche, sont récupérées par des recepteurs. C. Cube sismique : Résultat d'une prospection sur le delta du Niger (Cliché Shell Petroleum Development, Nigeria).*

de l'énergie de l'onde est renvoyée vers la surface. L'*impédance acoustique* d'une roche est égale au produit de la densité et de la vitesse sismique de la roche et est directement reliée aux propriétés pétrophysiques de la roche (porosité, cohérence, ...). L'onde réfléchie aura une amplitude d'autant plus grande que la différence d'impédance acoustique entre les deux milieux que sépare l'interface est importante ; on parle de *pouvoir réflecteur*. Après réflexion, les récepteurs reçoivent un signal qui dépend des mouvements du sol induits par les ondes. Les temps d'arrivée des ondes sont donc connus. Ce signal est amplifié et filtré ; chaque mesure amplifiée constitue une *trace*. La trace définit les variations des mouvements du sol avec le temps. Un *événement* est l'arrivée d'une énergie réfléchie qui varie systématiquement d'une trace à une autre. Ce qui distingue un événement d'un autre est non seulement l'amplitude de l'énergie réfléchie, mais aussi le fait que l'on retrouve cette énergie d'une trace à une autre.

Ainsi, un bon réflecteur doit être entre deux milieux d'impédances acoustiques différentes et doit être continu. Un exemple de réflecteur est un horizon. En effet, nous avions défini un horizon comme une interface entre deux couches de natures

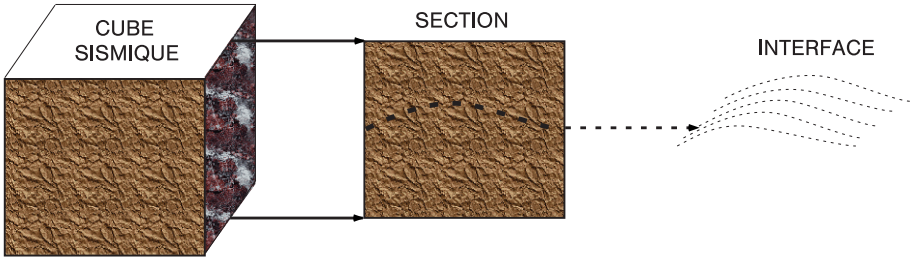


Figure 1.3 *Technique du pointé sismique : à partir d'un cube sismique, le géologue va numériser une série de points sur plusieurs sections. Ces points forment ce que l'on appelle une interface (ici un horizon).*

différentes. Un horizon est un bon réflecteur si la différence d'impédance acoustique entre les deux couches qu'il sépare est importante et s'il est suffisamment continu. Il est alors bien visible sur les données sismiques. Les failles peuvent également agir en tant que réflecteurs puisqu'en déplaçant des couches, elles peuvent devenir des interfaces entre des milieux d'impédances acoustiques différentes. Cependant, ceci n'est valable qu'en certains morceaux de la faille, qui n'est alors pas définie de manière continue. Les failles sont en fait surtout visibles grâce à la discontinuité qu'elles génèrent sur les horizons.

Les traces sismiques vont devoir subir différents traitements avant de pouvoir être interprétées. En effet, l'axe vertical ne correspond pas aux altitudes en profondeur mais aux altitudes en temps. Pour transformer les données en profondeur, l'opération consiste à *migrer* l'ensemble des données sismiques. Ces données peuvent alors être interprétées directement dans l'espace profondeur. Cependant, cette opération est coûteuse en temps et en mémoire. Une solution consiste à interpréter d'abord en temps puis à transformer le résultat de l'interprétation dans l'espace profondeur.

Les études sismiques peuvent être réalisées en deux ou trois dimensions (voir figure 1.2-C). Différents exemples de l'apport de la sismique 3D se trouvent dans [**?**], on évoquera par exemple, ...

1.1.3 Comment exploiter ces données sismiques ?

A partir du *cube sismique* les géologues recherchent les interfaces délimitants deux couches (horizons ou failles). Ces interfaces sont dans un premier temps extraites sous forme de semis de points puis finalement transformées en surfaces. Ces semis de points sont obtenus par *pointé manuel* ou *automatique*.

Dans le cas du pointé manuel, le géologue numérise les interfaces comme des séries de points sur une section du cube sismique, en général verticale comme le montre la figure 1.3.

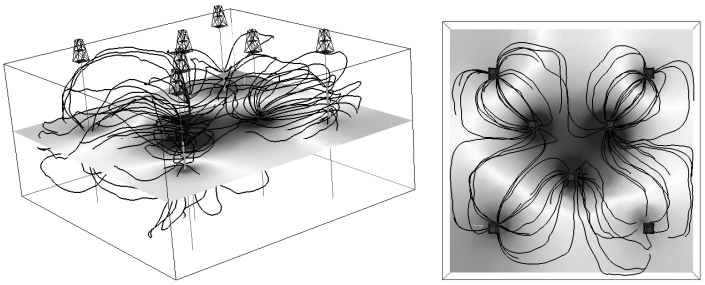


Figure 1.4 *Simulation de l'évolution thermique du site de Soultz au nord de l'Alsace. Les quatre puits périphériques sont des puits injecteurs d'eau froide. Les trois puits centraux récupèrent de l'eau chaude. Les courbes noires représentent les lignes de courant des fluides. Les niveaux de gris correspondent aux pressions de l'eau aux différentes localisations du modèle. (images de J.C. Voillemont).*

Le pointé manuel peut être associé à un pointé automatique[2]. Si la structure est simple, il sera entièrement automatique. Sinon, au passage de certaines failles il est préférable de guider le pointé automatique. Ainsi le géologue servira de guide de manière ponctuelle.

Une faille peut être interprétée à partir du cube sismique, ou bien être construite à partir des bords des horizons.

A partir de ces failles et horizons, les blocs sont retrouvés par *collage* de surfaces. La section 1.4.4 page 43 explique plus en détail la construction de ces blocs. Ils définissent une partition de l'espace et sont ensuite remplis par des propriétés (il peut par exemple s'agir de la porosité des roches ou de la vitesse de propagation des ondes). Ces données sont obtenues généralement de manière ponctuelle (par forage, par exemple) puis sont interpolées sur l'ensemble du domaine.

1.1.4 Quelles sont les applications géologiques ?

A partir d'un modèle sismique issu d'une campagne sismique, différents modèles géologiques peuvent être créés suivant l'application que l'on veut faire.

Le *modèle structural* est constitué d'un ensemble d'horizons et failles. Il permet au géologue de visualiser la géométrie des couches du sous-sol ainsi que les relations entre les différentes interfaces qui les séparent.

Le *modèle de vitesse* permet de traiter et de valider le cube sismique. Chaque couche correspond à un milieu géologique (sable, ...) et possède des lois de propagation des ondes. Des propriétés de vitesse sont rattachées à ce modèle pour

[2]les séries de points sont obtenues par un algorithme qui détectent les frontières de deux milieux d'impédance acoustique différente.

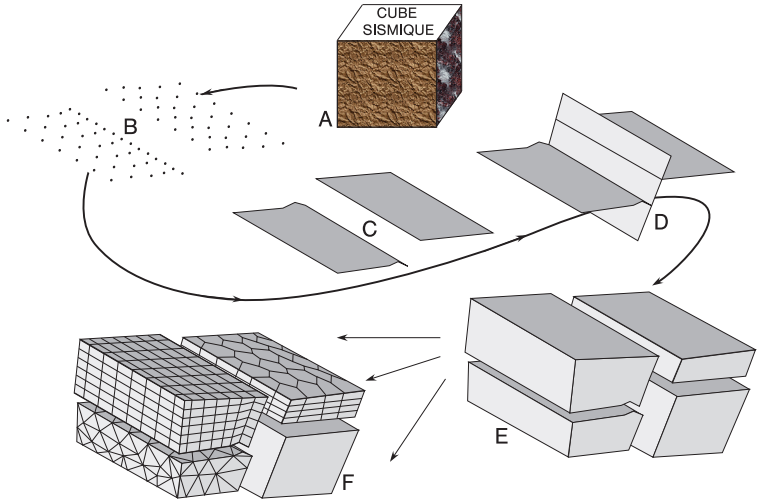


Figure 1.5 *(A) A partir d'un cube sismique, (B) des ensembles de points sont* pointés *par le géologue ou automatiquement. Chaque ensemble de points appartient à un horizon. (C) Cet horizon va être matérialisé par une surface. A partir des horizons, il est possible de construire (D) les failles. (E) La construction des régions se fait à partir des surfaces qui délimitent le bord du modèle. (F) Ces régions peuvent ensuite être remplies par des grilles hexaédriques, des tétraèdres ou des polyèdres pour y rattacher des propriétés.*

permettre d'effectuer du tracé de rayon. Ce tracé simule en fait une détonation qui se répercute sur les horizons et failles du modèle. En sortie, les ondes sont capturées pour former un cube sismique simulé. Ce cube est alors comparé au cube réel. Si les cubes *se ressemblent*, le géologue validera ce modèle. Au contraire, si les deux cubes sont différents, cela signifie que le modèle ne semble pas correct. Le modèle de vitesse devra alors être corrigé pour donner lieu à une nouvelle simulation.

Le *modèle de propriétés* qui associe à des objets géologiques des propriétés physiques : porosité, champs de pressions, champs de vitesses d'écoulement pour faire des simulations, des calculs d'incertitude pour mieux connaître le domaine. Par exemple, il est intéressant de récupérer la chaleur qui se trouve dans les roches du sol pour en faire une ressource d'énergie. En injectant de l'eau froide dans des puits, un ou plusieurs puits récupèrent de l'eau chaude. Des études sont effectuées pour simuler l'évolution thermique du domaine sur plusieurs dizaines d'années. En fonction des résultats, le site sera exploité ou non. La figure 1.4 montre l'exploitation virtuelle du site. Une autre application reliée au modèle de propriété est l'estimation du volume de pétrole stocké dans le sol. Pour cela, il faut connaître les propriétés de porosité des roches. De cette façon, il devient possible d'estimer le volume de pétrole qui peut se trouver dans le domaine étudié en multipliant le volumes des cellules par le pourcentage de porosité de ces dernières.

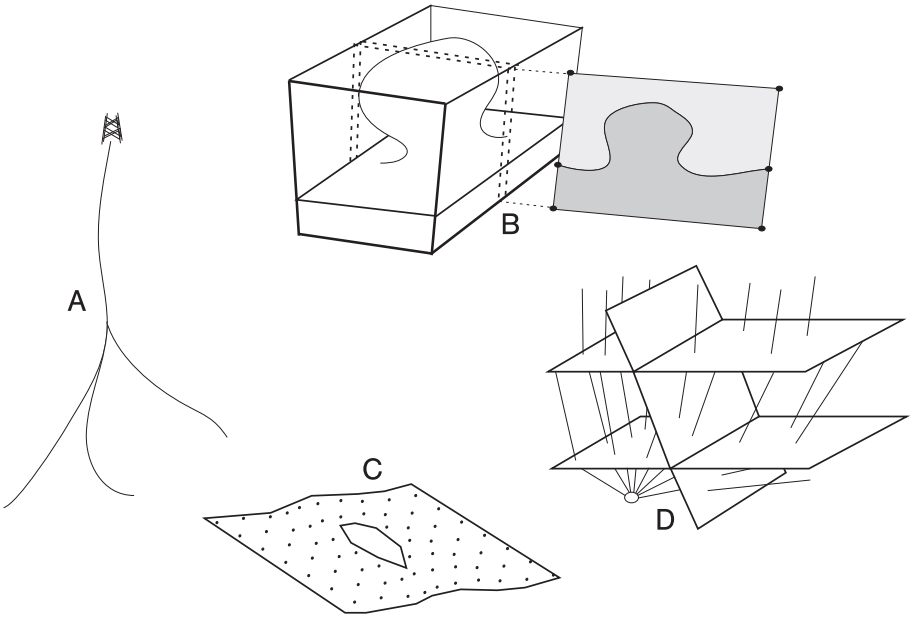


Figure 1.6 *Les différentes utilisations des lignes : (A) un puits, (B) une section d'un modèle 3D, (C) délimitation du bord d'un horizon ou (D) des tracés de rayons*

1.1.5 Comment traiter ces données en CAO ?

D'après [**?**], un modèle géologique décrit des objets géologiques par leur géométrie, leur topologie ou relations de voisinages et leurs propriétés. En informatique, ces modèles contiennent une hiérarchie de composantes telles que :

- des points,
- des lignes,
- des surfaces (horizons ou failles),
- des régions closes (blocs),
- des grilles (régulières ou stratigraphiques),
- des propriétés (vitesse sismique, porosité, ...).

Les points

Les points sont les éléments les plus simples. Ils sont obtenus en interprétant des cubes sismiques comme nous l'avons déjà expliqué ci-avant. Ces points décrivent les horizons ou les failles du modèle.

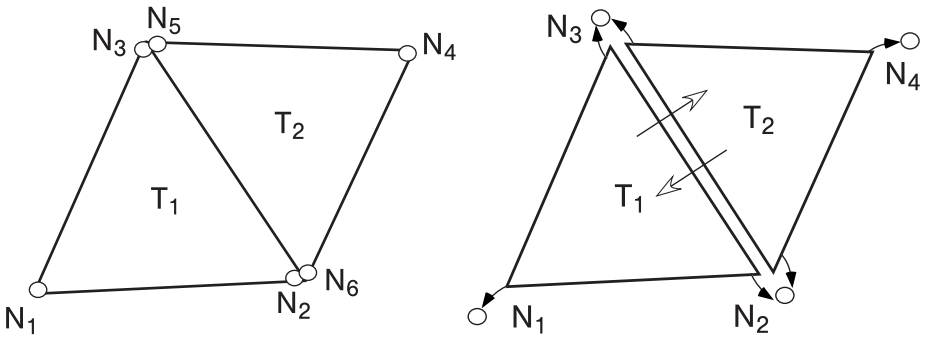


Figure 1.7 *Structure géométrique et topologique. (A) Une surface composée de triangles définis uniquement par leur géométrie : aucun lien de voisinage ne peut être déduit. (B) Ces deux triangles sont reliés topologiquement entre eux. L'arête qu'ils ont en commun est définie par les noeuds communs N_2 et N_3.*

Les lignes

Une ligne est définie par un ensemble de points reliés deux à deux par des segments ou par une autre équation géométrique. Ces lignes vont servir à représenter des tracés de rayons pour simuler la sismique, des intersections entre horizons/failles et failles/failles ou des trajectoires de puits (voir figure 1.6). Dans certains domaines, les modèles sont visualisés et manipulés sous forme d'une succession de sections 2D formées de lignes.

Les surfaces

Une surface peut être représentée par un ensemble de triangles connectés entre eux le long de leurs segments ou par une forme continue (spline, bézier, ...). Nous détaillons ces deux types de représentations dans la section suivante. Ces surfaces vont permettre de représenter de manière assez précise les horizons et les failles d'un modèle géologique et avec flexibilité.

La topologie

Dans le domaine de la science de la terre, la topologie est définie comme un ensemble de relations entre des objets géologiques. Mais la topologie peut aussi jouer le rôle de structure de données pour décrire et stocker ces objets comme nous le verrons plus tard. Ces relations entre les objets correspondent à une notion de voisinage : on cherche à déterminer les adjacences entre les sommets, arêtes et faces d'une surface par exemple.

Si l'on considère une surface triangulée comme l'ensemble de triangles définis par leur géométrie, il est difficile de connaître les triangles qui sont en contact avec un triangle donné. En effet, pour savoir si deux triangles sont *en contact* le

long d'une de leur arête, il faut comparer chaque couple d'arêtes provenant des deux triangles et tester si les sommets sont identiques. Dans l'exemple donné sur la figure 1.7-A, le triangle T_1 est en contact avec le triangle T_2 car les sommets N_2 et N_3 sont confondus avec les sommets N_5 et N_6. D'une part, ce test n'est pas élégant, mais d'autre part, il devient très long de savoir quels triangles sont en contact avec un triangle donné dans une surface qui en comporte plusieurs milliers par exemple. Il devient alors primordial de connecter topologiquement ces triangles entre eux pour ne pas être obligé de faire des calculs géométriques sur tous les triangles à chaque fois. Ainsi, pour chaque triangle, les triangles qui sont en contact avec ce dernier sont *stockés* explicitement (voir figure 1.7-B). La notion de relation ne concerne pas uniquement les triangles mais toutes les cellules d'un objet (sommets, lignes, blocs, ...).

Les blocs

Les blocs sont des régions de l'espace délimitées par des surfaces. Un bloc décrit une partie du sous-sol qui possède les mêmes caractéristiques.

Les grilles

Nous considérons 7 types de grilles différentes qui peuvent être séparés en deux groupes : les grilles structurées et les grilles non structurées.

Les grilles structurées :

- Grilles topologiquement et géométriquement régulières (figure 1.8-A) : Ces grilles sont constituées de cubes organisés selon trois axes. Leurs localisations sont définies par troix index et il est trivial d'accéder à leur voisinage en incrémentant ces index. La principale qualité de ce genre de grille est qu'elles sont compactes en mémoire, ce qui permet de stocker facilement de nombreux détails sur les modèles représentés par de telles grilles.

- Grilles stratigraphiques (figure 1.8-B) : ces grilles sont régulières topologiquement parlant mais sont déformées géométriquement pour suivre la géométrie de l'objet représenté. Une grille peut aussi être parallèle à la stratigraphie définie par un horizon. La géométrie devra être stockée pour chaque noeud tandis que son voisinage est toujours accessible par des index. Les points des surfaces entourant la région ne sont pas pris directement en compte pour construire la grille. Cependant, les bords de la grille respectent au mieux la géométrie des différentes surfaces. Elles sont utilisées pour stocker des propriétés, pour faire de la géostatistique ou des simulations d'écoulement.

- Grilles stratigraphiques faillées (figure 1.8-C) : ces grilles sont identiques au type de grilles citées ci-avant excepté que leur topologie est localement irrégulière. En effet, avec des grilles stratigraphiques standard, il est impossible de représenter des discontinuités comme le tracé d'une faille. Cette

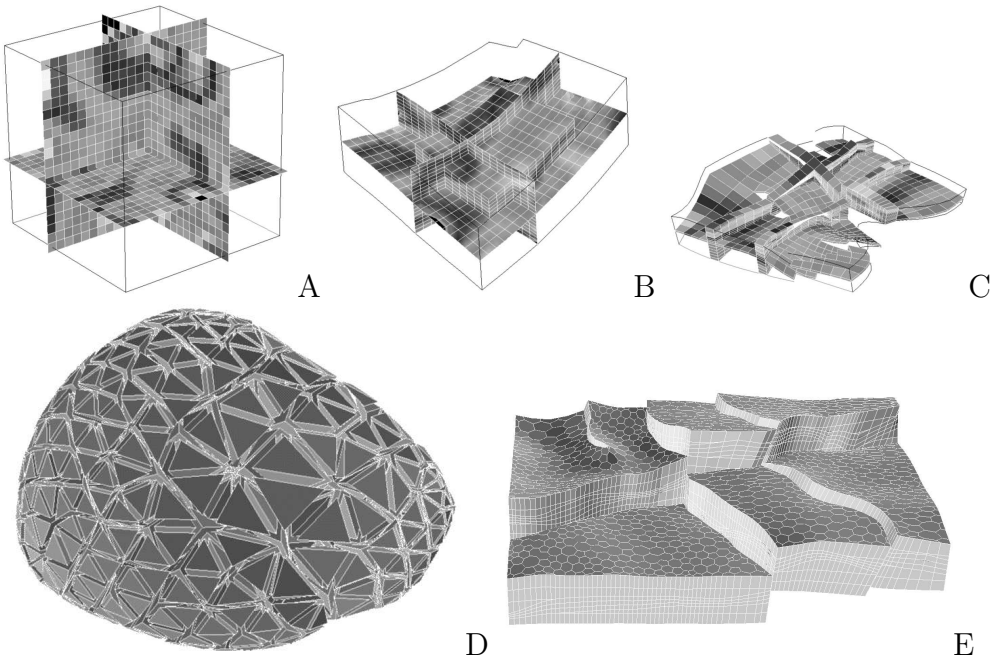


Figure 1.8 *A. Grille régulière B. Grille stratigraphique C. Grille stratigraphique faillée D. Grille tétraédrique E. Grille polyèdrique*

approche consiste à avoir des grilles dont certaines cellules peuvent être déconnectées les unes des autres. Il est à noter que même si les cellules sont déconnectées au niveau des failles, il pourra être nécessaire de reconstituer des relations de voisinages entre les cellules qui se touchent le long des failles.

Les grilles non structurées :

- Grilles tétraédriques (figure 1.8-D) : les surfaces triangulées qui délimitent la région vont servir de support pour la tétraédrisation de ce volume. Les triangles des surfaces deviendront des faces de tétraèdres. Ce type de grilles sera utilisé dans le domaine du tracé de rayon. Cela permet de valider les modèles 3D en comparant les résultats du tracé de rayon avec ceux de l'acquisition sismique. Cette tétraédrisation n'est malheureusement pas toujours très facile à obtenir [**?**, **?**, **?**, **?**].

- Grilles mixtes : ces grilles possèdent deux types de cellules : des hexaèdres et des tétraèdres. Elles sont donc régulières là où les cellules sont hexaédriques et permettent de mieux représenter certaines formes en utilisant des tétraèdres localement [**?**, **?**, **?**].

- Grilles hexaédriques non structurées : pour une grille régulière hexaédrique,

chaque sommet qui se trouve à l'intérieur de la grille est connecté à exactement huit hexaèdres. Les grilles hexaédriques non structurées ne possèdent pas cette caractéristique. Le degré[3] des sommets varie. Ces grilles sont plus souples que les grilles strictement régulières.

- Grilles polyédriques ou non structurées hétérogènes (figure 1.8-E) : ces grilles sont totalement irrégulières, aussi bien topologiquement que géométriquement. Toutes les grilles que nous avons vues précédemment étaient des grilles homogènes ou semi-homogènes (utilisant 2 types de cellules). Ce type de grille possède des cellules totalement hétérogènes. Elles font l'objet d'une nouvelle génération de simulateur de flux et de calcul de géostatistique. Elles permettent aussi de faire du tracé de rayon. Cependant, la flexibilité de ce genre de grille requiert plus de place en mémoire qu'une grille régulière. En effet, la géométrie est stockée explicitement ainsi que les relations de voisinage entre les cellules.

Les propriétés

Un modèle géologique peut contenir des propriétés stockées sur les points, lignes, surfaces, blocs et grilles. Il existe deux sortes de propriétés :

- Les propriétés dites *continues* sont définies par une fonction en tout point de l'espace : fonction constante, linéaire (selon la profondeur) ou plus compliquée. Par exemple, dans une roche homogène, la vitesse de la sismique est de la forme : $v = V0 + k * Z$ où Z représente la profondeur, $V0$ la vitesse lorsque $Z = 0$ et k une constante qui dépend de la roche. D'autres propriétés continues existent telles que la pression, la température, ...

- Les propriétés dites *discrètes* sont définies en certains points seulement, par exemple sur les noeuds de la grille qui est enfermée dans une région donnée. Ces propriétés ne peuvent pas être connues en tout point par des fonctions mathématiques. Elles sont connues en certains endroits dus à des forages et sont ensuite interpolées dans tout le modèle. Porosité et perméabilité sont d'autres propriétés discrètes.

1.2 Modeleurs 3D en géologie

Les premières méthodes de modélisation furent tout d'abord fondées sur un support 2D (carte en courbe de niveau, sections, surfaces isométriques). Ces méthodes ont été améliorées pour obtenir des résultats en 2,5D [**?**, **?**], c'est-à-dire

[3]degré d'un sommet : le nombre de cellules qui le partage. Il est de 8 pour un sommet qui se trouve à l'intérieur d'une grille régulière par exemple.

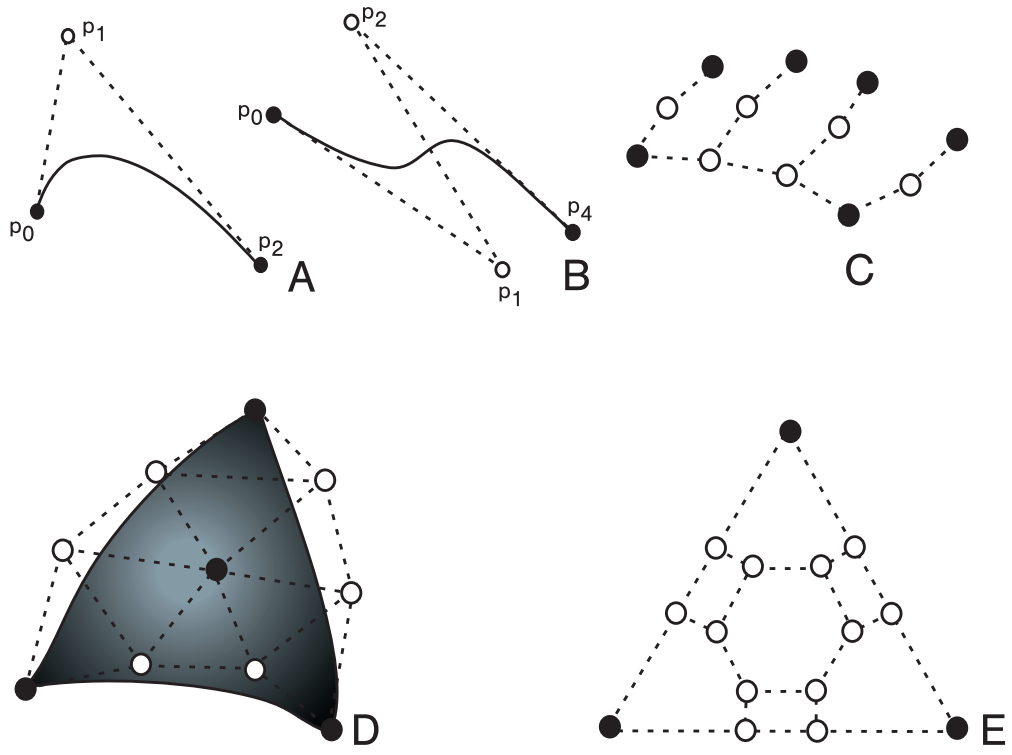


Figure 1.9 *A. Courbe de Bézier de degré 2. B. Courbe de Bézier de degré 3. C. Points de contrôle d'un carreau de Bézier. D. Carreaux triangulaires de Bézier. E. Carreaux triangulaires de Gregory.*

par empilement de cartes 2D. Mais les modèles pouvant être représentés de cette manière devaient être peu complexes, ce qui était limitatif en géologie.

Deux approches se sont alors dégagées :

- la représentation paramétrique des surfaces sous formes d'éléments de surfaces ou *patches* [**?**, **?**] qui constitue la base des méthodes de CAO classique (*approche continue*),

- des méthodes fondées sur des représentations 3D [**?**] (*approche discrète*).

1.2.1 Approche continue

Ces approches consistent à utiliser des courbes et surfaces polynomiales. Le lecteur pourra se référer à [**?**, **?**, **?**] pour obtenir une vue d'ensemble de ce type d'approche. Dans la suite de cette section, nous présentons différentes approches continues.

Bézier

Les courbes de Bézier ont été développées simultanément par Bézier [**?**, **?**] et de Casteljau [**?**] à la fin des années cinquante.

Une courbe de Bézier de degré n est définie par une équation mathématique dérivée de $n+1$ points caractéristiques de la courbe. Ces points sont appelés *points de contrôle*. L'équation d'une telle courbe est de la forme :

$$b(u) = \sum_{i=0}^{n} B_i^n(u).\mathbf{p}_i$$

avec :

$\mathbf{p}_i$: les points de contrôle,
$u \in [0,1]$ et
$B_i^n(u) = \frac{n!}{i!(n-i)!}u^i(1-u)^{n-i}$ (1.1)

Une courbe de Bézier est entièrement définie dans l'enveloppe convexe de ses points de contrôle et passe par le premier et dernier point de contrôle (c'est-à-dire que $b(0) = \mathbf{p}_0$ et $b(1) = \mathbf{p}_n$) comme le montre la figure 1.9-A-B.

On définit un *carreau* de Bézier par le produit cartésien de deux courbes de Bézier de même origine (voir figure 1.9-C). Une surface de Bézier est définie par un *réseau* de courbes de Bézier. Les points de contrôle forment une grille de contrôle. A nouveau, la surface de Bézier est entièrement définie dans l'enveloppe convexe de ses points. Il existe aussi des carreaux triangulaires (figure 1.9-D) de Bézier [**?**, **?**, **?**] que l'on peut généraliser à des n-simplexes [**?**].

Spline et BSpline

Les Splines [**?**, **?**, **?**, **?**] ont été définies pour contourner certaines limitations dues aux carreaux de Bézier.

Une BSpline [**?**, **?**], par exemple, est une figure linéaire constituée d'une chaîne continue de courbes de Bézier de degré 3. Chaque courbe est tangente aux autres en ses points de début et de fin. Contrairement aux courbes ou surfaces de Bézier, la modification d'un point de contrôle ne se répercute pas sur tout l'objet. En effet, ces courbes et surfaces fournissent un contrôle local, mieux adapté à un environnement interactif (industrie automobile, ...).

Nurbs et Gregory

Les applications géométriques ont besoin d'atteindre la continuité[4] G^1. Pour atteindre cette continuité, les carreaux de Bézier, par exemple, nécessitent des

[4]La continuité G^1 correspond à des surfaces pour lesquelles la direction du vecteur normal est continue.

polynômes de haut degré, ce qui crée souvent des problèmes de stabilité numérique et d'oscillations. Pour limiter ces degrés, des approches ont consisté à ajouter des termes en fraction rationnelle.

Les NURBS (Non Uniform Rational B-Spline) [**?**] font parties de ces approches et sont utilisées pour représenter des formes complexes. Certains logiciels de rendu comme rhinocéros 3D (http://www.rhino3d.com) ou Nurbana (http://www.nurbana.cx) pour ne citer que deux exemples, se basent entièrement sur des NURBS pour modéliser des objets.

Une autre famille de carreaux nommés carreaux de Gregory [**?**, **?**, **?**, **?**] (voir figure 1.9-E) est appréciée en géologie pour, par exemple, le tracé de rayons sismiques qui nécessite une continuité G^1. Les carreaux de Gregory ont été généralisés pour des cellules ayant un nombre arbitraire de côtés par Gregory et en dimension arbitraire par Bechmann *et. al.* [**?**] .

Récapitulatif

Les techniques utilisées pour ces approches continues sont bien connues et font l'objet de nombreuses publications ([**?**, **?**]) car ces objets mathématiques sont largement utilisés, notamment en CAO dans l'industrie automobile, aéronautique, ... En effet, les objets manufacturés ont souvent des formes lisses qui sont pleinement contraintes et bien plus simples à représenter à l'aide de ces surfaces à approche continue qu'avec un autre type d'objets. Les fonctions mathématiques utilisées ont des propriétés bien connues et il est aisé de calculer différents paramètres de celles-ci sans faire d'approximations (normale, dérivée, courbure, ...).

Cependant, il est difficile de représenter des discontinuités. Celles-ci sont en fait prises en compte en modifiant le domaine de définition de la fonction. Mais de tels domaines de définition morcelés demandent la modification, quelquefois de manière complexe, de la plupart des algorithmes. Pour des surfaces naturelles telles que les interfaces géologiques, la taille mémoire requise pour représenter de nombreux détails à l'aide d'une surface polynomiale peut être exorbitante.

1.2.2 Approche discrète

En CAO, les logiciels qui n'utilisent pas des surfaces polynomiales comme celles que nous venons de citer, utilisent des surfaces discrètes telles que des grilles à base carrés ou des surfaces triangulées [**?**, **?**, **?**, **?**, **?**, **?**].

L'avantage d'utiliser des surfaces triangulées est que leurs éléments, les triangles, sont toujours plans dans $\mathbb{R}^3$ (ce qui n'est pas forcément le cas pour des grilles) et sont toujours convexes (ce qui ne serait pas le cas en utilisant des polygones quelconques). Tout objet surfacique peut être représenté à l'aide d'une surface triangulée. Ainsi, des horizons et failles, aussi compliqués soient-ils, peuvent être représentés de manière fidèle en prenant compte des topologies comportant

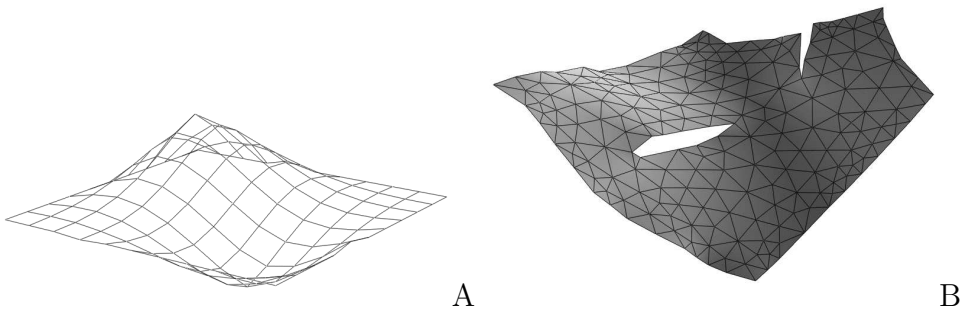


Figure 1.10 *A. Grille : les coordonnées X et Y sont stockées implicitement, seule la coordonnée Z varie. B. Surface triangulée : chaque sommet est stocké explicitement.*

des trous. Un horizon peut aussi être discontinu en étant coupé par de multiples failles. La surface triangulée peut présenter un maillage fin dans les zones où il est nécessaire d'avoir de nombreux détails alors qu'un maillage lâche peut être construit dans les zones plus planes.

Cependant, les surfaces triangulées ne sont pas lisses, c'est-à-dire qu'elles ne possèdent pas de bonnes propriétés mathématiques telle que la continuité G^1 et cela peut être gênant pour des applications qui supposent que les surfaces sont lisses. Segonds [**?**] montre que de telles surfaces posent des problèmes pour les applications de lancer de rayon par exemple. Une surface géométrique simple, telle qu'une sphère, peut être représentée par une équation mathématique simple alors qu'il faut de nombreux triangles pour en obtenir une représentation *fidèle*.

Le fait que les surfaces triangulées ne soient pas lisses et ne puissent représenter efficacement des objets géométriques tels que la sphère n'est pas vraiment déterminant en géologie car les horizons modélisés n'ont pas besoin d'être lisses et nécessiteraient des surfaces de degré très élevé pour représenter toutes les aspérités d'un horizon.

Nous verrons dans le chapitre 2 page 85 une autre approche dite *surfaces de subdivisions* qui fait le lien entre l'approche discrète et l'approche continue. En effet, ces surfaces sont discrètes mais la géométrie de ces surfaces tend vers la géométrie d'une surface continue.

1.2.3 Quelques géomodeleurs

Dans cette section, nous allons détailler quelques modeleurs existants ou ayant existé dans le domaine de la géologie et du pétrole. La plupart de ces géomodeleurs ont un noyau topologique qui n'est pas dévoilé au public. Il n'est donc pas possible de faire une comparaison entre ces logiciels. Cependant, nous pouvons lister leurs activités. Les différents géomodeleurs cités ci-dessous sont classés par ordre d'apparition.

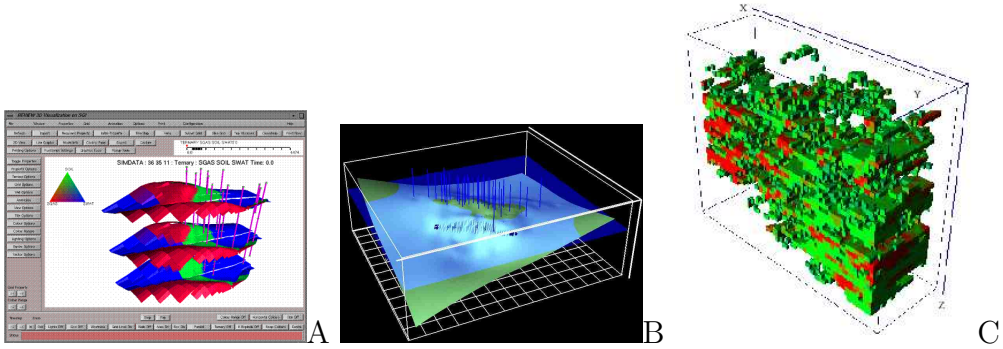


Figure 1.11 *Quelques images écran de Geovisual*

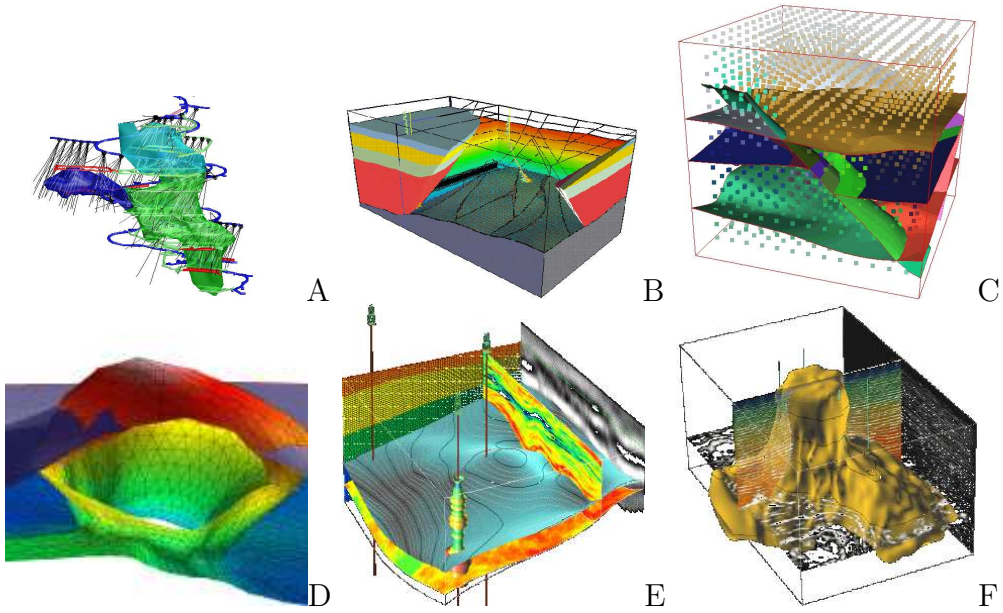


Figure 1.12 *A. Niagara (Surpac) B. Modèle géologique 3D (Xox) C. Modèle de propriétés (Xox) D. Modèle géologique 3D (Gocad) E. Modèle de réservoir (Gocad) F. Dome de sel 3D reconstruit à partir de l'interprétation sismique et des données de puits* Unocal *(Gocad)*

- 1970 : **Earth Vision** de *Dynamic Graphics* sur http://www.dgi.com
  Earth Vision regroupe un ensemble de logiciels basés sur une approche 3D. Il permet de définir une hiérarchie des failles et les impacts qui se répercutent dans la subdivision volumique. Il fournit la possibilité de construire des surfaces à partir de grilles horizontales.

- 19?? : **GeoSec3D** de *Paradigm Geophysical* sur http://www.paradigmgeo.com
  GeoSec et GeoSec3D sont des applications de modélisations géologiques qui peuvent être utilisées pour la construction de coupes géologiques. Ils per-

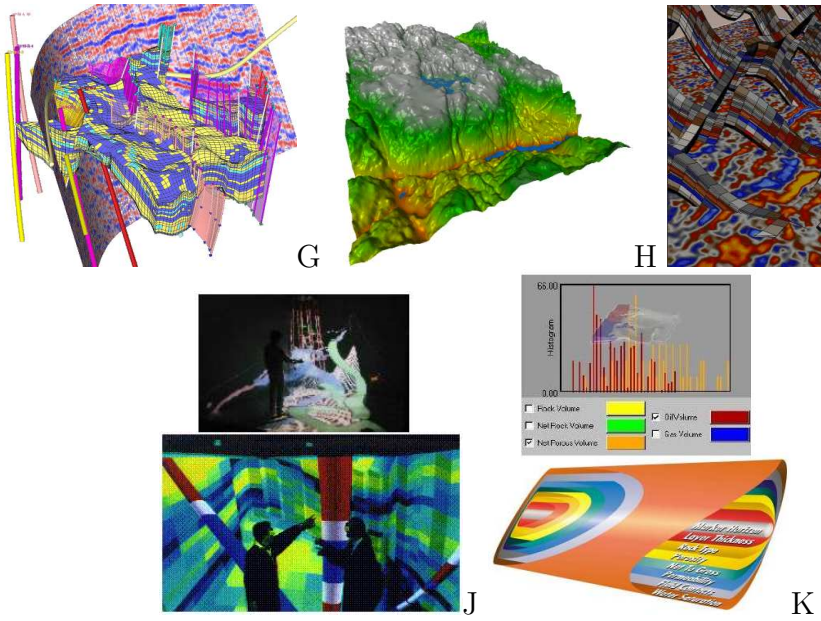


Figure 1.13 *G. Modèle de réservoir (Petrel) H. Surface topographique (Petrel) I. Modèle de réservoir (Petrel) J. Réalité virtuelle dans une CAVE(r)* Arco *et dans un visonarium de* SGI *(T-Surf) K. Analyse des incertitudes sur des volumes rocheux et leur capacité réservoir (T-Surf)*

mettent de les valider et de restaurer ces coupes dans leur état initial. Il est aussi possible de construire des surfaces, de les éditer et de faire des courbes de niveaux, et de les analyser soit sur une carte soit dans un modèle volumique. Paradigm Geophysical couvre également le marché de la sismique jusqu'au réservoir.

- 19?? : **RMS-Geoform** de *Roxar-Smedvig* sur http://www.roxar.com
  RMS-Geoform est un outil de modélisation pour créer le squelette structural d'un réservoir. Il permet de visualiser une panoplie de cartes 2D en 3D avec des fonctionnalités d'édition et manipulation de surfaces. Les horizons sont grillés à partir de données telles que des lignes sismiques, des points, des surfaces existantes et des données de failles. Il fournit des vues 2D et 3D multiples de données simples ou groupées telles que des cubes sismiques, surfaces, données de puits et information culturelle.

- 1982 : **Surpac Vision** de *Surpac Software International* sur http://www.surpac.com/ Surpac Vision est un logiciel qui traite des évaluations de gisements miniers en surface ou en profondeur.

- 1984 : **FrameWork3D** de *Schlumberger* sur http://www.slb.com
  Leur logiciel GeoFrame délivre une panoplie d'applications dans l'exploration

et la production qui sont unifiées..., une carte de base partagée, une zone de travail tridimensionnelle et une base de données de projets. FrameWork3D est une application de modélisation et de tracer, un constructeur automatisé d'architecture de réservoir. La compagnie Schlumberger couvre entièrement le marché de la sismique jusqu'au réservoir. Par contre, ils ont une approche 2,5D, ce qui signifie qu'ils n'ont pas d'objets tridimensionnels à proprement parler : ils utilisent des successions de surfaces 2D.

- 1985 : **3D Move** de *Midland Valley* sur http://www.mve.com
  3D Move permet de construire et éditer des modèles structuraux. Il permet aussi de faire des analyses de failles (calcul de trajectoire de faille, carte de juxtaposition) et il fournit aussi des techniques de restauration 3D. Ses autres fonctionnalités sont les attributs de carte sismique, analyse de fractures and analyse de migration.

- 1985 : **Shapes** de *Xox* sur http://www.xox.com
  Ce logiciel permet de représenter des modèles géologiques de la structure de la terre en rapport avec la sismique et l'exploration pétrolière. Cette compagnie utilisait une topologie robuste. Cependant, à la fin de l'année 2000, elle se retire du domaine du pétrole.

- 1989 : **Gocad** de *ASGA* sur http://www.ensg.inpl-nancy.fr/GOCAD/
  Gocad permet d'utiliser des ensembles de points, des lignes ainsi que des surfaces triangulées plongées en 3D. Ils utilisent aussi de nombreux types de grilles suivant les activités choisies.

- 1996 : **Petrel** de *TechnoGuide* sur http://www.technoguide.com/main.htm
  Petrel est un modeleur géologique basé sur un noyau de grilles 3d. La faille est représentée par un squelette 3D qui est facilement éditable par mécanisme de colonne. Les horizons sont construits dans le squelette 3D en interprétant aussi bien les lignes sismiques que des points ou des cartes grillées. Les horizons et les failles sont des parties d'un modèle de grilles 3D. Il fournit des fonctionnalités telles que calcul de volumes, modélisation de propriétés pétrophysiques et conversion en profondeur. TechnoGuide s'est focalisée sur la modélisation de grilles stratigraphiques.

- 1997 : **Gocad** de *T-Surf*[5] sur http://www.t-surf.com/
  T-Surf est une compagnie de consultation et développement logiciel spécialisée dans la technologie de la modélisation du sous-sol. Fondée en 1997, T-Surf se base sur le logiciel Gocad pour la modélisation structurale jusqu'au réservoir. Nous décrivons plus précisément le logiciel Gocad dans la section 1.2.4.

[5]Le logiciel Gocad cité précédemment est commercialisé depuis 1997 par la société T-Surf.

1.2.4 Le géomodeleur Gocad

Le projet Gocad (**G**eological **O**bjects **C**omputer **A**ided **D**esign) a vu le jour en 1989 par l'équipe informatique de l'école nationale supérieure de géologie (ENSG) à Nancy. Ce projet est actuellement mené en collaboration avec l'institut national polytechnique de lorraine (INPL) et le centre de recherches pétrographiques et géochimiques (CNRS).

Le but de ce projet est de développer une approche informatique pour la modélisation d'objets géologiques. Cette approche est spécifiquement adaptée pour des applications géophysiques, géologiques et réservoir.

Gocad est géré par l'association scientifique pour la géologie et ses applications (ASGA) et dirigé par le professeur Jean-Laurent Mallet.

Ce projet est sponsorisé par un consortium international ouvert pour toutes compagnies pétrolières et instituts de recherche.

Les origines de Gocad

Dans les années 80, il est devenu clair que, malgré le succès de la modélisation de surfaces simples, les systèmes de cartographies automatiques traditionnels ne pourraient pas générer des surfaces complexes et, plus généralement, des volumes géologiques complexes. A cette époque, les logiciels informatiques traditionnels developpés pour l'industrie automobile étaient inutilisables pour traiter des données de géosciences complexes.

Pour cette raison, dans le cadre du projet Gocad, une stratégie radicalement différente, utilisant la modélisation discrète d'objets naturels, a été proposée en 1989. Dans cette approche discrète:

- la géométrie de tout objet est définie par un ensembles finis de noeuds (points) dans $\mathbb{R}^3$
- sa topologie est modélisée par des liens entre les noeuds,
- ses propriétés physiques sont attachées aux noeuds.

Par exemple, si l'objet à modéliser est une surface, les liens entre les noeuds peuvent être arrangés de manière à obtenir une surfaces à facettes triangulaires. De même, il est possible de générer des courbes ou des volumes.

En pratique, une telle discrétisation est sans intérêt sans un outil mathématique puissant permettant d'interpoler les propriétés physiques et la localisation (x, y, z) des noeuds définissant l'objet dans $\mathbb{R}^3$. Pour cette raison, Jean-Laurent Mallet a proposé une méthode appelée "Discrete Smooth Interpolation" (DSI) [**?**] ou interpolateur lisse et discret, qui est au coeur de Gocad. Cet interpolateur a été spécifiquement conçu pour modéliser des objets naturels en prenant en compte plusieurs types de données complexes et plus ou moins précises. Nous ne citerons que quelques unes de ces contraintes :

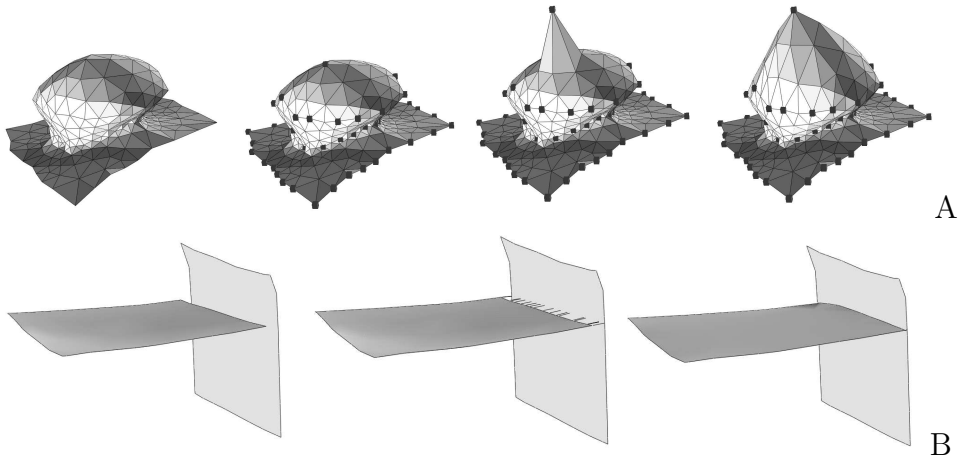


Figure 1.14 *(A) Cette surface triangulée représente un corps géologique : un dôme de sel. Les noeuds symbolisés par des cubes foncés sont* ancrés *: les coordonnées X, Y, Z de ces noeuds ne bougeront pas lors de l'interpolation. Un noeud est déplacé manuellement. Les noeuds vont maintenant être interpolés. Le résultat : les noeuds ont bougé en fonction des noeuds ancrés. (B) A cause de la précision du pointé sismique, cet horizon ne touche pas la faille qui le coupe. Des constraintes horizon/faille sont installées sur certains noeuds du bord de manière à prolonger l'horizon vers la faille. Après interpolation, le contact est quasi parfait.*

- points fixes : L'exemple de la figure 1.14-A montre une utilisation possible de DSI. La surface triangulée possède des contraintes *fortes* qui empèchent certains noeuds de bouger. Les noeuds *libres* vont se déplacer pour lisser la surface. D'autres contraintes ont été ajoutées depuis la naissance de DSI.

- contrainte horizon/faille : lors du pointé sismique, la frontière de l'horizon au contact de la faille est difficile à repérer. Il arrive souvent que le géologue arrête son pointé avant de rencontrer la faille. Lors de la modélisation, on obtient un horizon qui ne touche pas la faille alors que ceci devrait être le cas. Une contrainte horizon/faille va permettre aux points du bord de l'horizon de se mettre en contact le long de la faille (voir figure 1.14-A).

- contrainte points : certains points d'un horizon sont plus ou moins connus. La contrainte points permet à la surface de passer par un maximum de ces points. Une illustration de cette contrainte est donnée sur la figure 1.15-A.

- contrainte verticale : seules les coordonnées Z des noeuds contraints peuvent variées. La figure 1.15-B nous montre un exemple d'utilisation de cette contrainte.

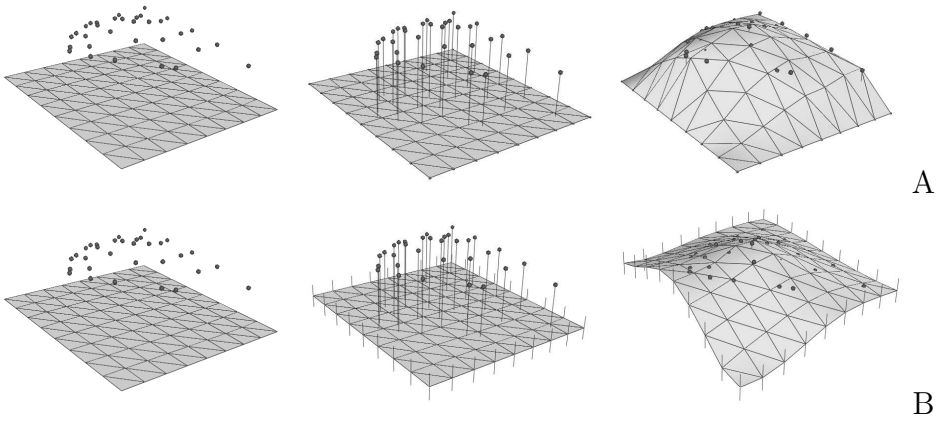


Figure 1.15 *(A) Les contraintes points : une surface triangulée et un ensemble de points à respecter. La surface doit respecter au mieux l'ensemble des points (ici, les points du bords sont fixes). La figure du milieu montre les contraintes points (traits entre les points et la surface). Avec cette première méthode, on obtient une surface très bombée. (B) Les contraintes verticales : avec la même surface et le même ensemble de points, on autorise aux points du bord à pouvoir se déplacer selon l'axe Z. La figure du milieu nous montre cette nouvelle contrainte (traits de part et d'autres des points du bord. Avec cette méthode, on obtient une surface plus lisse.*

Le consortium

Depuis 1990, l'évolution de Gocad s'est faite en trois phases.

Pendant la première phase de 1989 à 1994, le seul objectif était d'explorer les possibilités qu'offrait une telle approche discrète associée à la méthode DSI. Par ailleurs, le groupe de recherche de Gocad produisait un logiciel pour démontrer l'efficacité des méthodes utilisées.

Les sponsors ont commencé à utiliser le prototype Gocad pour leur propre compte. Rapidement, des problèmes de maintenance se sont fait sentir. Pour cette raison, en septembre 1994, la phase 2 commença. La grande différence avec la phase 1, était que, en plus du groupe de recherche Gocad, une équipe de maintenance et industrialisation a été créée.

Grâce à la création de l'entité de maintenance, plusieurs compagnies se sont joints au consortium et ont utilisé de plus en plus le logiciel Gocad. Ce succès a produit une demande de plus en plus forte d'une coopération industrielle avec les sponsors. La structure du consortium ne pouvait pas faire face à une telle demande. La phase 3 a été lancée en septembre 1998. Dans cette phase, l'entité de maintenance devient une compagnie commerciale indépendante nommée T-Surf. T-Surf reçoit de l'ASGA le droit de maintenir, developper et commercialiser la version officielle du logiciel Gocad. Cette compagnie devient membre du consortium et incorpore les recherches du laboratoire.

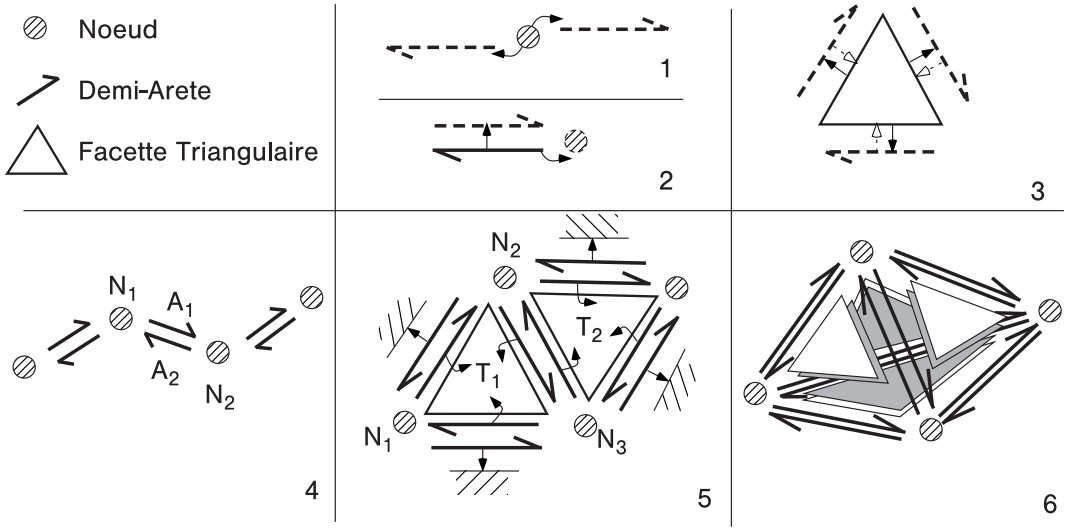


Figure 1.16 *La structure de Weiler implantée dans Gocad. (1) Un noeud connaît la liste des demi-arêtes qui l'entoure. (2) Une demi-arête connaît le noeud d'origine et sa demi-arête associée. (3) Une facette triangulaire possède trois demi-arêtes. Les demi-arêtes des surfaces possède un pointeur sur la facette triangulaire qu'elles constituent. (4) Une courbe polygonale. (5) Une surface triangulée : les demi-arêtes qui se trouvent au bord "pointent" dans le* vide. *(6) Un tétraèdre : chaque facette triangulaire est doublée.*

La structure topologique

La structure topologique de Gocad utilise la topologie de Weiler [**?**]. Cette topologie va permettre de construire des ensembles de points, des lignes polygonales, des surfaces triangulées ou des volumes tétraédrisés.

La brique de base est le *noeud* : il stocke à la fois sa géométrie ainsi que d'éventuelles propriétés physiques. Le noeud connaît toutes les arêtes qui l'entourent. L'arête est décomposée en fait en deux *demi-arêtes* orientées. Chaque demi-arête connaît son origine et sa demi-arête associée. Elles sont spécialisées pour chaque objet : il existe ainsi une demi-arête pour les lignes, les surfaces et les volumes. Dans l'exemple 4 de la figure 1.16, pour accéder aux noeuds d'une arête A_1, on récupère directement son origine (N_1) et on récupère l'origine N_2 de la demi-arête associée A_2. Enfin, la *facette triangulaire* permet de construire des surfaces triangulées et est aussi utilisée pour composer des tétraèdres. D'après la figure 1.16-5, pour connaître les trois noeuds du triangle T_1, il faut passer par les trois demi-arêtes rattachées à ce dernier. Ainsi, en récupérant les origines de ces demi-arêtes, on peut accéder aux noeuds N_1, N_2 et N_3. De même, pour accéder aux triangles voisins de T_1, il faut passer par les demi-arêtes du triangle. Chaque demi-arête connaît la demi-arête associée et chaque demi-arête associée possède un pointeur sur un triangle ou sur le *vide*. Dans ce dernier cas, cela signifie

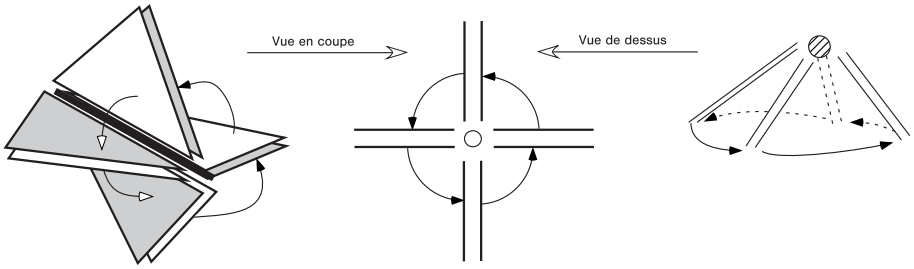


Figure 1.17 *Structure d'arête radiale. A gauche, quatre triangles se rejoignent le long d'une arête. A droite, quatre segments se rejoignent en un noeud.*

que le triangle considéré se trouve au bord de la surface. Dans notre exemple, l'arête $[N_1, N_3]$ se trouve au bord tandis que par l'arête $[N_2, N_3]$ il est possible d'atteindre le triangle T_2. A partir de cette structure, il est également possible de construire des tétraèdres. Chaque facette triangulaire d'un tétraèdre connaît les trois demi-arêtes qui la constitue et possède un lien vers la facette triangulaire associée. Cette dernière facette correspond à la facette d'un tétraèdre voisin. La figure 1.16-6 montre un exemple d'un tétraèdre isolé : chacune de ces quatre facettes est liée avec une facette dépourvue de tétraèdre.

Cependant, si l'on veut rattacher plusieurs triangles d'une surface le long d'une arête, il faut mettre en place une structure supplémentaire. Weiler a introduit cette structure sous le nom de RadialEdge ou arête radiale dans [**?**] (voir aussi la figure 1.17). Cette structure permet de tourner autour des triangles qui possèdent une arête en commun ou autour des segments (RadialNode) qui possèdent un noeud en commun. Grâce à cette structure, il est possible de construire les couches du sous-sol. Nous verrons plus en détail dans la section 1.4.4 comment est utilisée cette structure pour construire des modèles géologiques.

Après cette introduction au monde de la géologie, nous allons présenter la modélisation à base *topologique*.

1.3 Modélisation géométrique à base topologique

La topologie est une branche relativement récente des mathématiques, qui a fait son apparition au début des années 1900 avec Henri Poincaré.

Elle se compose de plusieurs sous-domaines parmi lesquels on distingue la topologie *combinatoire* qui permet de déterminer certaines propriétés et théorèmes concernant la manière dont les éléments des objets discrétisés (sommets, arêtes, faces, ...) sont connectés entre eux.

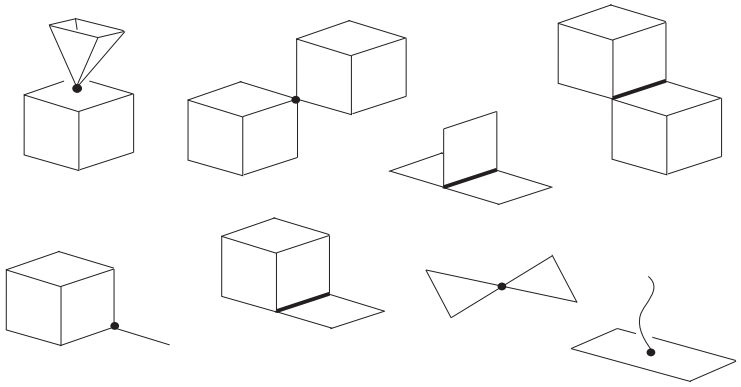


Figure 1.18 *Exemples de non-variétés.*

1.3.1 Quelques notions topologiques

Pour la suite de ce mémoire, nous définissons à présent quelques notions topologiques que nous utiliserons ultérieurement.

Notion de variété

Un objet est dit n-variété si le voisinage de tout point lui appartenant est homéomorphe[6] à une sphère de dimension n. Un objet non-variété est un objet qui ne respecte pas la précédente définition.

Sur la figure 1.18, les objets ne sont pas variétés à cause des zones dessinées en gras.

La réalisation d'un objet non-variété de dimension 2 implique, dans la plupart des modèles existants, un grand nombre de types de structures (onze pour Weiler). Maintenir en cohérence toutes ces structures par des algorithmes qui les modifient s'avère fastidieux et difficile.

La structure utilisée par Lévy [**?**] nous permettra d'obtenir des objets non-variétés d'une manière plus simple. Nous expliquerons cette structure dans la section 1.4.2 page 39.

Notion de n-cellule

Nous nous intéressons à des objets pouvant être décomposés en un nombre fini d'objets élémentaires appelés *cellules*. Ces cellules sont définies de la manière suivante :

[6]On dit que deux espaces topologiques E et F sont homéomorphes s'il existe une bijection f entre E et F qui à tout ouvert de E fait correspondre un ouvert F, et réciproquement. En géométrie, l'homéomorphie de deux figures correspond à la notion intuitive de *transformation par déformation continue*, par exemple entre une sphère et un cube.

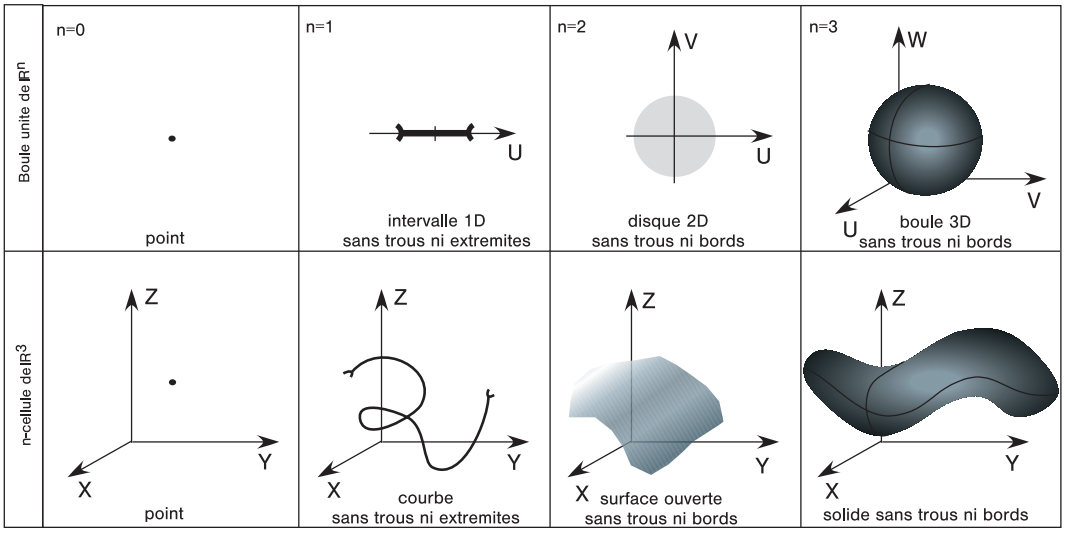


Figure 1.19 *Les n-cellules de $\mathbb{R}^3$ sont homéomorphes à des n-boules.*

Une n-cellule est définie comme un ensemble de points pouvant être mis en correspondance avec la boule unité de $\mathbb{R}^n$ par une fonction continue ou plus précisément, par un homéomorphisme.

Par exemple, comme nous pouvons le voir figure 1.19 :

- une 0-cellule appelée sommet et correspondant à un point isolé de $\mathbb{R}^3$.
- une 1-cellule appelée arête et correspondant à une courbe simplement connexe (c'est-à-dire en un seul morceau) de $\mathbb{R}^3$ sans extrémité.
- une 2-cellule appelée face et correspondant à une surface ouverte simplement connexe (c'est-à-dire sans trou) de $\mathbb{R}^3$ sans bord.
- une 3-cellule appelée volume et correspondant à un solide ouvert simplement connexe (c'est-à-dire sans trou) de $\mathbb{R}^3$ sans bord.

Ces cellules vont nous permettre de définir une structure de données pour représenter un objet discrétisé. Nous nous intéresserons également aux relations de voisinages entre ces cellules.

1.3.2 Différents modèles topologiques

Avant de parler de la structure que nous utilisons et d'expliquer l'apport que nous y avons fait, nous définissons brièvement les trois principales sortes de modèles existants et nous décrivons leurs avantages et inconvénients. Pour plus de détails, le lecteur pourra se référer à [?, ?, ?, ?, ?].

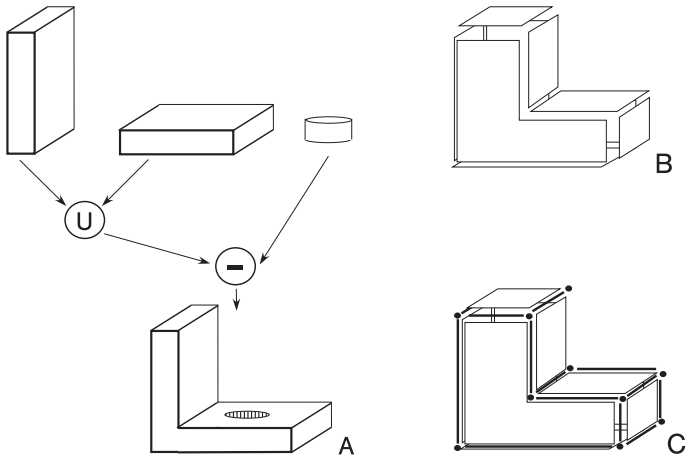


Figure 1.20 *A. Représentation booléennes B. Représentation par frontières C. Représentation par* cellules *(sommets, arêtes, faces)*

Modèles booléens

La première catégorie est appelée Modèles Booléens [**?**, **?**, **?**, **?**, **?**, **?**, **?**] ou **CSG** (Constructive Solid Geometry). Elle est principalement utilisée par les industries mécaniques. Elle permet par exemple de piloter des machines outils pour sculpter des objets en trois dimensions. Les modèles sont obtenus en appliquant des opérations booléennes (union, intersection et différence) entre des objets simples (cube, sphère, cylindre, ...). La figure 1.20-A représente un objet construit avec des opérations booléennes : à partir de deux parallélépipèdes et d'un cylindre, les opérations d'union et différence sont combinées pour obtenir l'objet final. Pour certaines applications, les opérations booléennes définies dans ces modèles ne sont ni précises ni adaptées. Par exemple en géologie, il n'est pas pratique de construire un horizon ou une faille par cette approche. De plus, la modification interactive de la forme d'un objet demande une réévaluation coûteuse de l'arbre booléen (le lecteur souhaitant plus de détails sur ce sujet se réfèrera à l'article de Voelcker et Requicha dans [**?**]).

Représentation par frontières

La seconde catégorie représente les objets par leurs frontières et est nommée *Boundary Representation* ou B-Rep en abrégé [**?**, **?**, **?**, **?**, **?**]. La représentation à base d'*arêtes ailées* [**?**] et la structure des *quad-edge* de Guibas et Stolfi [**?**] sont des exemples de B-Rep. Pour l'arête ailée, l'arête est la structure de base. Chaque arête connaît les deux faces qui l'entourent ainsi que ses deux extrémités. Pour parcourir la structure, l'arête connaît les deux arêtes qui se trouvent connectées à sa gauche et les deux de droite (voir figure 1.24). Weiler [**?**] a amelioré cette

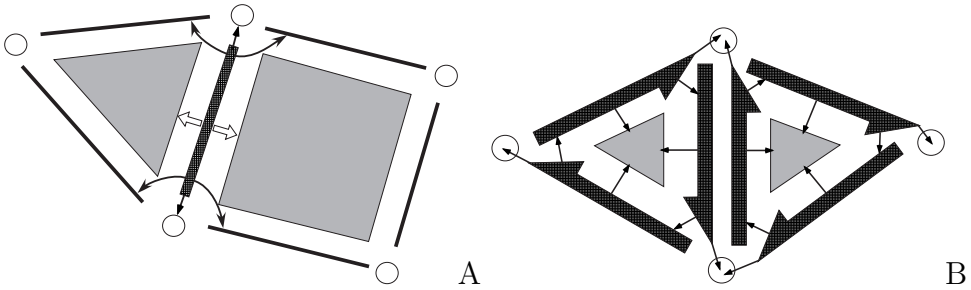


Figure 1.21 *A. Arête ailée (Winged edge) : chaque arête connaît deux faces (flèches blanches) ses deux extrémités (petites flèches) et quatre arêtes (flèches courbes). B. Demi-arête : chaque demi-arête connaît une face, un sommet, la demie-arête suivante et la demi-arête voisine.*

structure en utilisant des demi-arêtes. Cette structure, largement utilisée, est très connue [**?**, **?**].

Le principal inconvénient de cette catégorie de modèle est de ne pas avoir une approche pratique pour représenter les objets non-variétés de dimension 1 ou 2. Les solutions proposées se fondent plus sur un ensemble de structures de données informatiques complexes que sur une théorie robuste. Ainsi, Weiler a introduit la représentation d'arêtes radiales dans [**?**] afin de pouvoir représenter des objets non-variétés 2d (voir aussi la figure 1.17). Mais cette représentation ne compte pas moins de onze structures différentes. Malgré un formalisme assez rigoureux, cette structure rend les algorithmes agissant difficiles à écrire et à maintenir. De plus, en utilisant ces modèles, il est difficile d'avoir une décomposition d'un objet en cellules arbitraires (sommets, arêtes, polygones, polyèdres, ...) car cela requiert l'ajout de nouveaux éléments.

Pour s'affranchir de ces problèmes, Brisson [**?**] a défini une structure mathématique dite *tuples de cellules* (cell-tuples en anglais). Ce modèle rentre dans la dernière catégorie que nous developpons dans le paragraphe suivant.

Modèles cellulaires

La dernière catégorie, les modèles cellulaires, est fondée sur une décomposition des objets en ensembles de cellules. Une théorie solide, connue sous le nom de topologie algébrique, assure la validité de cette décomposition. Lienhardt [**?**] a complété l'étude de Brisson et a défini une structure purement combinatoire appelée G-Carte (G-Map en anglais) qui signifie carte généralisée.

Les relations mathématiques entre les cellules qui composent l'objet sont spécifiées. Ensuite, ces relations sont directement traduites en structures de données informatiques. De cette manière, des preuves mathématiques peuvent être appliquées aux algorithmes agissant sur de telles structures [**?**]. Une autre conséquence de cette structure mathématique est que les implantations de cette

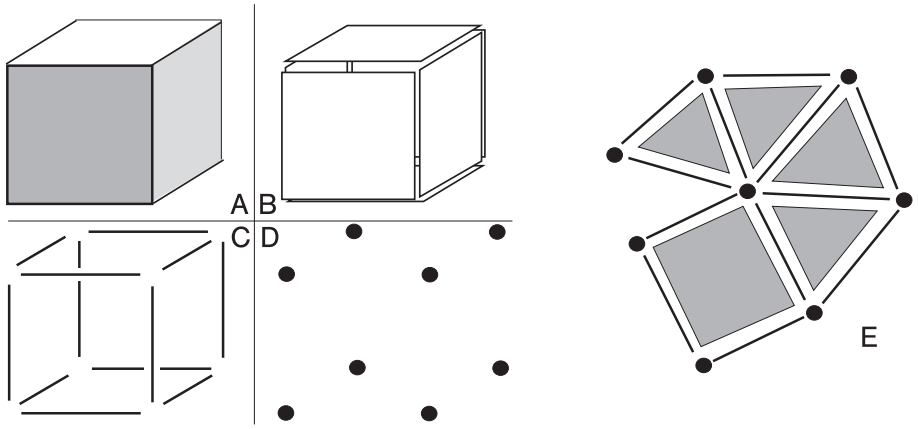


Figure 1.22 *Décomposition cellulaire d'un cube : A. Polyèdre. B. Faces. C. Arêtes. D. Sommets. E. Modèle cellulaire d'une surface décomposée en 8 sommets, 12 arêtes et 5 faces.*

théorie consistent en un noyau topologique petit et robuste. Borianne et Bertrand [**?**, **?**] ont été les premiers à créer un modeleur avec des G-Cartes.

Une présentation plus complète des modèles cellulaires se trouve dans [**?**, **?**, **?**, **?**].

Les modèles les plus récents permettent la représentation de subdivisions d'espaces de dimension quelconque, variétés [**?**, **?**, **?**, **?**, **?**, **?**] ou non, à bords complets ou incomplets, homogènes ou hétérogènes [**?**, **?**, **?**, **?**, **?**].

En géologie, la modélisation des objets du sous-sol ne demande pas l'utilisation de modèles aussi complets. En effet, le monde réel est "construit" uniquement à partir d'objets variétés 3D. Cependant, en informatique, nous construirons des objets non-variétés 2D à partir de surfaces telles que des horizons ou des failles. Nous verrons en section 1.4.4 comment obtenir des objets variétés 3D à partir de surfaces.

1.3.3 Classement des modèles

Depuis plusieurs dizaines d'années, de nombreux modèles topologiques ont été réalisés dans le domaine de la représentation et la manipulation d'objets géométriques en informatique.

Nous avons classé ces modèles selon cinq critères.

Dimension (2d-3d)

Certains modèles utilisent des séries de coupes 2D (médecine, industrie pétrolière). L'intérêt d'utiliser de tels modèles était surtout dû au fait que les ordinateurs de l'époque n'étaient pas capables d'afficher des modèles tridimensionnels car la mémoire de la machine n'était pas suffisamment grande. En géologie, avant

d'utiliser des ordinateurs, il n'y avait pas moyen de construire un modèle 3D, les géologues avaient des sections de leur modèle sur des feuilles de papier. Et c'est pour cela que certains modeleurs (et notamment en géologie) continuent de travailler en 2D, tandis que d'autres sont passés à la 3D. En médecine, il est maintenant possible de visualiser en 3D tous les vaisseaux sanguins d'une partie du corps : le médecin n'est plus obligé d'imaginer son modèle à partir de séries de coupes 2D.

Il est à noter qu'il faut faire une distinction entre dimension topologique et dimension géométrique d'un objet. Par exemple, si l'on considère une courbe, sa topologie est de dimension 1, Mais cette courbe peut être définie dans $\mathbb{R}^n$. La dimension géométrique de la courbe est alors n.

Représentation simpliciale ou cellulaire

Les modèles sont définis par des éléments qui sont :

- soient des simplexes de dimension i ou i-simplexes (0-simplexe pour un noeud, 1-simplexe pour une arête, 2-simplexe pour un triangle, 3-simplexe pour un tétraèdre, ...). [?, ?, ?, ?, ?, ?]
- soient des éléments plus généraux, des cellules ou i-cellules. (0-cellule pour un noeud, 1-cellule pour une arête, 2-cellule pour un polygone, 3-cellule pour un polyèdre, ...). [?, ?, ?, ?]

Ces éléments sont connectés les uns aux autres impliquant des relations d'incidences et d'adjacences entre cellules.

Variété ou non-variété

La notion de variété a déjà été expliquée en section 1.3.1 page 30 (voir aussi figure 1.18).

Représenter des objets variétés permet d'utiliser une structure plus légère mais la plupart des structures ne sont pas génériques et ne peuvent pas être étendues facilement à des objets de dimension n. Par exemple, une structure utilisée pour représenter des surfaces est difficilement réutilisable avec peu de modifications pour représenter des volumes. Les G-Cartes permettent de présenter une classe d'objets appelés des quasi-variétés[7] pour toute dimension.

Les structures nécessaires pour représenter des objets non-variétés sont souvent complexes et plus lourdes : cela est dû au fait que plusieurs n-cellules peuvent être rattachées à une même $n-1$ cellule.

Lévy [?] présente une évolution des G-Cartes en H-G-Cartes (H pour hiérarchique) qui permet de représenter des objets non-variétés. Dans la suite de ce chapitre,

[7] Un objet de dimension n est quasi-variété si toute $(n-1)$-cellule est au bord d'au plus deux n-cellules.

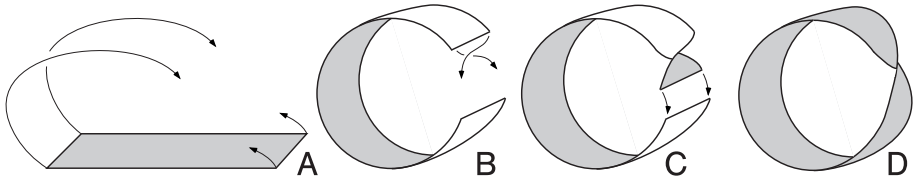


Figure 1.23 *Comment construire un ruban de Moebius (surface non orientable) : A. Prendre une bande de papier et former un cylindre avec. B. Tourner une extrémité du papier de 180 degrés. C. Coller les deux extrémités. D. Au niveau de la jonction, le recto se trouve du même côté que le verso. En* prolongeant *la couleur grise du recto, tout l'objet devient gris. En fait, cet objet n'a ni recto, ni verso.*

nous utiliserons les H-G-Cartes pour construire des modèles géologiques. La section 1.4.2 page 39 donne plus de détails sur cette structure.

Orientabilité

Un objet de dimension i ou i-objet est orientable s'il possède deux faces de dimension i ou i-faces différentes.

En général, on cherche à modéliser des objets orientables car les objets qui nous entourent sont orientables. Cependant, il est très simple d'obtenir un objet non-orientable en informatique. La figure 1.23 représentant un ruban de Moebius nous permet de comprendre visuellement ce qu'est une surface non-orientable. L'idée est que si l'on veut peindre d'une couleur grise la face que l'on considère être le recto, du moins localement, nous allons aussi peindre celle qui correspondrait au verso. En fait, la notion de recto et verso n'a aucun sens pour un objet non orientable. Dans le monde réel, un ruban de Moebius n'existe pas car, aussi fine soit la feuille de papier que l'on utilise pour construire le ruban, il faut toujours considérer son épaisseur : nous sommes donc en face d'un objet volumique et non surfacique. Une feuille de papier qui est tournée de manière à former un ruban de Moebius est en réalité un tore 3D.

Dans "le géométricon" de J.P. Petit [**?**] page 54, Anselme Lanturlu tente en vain de planter des clous dans un ruban de Moebius afin de délimiter l'intérieur de l'extérieur mais, l'opération se solde par un échec étant donné que cette bande n'a ni intérieur, ni extérieur.

Hiérarchique ou non hiérarchique

Les structures hiérarchiques ont déjà été utilisées dans différents aspects de la modélisation géométrique. Par exemple, De Floriani et Falcidieno [**?**] représentent des objets industriels en CAD avec un arbre de structures topologiques : chaque noeud de l'arbre décrit une caractéristique de l'objet tandis que les branches représentent la manière dont ces caractéristiques sont connectées les unes aux

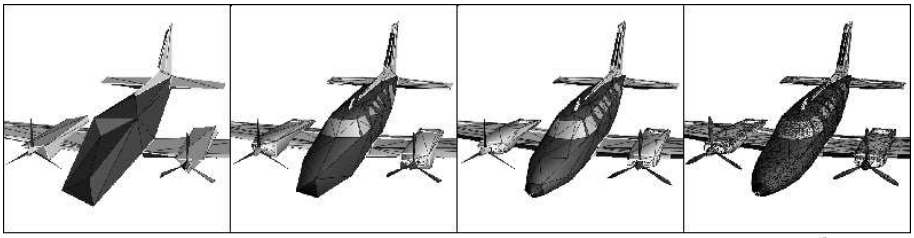


Figure 1.24 *Représentation d'un avion avec une hiérarchie de maillage (Images de Hugues Hoppe obtenues à partir de http://www.research.microsoft.com/~hoppe/) De gauche à droite : l'objet possède 150, 500, 1000 et 13546 faces.*

autres. De Floriani et Puppo [**?**] proposent une structure pour des surfaces triangulées multi-résolution : chaque triangle de la surface peut être représenté par une triangulation plus fine et plusieurs niveaux de détails peuvent ainsi être obtenus. Dans ses travaux, Hoppe [**?**] manipule des structures hiérarchiques à base de triangles. A partir d'un objet possédant un maillage détaillé, le principe est de réduire le nombre de triangles au fur et à mesure que l'objet s'éloigne du point de vue. Cette méthode est utilisée par exemple dans les jeux vidéos : les objets sont très détaillés lorsqu'ils se trouvent très proches de la caméra et ceux qui se trouvent au fond de la scène sont beaucoup moins détaillés. Cela permet d'afficher beaucoup d'objets sans pour autant afficher beaucoup de triangles.

Le modèle que nous utilisons

Pour traiter des problèmes d'ordre topologique, nous avons vu que les modèles classiques utilisent des structures de données complexes. En fait, ces structures de données sont très interdépendantes et donc difficiles à gérer. Il en résulte une faible robustesse théorique et une complexité accrue pour les structures de données de topologies non-variétés.

Notre définition d'un modèle fondé sur la représentation topologique connue sous le nom de G-Carte [**?**] fait une claire distinction entre la connectivité et la subdivision des objets (voir explication détaillée page 39). La structure implantant ces deux sortes de représentations sont respectivement appelées *Cadre* et *Toile* (voir figure 1.27-B). Le cadre est une simple représentation du bord de l'objet, tandis que la toile est attachée à un cadre et représente la décomposition cellulaire en sommets, côtés, polygones, ... de l'objet.

La séparation des deux structures permet de construire des modèles où les interfaces entre les couches géologiques peuvent être des surfaces de différents types, comme un maillage polygonal ou une surface paramétrique. Un autre avantage important de cette structure est que les algorithmes peuvent être prouvés. En effet, la structure des G-Cartes est non seulement une structure de données informatique mais aussi une structure mathématique sur laquelle des preuves

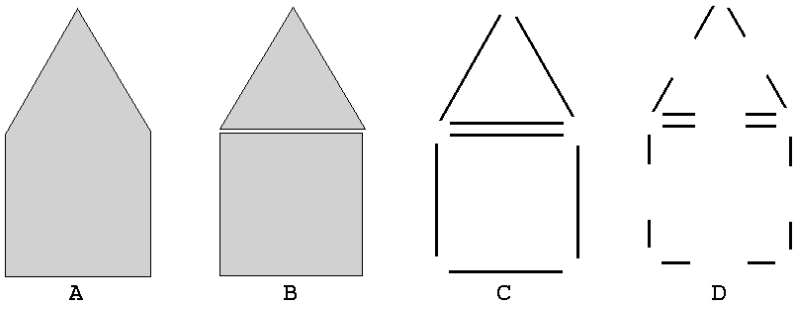


Figure 1.25 *Analyse cellulaire. A. La surface d'étude représente une maison. B. Nous imposons que cette maison soit construite à partir de deux polygones (2-cellules) : un carré et un triangle. C. Chaque polygone est constitué d'arêtes (1-cellules). D. Chaque arête a deux sommets (0-cellules). Ces sommets sont représentés par des bouts d'arêtes de manière à visualiser la forme de l'objet de départ.*

mathématiques peuvent être effectuées.

Pour récapituler, le modèle que nous allons utiliser permet de construire des objets de dimension n non-variétés composés de cellules arbitraires homéomorphes à des n-boules. Il sera possible de détecter si un objet est orientable ou non.

Nous allons maintenant définir ce qu'est une G-Carte puis une H-G-Carte et pourquoi nous utilisons ce noyau topologique.

1.4 Noyau topologique

Dans cette section, nous introduisons la théorie des cartes généralisées ou G-Cartes qui, comme son nom l'indique généralisent les cartes [**?**, **?**, **?**, **?**, **?**]. Pour plus de détails, le lecteur pourra consulter [**?**, **?**, **?**].

1.4.1 Définition des G-Cartes

Nous donnons une approche intuitive des G-Cartes avant de les définir mathématiquement.

Considérons l'objet de la figure 1.25-A représentant un maison en 2D. Décomposons cet objet en deux polygones : un triangle formant le toit de la maison et un carré représentant sa façade (figure 1.25-B). Ces polygones sont eux-mêmes composés respectivement de trois et quatre arêtes (figure 1.25-C). Finalement, chacune des arêtes se décompose en deux extrémités (figure 1.25-D). Chacune de ces extrémités correspond à un *brin*. Ainsi, un brin correspond à un sommet *vu* d'un côté, lui même *vu* d'un polygone.

A l'inverse, reconstruisons les relations entre les n-cellules (sommets, arêtes, polygones, ...) composant les objets en connectant les brins entre eux (figure

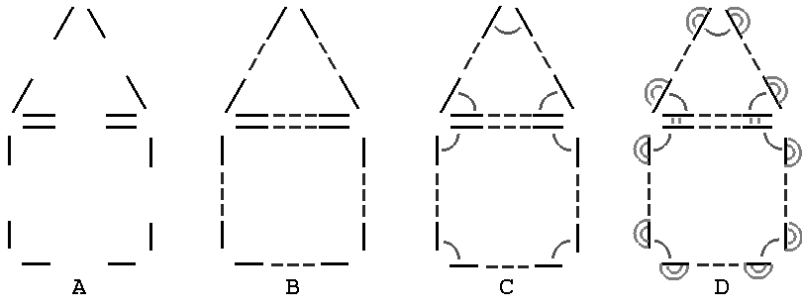


Figure 1.26 *Synthèse cellulaire. A. Les quatorze éléments qui constituent cet objet sont appelés des brins et représentent les extrémités des arêtes de la surface. B. En connectant deux à deux les brins par un lien α_0, nous obtenons des 1-cellules. C. En connectant les 1-cellules le long des brins formant leur bord par des liens α_1, nous obtenons des 2-cellules. D. En connectant les 2-cellules le long des brins composant leur bord par des liens α_2, nous obtenons notre objet d'étude : la maison.*

1.26-A). Chaque couple de brins qui forment les deux bouts d'une arête sera connecté par des liens de dimension 0 appelés liens α_0 (figure 1.26-B). Les arêtes des polygones sont connectées entre elles par des liens de dimension 1 appelés liens α_1 (figure 1.26-C). Enfin, les deux polygones sont reliés entre eux par des liens de dimension 2 appelés liens α_2 (figure 1.26-D).

Nous pouvons remarquer que ces liens connectent uniquement deux brins entre eux. En fait, les liens α_0, α_1 et α_2 sont définis mathématiquement comme des involutions agissant sur les brins. C'est-à-dire que pour tout brin b appartenant à l'ensemble des brins de l'objet, $\alpha_i \circ \alpha_i(\mathrm{b}) = \mathrm{b}$.

Brins et liens α sont les deux éléments nécessaires pour former une G-Carte. D'une façon plus formelle, Lienhardt [**?**] définit une n-G-Carte comme suit :

Soient $n \geq -1$ un entier donné et $G = (B, \alpha_0, \alpha_1, \ldots, \alpha_n)$ un $(n+2)$-uplet tel que B est un ensemble fini de brins et α_i des involutions sur B. Par définition, G est une G-Carte de dimension n, ou plus simplement une n-G-Carte, si, et seulement si, la condition suivante est respectée :

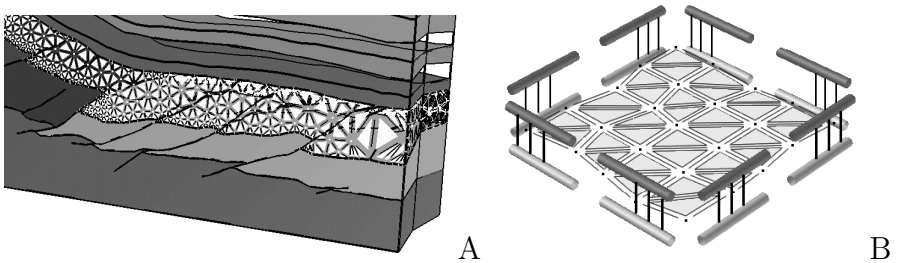
- $\alpha_i \circ \alpha_j$ est une involution $\forall 0 \leq i < i+2 \leq j \leq n$

Après avoir introduit la définition des G-Cartes, nous allons définir la structure que nous allons utiliser pour représenter des modèles géologiques du sous-sol.

1.4.2 Définition des H-G-Cartes

Un objet peut être considéré sous trois différents aspects (voir figure 1.27-A) :

- La *connectivité* correspondant à la façon dont l'objet est connecté aux autres objets du modèle,



A B

Figure 1.27 *(A) Les régions du modèle sont connectées le long des lignes noires. Chaque région est subdivisée par un ensemble de cellules (ici sommets, arêtes, triangles). (B) La toile (discrétisation) et le cadre (tubes) d'une surface.*

- La *subdivision* décomposant l'objet en composantes (ou cellules),
- Le *plongement* représentant la géométrie et les propriétés attachées aux objets.

Les méthodes classiques ne font pas une distinction claire entre ces trois points. Weiler [**?**, **?**] a été l'un des premiers à mentionner l'importance de séparer la géométrie (à savoir le plongement) de la combinatoire (connectivité et subdivision). En revanche, il n'établit pas la distinction entre subdivision et connectivité, ce qui impose des structures plus lourdes à implanter et maintenir comme nous l'avions déjà mentionné auparavant.

L'approche proposée sépare complètement ces trois points. Un objet est défini par une *toile* représentant sa subdivision et par un *cadre* représentant sa connectivité. La toile d'un objet représente les relations entre ses cellules (sommets, arêtes, polygones, ...). Cette toile est une G-Carte de la dimension de l'objet (1-G-Carte pour une courbe, 2-G-Carte pour une surface, ...). Le cadre quant à lui permet de connecter les objets entre eux et est représenté par une (n+1)-G-Carte. Il est composé de deux n-cellules reliées entre elles par des liens α_{n+1}. Ces deux cellules se font face et englobent l'objet auquel elles sont rattachées (voir figure 1.27-B). Nous verrons plus loin comment connecter les brins du cadre pour former des objets non-variétés.

L'avantage principal de cette séparation est la possibilité de plonger un cadre dans différentes sortes d'objets. Ainsi, une surface peut être représentée de différentes manières. Par exemple, la surface peut être une surface triangulée, paramétrique ou avoir une autre représentation (figure 1.28).

De plus, comme il a été dit précédemment, un cadre est une G-Carte. D'un point de vue pratique, les mêmes fonctions informatiques peuvent être utilisées pour les deux aspects, ce qui réduit la taille du code de manière notable. En d'autres termes, le code source de base utilisé pour le cadre **est le même** que celui utilisé pour la toile. Ainsi, de nombreuses opérations de base sont utilisées par les deux représentations telles que :

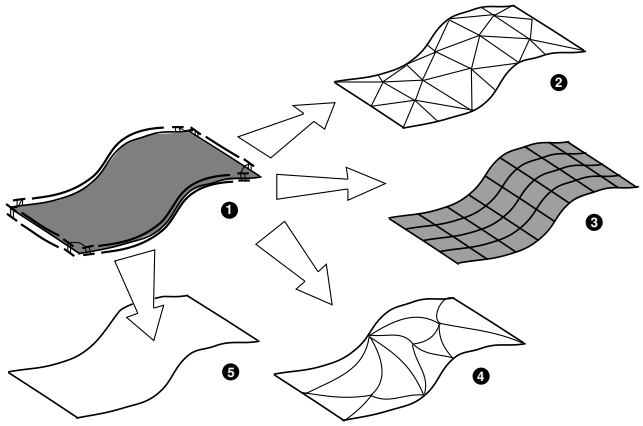


Figure 1.28 *La surface (1) peut être plongée de différentes manières : (2) surface triangulée, (3) surface paramétrique, (4) surface de bézier, (5) autres représentations.*

- parcours de cellules (sommets, arêtes, polygones, ...)
- parcours de composantes connexes (courbes, surfaces, régions, ...),
- modifications topologiques de base (insertion/destruction de cellules, ...),
- vérification de cohérence (orientabilité, ...).

Les structures cadre et toile correspondent donc à deux G-Cartes imbriquées l'une dans l'autre. Cette imbrication de G-Cartes se nomme H-G-Carte (carte généralisée hiérarchique) et a été définie dans [**?**]. Dans son mémoire, Lévy décrit cette structure et explique comment *assembler* des H-G-Cartes entre elles. Nous allons utiliser cette structure pour construire des modèles complets.

1.4.3 Structure d'un modèle

Pour construire un n-modèle, nous disposons d'un ensemble d'objets de dimension $n-1$ servant de murs-diviseurs (en géologie, un modèle volumique sera construit à partir d'un ensemble de surfaces). Les bords des murs sont collés les uns aux autres, ce qui permet de subdiviser l'espace nD en objets de dimension n que nous appelons *régions*. L'exemple de modèle donné par la figure 1.29 est un 2-modèle ; les murs-diviseurs sont des lignes et les régions sont représentées par des polygones. Une région est connexe mais peut cependant contenir des trous (exemple : la région R_2 englobe la région R_5). Par ailleurs, il existe une région, ici R_0, qui englobe toutes les autres régions : on nomme également cette région *univers*.

Coller des murs le long de leurs bords peut engendrer des configurations non-variétés. Dès lors que plus de deux murs diviseurs sont assemblés le long d'un

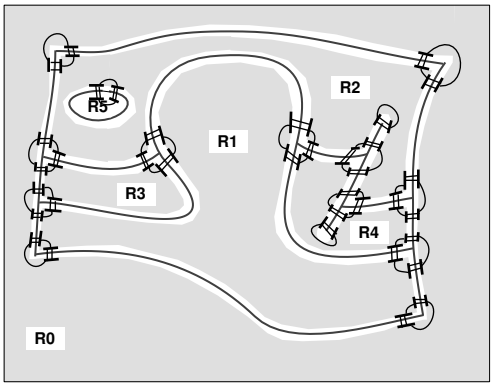


Figure 1.29 *Différentes sortes de régions. La région R_0 correspond à l'espace extérieur du modèle. La région R_2 possède un bord interne correspondant à celui de la région R_5.*

même bord, la configuration devient non-variété. Un exemple est donné sur la figure 1.30-A. Comme nous l'avons mentionné dans l'introduction, les structures de données de topologies non-variétés sont plus complexes que les structures variétés. Nous proposons donc de représenter ces murs sous une autre forme pour obtenir des objets variétés et afin d'alléger la structure. Ainsi, au lieu de considérer les murs comme des objets de dimension $n-1$, nous allons les interpréter comme des interfaces entre deux objets de dimension n. Par exemple, dans la figure 1.30-B les murs jouent le rôle d'interface entre des polygones. Donc, au lieu de coller des lignes entre elles, ce qui engendre des configurations non-variétés 1D, ce sont des frontières de polygones qui sont collées. Nous obtenons ainsi un objet variété 2D.

Il serait intéressant de posséder une structure de données qui peut à la fois contenir la topologie du mur ainsi que celle qui le représente en tant qu'interface. La structure des H-G-Cartes possède ces deux niveaux topologiques. La toile de la H-G-Carte représente le mur de dimension $n-1$ tandis que son cadre représente les deux frontières des régions de dimension n en contact le long de ce mur. La figure 1.30 montre ces deux représentations ; le mur (ligne) qui sépare les régions R_1 et R_2 est en fait interprété comme les deux bords connectés (arêtes) des polygones P_1 et P_2.

Il est à signaler que ce système ne permet cependant pas de représenter toutes les configurations non-variétés, mais suffit à prendre en compte ceux existants en géologie.

Il reste encore à définir comment connecter ces cadres pour former des modèles.

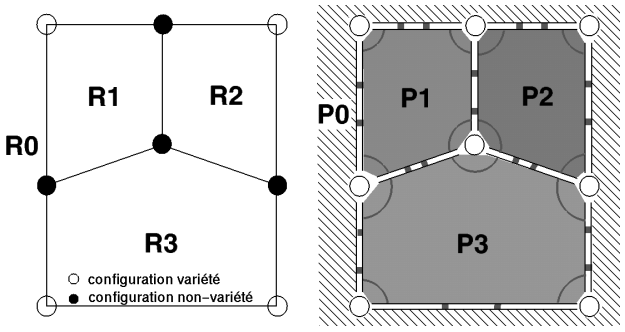


Figure 1.30 *Modèle surfacique. A. Ensemble de lignes (toile) engendrant des configurations non-variétés. B. Ensemble de polygones (cadre) formant un objet variété.*

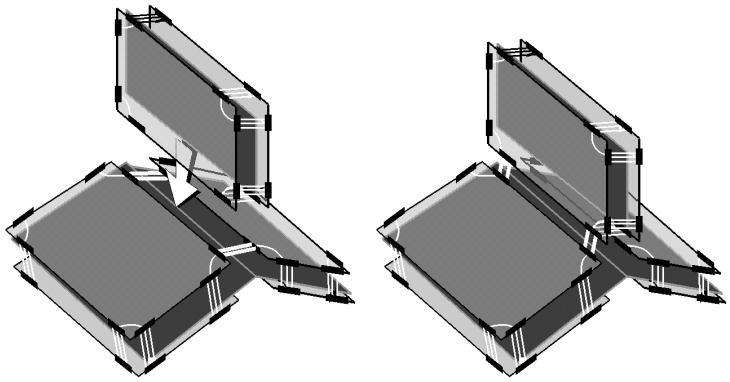


Figure 1.31 *Structure non-variété le long d'une ligne radiale (surfaces grisées). Les brins du cadre sont en noir. Les α_0 ne sont pas dessinés. Les α_i sont représentés par i traits blancs. - Avant collage et après collage d'un nouveau cadre.*

1.4.4 Construction d'un modèle

La construction d'un modèle implique la définition d'opérations spécifiques sur les brins du cadre. Ces opérations consistent à *coller* et *décoller* plusieurs cadres entre eux.

En deux mots, coller un cadre d'un objet de dimension n à un ensemble de cadres de même dimension déjà collés entre eux le long d'une $(n-1)$-cellule revient à insérer ce cadre entre deux cadres comme nous insèrerions une feuille entre deux pages d'un livre le long de la reliure (voir figure 1.31). De la même manière, décoller un cadre d'un ensemble de cadres consiste à *enlever* une page d'un livre.

Un cadre se compose de deux n-cellules décrites par des bords de dimension $n-1$. Coller un cadre c_1 à un cadre c_2 revient à coller un bord de c_1 avec un bord de c_2 ou plus précisement, relier les brins d'un bord de c_1 aux brins d'un

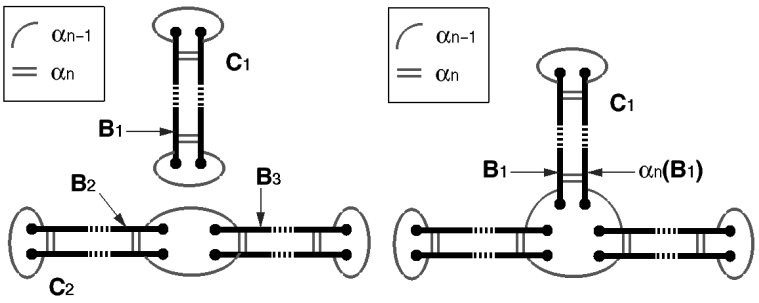


Figure 1.32 *(De gauche à droite) Coller un cadre c_1 à un cadre c_2. (De droite à gauche) Décoller un cadre c_1.*

bord de c_2.

Nous allons donner une définition plus formelle de ces deux opérations. Le lecteur pourra se référer à [**?**, **?**] pour une description plus détaillée.

Opération de collage

Soit b_1 et b_2 deux brins appartenant respectivement aux bords de deux n-cadres c_1 et c_2 d'un modèle. Le bord à coller du n-cadre c_1 doit être seul, c'est-à-dire qu'il ne doit pas être rattaché à un autre cadre. Ceci se traduit par $\alpha_{n-1}(b_1) = \alpha_n(b_1)$.

L'algorithme consiste alors à :

- Stocker le brin $\alpha_{n-1}(b_2)$ dans b_3.
- Connecter par α_{n-1} b_1 avec b_2 et $\alpha_n(b_1)$ avec b_3 (voir figure 1.32).
- Coller tous les brins du bord de la même manière.

Opération de décollage

Soit b_1 le brin du n-cadre c_1 d'un modèle.

L'algorithme consiste à :

- Déconnecter par α_{n-1} les brins b_1 et $\alpha_n(b_1)$ et restaurer les liens α_{n-1} (voir figure 1.32).
- Décoller tous les brins du bord de la même manière.

Cette structure et ces opérations de base nous permettent d'ores et déjà de construire des modèles géologiques.

Nous allons maintenant nous intéresser aux relations qui relient les brins de la toile à ceux du cadre.

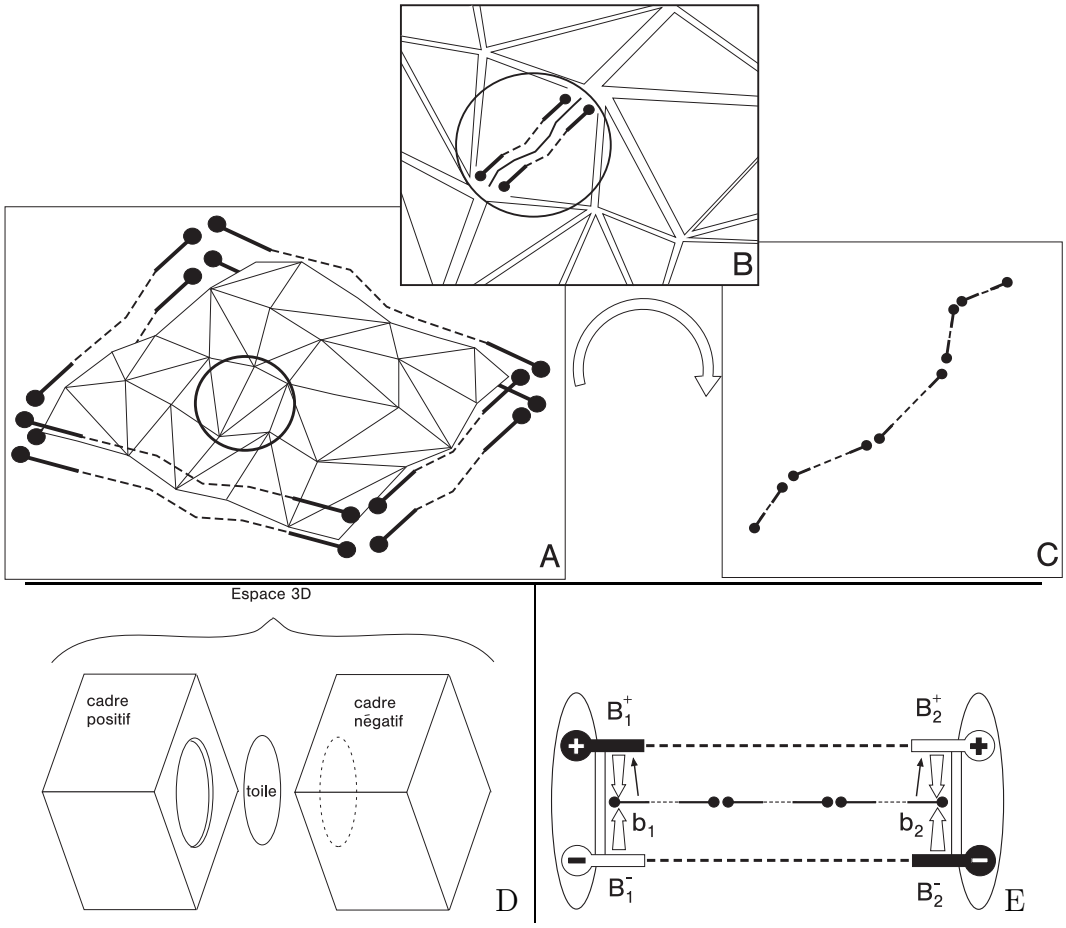


Figure 1.33 H-G-Carte à 3 niveaux. *A. Les brins représentent une 3-G-Carte englobant une surface. B. Les brins représentent une 2-G-Carte englobant une courbe. C. Les brins représentent une 1-G-Carte.* **Liens entre toile et cadre.** *D. Le cadre représente l'espace 3D entourant sa toile. E. Les brins du cadre (*brins gras*) connaissent un brin de la toile (*brins fins*). Chaque demi-cadre est signé. Par exemple, B_1^+ et B_1^- référencent b_1.*

1.4.5 Liens entre toile et cadre

La toile et le cadre sont tous deux des G-Cartes, le cadre étant une G-Carte de dimension directement supérieure à celle de la toile. Pour une ligne, G-Carte de dimension 1, un cadre sera représenté par une 2-G-Carte (cf figure 1.33).

Rappelons que le cadre est en fait l'union de demi-cadres cousus entre eux par des liens α_n. Il existe un demi-cadre que nous noterons *positif* et un demi-cadre *négatif*. Chaque demi-cadre *fait face* au *demi-espace* qui l'entoure (voir la figure 1.33-D). Chaque couple de brin $(B,\alpha_n(B))$ du cadre référence un brin b du bord de la toile comme le montre la figure 1.33-E.

Cette structure hiérarchique ne possède que deux niveaux, ce qui est suffisant

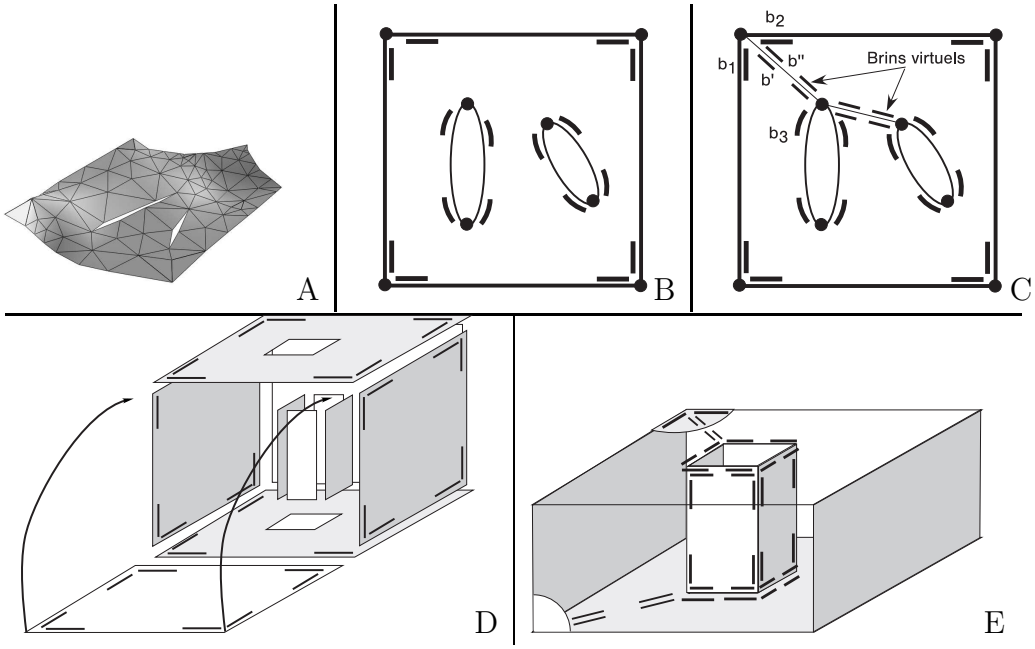


Figure 1.34 *A. Horizon troué : les trous proviennent des intersections avec les failles. B. Cadre de l'horizon C. Les 3 composantes connexes formant ce cadre sont reliées entre elles par des arêtes virtuelles (seules les brins B^+ du cadre sont représentés ici). D. Toiles d'un modèle : les toiles du sommet et de la base sont troués. La toile de devant a été mise à plat de façon à pouvoir voir l'intérieur du modèle. E. Construction du modèle : la région interne est traversée entièrement par un cylindre cubique. Pour obtenir cette région, les brins virtuels sont parcourus pour passer d'un bord à un autre des toiles formant le toit et la base de ce modèle.*

pour l'utilisation que nous en faisons. Mais il est tout à fait possible d'accroître le nombre de niveaux. En effet, considérons un cadre de dimension 3 rattaché à une toile de dimension 2 comme le montre la figure 1.33-A. Cette toile peut être considérée comme un ensemble de cadres de dimension 2 auquels on rattache des toiles de dimension 1 (voir figure 1.33-B). Chaque arête de l'objet de dimension 2 est en fait un plongement plus détaillé qui est stocké dans les objets de dimension 1 (voir figure 1.33-C).

Inclusion

Lorsqu'une surface est trouée, par exemple par l'intersection d'une faille qui la traverse, son nombre de bords augmente (voir la figure 1.34-A). Si l'on considère le cadre d'une telle surface représenté sur la figure 1.34-B, il est composé de plusieurs composantes connexes. Chaque composante représente un bord de la toile. L'horizon de la figure 1.34-A est composé d'un bord externe et de deux bords internes : il possède donc trois composantes connexes formant son cadre.

Si l'on veut parcourir tous les brins du cadre de cet horizon, il est nécessaire d'avoir une structure supplémentaire de type arbre d'inclusion pour rattacher les composantes internes à la composante externe. Une autre solution consiste à relier ces composantes par des *arêtes virtuelles* comme le montre la figure 1.34-C.

Ainsi, si l'on veut parcourir les brins du bord externe de la figure 1.34-C, il faut *ignorer* les brins b' et b''. Pour cela, la fonction qui donne l'α_1 d'un brin du bord est modifiée. Etant donné un brin b d'une n-G-Carte g, si $\alpha_{n-1}(b)$ est un brin virtuel b' alors $\alpha_{n-1}(b)$ est égal à $\alpha_{n-1} \circ \alpha_n(b')$. Dans notre exemple, $\alpha_1(b_1)$ ne doit pas retourner b' mais doit retourner $\alpha_1 \circ \alpha_2(b') = b_2$. Par contre, si l'on veut parcourir tous les brins de toutes les composantes connexes d'un cadre, la fonction qui donne l'α_1 d'un brin est à nouveau modifiée. Etant donné un brin b d'une n-G-Carte g, si $\alpha_{n-1}(b)$ est un brin virtuel b' alors $\alpha_{n-1}(b)$ est égal à $\alpha_{n-1} \circ \alpha_0(b')$. Dans notre exemple, $\alpha_1(b_1)$ doit retourner $\alpha_1 \circ \alpha_0(b') = b_3$.

Lors de la construction d'un modèle et plus particulièrement de la détermination de régions de ce dernier, il est nécessaire de parcourir les brins du cadre pour accéder à tous ceux qui définissent une région donnée. Dans l'exemple de la figure 1.34-D, on constate que si l'on ne parcourt que les brins des bords externes des différents cadres présents, le cylindre cubique interne de ce modèle est ignoré et ne ferait donc pas partie de la région. En *passant* par les brins virtuels, la région est complétée par le cylindre cubique comme le montre la figure 1.34-E.

1.4.6 Plongements des cellules

Dans cette section, nous allons parler des plongements ou propriétés que l'on peut rattacher aux cellules. En effet, l'utilisateur rattache aux cellules de son modèle des propriétés sur lesquelles il va vouloir travailler. En géologie, ces propriétés sont nombreuses (densité, porosité, vélocité, ...). Elles varient selon le milieu où l'on se trouve. Nous avons déjà expliqué dans la section 1.1.5 l'utilité de ces propriétés.

D'un point de vue informatique, il faut associer ces informations sur un objet topologique ou géométrique existant. La plupart des modèles raccrochent ces données aux noeuds du maillage. Avec la structure que nous utilisons, il est très facile de raccrocher ces informations sur n'importe quel type de cellules. Dans la figure 1.35, des propriétés ont été rattachées aux noeuds (figure 1.35-A), aux arêtes (figure 1.35-B), aux faces (figure 1.35-C) et même aux noeuds des faces (figure 1.35-D) d'une surface représentant un dôme de sel.

Dans le chapitre suivant, nous montrons entre autre comment définir des cellules à partir d'un ensemble de brins.

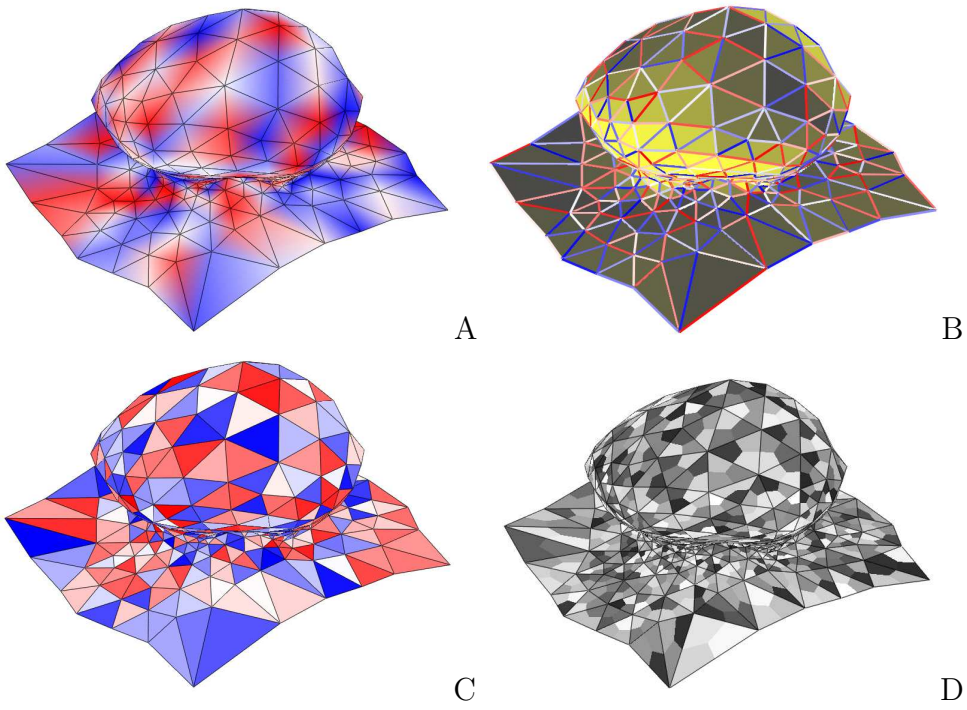


Figure 1.35 *Propriétés sur les cellules d'une surface représentant un dôme de sel. (A) Les propriétés ont été rattachées aux noeuds et ont été interpolées sur toute la surface. (B) Les propriétés sont sur les arêtes. (C) Les propriétés ont été stockées sur les faces de la surface. (D) Les propriétés sont stockées sur les noeuds des faces : un même noeud a des propriétés différentes selon le triangle sur lequel on se trouve.*

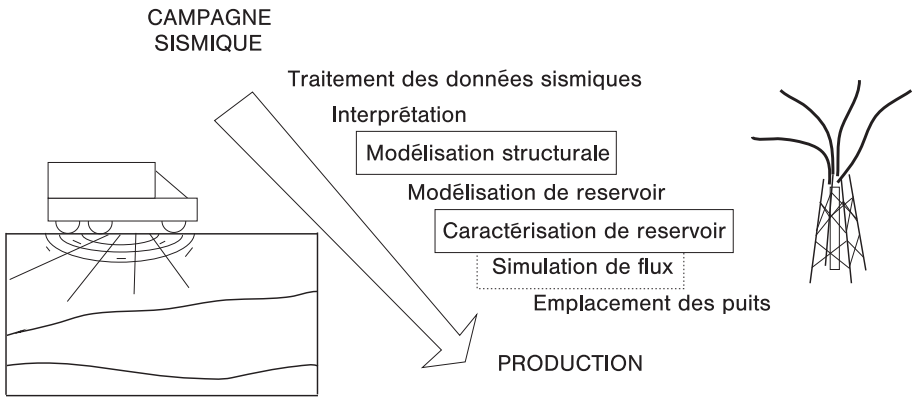


Figure 1.36 *Suite d'étapes de géomodélisation*

1.5 Conclusion

En géologie et notamment dans l'exploration pétrolière, la modélisation 3D a pris une place prépondérante. En effet, l'acquisition de données et son interprétation nécessitent des coûts exorbitants. De nos jours, il est difficile de découvrir de nouvelles réserves pétrolières. La recherche en modélisation 3D est une solution à ce problème. Les données extraites du sous-sol ne donnent qu'une image floue et partielle du terrain. A partir de ces images, les compagnies pétrolières cherchent à avoir une représentation la plus précise possible.

Le schéma 1.36 montre les différentes étapes du domaine de la géoscience en partant du traitement des données sismiques jusqu'à la production. Parmi toutes ces étapes, nous nous intéressons à 3 d'entre elles :

- Modélisation structurale : cette étape nécessite de représenter les couches du sous-sol ainsi que les interactions entre les différents objets géologiques (horizons, failles, ...). La structure topologique que nous utilisons pour cette représentation se nomme H-G-Carte. Cette structure permet à la fois de représenter les horizons et failles du modèle en tant que surface mais aussi les régions du modèle en tant que cellules volumiques.

- Caractérisation de réservoir : lors de cette étape, on cherche à rattacher des propriétés physiques à l'ensemble du domaine pour effectuer différents calculs. Par exemple, on va évaluer la quantité de pétrole qui est stockée dans la partie du sous-sol qui est traitée. Pour rattacher des propriétés de porosité à la région volumique considérée, il est nécessaire de construire un support tridimensionnel qui remplit l'espace étudié. Ce support peut être une grille régulière ou un volume tétraédrisé ou polyédrisé. Les propriétés seront rattachées aux noeuds, aux arêtes, aux faces ou aux polyèdres de l'objet. La généricité des G-Cartes nous permet d'obtenir de tels supports.

- Simulateur de flux : Cette étape cherche à étudier les déplacements de liquides dans le sous-sol. Par exemple, que se passe-t-il si l'on injecte de l'eau dans un puits ? Est-ce que du pétrole pourrait ressortir du sous-sol par un autre puits ? Les techniques utilisées prennent en considération des grilles régulières cubiques. Cependant, avec les moyens d'aujourd'hui, il devient possible d'utiliser des grilles à base polyédriques. Suivant la méthode utilisée pour la construction de ces grilles, la géométrie des cellules permet de faire des calculs de flux plus facile. Comme précédemment, grâce aux G-Cartes, ces grilles peuvent être construites et utilisées pour cette technique.

Il est à noter qu'au moment où cette thèse est écrite, une équipe du laboratoire se penche sur le problème du traitement des données sismiques. Une méthode envisagée reprend les travaux de Fiorio [**?**] sur les analyses d'images. On considère l'image 3D sismique comme un ensemble de voxels (pixel de dimension 3) et on regroupe les voxels de même couleurs entre eux pour former des cellules de dimension 3. Ces cellules, totalement arbitraires sont créées à l'aide de 3-G-Cartes. Fiorio utilisait des 2-Cartes pour des images 2D et avait déjà évoqué une solution pour passer en dimension supérieure.

Dans le chapitre suivant, nous allons détailler plus amplement le noyau topologique des G-Cartes que nous utiliserons dans le modeleur. Nous verrons aussi les opérations *légères* et *lourdes* qui agissent sur ce noyau.

Finalement, nous mettrons en avant une interface topologique de haut niveau plus intuitive que les G-Cartes.

Chapitre 2

Opérations topologiques

Dans le chapitre précédent, nous avons mis en avant les aspects génériques des G-Cartes du point de vue de leur structure. Le but de ce chapitre est de montrer comment ces aspects génériques se trouvent dans les algorithmes de manipulations de la topologie. Les sections 2.1 et 2.2 considère les opérations de bases sur les G-Cartes ainsi l'implantation que nous en avons faite. Puis en section 2.3, nous étudions les opérations dites *légères* introduites par Elter [**?**] ; ces opérations correspondent à des éditions de maillages : nous expliquons pourquoi et comment nous avons implanté une partie de ces opérations. Finalement, nous introduisons de nouvelles opérations dites *lourdes* telles que l'extrusion, certaines subdivisions surfaciques et volumiques. Nous expliquons dans la section 2.5 comment simplifier l'écriture de certains algorithmes à base de G-Carte. Et pour finir, nous évoquons comment utiliser un noyau topologique à base de cellules intuitives (sommet, arête, face, ...) tout en gardant une structure mathématique robuste dans la section 2.6.

2.1 Opérations de bases sur les G-Cartes

Dans cette section, nous définissons un ensemble d'opérations de base permettant de créer et modifier une G-Carte. Ces opérations sont les suivantes :

- création d'une n-G-Carte vide (section 2.1.1),
- création et suppression de brins (section 2.1.2),
- coudre et découdre des brins (section 2.1.3) et
- parcours de brins (section 2.1.4).

Borianne et Lienhardt [**?**, **?**] ont réalisé d'autres études sur ces opérations de base sur les 2-G-Cartes, Bertrand [**?**] sur les 3-G-Cartes et [**?**, **?**, **?**, **?**] sur les n-G-Cartes.

2.1.1 Création d'une n-G-Carte vide

CreerGCarteVide(n) définit une n-G-Carte G comme suit : $G = (B, \alpha_0, \ldots, \alpha_n)$ avec $B = \emptyset$.

Cette opération permet de définir une n-G-Carte vide, c'est-à-dire une n-G-Carte qui ne possède pas de brins.

2.1.2 Création et suppression de brins

Soit une n-G-Carte $G = (B, \alpha_0, \ldots, \alpha_n)$, CreerBrin($G$) crée un brin libre b de dimension n (i.e. $\forall i \in [0, n]\ \alpha_i(b) = b$) et définit une n-G-Carte $G' = (B', \alpha'_0, \ldots, \alpha'_n)$ avec :

$$B' = B \cup b \text{ avec } b \notin B \text{ et } \forall b' \in B', \forall i \in [0, n], \alpha'_i(b') = \begin{cases} b' & \text{si } b' = b \\ \alpha_i(b') & \text{si } b' \in B \end{cases}$$

Cette opération consiste à créer un brin libre b de la dimension de la G-Carte, Le brin b est ajouté à l'ensemble des brins de la G-Carte.

Soit une n-G-Carte $G = (B, \alpha_0, \ldots, \alpha_n)$ et b un brin libre de B, DetruireBrin(G, b) détruit le brin libre b de G et définit une n-G-Carte $G' = (B', \alpha'_0, \ldots, \alpha'_n)$ (i.e. $\forall i \in [0, n], \alpha'_i(b) = b$) avec :

$$B' = B/b \text{ et } \forall b' \in B', \forall i \in [0, n], \alpha'_i(b') = \alpha_i(b')$$

A l'inverse, cette opération consiste à enlever un brin b de l'ensemble des brins de la G-Carte. Ce brin aura été préalablement libéré sur toutes ses dimensions (c'est-à-dire que $\forall i \in [0, n], \alpha_i(b) = b$).

2.1.3 Coudre et découdre des brins

Soit une n-G-Carte $G = (B, \alpha_0, \ldots, \alpha_n)$, $b_1 \in B$, $b_2 \in B/b_1$, et $i \in [0, n]$, CoudreBrins(G, b_1, b_2, i) relie les brins S des (i-1)-cellules désignées par b_1 et b_2 par α_i (b_1 et b_2 doivent être des cellules isomorphes[1]) et défini une n-G-Carte $G' = (B, \alpha_0, \ldots, \alpha'_i, \ldots, \alpha_n)$ où seules les liaisons α_i des brins de S ont été modifiées de telle façon que :

$$\forall b \in S, \alpha'_i(b) \neq \alpha_i(b),$$

$$\forall b \in S, \forall j \in [0, \ldots, i-1, i+1, \ldots, n], \alpha_j(b) \text{ n'a pas changé et}$$

$$\forall b \in B/S, \forall k \in [0, n], \alpha_k(b) \text{ n'a pas changé.}$$

[1]Dans notre cas, il faut que les deux cellules soient de même nature : deux polygones à k cotés, deux segments, ...En aucun cas, il n'est possible de coudre un carré avec un triangle avec des liaisons α_3.

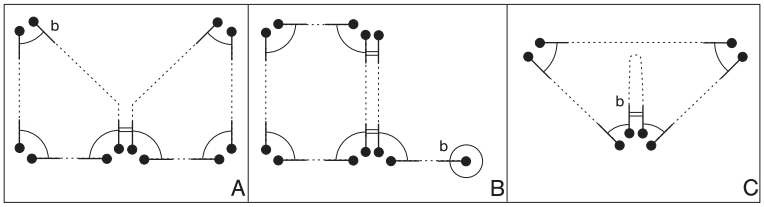


Figure 2.1 *Configurations interdites : (A) Couture incomplète. (B) Bords à dimension hétérogènes. (C) Arête repliée sur elle-même.*

L'opération de couture consiste à relier les brins de deux (i-1)-cellules entre eux par α_i. Par exemple, si l'on veut coudre deux brins d'une 3-G-Carte par α_3, nous allons en fait relier tous les brins des deux faces désignées par ces deux brins par α_3. Remarquez que coudre deux brins entre eux par α_i implique que ces brins doivent être libres en α_i.

Soit une n-G-Carte $G = (B, \alpha_0, \ldots, \alpha_n)$, $i \in [0, n]$, $b_1 \in B$ et $b_2 = \alpha_i(b_1)$, DecoudreBrins(G, b_1, b_2, i) libère les brins S des (i-1)-cellules désignées par b_1 et b_2 de la liaison α_i et défini une n-G-Carte $G' = (B, \alpha_0, \ldots, \alpha'_i, \ldots, \alpha_n)$ où seules les liaisons α_i des brins de S ont été modifiés de telle façon que :

$$\forall b \in S, \alpha_i(b) = b,$$

$$\forall b \in S, \forall j \in [0, \ldots, i-1, i+1, \ldots, n], \alpha_j(b) \text{ n'a pas changé et}$$

$$\forall b \in B/S, \forall k \in [0, n], \alpha_k(b) \text{ n'a pas changé.}$$

L'opération inverse de la couture consiste à détacher les brins de deux (i-1)-cellules qui ont été préalablement reliées entre elles par α_i.

Ces deux opérations vont permettre de construire et modifier un objet topologique.

Certaines règles sont à respecter du fait de la définition des G-Cartes. Par exemple, il est interdit de réaliser une couture incomplète (voir figure 2.1-A). Toutes les configurations interdites se trouvent figure 2.1 [**?**]. Cependant, nous verrons dans la section 2.5 page 93 qu'il est possible de contourner ces règles localement tout en s'assurant d'obtenir une G-Carte valide à la fin du traitement.

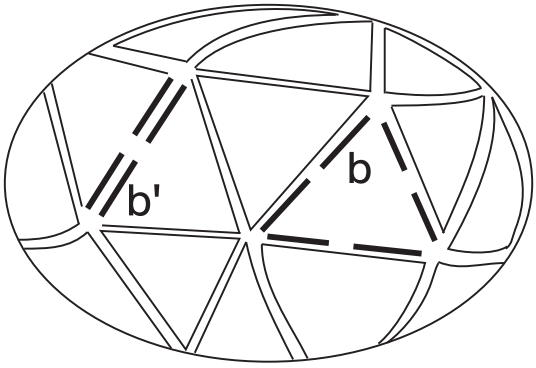


Figure 2.2 *Parcours d'involutions :* $< \alpha_0, \alpha_1 > (b)$ *consiste à accéder à tous les brins du polygone contenant* b *et* $< \alpha_0, \alpha_2 > (b')$ *consiste à accéder à tous les brins de l'arête contenant* b'.

2.1.4 Parcours de brins

Soit une n-G-Carte $G = (B, \alpha_0, \ldots, \alpha_n)$, $b \in B$ et $A = (\alpha_{i_1}, \ldots, \alpha_{i_m})$ un ensemble d'involutions appartenant aux involutions de G, ParcoursBrins$< \alpha_{i_1}, \ldots, \alpha_{i_m} > (b)$ construit le sous-ensemble S de B tel que $\forall b' \in S, \alpha_{i_{j_1}} \circ \alpha_{i_{j_2}} \circ \ldots \circ \alpha_{i_{j_l}}(b) = b'$ avec $\alpha_{i_{j_k}} \in A$ et $k \in [1..m]$. En d'autres termes, b' est accessible à partir de b par une succession finie d'alphas appartenant au sous-ensemble A.

Cette opération est légèrement plus complexe que les autres mais très utilisée. Elle consiste à parcourir un sous-ensemble de brins connectés les uns aux autres par des involutions ou série d'involutions.

Ces parcours vont permettre par exemple de définir des ensembles de brins qui appartiennent à une même i-cellule.

Par exemple, soit $G = (B, \alpha_0, \alpha_1, \alpha_2)$ une 2-G-Carte représentant une surface triangulée fermée connexe et b et b' deux brins de B (voir figure 2.2). Parcourir $< \alpha_0, \alpha_1 > (b)$ revient à parcourir les 6 brins qui constitue le triangle contenant b. Parcourir $< \alpha_0, \alpha_2 > (b)$ revient à parcourir les 4 brins qui constitue l'arête contenant b commune à deux triangles. Parcourir $< \alpha_0, \alpha_1, \alpha_2 > (b)$ revient à parcourir tous les brins de la surface.

Parcourir tous les brins d'une i-cellule d'un n-maillage à partir d'un brin b revient à parcourir $< \alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n > (b)$, c'est-à-dire à parcourir toutes les involutions sauf l'involution de la dimension de la cellule recherchée.

Nous verrons plus en détail les différents types de parcours existants suivant les α sélectionnés dans la section 2.2 qui traite de l'implantation des G-Cartes.

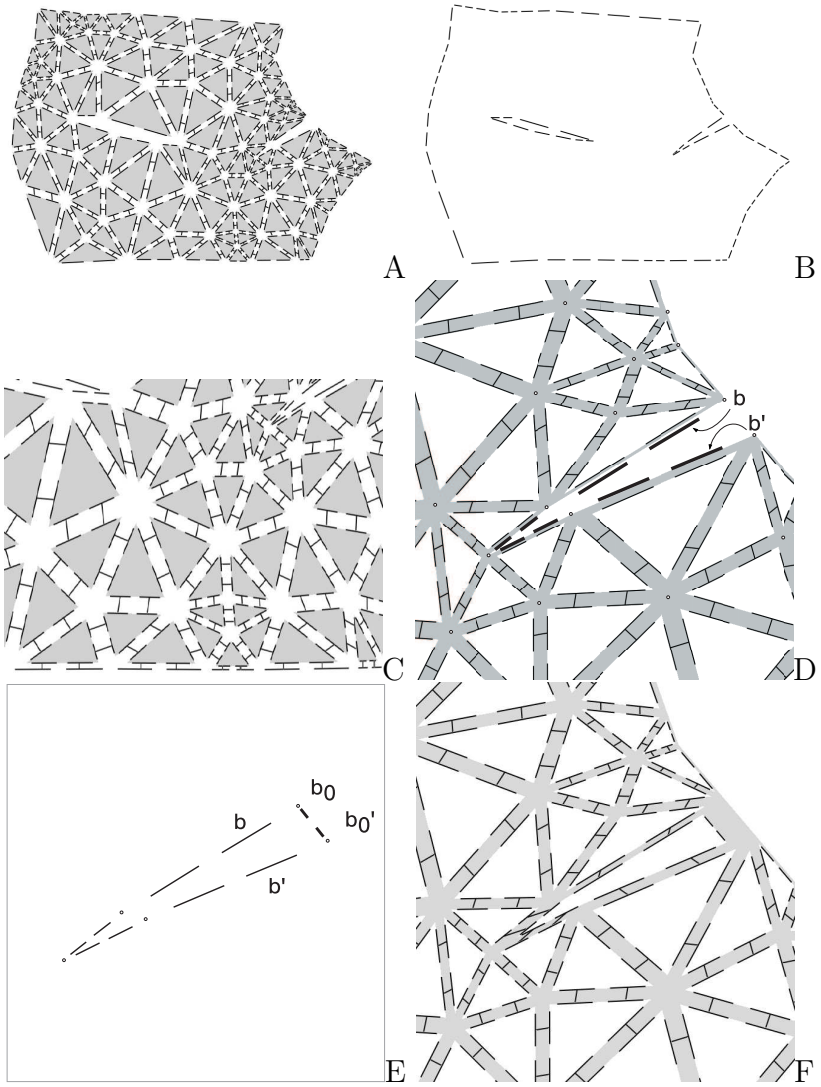


Figure 2.3 *A. Surface triangulée (2-G-Carte), seules les liaisons α_2 sont représentées ici. B. Courbe représentant la 1-G-Carte du bord de la surface. C. Fermeture de la 2-G-Carte. Les brins du bord externe (ici, les 8 brins qui sont au bas de la figure) sont reliés aux brins libres en α_2 de la surface par α_2. D. Bord des brins formant l'entaille du bord externe de la surface : les brins b et b′ sont libres en α_1. E. Fermeture de la courbe en insérant l'arête b_0 b'_0 entre b et b′. F. Résultat final.*

2.1.5 Fermeture d'une n-G-Carte

La dernière opération que nous étudions se nomme l'opération de *fermeture* [**?**, **?**]. Cette opération sera utilisée dans les opérations légères qui forment la section 2.3 de ce chapitre en page 66.

Elle peut s'appliquer à toute une n- G-Carte ou seulement à un ensemble de

ses brins. Cette opération consiste à lier avec de nouveaux brins tous les brins qui sont libres pour α_n.

Pour fermer une n-G-Carte G, une méthode consiste à créer le bord G' de G et de coudre par α_n les brins de G' aux brin de G, libres en α_n.

Le bord d'une n-G-Carte $G = (B, \alpha_0, \ldots, \alpha_n)$ est une (n-1)-G-Carte $G' = (B', \alpha'_0, \ldots, \alpha'_{n-1})$ tel que (voir exemple sur la figure 2.3-B) :

$$\begin{cases} & B' & = & \{b \in B \text{ avec } \alpha_n(b) = b\} \\ \forall 0 \leq i \leq n-2, \forall b \in B', & \alpha'_i(b) & = & \alpha_i(b) \\ \forall b \in B', & \alpha'_{n-1}(b) & = & b' \text{ avec } b' \in < \alpha_n, \alpha_{n-1} > (b) \text{ et } b' \neq b \end{cases} \tag{2.1}$$

Ainsi, pour chaque brin b' de la $(n-1)$-G-Carte qui a été construit à partir d'un brin b de la n-G-Carte, il faudra relier ces deux brins ensemble par α_n (voir figure 2.3-C).

La surface dessinée sur la figure 2.3-A possède deux bords : un bord externe qui correspond au grand polygone de la figure 2.3-B et un bord interne qui correspond au petit polygone de la même figure. Il est possible de n'appliquer la fermeture qu'à l'un de ses bords. Par exemple, comme le montre la figure 2.3-C, c'est le bord externe qui a été fermé. Une autre option consisterait à n'appliquer l'opération que sur certains brins, comme ceux qui forment l'entaille dans le bord interne de la surface de la figure 2.3-D. De cette manière, le bord ajouté est une courbe ouverte : les deux brins extrémités b et b' sont libres en α_1 (figure 2.3-D). Pour obtenir une 2-G-Carte correcte, il va falloir fermer ce polygone ouvert. On applique la fermeture au bord de notre courbe. Ce bord est une 0-G-Carte constituée de deux brins b_0 et b'_0 reliés entre eux par α_0 (figure 2.3-E). Chacun de ces brins sera relié par α_1 à un brin extrémité de la courbe pour obtenir un polygone fermé (figure 2.3-F). Pour retrouver le brin extrémité de b, nous utilisons la fonction AutreBrinLibre(b : Brin, n : Entier) qui retourne le brin b' tel que b' est libre en α_{n-1} et $b' = [\alpha_n \circ \alpha_{n-1}]^k(b)$ (d'autres exemples sont illustrés sur la figure 2.4).

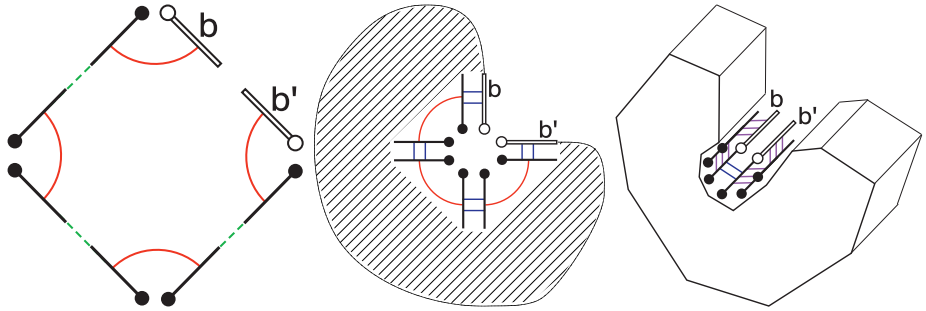


Figure 2.4 *AutreBrinLibre(b) = b′ : pour une 1, 2 et 3-g-carte.*

```
Procédure Fermer(G : n-g-carte)
   Pour chaque brin b du bord de G Faire
      b′ : Brin
      b′ ← CreerBrin(G)
      relier(G,b,b′,n)
   FinPour
   Pour chaque nouveau brin b′ Faire
      Pour i de 0 à n − 2 Faire
         relier(G,b′,alpha(alpha(alpha(b′,n),i),n),i)
      FinPour
      relier(G,b′,AutreBrinLibre(alpha(b′,n),n),n − 1)
   FinPour
FinFermer
```

Une autre méthode de fermeture du bord se trouve dans la section 2.5 page 93.

Etudions maintenant comment nous avons implanté de telles structures.

2.2 Implantation : topologie et propriété

Nous avons utilisé un langage supportant la programmation générique (ANSI C++ [**?**]) pour implanter la structure des G-Cartes. Dans cette section, nous utilisons des écritures qui se rapportent à la programmation générique. Ces écritures sont décrites au fur et à mesure de leur apparition.

2.2.1 Topologie

Rappelons qu'une n-G-Carte G est définie par un $n+2$ uplet comme suit : $G = (B, \alpha_0, \ldots, \alpha_n)$ avec B un ensemble de brins et les α_i des involutions agissant sur

les brins de B. Définissons les structures **Brin** et **GCarte**.

D'un point de vue topologique, un brin, dont la structure est présentée sur la figure 2.5, stocke $n+1$ liens vers des brins qui correspondent aux involutions α_i. Deux fonctions, alpha et positionner_alpha, permettent d'accéder et modifier les α_i des brins. La fonction est_libre permet de savoir si un brin b est libre pour la dimension i donnée. En d'autres termes, cette fonction teste si $\alpha_i(b)$ est égal à b ou non.

Une G-Carte (figure 2.6) connaît sa dimension et la liste des brins qui la constituent. En programmation générique, une liste d'un élément quelconque ELT s'écrira Liste<ELT>. La G-Carte s'initialise par la fonction CreerGCarteVide en passant en argument sa dimension. C'est elle qui s'occupe de créer et détruire ses brins. Finalement, c'est en appelant CoudreBrins et DecoudreBrins que les involutions α_i des brins sont modifiées. La fonction RelierBrins sert à relier uniquement deux brins entre eux et est utilisée par la fonction CoudreBrins. Par exemple, pour coudre deux triangles le long d'une arête, CoudreBrins appellera la fonction RelierBrins deux fois : une pour les brins b et b' et une autre pour les brins $\alpha_0(b)$ et $\alpha_0(b')$ (b étant un brin de l'arête du premier triangle et b' le brin opposé du deuxième triangle).

Grâce à ces deux structures et leurs ensembles de fonctions, il est possible de créer des objets. Dans l'exemple de la figure 2.8, nous construisons une surface triangulée contenant deux triangles. Ces triangles seront collés le long d'une arête (cf figure 2.7). L'algorithme se trouve figure 2.8.

Remarquez qu'aucune propriété géométrique n'a été affectée aux noeuds de la surface.

2.2.2 Propriétés

Nous complétons maintenant la structure des brins que nous avions faite précédemment pour y intégrer des propriétés. La plupart du temps, l'utilisateur n'attachera que des propriétés aux sommets de son objet (coordonnées dans l'espace, vecteurs normaux mais aussi porosité, température, ...). Cependant, il peut être intéressant de rattacher des propriétés à tout type de cellules (aux sommets, mais aussi aux

Structure Brin
Brin alpha(Brin brin, Entier dimension) Procédure positionner_alpha(Brin brin, Entier dimension, Brin voisin) Booleen est_libre(Brin b, Entier dimension)
Brin alphas_[n+1]

Figure 2.5 *La structure Brin*

Structure GCarte
GCarte CreerGCarteVide(Entier dimension) Brin CreerBrin(GCarte g) Procédure DetruireBrin(GCarte g, Brin b) Procédure CoudreBrins(GCarte g, Brin b1, Brin b2, Entier dimension) Procédure DecoudreBrins(GCarte g, Brin b1, Brin b2, Entier dimension) Procédure RelierBrins(GCarte g, Brin b1, Brin b2, Entier dimension)
Entier dimension_ Liste<Brin> brins_

Figure 2.6 *La structure G-Carte*

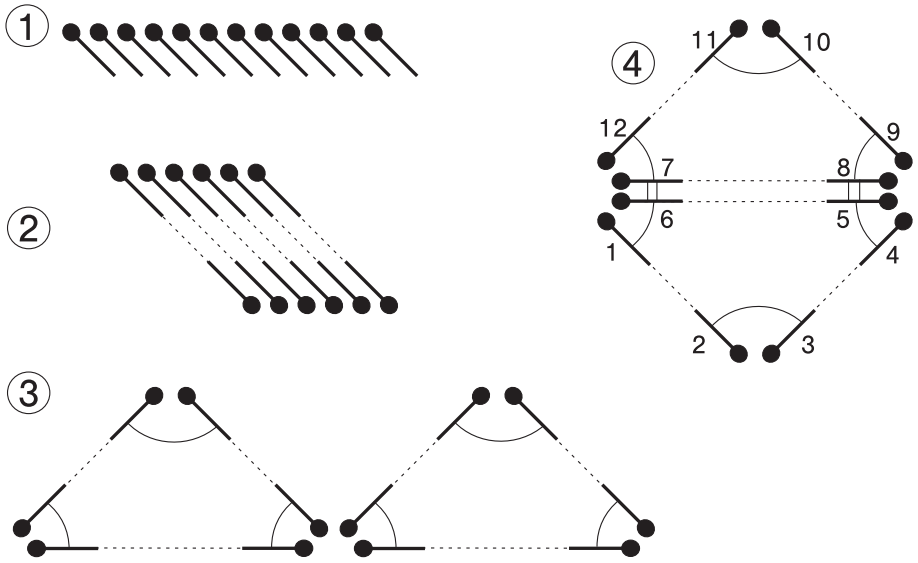


Figure 2.7 *Deux triangles ayant une arête commune. Etape 1 : création des 12 brins. Etape 2 : les brins sont reliés en α_0. Etape 3 : les brins sont reliés en α_1. Etape 4 : les brins sont reliés en α_2.*

arêtes, polygones, polyèdres, ...) et même à des orbites quelconques (arêtes de faces, brins, ...). La figure 2.9 illustre schématiquement comment les propriétés sont rattachées aux cellules de l'objet. Par exemple, il sera possible de rattacher une chaîne de caractères à chaque 3-cellule d'un volume pour signifier que tel polyèdre est une roche calcaire, du grès ou tout autre élément du sous-sol.

Dans ce qui suit, le terme plongement désigne l'ensemble des propriétés que

```
// Construction d'une 2-G-Carte vide (cf figure 2.7)
gcarte : GCarte
gcarte ← CreerGCarteVide(2);
// Construction des 12 brins nécessaires à l'opération
// Voir Etape 1 de la figure 2.7
Brin brins[12];
Pour i de 1 à 12 Faire
   brins[i] ← CreerBrin(gcarte);
FinPour
// Formation des arêtes des deux triangles
// Voir Etape 2 de la figure 2.7
Pour j de 1 à 6 Faire
   CoudreBrins( gcarte, brins[j x 2-1], brins[j x 2], 0 );
FinPour
// Formation des triangles
// Voir Etape 3 de la figure 2.7
CoudreBrins( gcarte, brins[1], brins[6], 1 );
CoudreBrins( gcarte, brins[3], brins[2], 1 );
CoudreBrins( gcarte, brins[5], brins[4], 1 );
CoudreBrins( gcarte, brins[7], brins[12], 1 );
CoudreBrins( gcarte, brins[9], brins[8], 1 );
CoudreBrins( gcarte, brins[11], brins[10], 1 );
// Formation de la surface
// Voir Etape 4 de la figure 2.7
CoudreBrins( gcarte, brins[7], brins[6], 2 );
```

Figure 2.8 *Construction de deux triangles ayant une arête commune.*

l'utilisateur attache aux cellules. La structure brin stocke également $n+1$ plongements qui correspondent aux $n+1$ cellules auxquelles appartient le brin. Comme pour les alphas, deux fonctions permettent d'accéder et modifier le plongement des brins. Une information supplémentaire semble nécessaire à cette structure. En effet, les propriétés d'une i-cellule sont rattachées à tous les brins formant cette i-cellule. Pour résoudre certains conflits, un brin de chaque cellule est désigné en tant que porteur officiel de l'information [**?**].

Pour expliquer comment les informations de plongement vont être stockées, nous étudions l'exemple suivant. Sur la figure 2.11-A, rattacher un point géométrique à un sommet de la surface consiste à rattacher ce point à tous les brins de la cellule (ici la cellule est un sommet). Si l'on laisse le client faire ce travail, cette

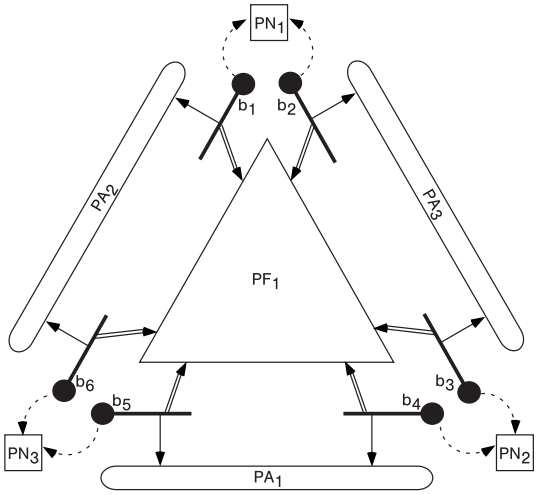


Figure 2.9 *Des propriétés peuvent être attachées à tout type d'orbite. Ici, nous représentons des propriétés cellulaires. Les brins b_1 et b_2 qui forment un sommet pointent sur la propriété PN_1. Les brins b_2 et b_3 pointent sur la propriété d'arête PA_3. Tous les brins possèdent une propriété de face PF_1.*

Structure Brin
Brin alpha(Brin brin,Entier dimension) Procédure positionner_alpha(Brin brin,Entier dimension,Brin mate) Booleen est_libre(Brin b,Entier dimension) **Plongement plongement(Brin brin,Entier dimension)** **Procédure positionner_plongement(** **Brin brin,Entier dimension,Plongement plongement** **)** **Booleen est_porteur_officiel(Brin brin,Entier dimension)**
Brin alphas_[n+1] **Plongement plongement_[n+1]** **Booleen porteur_officiel_[n+1]**

Figure 2.10 *Nouvelle structure Brin avec gestion des propriétés*

méthode risque d'être source d'erreur. En effet, l'utilisateur pourra faire pointer le plongement de deux brins d'un même sommet sur deux points différents. Afin d'éviter ce travail assez fastidieux, nous avions déjà dit qu'un brin de chaque cellule sera désigné en tant que porteur officiel de l'information. Une séquence de mise à jour est mise en place pour distribuer l'information aux autres brins de la

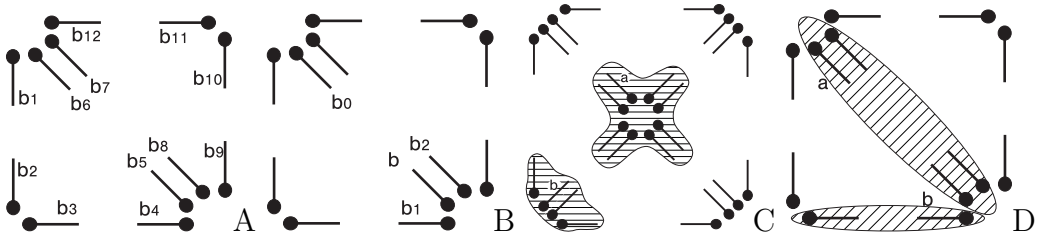


Figure 2.11 *(A)Les brins de deux triangles possèdant une arête en commun. Les liens α_i ne sont pas représentés ici. (B) Exemple de parcours à une involution (α_0, α_1 ou α_2). (C) Exemple de parcours à deux involutions successives (α_1 et α_2). (D) Exemple de parcours à deux involutions non successives (α_0 et α_2).*

cellule automatiquement.

De cette manière, dans la figure 2.11-A, seuls les brins numéros 1, 3, 8 et 11 par exemple vont être désignés par l'utilisateur pour être porteurs de l'information.

La séquence de mise à jour fonctionne de la manière suivante. En parcourant tous les brins de notre objet, chaque 0-cellule sera parcourue pour trouver son porteur officiel et distribuer ses informations aux autres brins de la cellule. Ainsi, les plongements des brins b_6, b_7 et b_{12} pointeront tous sur celui détenu par le brin porteur officiel b_1.

Pour parcourir les cellules, une série d'itérateurs a été implantée pour faciliter leur utilisation. Nous les définissons dans le paragraphe suivant.

2.2.3 Parcours

Rappelons que parcourir une cellule de dimension i d'un objet n à partir d'un brin b consiste à accéder à tous les brins en partant de b et en utilisant les involutions $\alpha_0, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_{n-1}, \alpha_n$.

Pour la figure 2.11-A, les autres brins de la 2-cellule du brin b_8 sont : b_7, b_9, b_{10}, b_{11} et b_{12}.

L'algorithme de parcours peut être vu comme suit :

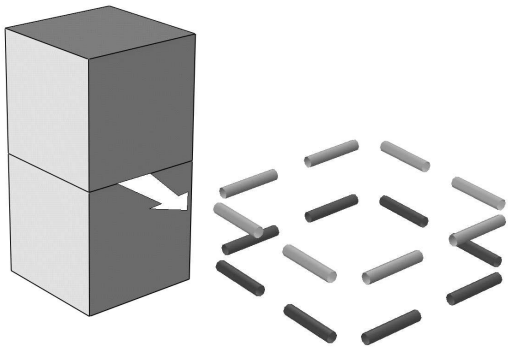


Figure 2.12 *Exemple de parcours à trois involutions ($\alpha_0, \alpha_1, \alpha_3$). (A gauche) Deux cubes reliés le long d'une face. (A droite) Les brins de la face commune. Les brins clairs sont reliés aux brins foncés par des liens α_3.*

```
Procédure ParcoursBrins< α_{i_1}, ..., α_{i_j} >(b : Brin)
   P : Pile<Brin>
   P ← b
   TantQue P n'est pas vide Faire
      b ← sommet de P
      Si b n'a pas encore été marqué Faire
         Marquer b
         Pour chaque involution α_{i_k} avec k = [1..j] Faire
            Mettre α_k(b) dans P
         Fin Pour
      Fin Si
   Fin TantQue
FinParcoursBrins
```

Cet algorithme est général mais non optimal. En effet, pour la plupart des parcours, le passage par une pile ne s'avère pas nécessaire.

Si le parcours ne possède qu'une seule involution, seuls deux brins seront visités au maximum. Dans l'exemple de la figure 2.11-B, c'est le brin b qui est le brin de départ. Le parcours correspondant à l'involution α_i possède les brins b et b_i pour $0 \leq i \leq 2$.

Pour les parcours possédant deux involutions, il faut les diviser en deux catégories :

- La première catégorie regroupe les parcours dont les deux involutions se succèdent (c'est-à-dire α_i et α_{i+1}) (Figure 2.11-C). Ce genre de parcours "tourne" autour d'une cellule. Par exemple, les involutions α_0 et α_1 per-

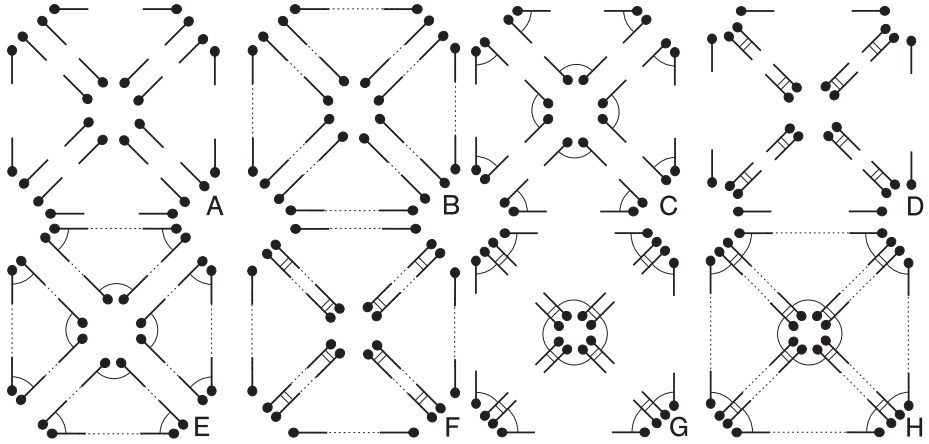


Figure 2.13 *Parcours d'orbites : (A) Pas d'involutions. (B) Involution α_0 (arêtes de polygone). (C) Involution α_1 (sommets de polygone). (D) Involution α_2. (E) Involutions α_0 et α_1 (polygones). (F) Involutions α_0 et α_2 (arêtes). (G) Involutions α_1 et α_2 (sommets). (H) Involutions α_0, α_1 et α_2 (surface).*

mettent d'obtenir les brins constituant un polygone d'une surface.

- La deuxième catégorie regroupe les parcours dont les deux involutions ne se succèdent pas (c'est-à-dire α_i et α_j avec j $\geq$ i+2 (cf figure 2.11-D). Par exemple, les involutions α_0 et α_2 permettent d'obtenir les brins constituant l'arête d'une surface. Dans ce cas, quatre brins au maximum peuvent être atteints. Ceci est dû au fait que l'association de ces deux involutions forme une nouvelle involution.

Pour les parcours à trois involutions, à nouveau plusieurs catégories se distinguent.

- La première catégorie regroupe les parcours dont les trois involutions se succèdent (c'est-à-dire α_i, α_{i+1} et α_{i+2}). Par exemple, les involutions α_0, α_1 et α_2 permettent de parcourir tous les polygones d'une surface.

- La deuxième catégorie regroupe les parcours dont deux involutions se succèdent mais pas la troisième. Par exemple, les involutions α_0, α_1 et α_3 permettent de parcourir tous les brins constituant une face d'un solide (cf figure 2.12). Ce parcours se fait en trois étapes :

  – Parcours avec les involutions qui se succèdent (les brins clairs de la figure 2.12 sont parcourus).
  – Accéder à un brin par la troisième involution.
  – Parcours avec les involutions qui se succèdent (les brins foncés de la figure 2.12 sont parcourus).

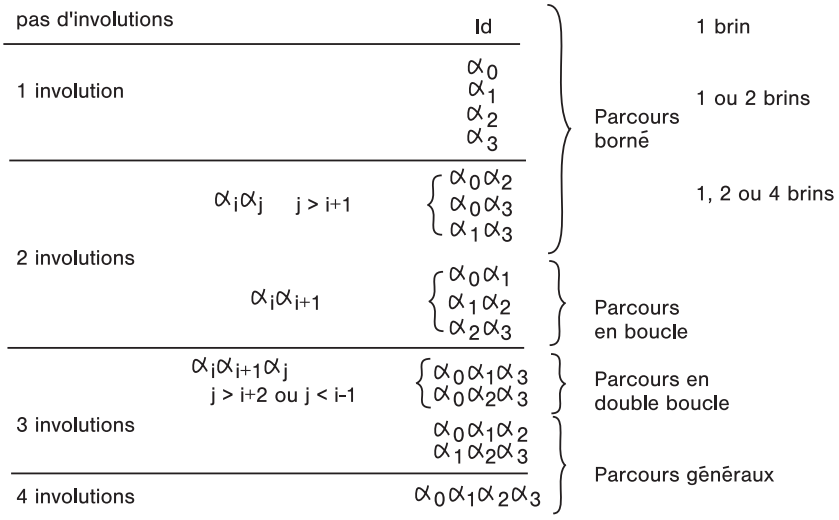


Figure 2.14 *Un parcours général utilisant une pile n'est nécessaire que pour les trois derniers parcours. Les autres parcours sont bornés (c'est-à-dire que l'on ne peut accéder qu'à un nombre maximum connu de brins) ou sont des parcours en boucles (dans ce cas, on accède aux brins en altérnant successivement deux involutions).*

Ce type de parcours est dû au fait que deux couples de liaisons alphas parmi les trois forment une involution (ici $\alpha_1 \circ \alpha_3$ et $\alpha_0 \circ \alpha_3$).

Pour des objets volumiques, il n'existera qu'un seul parcours à quatre involutions. Ce parcours possède les involutions α_0, α_1, α_2 et α_3 et permet d'atteindre tous les brins d'une composante connexe d'un volume.

Ces parcours vont nous permettre de localiser tous les brins d'une i-cellule donnée. Nous allons également les utiliser pour *élire* un brin en tant que porteur officiel du i-plongement associé. Lors de la mise à jour d'un objet, tous les brins de ce dernier sont parcourus. Pour chaque dimension de cellule, un parcours de cellule est effectué. Suivant le nombre de porteur officiel détecté, un traitement est effectué sur le brin courant.

- 0 brin porteur : le brin courant est élu porteur officiel.
- 1 brin porteur : le plongement du brin courant pointe sur celui du brin porteur
- 2 brins porteurs ou plus : cela signifie que des cellules ont dû être fusionnées, le brin courant devient porteur officiel et les anciens brins porteurs sont déclassés en tant que brin *normal*.

Nous étudions maintenant des opérations dites légères sur les G-Cartes.

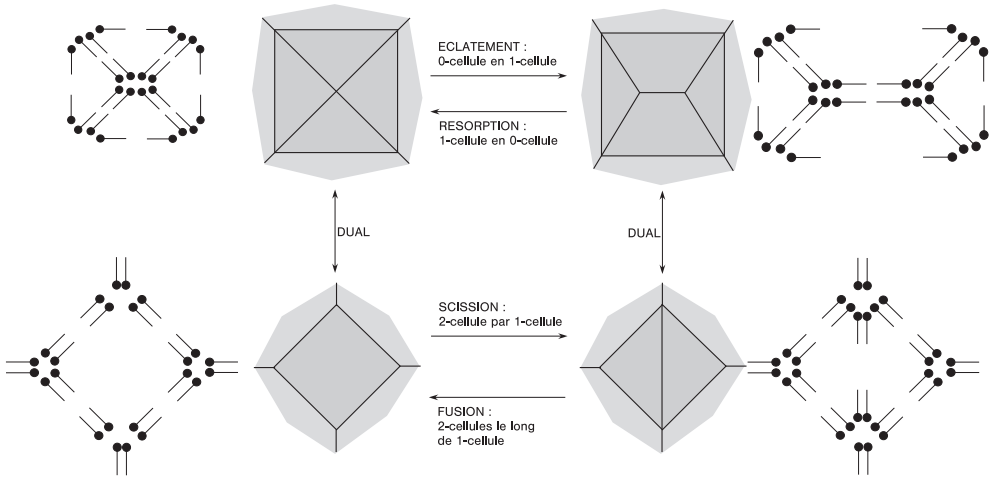


Figure 2.15 *Présentation de 4 opérations par Elter : éclatement, résorption, scission et fusion.*

2.3 Opérations légères sur les G-Cartes

Dans cette section, nous allons énumérer une large partie d'opérations qui permettent d'éditer des maillages.

Dans la thèse d'Elter [**?**], des opérations de haut niveau sont définies sur les n-G-Cartes. S'inspirant des opérateurs dits *classiques* en modélisation de solides (chanfreinage et division (*split* en anglais)), ces nouvelles opérations se nomment éclatement, résorption, scission et fusion. Le lecteur pourra se rapporter aux pages 85 à 131 de [**?**] pour de plus amples précisions.

Dans sa thèse, Elter est parti de l'opération de chanfreinage [**?**, **?**, **?**, **?**] pour expliquer ensuite sa généralisation (l'éclatement). Les travaux sur l'éclatement peuvent se trouver dans [**?**, **?**, **?**, **?**, **?**]. A partir de cette opération, les trois autres se définissent de la manière suivante. La résorption est son inverse, la scission correspond au dual de l'éclatement et finalement la fusion est l'inverse de la scission (c'est-à-dire l'inverse du dual de l'éclatement).

Toutes ces opérations sont génériques et manipulent des i-cellules et des j-cellules. Selon les opérations, nous n'avons choisi qu'un sous-ensemble des opérations décrites par Elter, par exemple, la résorption que nous avons implanté consiste toujours à changer des i-cellules en 0-cellules et non pas en j-cellules. Les différentes opérations que nous avons développées sont :

- chanfreinage d'une i-cellule.
- destruction de m-cellules d'un n-maillage
- résorption de m-cellules en 0-cellule

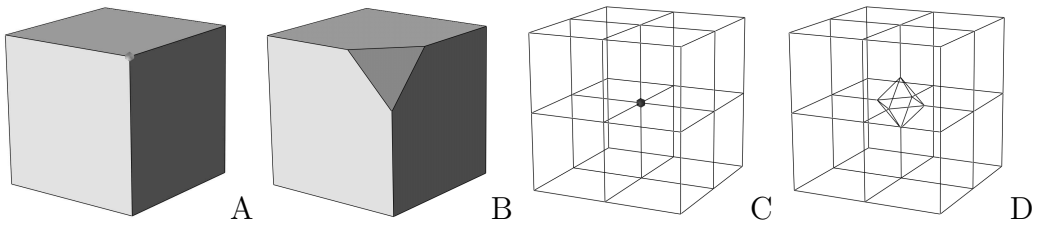


Figure 2.16 *A. Chanfreinage d'un sommet d'un cube isolé. B. Résultat : le sommet a été coupé. C. Chanfreinage du sommet* interne *d'un volume formé de huit cubes. D. Résultat : le sommet a été remplacé par un octaèdre.*

- scission de n-cellules d'un n-maillage
- fusion de n-cellules d'un n-maillage le long d'une m-cellule

Nous verrons quelques opérations isolées comme l'ajout d'une arête dans un polygone, l'insertion d'une structure régulière de i-cellules dans un i-maillage, la subdivision de maillages, ...

2.3.1 Chanfreinage

Le chanfreinage consiste à *couper* une cellule par un *plan de coupe*. Par exemple, chanfreiner le sommet d'un cube va consister à couper le coin portant ce sommet (voir figure 2.16-A-B). Si le sommet à chanfreiner se trouve *dans* le n-maillage, ce sommet sera *remplacé* par une n-cellule. Ainsi, chanfreiner un sommet interne d'un volume va consister à remplacer ce sommet par un polyèdre comme le montre la figure 2.16-C-D.

Le chanfreinage d'une i-cellule se fait en plusieurs étapes :

Découdre les (i+1)-liaisons

On découd par α_{i+1} tous les brins de la i-cellule (voir figure 2.17-A-B).

Introduction des (i+1)-cellules

Pour chaque couple de brin (b_1, b_2) qui étaient reliés entre eux par α_{i+1}, créer deux brins b'_1 et b'_2 reliés entre eux par α_i qu'il faut relier à b_1 et b_2 par α_{i+1} (voir figure 2.17-C-D).

Introduction des (i+2)-cellules,. . . ,n-cellules

Les brins b'_1 et b'_2 créés précédemment sont libres en α_{i+2} et forment un (i+2)-bord. L'étape consiste à refermer ce bord en introduisant les brins b''_1 et b''_2, puis, itérer

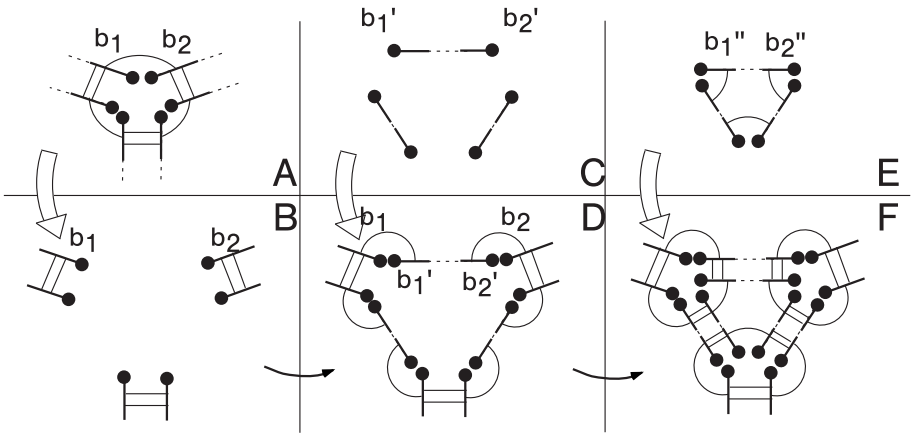


Figure 2.17 *Chanfreinage d'un sommet d'une surface : A. Le sommet est de degré 3. B. La première étape consiste à découdre tous les brins par α_1. C. La deuxième étape consiste à introduire des 1-cellules. D. Les 1-cellules sont rattachées aux brins libres en α_1 (B). E. La dernière étape consiste à introduire la 2-cellule. F. La 2-cellule est rattachée aux brins introduits précédemment.*

cette opération jusqu'à l'introduction des n-cellules en changeant progressivement la dimension du bord à fermer (voir figure 2.17-E-F).

2.3.2 Destruction d'une cellule

Exemple 2 : destruction d'un polyèdre d'un solide

Détruire une i-cellule d'un n-maillage consiste à détruire toutes les n-cellules incidentes à la i-cellule désignée.

Exemple 1 : destruction d'un noeud d'une surface

Les polygones incidents à ce noeud vont être détruits.

Pour tous les brins du noeud sélectionné, c'est-à-dire tous les brins que l'on peut atteindre par α_1 et α_2, les stocker. Stocker par la même occasion les n-cellules de ces brins, c'est-à-dire les brins que l'on peut atteindre par α_0 et α_1.

Il reste alors à isoler la cellule en décousant tous les brins stockés par α_2 puis détruire ces brins.

Destruction d'une m-cellule d'un n-maillage

Pour tous les brins de la m-cellule sélectionnée, c'est-à-dire tous les brins que l'on peut atteindre par $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}, \alpha_{m+1}, \ldots, \alpha_{n-1}, \alpha_n$, les stocker. Stocker aussi par la même occasion les n-cellules de ces brins, c'est-à-dire les brins que

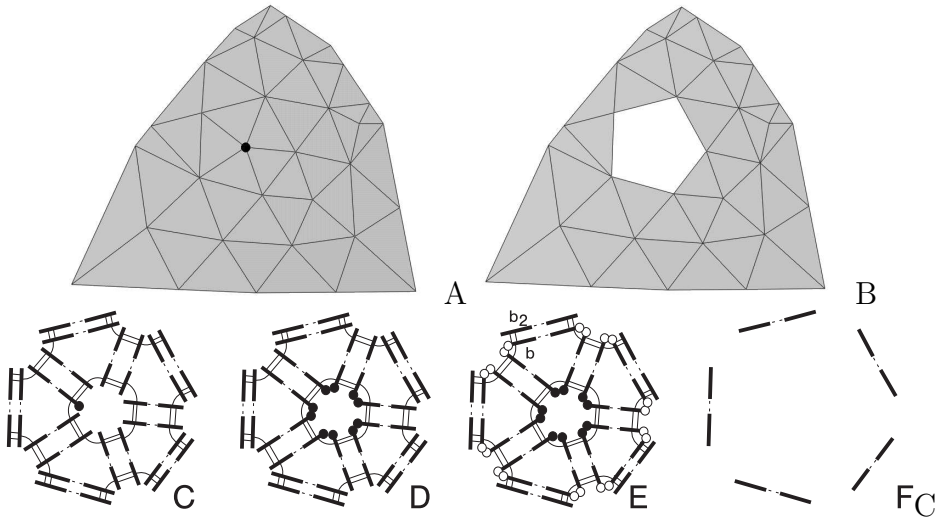


Figure 2.18 *(A) Une surface dont un noeud central va être détruit. (B) Le résultat. (C) De gauche à droite : Nous représentons ici les brins qui vont jouer un rôle lors de la destruction du noeud. Le brin qui possède une boule en son extrémité est le brin de départ. A partir du brin de départ, tous les brins du sommet sont sélectionnés. A partir des brins sélectionnés, on sélectionne les brins des faces que l'on peut atteindre, (ici, les brins trouvés sont ceux qui possède une boule blanche). Finalement, tous les brins sélectionnés sont détachés et supprimés. On représente dans la dernière figure uniquement les brins qui étaient en contact avec les cellules supprimées.*

l'on peut atteindre par $\alpha_0, \alpha_1, \ldots, \alpha_{n-2}, \alpha_{n-1}$. Il reste à isoler les brins stockés en les décousant par α_n puis détruire ces brins. L'algorithme de cette opération se trouve figure 2.19.

2.3.3 Résorption de cellules

Résorber des i-cellules d'un n-maillage en un point consiste à identifier tous les noeud des i-cellules à un seul et unique noeud.

Exemple 1 : résorption d'une arête d'une surface

Les deux noeuds d'une arête vont se rassembler en un seul point. Pour cela, il faut calculer le barycentre de l'arête, supprimer les brins formant l'arête (voir figure 2.20-D-E. En supprimant les brins b et b_0, relier les brins b_1 et b_{01} entre eux par α_1. La surface de la figure 2.20-E possède deux polygones plats. Pour remédier à ce problème, relier les brins b_{21} et b_{201} directement par α_2 et supprimer les brins b_1 et b_{01} (voir figures 2.20-E-F).

```
DestructionCellule(g : n-G-Carte, c : m-cellule)
   Pour chaque brin b de la m-cellule Faire
      Si b n'a pas encore été stocké Faire
         stocker chaque brin de la n-cellule contenant b
      FinSi
   FinPour
   Pour chaque brin b stocké Faire
      g→delier( n, b, α_n(b) )
   FinPour
FinDestructionCellule
```

Figure 2.19 *Destruction de cellules avec g une G-Carte de dimension n, et c une cellules de g de dimension m ($m \leq n$).*

Exemple 2 : résorption d'un polygone d'une surface

Tous les noeuds d'un polygone vont se rassembler en un seul point. Pour cela, il faut calculer le barycentre du polygone, stocker les brins reliés aux brins du polygone par α_2. Ces brins correspondent aux brins à tête blanche de la figure 2.21-D. Ensuite, il faut supprimer le polygone initial et, pour chaque arête formée par b et b_0, supprimer ces deux brins et relier les brins b_1 et $b_0 1$ entre eux par α_1 comme le montre les figures 2.21-E-F. Dans notre exemple, il n'y a pas de cellules dégénérées à supprimer, l'opération est donc terminée.

Résorption d'une m-cellule d'un n-maillage

Dans [**?**], la résorption correspond à l'inverse de l'éclatement. La première étape consiste à supprimer la m-cellule sélectionnée, puis les (m-1)-cellules aux 1-cellules que l'on peut atteindre à partir de la m-cellule et enfin la dernère étape consiste à restaurer les liens α_1. Cependant, en résorbant une m-cellule, certaines (m+1)-cellules peuvent devenir dégénérées. Par exemple, en résorbant une arête d'une surface triangulée, les deux triangles qui étaient en contact avec cette arête deviennent des polygones à deux côtés. Il est alors nécessaire de faire une deuxième passe pour supprimer ces cellules. L'algorithme de cette opération se trouve figure 2.22.

2.3.4 Scission d'une cellule

Dans [**?**], l'opération de scission de m-cellules par des i-cellules ($0 \leq i < m \leq n$), appelée plus classiquement *split*, est définie à partir de l'opération d'éclatement et consiste à diviser une m-cellule en plusieurs m-cellules en insérant une ou plusieurs i-cellules.

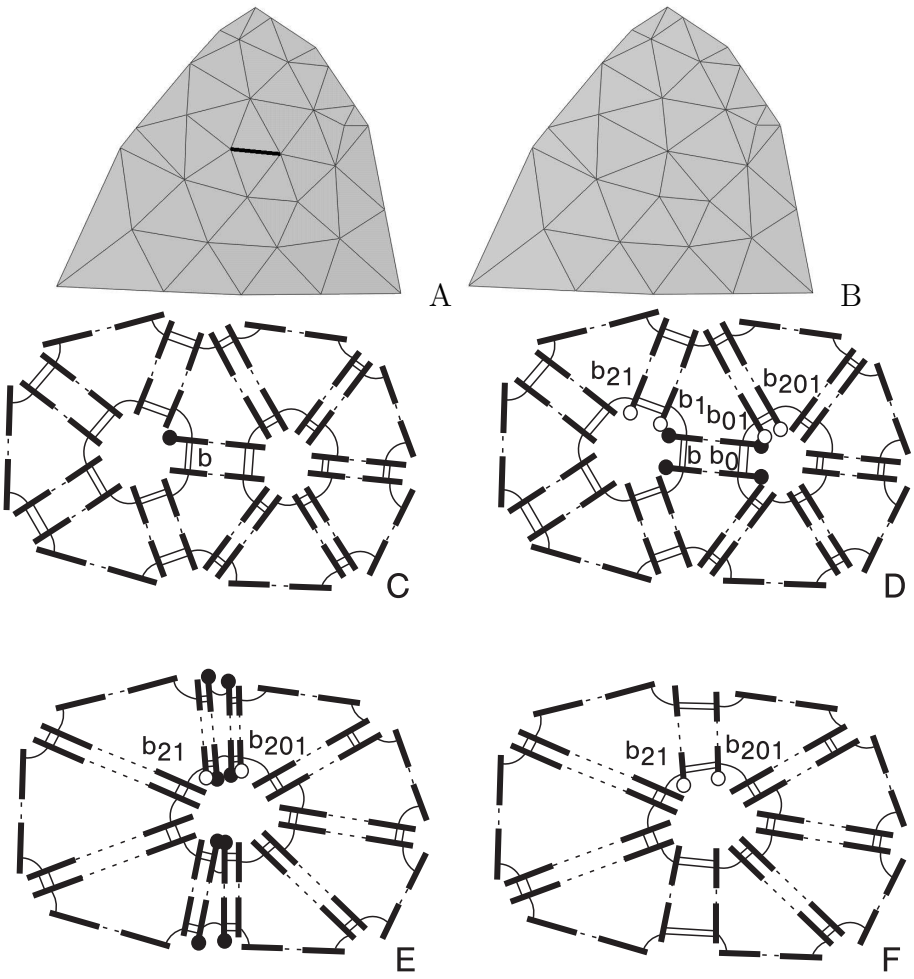


Figure 2.20 *(A) Une arête va être résorbée. (B) Le résultat. En détail : (C) L'arête contenant le brin b va être résorbée. (D) Les brins de l'arête sont sélectionnés (4 brins à tête noire). Pour chaque couple de brin b et b_0, (E) relier b_1 et b_{01} entre eux par α_1 puis supprimer b et b_0. Il reste des cellules dégénérées. En les supprimant, (F) on obtient le résultat final.*

Notre opération de scission est une scission barycentrique. Seul un point est ajouté au centre de la m-cellule. Des n-arêtes relient ce point aux points existants de la m-cellule.

Exemple 1 : scission d'un polygone d'une surface

Le polygone sélectionné va être divisé en n triangles avec n étant le nombre de noeuds du polygone (figure 2.23 à droite).

Pour chaque noeud symbolisé par le brin b_1 du polygone sélectionné, une arête double va être ajoutée. Pour cela, on va coudre une arête constituée de deux brins

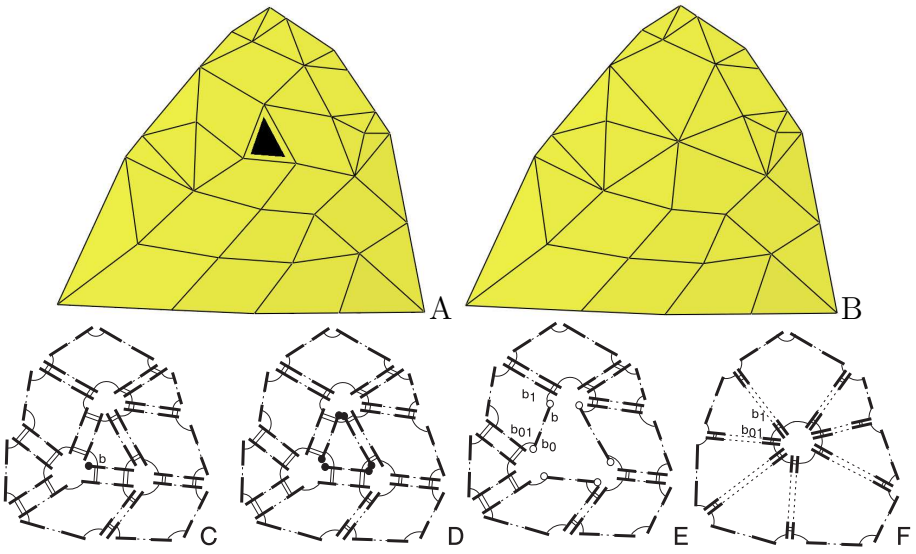


Figure 2.21 *(A) Un polygone va être résorbé. (B) Le résultat. En détail : (C) Le polygone contenant le brin b va être résorbé. (D) Les brins du polygone sont sélectionnés (tête noire). (E) On a stocké les brins (tête blanche) reliés par α_2 aux brins à tête noire puis on a supprimé ces derniers. Pour chaque arête b,b_0, on relie les brins b_1 et b_{01} entre eux par α_1 et on supprime b et b_0. Aucune cellule n'est dégénérée, l'opération est terminée.*

b_4 et b_5 au brin b_6 par α_1 ainsi qu'une autre arête constituée des brins b_2 et b_3 au brin b_1. Ces deux arêtes seront ensuite cousues entre elles par α_2. Auparavant, on aura vérifié si $\alpha_0(b_1)$ a déjà été traité. Si c'est le cas, il faudra coudre le brin $\alpha_0 \circ \alpha_1 \circ \alpha_0(b_1)$ avec b_3 par α_1. De la même manière, si le brin $\alpha_0(b_6)$ a déjà été traité, il faudra coudre le brin $\alpha_0 \circ \alpha_1 \circ \alpha_0(b_6)$ avec b_5 par α_1. De ce fait, l'arête double est fermée à ses extrémités.

Exemple 2 : scission d'un polyèdre d'un solide

Le polyèdre sélectionné va être divisé en n pyramides (n étant le nombre de polygones du polyèdre) (figure 2.24 à droite).

On crée une arête volumique qui va relier chaque sommet du polyèdre en son barycentre.

Scission d'une m-cellule d'un n-maillage

C'est la subdivision barycentrique détaillée dans la section 2.5.

ResorptionCellule(g : n-G-Carte, c : m-cellule)
Calculer le barycentre de la m-cellule
Stocker les brins connectés à la m-cellule par α_m dans Liste
Découdre et Supprimer les brins de la m-cellule
Pour i **de** m-1 **à** 1 **Faire**
Stocker les brins connectés aux i-cellules que l'on peut atteindre par les brins de Liste par α_i dans Liste'
Découdre et Supprimer les i-cellules
Vider Liste et echanger les deux listes
FinPour
Restaurer les liens α_1
Suppression des éventuelles cellules dégénérées
FinResorptionCellule

Figure 2.22 *Résorption de cellules avec g une G-Carte de dimension n et c une cellule de g de dimension m ($m \leq n$).*

2.3.5 Fusion de cellules

Fusionner les n-cellules d'un n-maillage le long d'une i-cellule consiste à enlever la i-cellule du n-maillage **sans faire de trous**.

Exemple 1 : fusion de polygones le long d'un noeud d'une surface

Les polygones incidents au noeud vont fusionner (figure 2.25).

Pour réaliser la fusion, il faut stocker dans une liste L tous les brins b' cousus par α_0 aux brins b du noeud sélectionné (figure 2.25-E), supprimer tous les brins b du noeud, puis pour chaque brin b' stocké dans L, si $\alpha_2(b')$ est un brin stocké, alors relier par α_1 les brins $\alpha_1(b')$ et $\alpha_2(\alpha_1(b))$. Isoler $\alpha_2(b')$ et b' par α_1, enfin supprimer les brins b' stockés dans L (figure 2.25-F).

Exemple 2 : fusion de polygones le long d'une arête d'une surface

Les deux polygones incidents à l'arête vont fusionner (figure 2.26).

Pour cela, il faut stocker tous les brins de l'arête (figure 2.26-D), pour tous les brins stockés, si α_2(b) est un brin stocké, alors relier par α_1 les brins α_1(b) et $\alpha_2(\alpha_1$(b)) (figure 2.26-E), isoler α_2(b) et b par α_1 et finalement supprimer les brins stockés (figure 2.26-F).

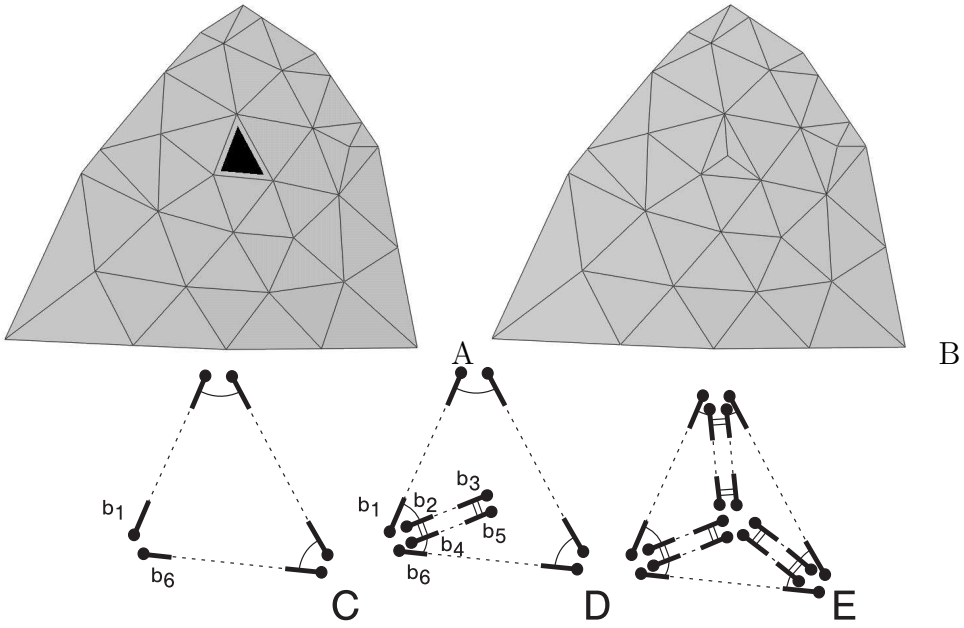


Figure 2.23 *(A) Une surface dont un polygone va être divisé. (B) Le résultat.*

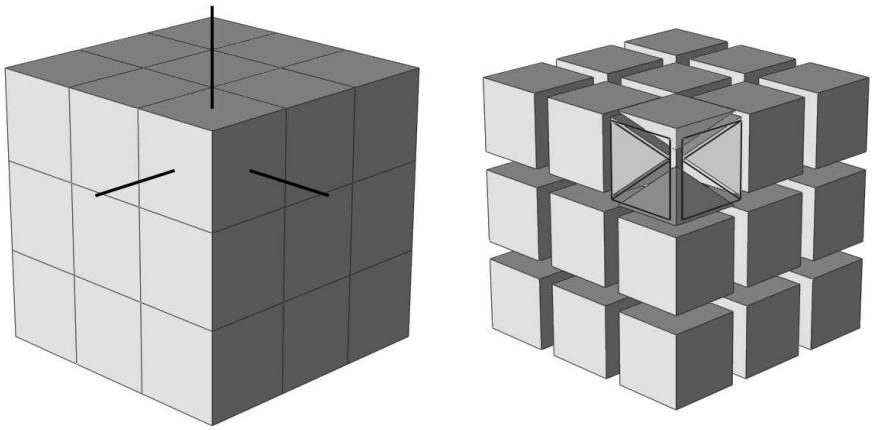


Figure 2.24 *Un solide dont le polyèdre pointé par les trois axes noirs va être divisé.*

Exemple 3 : fusion de polygones le long d'un polygone

Cette opération consiste à fusionner un polygone avec lui-même. En d'autres termes, cette opération ne fait rien. Cette opération est l'inverse de la résorption d'un noeud vers un noeud.

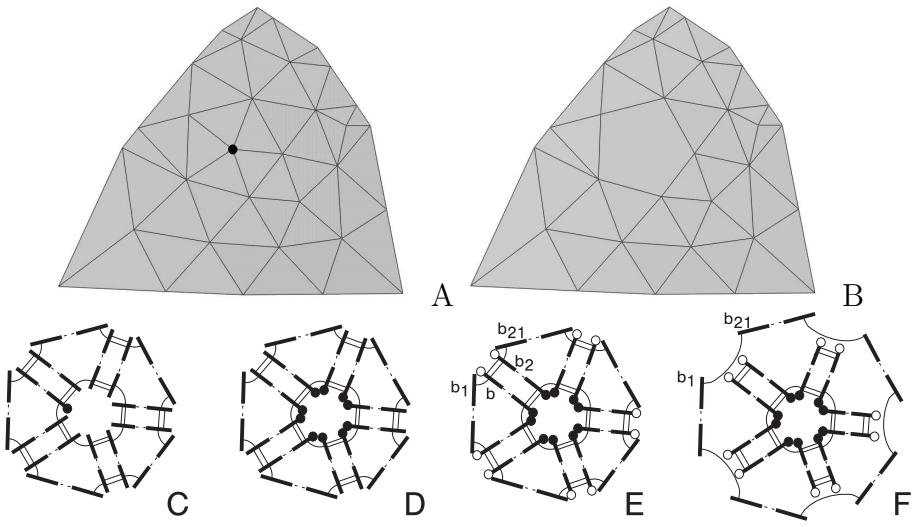


Figure 2.25 *A. Une surface dont un noeud central va être effacé. B. Le résultat. En détail : (C) Le noeud contenant le brin à tête noire va être effacé. (D) Tous les brins du noeud sont sélectionnés (brins à tête noire). (E) A partir de ces brins, on stocke tous les brins reliés par α_0 (brins à tête blanche). Pour chaque couple de brin (b b_2), (E) on relie leurs brins b_1 et b_{21} entre eux par α_1 et on déconnecte b et b_2. (F) Finalement, il reste à supprimer les brins sélectionnés, c'est-à-dire les brins à tête noire et les brins à tête blanche.*

Fusion de n-cellules d'un n-maillage le long d'une m-cellule

L'opération de fusion correspond au dual de l'opération de résorption [**?**]. L'algorithme de cette opération se trouve figure 2.27.

2.3.6 Récapitulatif

Pour finir, un récapitulatif de toutes les fonctions génériques citées ci-dessus se trouve sur la figure 2.28. Chaque fonction prend en argument deux objets. Le premier est le n-maillage M sur lequel on modifie la m-cellule c.

2.4 Opérations lourdes

Dans cette section, nous allons aborder les nouvelles opérations que nous avons implanté. La première est une opération connue qui consiste à calculer le dual d'un objet. L'apport que nous avons effectué consiste à obtenir un bord lisse. La deuxième opération est l'opération d'extrusion qui va permettre de construire des grilles non-structurées qu'utiliseront les ingénieurs réservoir. La dernière opération regroupe différentes subdivisions et insertions de maillages.

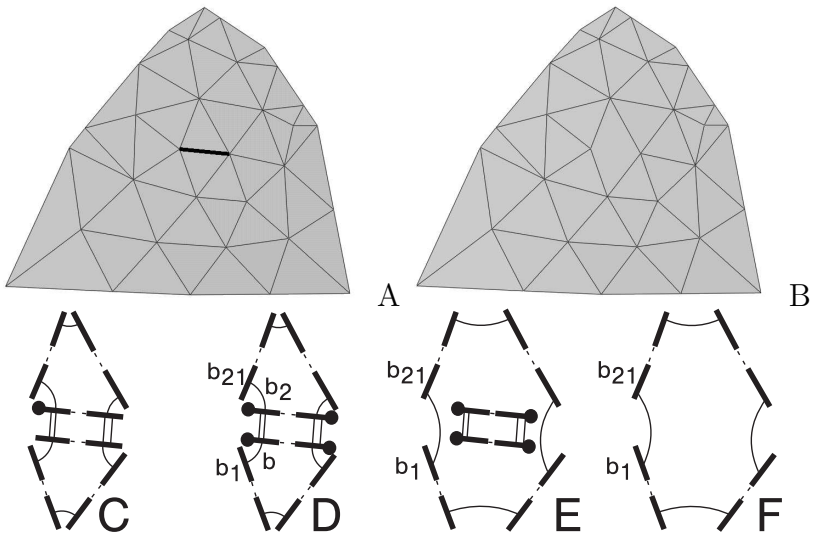


Figure 2.26 *A. Une surface dont une arête centrale va être effacée. B. Le résultat. En détail : (C) L'arête contenant le brin à tête noire va être effacée. (D) Tous les brins de l'arête sont stockés. Pour chaque couple de brin (b b_2), (E) on relie leurs brins b_1 et b_{21} entre eux par α_1 et on déconnecte b et b_2. (F) Finalement, il reste à supprimer l'arête sélectionnée.*

FusionCellule($\mathcal{G}$: n-G-Carte, c : m-cellule)
  Stocker les brins connectés à la m-cellule par α_m dans L
  Découdre et Supprimer les brins de la m-cellule
  Pour i **de** m+1 **à** n-1 **Faire**
    Stocker les brins connectés aux i-cellules de L par α_i
    Découdre et Supprimer les i-cellules que l'on peut atteindre par les brins de L
  FinPour
  Restaurer les liens α_{n-1}
FinFusionCellule

Figure 2.27 *Fusion de cellules avec $\mathcal{G}$ une G-Carte de dimension n et c une cellule de $\mathcal{G}$ de dimension m ($m \leq n$).*

2.4.1 Dualité

On définit le dual topologique d'une G-Carte comme suit :

- chaque i-cellule du primal devient une (n-i)-cellule du dual.

- les relations des j-cellules par rapport à la i-cellule deviennent des relations de (n-j)-cellules par rapport à la (n-i)-cellule.

Fonctions M : n-maillage c : m-cellule	Description
DestructionCellule(M, c)	enlève du maillage M une cellule c. En fait, toutes les n-cellules incidentes à c seront supprimées.
RésorptionCellule(M, c)	identifie les n-cellules incidentes à c en un point.
ScissionCellule(M, c)	scinde la m-cellule en k m-cellules; k étant le nombre de m-1-cellules incidentes à c.
FusionCellule(M, c)	efface la cellule c du maillage M.

Figure 2.28 *Fonctions légères*

Pour une surface triangulée, chaque triangle de la surface devient un noeud du dual. Chaque arête qui se trouve entre deux triangles devient une arête qui relie les deux noeuds du dual correspondant à ces deux triangles. Chaque noeud devient un polygone.

Pour un solide tétraédrisé, chaque tétraèdre devient un noeud du dual. Chaque face qui se trouve entre deux tétraèdres devient une arête qui relie les deux noeuds du dual correspondant à ces deux faces. Chaque arête devient une face. Chaque noeud devient un polyèdre.

L'algorithme du calcul du dual se généralise en dimension n.

```
Procédure Dual(G : n-G-Carte)
   Pour chaque brin b de G Faire
      Pour i de 0 à n/2 Faire
         échanger α_i(b) avec α_{n-i}(b)
      FinPour
   FinPour
FinDual
```

Bien qu'il soit généralisable en toute dimension, l'algorithme du calcul du dual reste très simple. Les structures de données classiques [**?**, **?**] requièrent des algorithmes plus complexes qu'il est difficile de généraliser, du fait de la multiplicité des éléments à manipuler.

D'un point de vue géométrique, chaque n-cellule c est transformée en sommet.

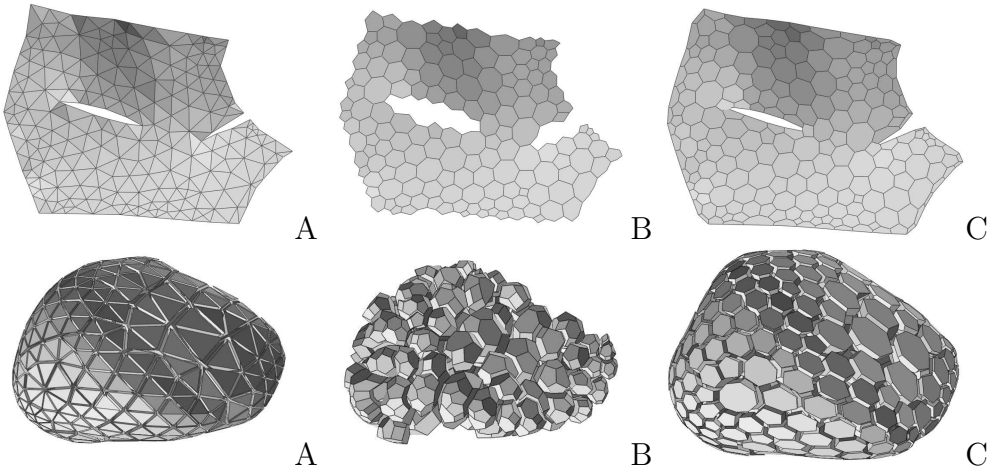


Figure 2.29 *A. Surface triangulée et solide tétraédrisé. B. Duaux : Les cellules dégénérées du bord ont été supprimées. C. Duaux à bord* lisse.

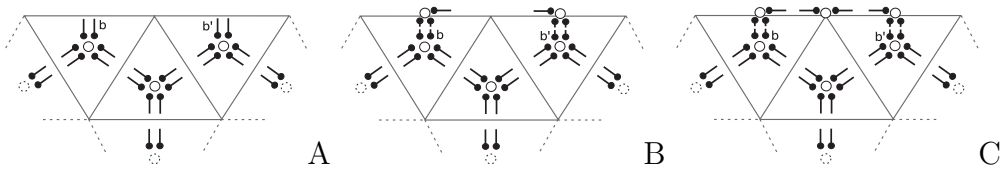


Figure 2.30 *A. En traits fins, la surface triangulée primale. Les brins du dual sont montrés en traits gras. Les brins b et b′, entre autres, sont libres en* α_0. *B. Première approche : on* prolonge *les arêtes de ces deux brins jusqu'au centre de l'arête du primal (jusqu'au barycentre du polygone du primal pour un volume) puis on ferme le polygone. C. Deuxième approche : on prolonge comme précédemment et on garde également le sommet existant du bord du primal. De cette manière, le bord est entièrement préservé contrairement à la première approche qui* coupe *les coins du bord (cette méthode n'a pas été généralisée pour les volumes).*

Les coordonnées de ce sommet peuvent correspondre par exemple au barycentre de la cellule ou au centre de la n-sphère circonscrite de la n-cellule c (dual de voronoï).

Les cellules du bord d'un objet dual sont dégénérées : dans la théorie, elles possèdent des sommets qui se trouvent à l'infini. Avec les G-Cartes, ces cellules sont caractérisées par des brins qui sont libres en α_0. Ils correspondent aux brins du primal qui étaient au bord (libre en α_n). Une première approche consiste à supprimer ces cellules. Nous obtenons un dual avec un bord *non lisse* (cf figures 2.29-B). La deuxième approche va consister à *fermer* ces cellules afin de préserver au mieux le bord initial de l'objet (voir figure 2.29-C). Pour cela, nous allons prolonger les arêtes ouvertes (symbolisées par les brins b libres en α_0) en

ajoutant un brin que l'on va coudre par α_0 (voir figure 2.30-A). Le plongement de ce brin correspond à la moyenne barycentrique de l'ancienne $(n-1)$-cellule du primal qui contenait le brin b. Ensuite, il suffit de fermer progressivement la G-Carte pour les dimensions 2 jusqu'à n (voir figure 2.30-B). L'opération de fermeture a été décrite dans la section 2.1.5 page 55. Une autre possiblité de fermeture consiste à garder les noeuds du bord du primal comme le montre la figure 2.30-C.

L'opération du dual nous permet de construire de nouvelles surfaces à partir des surfaces triangulées standards qui correspondent à un horizon d'un modèle. Ces nouvelles surfaces possèdent des polygones et ont à peu près le même bord que leur primal. L'opération d'extrusion décrite dans la section suivante utilise ces surfaces.

2.4.2 Extrusion

En géologie, l'utilisateur va vouloir attacher des propriétés en différents points dans son modèle. Les régions seront remplies, par exemple, par des grilles.

Des propriétés vont être rattachées aux noeuds de ces grilles. Ces propriétés ne sont pas connues en tout point du modèle (voir propriétés discrètes en section 1.1.5 page 17) : il est nécessaire d'interpoler ces données aux points dépourvus de propriétés. Une méthode d'interpolation utilisée en géostatistique s'appelle la méthode de *krigeage*. Nous n'allons pas détailler ici cette méthode. Il suffit de savoir que cette méthode se base sur des calculs de distances entre des points. Ces distances permettront de faire une moyenne pondérée pour obtenir un résultat.

Cependant, les grilles peuvent être déformées de manière à *suivre* la stratigraphie définie par les couches (figure 2.31). Utiliser une distance euclidienne dans une telle grille n'est pas concevable, il faudrait pouvoir retourner dans l'espace de dépôt, non déformé.

Pour résoudre ce problème, la solution consiste à plonger cette grille dans un espace qui la rend régulière géométriquement parlant de manière à pouvoir utiliser les distances euclidiennes (figure 2.31 en bas).

Du fait de la structure de la grille, cette opération est triviale. Les cellules sont ordonnées selon 3 axes (U, V et W), chaque axe possédant un nombre de noeuds n_i connu. Les nouvelles coordonnées des noeuds sont données de la manière suivante :

- les noeuds qui se trouvent aux extrémités des axes de la grille auront respectivement pour valeur 0 et 1.
- les noeuds intermédiaires auront une valeur linéaire qui correspond à l'emplacement de la cellule sur son axe :

$$\frac{n_j}{N_i} \text{ avec } \begin{cases} n_j & : \text{ numéro de la cellule et} \\ N_i & : \text{ nombre de cellules pour l'axe i} \end{cases}$$

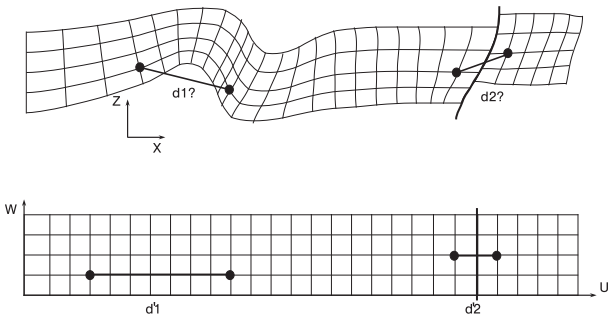


Figure 2.31 *Calcul de distance pour interpolation de propriétés : (En haut) calculer la distance entre deux points dans une couche stratigraphique plongée dans $\mathbb{R}^3$ ne peut pas se faire trivialement car le repère n'est pas euclidien. (En bas) En changeant l'espace de plongement, il est alors possible de calculer des distances.*

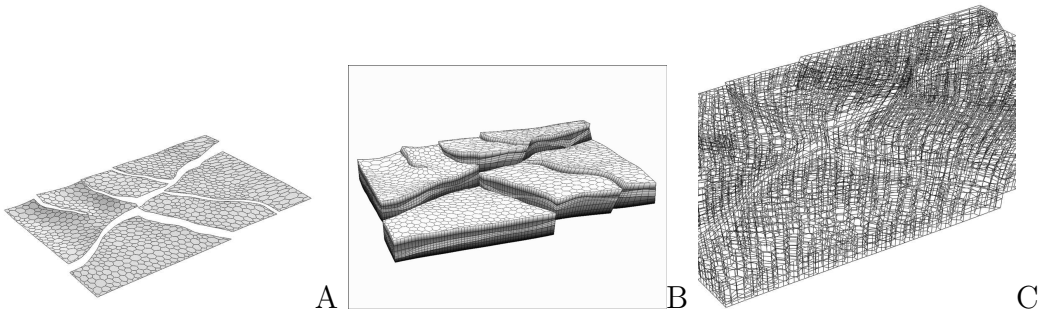


Figure 2.32 *(A) Horizon d'origine. (B) Grille finale. (C) Maillage de la grille.*

Une grille structurée n'est composée que d'une seule sorte d'élément, par exemple des cubes. Ces cubes sont organisés selon 3 axes que l'on appelle généralement U, V et W. Chaque cube est implicitement localisé par trois index (i, j et k). Du fait de cette structure, ces grilles ne sont pas très flexibles mais sont très faciles à utiliser pour faire des calculs en géostatistique.

Les grilles non-structurées sont composées de cellules polyédriques arbitraires. Cette flexibilité nous autorise à mieux suivre la stratigraphie des couches, à densifier localement certaines cellules, à obtenir des objets complexes avec peu de cellules, ...

Nous proposons de fournir une grille non structurée qui possède des propriétés U, V et W comme définies précédemment. L'opération d'extrusion va nous permettre de créer de telles grilles volumiques non-structurées.

A partir d'une surface (triangulée ou non) et d'un ensemble de vecteurs associés à chaque sommet de la surface, l'opération d'extrusion consiste à dupliquer la surface d'origine, de la déformer par le champ de vecteurs, puis de relier ces deux surfaces pour former des cellules volumiques.

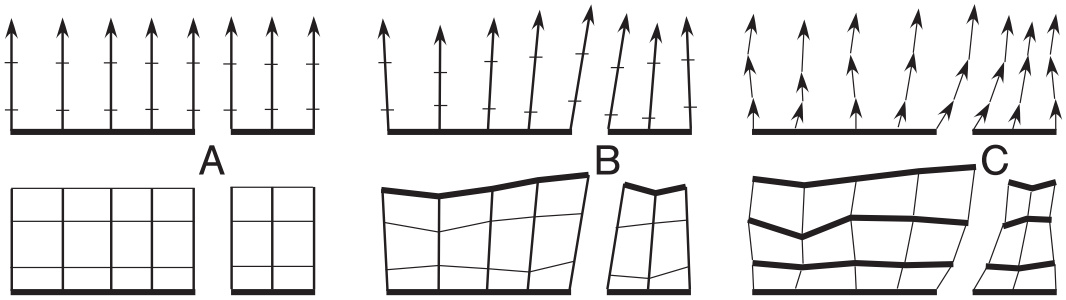


Figure 2.33 *(A) vecteur constant. (B) carte d'épaisseur et horizon supérieur connus. (C) ensemble de couches connu.*

Ces vecteurs sont définis différemment suivant les données que détient le géologue :

- à partir d'un seul horizon, le géologue peut utiliser une épaisseur constante ou bien utiliser une carte d'épaisseur obtenue à partir de données de puits (figure 2.33-A), les vecteurs sont tous parallèles les uns par rapport aux autres.

- à partir de deux horizons, une carte d'épaisseur est déduite (figure 2.33-B). Les vecteurs sont définis entre la surface du bas et la surface du haut.

- avec des horizons intermédiaires, il est possible de représenter des failles listriques (faille courbe montrée sur la figure 2.33-C). Les vecteurs sont décomposés en plusieurs segments permettant d'approcher la courbe de la faille.

De plus, chaque couche peut être subdivisée en sous-couches correspondant à un *style de couche*. Cette méthode est intéressante lorsque des informations sont plus abondantes dans une couche donnée. On compte trois moyens de subdivisions :

- les sous-couches peuvent être parallèles à l'horizon du bas comme le montre la figure 2.34 de gauche, la couche est alors tronquée au niveau de l'horizon supérieur (on parle alors de surface d'érosion),

- les sous-couches sont proportionnelles aux horizons de la couche principale comme le montre la figure 2.34 du milieu, l'épaisseur des sous-couches varie proportionnellement entre les deux horizons, (c'est le cas de sédimentation avec accomodation lente)

- ou les sous-couches peuvent être parallèles à l'horizon du haut comme le montre la figure 2.34 de droite (sédimentation avec accomodation rapide).

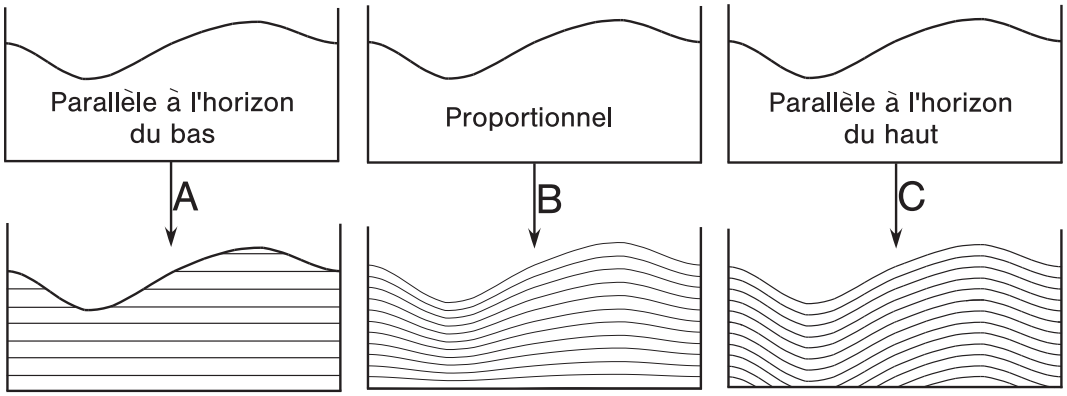


Figure 2.34 *Création de sous-couches : (A) parallèle à l'horizon du bas (B) proportionnel aux deux horizons (C) parallèle à l'horizon du haut*

Certaines cellules de la grille sont dégénérées (cellules plates, cellules avec des faces réduites à une arête, faces de cellules avec des arêtes réduites à un point), par exemple, lors de la modélisation de *discontinuités* (figure 2.35). L'agencement des couches les unes par rapport aux autres est contrôlé par des paramètres sédimentologiques, tels que la variation relative du niveau marin, la pente du talus sur lequel se déposent les sédiments, ...

Dans un modèle structural classique, le nombre de cellules dans les directions U, V et W doit être constant. Pour cette raison, les cellules dégénérées ne peuvent pas être supprimées. Ces cellules restent dans le modèle mais sont stockées en tant que cellules non existantes. Ceci permet de définir de manière cohérente le concept de voisinage et demande la maintenance d'une liste notifiant des connexions spéciales entre cellules. Par exemple, si deux cellules sont plates entre une cellule c_1 et une cellule c_2, la liste de connexions spéciales spécifiera que la cellule c_1 est en connexion directe avec la cellule c_2.

Etant donné que nos grilles possèdent une souplesse au niveau des formes des cellules, nous pouvons supprimer les cellules dégénérées et ne stocker que les cellules réelles.

Pour utiliser ce type de grilles, le géostatisticien a besoin de faire des calculs de distance entre différents points. Nous avons vu au début de cette section qu'il n'est pas possible de faire ces calculs avec la géométrie réelle de cette grille. Il va falloir la transformer en la plongeant dans un espace régulier. Deux étapes sont nécessaires à ce processus :

- La première étape consiste à plonger l'horizon de départ dans un espace régulier (figure 2.36-A). De tels travaux sont détaillés dans [**?**]. Nous avons pour chaque sommet de la surface, son plongement U, V correspondant (voir la figure 2.36-A).

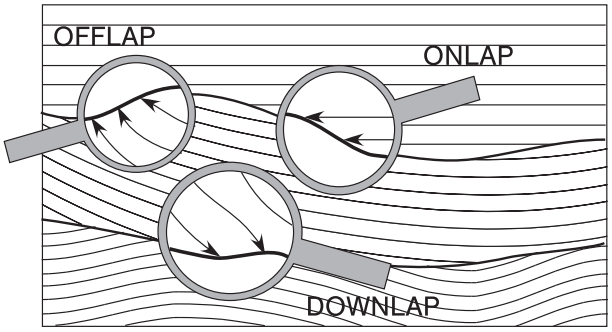


Figure 2.35 *Différents types de discontinuité.*

- La deuxième étape se déroule lors de l'extrusion. La valeur W de chaque noeud est donnée suivant le nombre de surfaces formant l'extrusion et le numéro de la surface (voir la figure 2.36-B).

Ces nouvelles grilles peuvent être utilisées en géostatistique comme le prouve la figure 2.37.

2.4.3 Insertion de maillages

Insertion d'une ligne entre deux bords de lignes existants

Insérer une ligne de n segments de long entre 2 points consiste à créer n arêtes reliées les unes aux autres entre ces deux points.

La figure 2.38 nous montre un exemple de ligne avant insertion, en cours d'insertion de segments et le résultat final. Une primitive CreerArete qui va construire une arête composée de deux brins reliés entre eux par α_0 et qui retourne un des deux brins, accélerera le processus.

L'algorithme d'insertion est donc le suivant, il faut créer n arêtes en appelant la fonction CreerArete et relier au fur et à mesure les arêtes entre elles par α_1. On aura cousu la première arête au premier brin de départ. Finalement, le brin libre en α_1 de la dernière arête sera cousu par α_1 au deuxième brin de départ.

Insertion d'une surface entre deux bords de surfaces existants

Insérer une surface de n segments de long sur p segments de large entre 2 bords constitué de p segments consiste à créer $n * p$ faces à base carrées reliées les unes aux autres pour former une grille 2D.

La figure 2.39 nous montre un exemple de surface avant insertion, en cours d'insertion de carrés et le résultat final. Une primitive CreerCarre qui va construire un carré en appelant par exemple quatre fois la primitive CreerArete décrite dans le paragraphe précédent, accélerera le processus.

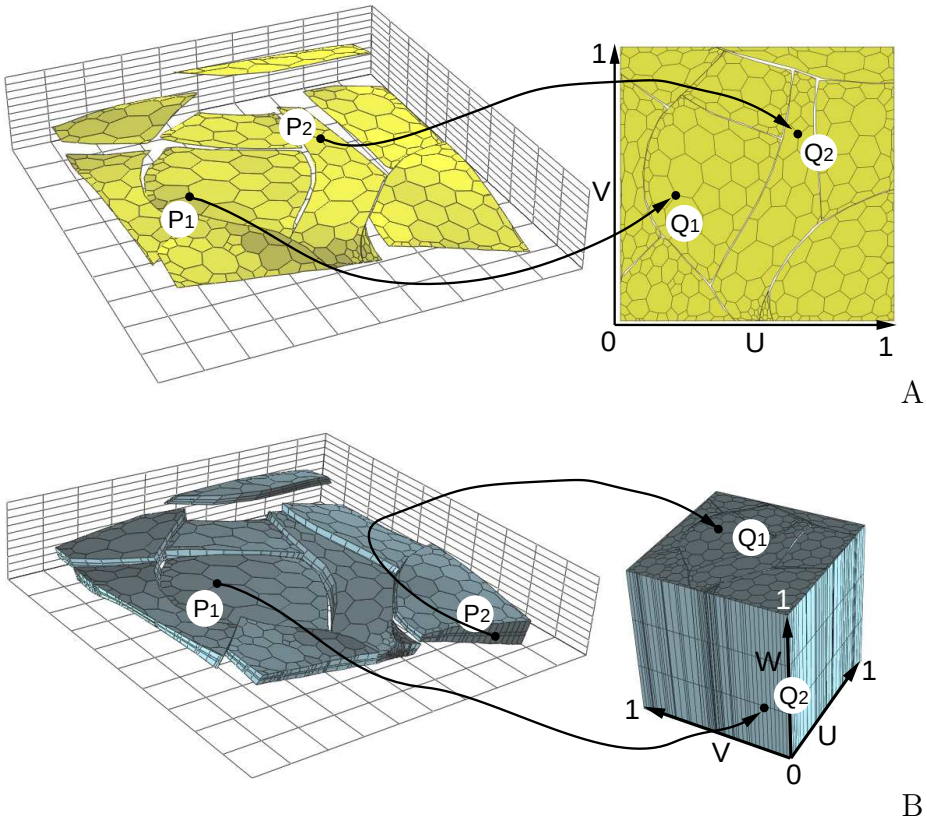


Figure 2.36 *A. Horizon de départ paramétrisé. B. Grille finale paramétrisée.*

L'algorithme d'insertion se fait en plusieurs étapes répétitives. La première étape consiste à créer n carrés en appelant la fonction CreerCarre et relier au fur et à mesure les carrés entre eux par α_2 de manière à former une colonne de carrés. On prendra soin de coudre aussi les carrés aux carrés de départ. Puis, on répète encore cette étape $p-1$ fois en prenant soin de coudre les nouveaux carrés à ceux créés juste précedemment. Finalement, les derniers carrés seront cousus par α_2 aux autres carrés de départ.

Insertion d'un solide entre deux bords de solides existants

Insérer un solide de n segments de long, p de large et q de haut entre 2 bords constitué de p segments de long et q de large consiste à créer $n * p * q$ solides à base cubique reliés les uns aux autres pour former une grille 3D.

La figure 2.40 nous montre un exemple de solide avant insertion, en cours d'insertion de cubes et le résultat final. Une primitive CreerCube qui va construire un cube en appelant par exemple six fois la primitive CreerCarre décrite dans le paragraphe précédent, accélerera le processus.

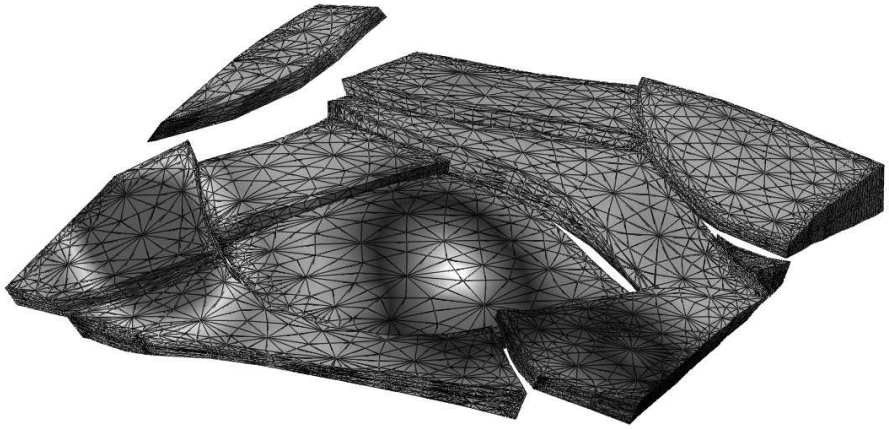


Figure 2.37 *Etude géostatistique sur un cas réel. Des propriétés de perméabilité ou porosité sont stockées sur les noeuds de ce maillage volumique non-régulier. Ces propriétés ont été interpolées sur les surfaces de l'objet. On observe des continuités ellipsoidale en certaines zones.*

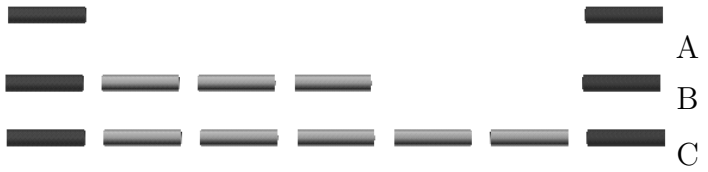


Figure 2.38 *(A) Insertion entre deux lignes existantes. L'extrémité utilisée de chaque ligne est dessinée en foncé. (B) Insertion en cours. Quelques segments ont été créés. (C) Insertion terminée. Une ligne relie maintenant les deux anciennes lignes.*

L'algorithme d'insertion se fait en plusieurs étapes répétitives. La première étape consiste à créer n cubes en appelant la fonction CreerCube et relier au fur et à mesure les cubes entre eux par α_3 de manière à former une colonne de cubes. On répète cette étape $p-1$ fois en prenant soin de coudre les nouveaux cubes aux n cubes créés juste précedemment. On aura cousu ces $n*p$ cubes aux cubes de départ. On répète ces deux étapes $q-1$ fois en prenant soin de coudre les nouveaux cubes aux $n*p$ cubes créés juste précedemment. Finalement, les derniers $p*q$ cubes seront cousus par α_3 aux autres cubes de départ.

2.4.4 Subdivision de maillages

Les surfaces de subdivision ont été simultanément mises en évidence en 1978 par Doo et Sabin [**?**], et Catmull et Clark [**?**].

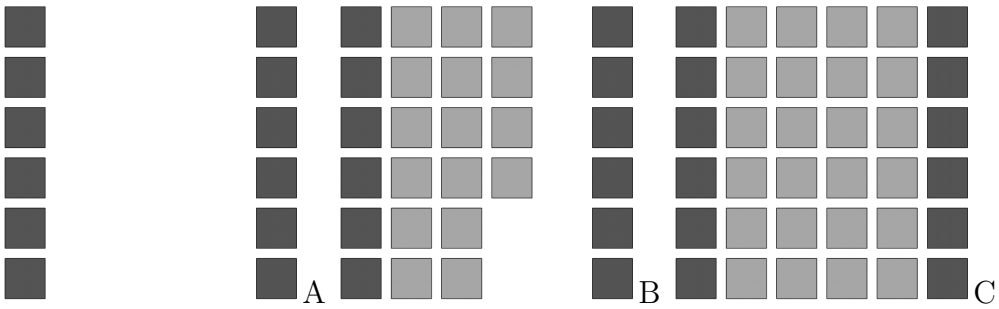


Figure 2.39 *(A) Insertion entre deux surfaces existantes. L'extrémité utilisée pour chaque surface est dessinée en foncé. (B) Insertion en cours. Quelques carrés ont été créés. (C) Insertion terminée. Une surface quadrillée relie maintenant les deux anciennes surfaces.*

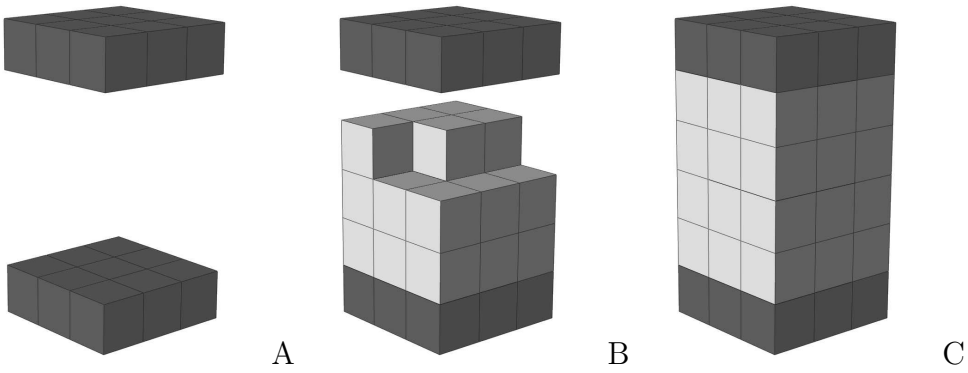


Figure 2.40 *(A) Insertion entre deux solides existants. L'extrémité utilisée de chaque solide est dessinée en foncé. (B) Insertion en cours. Quelques cubes ont été créés. (C) Insertion terminée. Un solide quadrillé relie maintenant les deux anciens solides.*

Le principe est simple. En partant d'un maillage initial M^0 donné, des règles de raffinement simples sont définies pour passer à un maillage d'ordre supérieur. Les coordonnées des sommets du maillage M^{i+1} sont calculées en fonction de ceux du maillage M^i de telle sorte que la suite des maillages converge vers une surface lisse M^∞. Une étude plus complète de cette famille de surfaces se trouve dans [**?**].

Dans cette section, nous allons étudier différentes subdivisions. Chacune de ces subdivisions permet d'obtenir des maillages plus ou moins réguliers ayant certaines caractéristiques : par exemple, le degré moyen des sommets accroît rapidement ou certaines cellules possèdent un nombre de sommet de plus en plus important. Suivant l'utilisation que l'on veut en faire, le géologue pourra subdiviser son maillage pour en obtenir un possèdant beaucoup de noeuds afin d'attacher plus d'informations géologiques.

Différentes subdivisions

La première est dite *subdivision des coins* : elle consiste à couper tous les coins des n-cellules du n-maillage et de les recoller à la n-cellule centrale. Avec une surface triangulée, le maillage reste homogène et le degré moyen des sommets est de 6. Cette subdivision permet d'accroitre le nombre de triangles de manière régulière sans pour autant obtenir des triangles *indésirables* comme des triangles plats. En utilisant des surfaces à base polygonale, cette subdivision crée un maillage dense autour des arêtes des cellules initiales (voir figures 2.42-B et 2.43-B).

La deuxième est dite *subdivision en hexaèdres* et consiste à diviser chaque arête en deux et relier ce nouveau point au point central de chaque cellule. Pour des polyèdres, il faut d'abord subdiviser les faces en hexaèdres puis relier les points centraux des faces au point central du polyèdre. Pour des surfaces, le maillage est hexaédrique et le degré moyen des sommets est de 4 (voir figures 2.42-C et 2.43-C).

La troisième est dite *subdivision barycentrique* et consiste à créer un nouveau noeud pour chaque type de cellule en son barycentre et de relier ces noeuds au noeud central de la cellule. Cette subdivision a la particularité d'accroitre rapidement le degré moyen des sommets (voir figures 2.42-D et 2.43-D).

La quatrième est dite *subdivision de Gregory*. Cette subdivision consiste à ajouter des sommets dans un polygone de manière à obtenir la configuration des points de contrôle des carreaux de Gregory. Grâce à cette subdivision, il est possible d'obtenir des polygones possédant un grand nombre de côtés comme le montre les figures 2.42-E.

La cinquième est dite *subdivision en grilles* et consiste à requadriller les cellules. Cette opération prend en compte le niveau de quadrillage demandé. Par exemple, la figure 2.41 décompose une subdivision à 5 niveaux. La première étape consiste à subdiviser chaque arête en 5 segments (figure 2.41-B). La deuxième étape construit un polygone plus petit à l'intérieur du polygone considéré puis relie ses points aux points du polygone extérieur. Finalement, l'opération se réitère avec le polygone intérieur jusqu'à obtenir un point ou un polygone qui possède le même nombre de côté que le polygone de départ. Pour des polyèdres, il faut étendre cette définition aux faces (voir figures 2.42-F et 2.43-E).

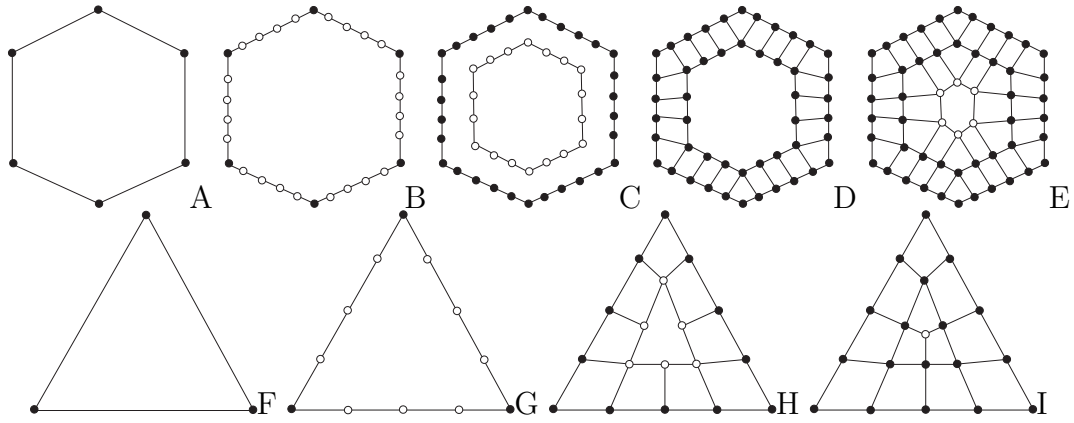


Figure 2.41 *Subdivisions en quadrilatères. Les sommets noirs sont les sommets de départ ou les sommets créés à l'étape précédente. Les sommets blancs sont les sommets créés à l'étape courante. (A) Subdivision d'un polygone en 5 niveaux. (B) La première étape découpe les arêtes en 5 segments. (C) La deuxième étape construit un polygone plus petit à l'intérieur du polygone courant. (D) La troisième étape relie les deux polygones. (E) Finalement, les deux dernières opérations sont répétées jusqu'à l'obtention d'un polygone central possédant le même nombre de côtés que le polygone de départ. (F) Subdivision d'un triangle en 4 niveaux. (G) La première étape découpe les arêtes en 4 segments. (H) Les deuxième et troisième étapes construisent un triangle plus petit à l'intérieur du triangle courant et le relie avec ce dernier. (I) Finalement, avec une subdivision de niveau pair, on obtient un sommet central.*

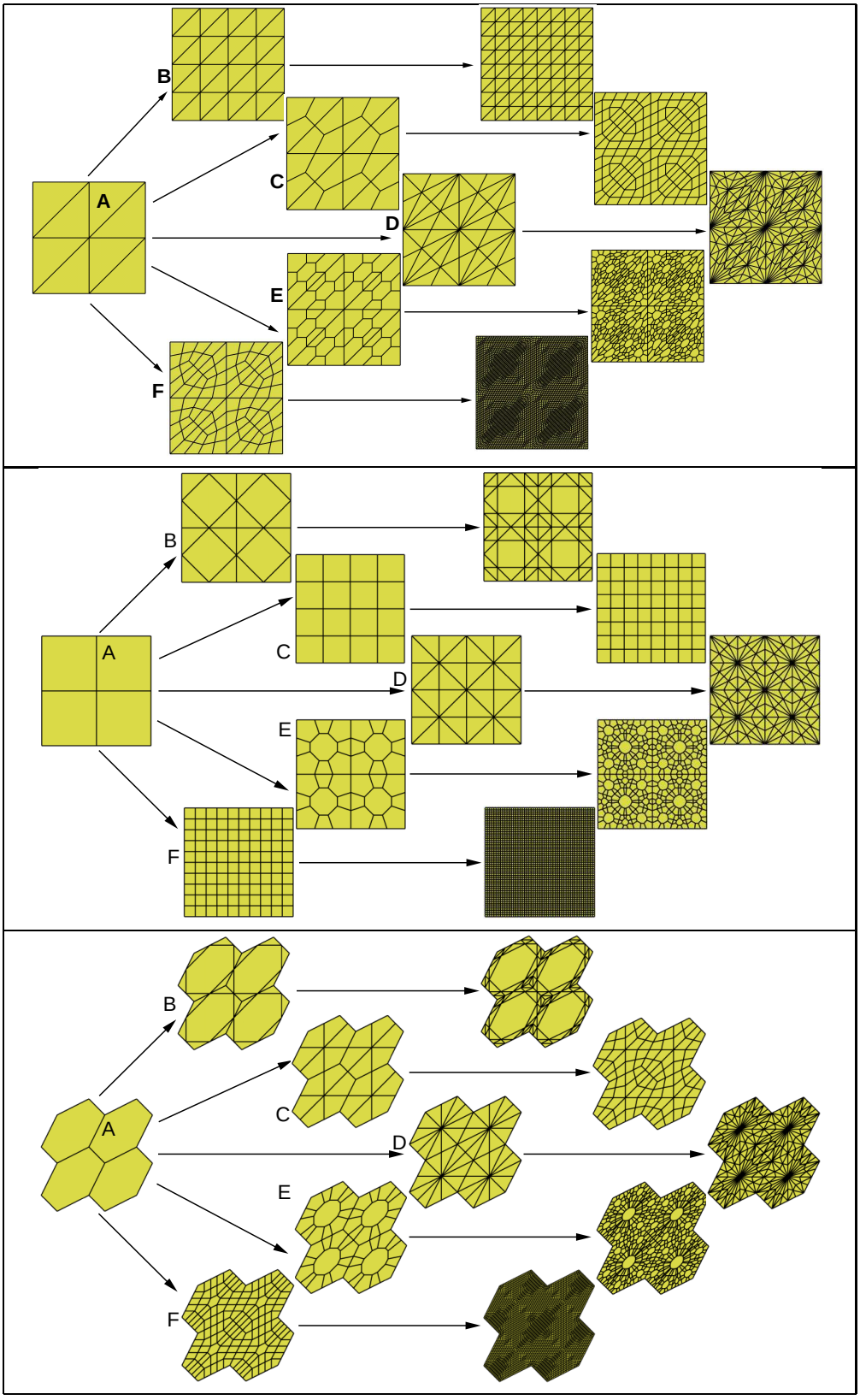


Figure 2.42 *Différentes subdivisions : A. Surface d'origine. B. Subdivision des coins. C. Subdivision en héxaèdres. D. Subdivision barycentrique. E. Subdivision version Gregory. F. Subdivision en quadrilatères (5 niveaux). Au milieu, une seule subdivision, à droite, deux subdivisions.*

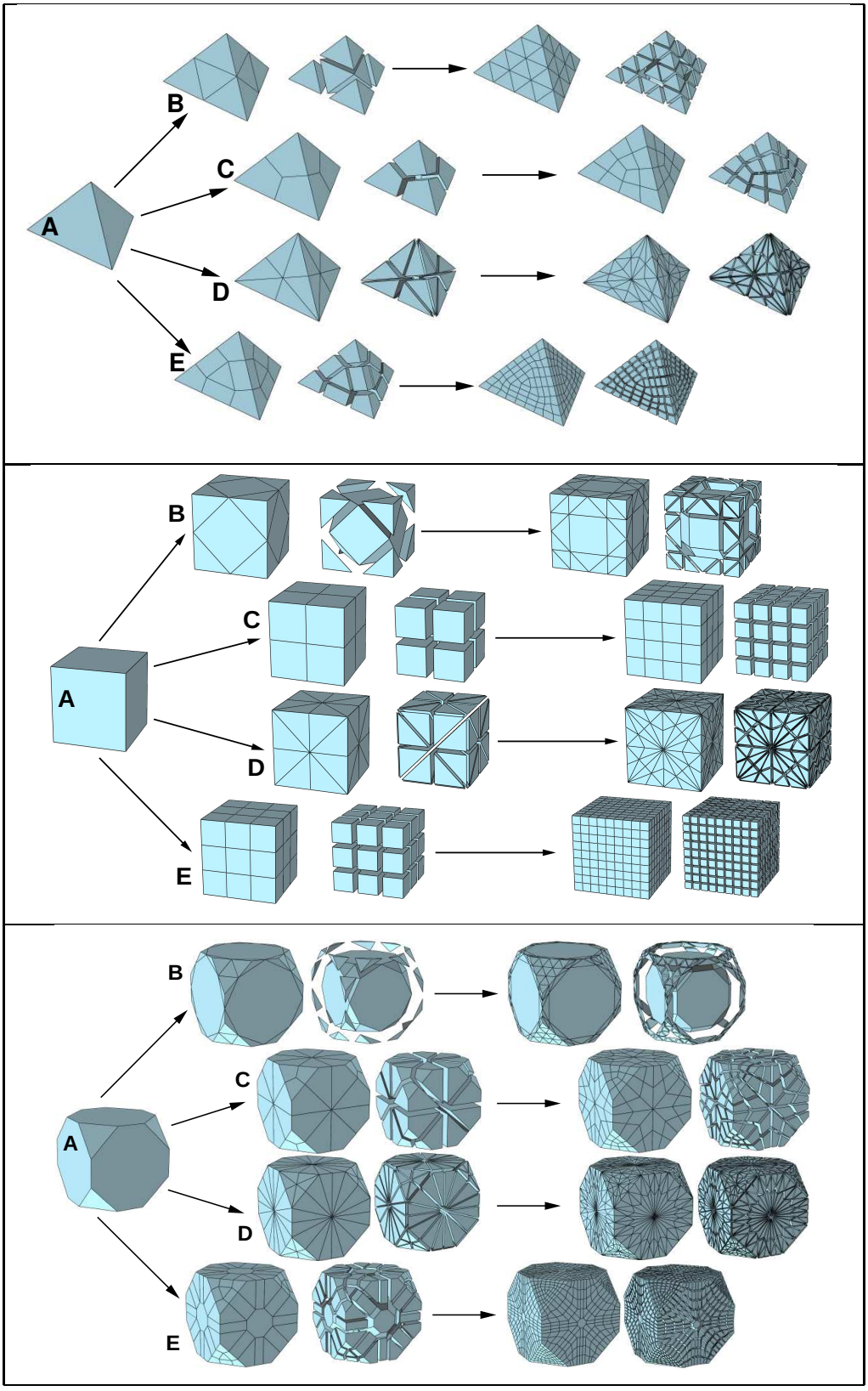


Figure 2.43 *Différentes subdivisions : A. Solide d'origine. B. Subdivision des coins. C. Subdivision en hexaèdres. D. Subdivision barycentrique. E. Subdivision en quadrilatères (3 niveaux). Au milieu : un niveau de subdivision. A droite, deux niveaux de subdivisions.*

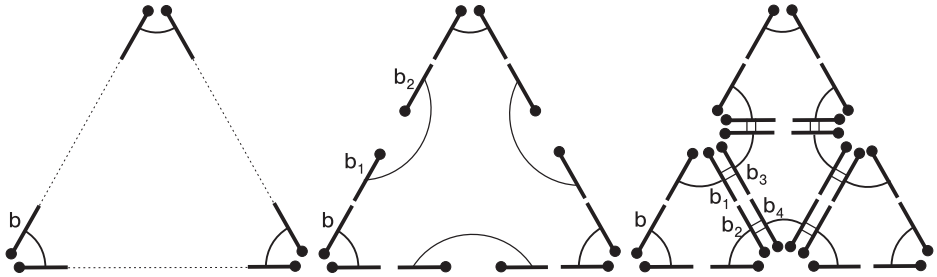


Figure 2.44 *Subdivision des coins. (A) Le triangle va être subdivisé. (B) La première étape divise les arêtes en deux. (C) La deuxième étape crée les arêtes qui forment le triangle du milieu.*

Exemple

Détaillons l'algorithme de la subdivision des coins pour une surface triangulée en se référant à l'algorithme donné en figure 2.45 et aux schémas de la figure 2.44.

- une première étape consiste à insérer un point au milieu de chaque arête :

  Soit b un brin de l'arête à subdiviser. L'étape consiste à créer deux brins b_1 et b_2 qui formeront le nouveau noeud de la subdivision comme le montre la figure 2.44-B. Ensuite, il reste à rattacher $\alpha_0(b)$ et b_2 par α_0 puis b et b_1.

- la deuxième étape consiste à ajouter une arête qui relie chaque couple de nouveaux points successifs entre eux :

  Soit b un brin d'un sommet du triangle initial. L'étape consiste à créer quatre brins b_1, b_2, b_3 et b_4 formant une arête double. C'est-à-dire que b_1 et b_2 sont liés entre eux par α_0 pour former une arête et il en est de même pour b_3 et b_4, de plus, les deux arêtes sont liées entre elles par α_2. Il reste ensuite à attacher les extrémités de cette arête aux nouveaux noeuds que l'on peut atteindre à partir de $\alpha_0(b)$ et $\alpha_0 \circ \alpha_1(b)$ comme le montre la figure 2.44-C.

Cet algorithme n'est pas très compliqué mais il a l'inconvénient de faire appel aux différentes cellules de l'objet. Il n'est donc pas trivial de construire l'algorithme équivalent en 3D et encore moins dans les dimensions supérieures.

Dans la section suivante, nous proposons une autre approche qui facilite la construction d'algorithmes en général et particulièrement ceux de subdivisions.

```
Procédure DiviserAretes(S : surface)
  Pour chaque arete A de la surface S Faire
    b,b1,b2 : Brin
    b ← brin(A)
    b1 ← CreerBrin(S)
    b2 ← CreerBrin(S)
    CoudreBrins(S,b1,b2,1)
    CoudreBrins(S,alpha(b,0),b2,0)
    CoudreBrins(S,b,b1,0)
    Si b n'est pas libre en 2 Faire
      b',b1',b2' : Brin
      b' ← alpha(b,2)
      b1' ← CreerBrin(S)
      b2' ← CreerBrin(S)
      CoudreBrins(S,b1',b2',1)
      CoudreBrins(S,alpha(b',0),b2',0)
      CoudreBrins(S,b',b1',0)
      RelierBrins(S,b1,b1',2)
      RelierBrins(S,b2,b2',2)
    FinSi
  FinPour
FinDiviserAretes

Procédure CreerTriangles(S : surface)
  Pour chaque ancien triangle T de la surface S Faire
    Pour chaque ancien Noeud N de T Faire
      b, b1, b2, b3, b4 : Brin
      b ← brin(N)
      b1 ← CreerBrin(S)
      b2 ← CreerBrin(S)
      b3 ← CreerBrin(S)
      b4 ← CreerBrin(S)
      CoudreBrins(S, b1, b2,0)
      CoudreBrins(S, b3, b4,0)
      CoudreBrins(S, b1, b3,2)
      CoudreBrins(S,alpha(alpha(b,0),1),b3,1)
      CoudreBrins(S,alpha(b,0),b1,1)
      CoudreBrins(S,b.alpha(alpha(alpha(b,1),0),1),b4,1)
      CoudreBrins(S,alpha(alpha(b,1),0),b2,1)
    FinPour
  FinPour
FinCreerTriangles
```

Figure 2.45 *Subdivision des coins par approche cellulaire*

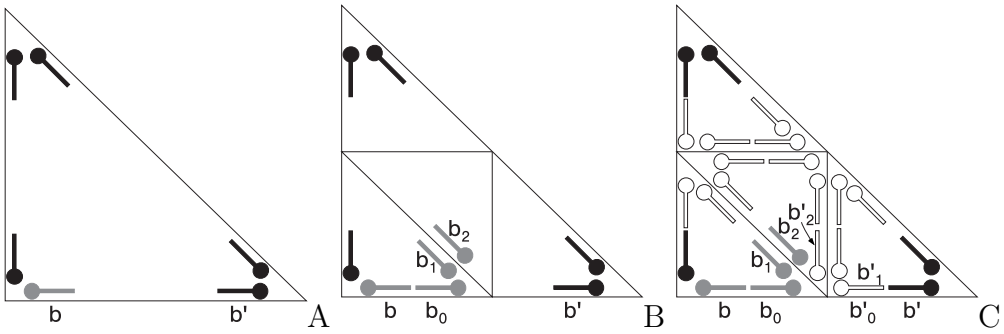


Figure 2.46 *(A) Subdivisons ce triangle d'origine. (B) Le traitement que nous allons appliquer à chacun de ces brins consiste à ajouter trois brins. Les trois brins créés sont respectivement appelés b_0, b_1 et b_2. Pour simplifier la visualisation, les liens n'ont pas été représentés sur la figure, cependant : le brin b_2 sera relié au brin b_1 par α_2, le brin b_1 sera relié au brin b_0 par α_1 et le brin b_0 sera relié au brin b par α_0. (C) Il reste maintenant à relier ces brins aux autres brins créés si ceux-ci existent déjà. En effet, nous remarquons que : On reliera par α_2 le brin b_0 si le brin b n'est pas libre en α_2 et si le brin $\alpha_0 \circ \alpha_2(b)$ est un nouveau brin. Dans ce cas, il sera relié au brin $\alpha_0 \circ \alpha_2(b)$. On reliera par α_0 le brin b_1 si le brin $\alpha_0 \circ \alpha_1(b)$ est un nouveau brin. Dans ce cas, il sera relié au brin $\alpha_1 \circ \alpha_0 \circ \alpha_1(b)$. On reliera par α_0 le brin b_2 si le brin b_1 a été relié par α_0. Dans ce cas, il sera relié au brin $\alpha_2 \circ \alpha_0(b_1)$. On reliera par α_1 le brin b_2 si l'ancien brin $\alpha_0(b)$ (c'est-à-dire b') est maintenant relié par α_0 à un nouveau brin. Dans ce cas, il sera relié au brin $\alpha_2 \circ \alpha_1 \circ \alpha_0(b')$.*

2.5 Algorithmes locaux

Pour la plupart des modèles classiques, l'écriture ou la modification d'algorithmes se rapportant à un type de cellule est rendue difficile car elle nécessite la connaissance des autres types de cellules. Ainsi, si l'on veut supprimer une cellule de type *face* d'un maillage 2D, il va falloir toucher aux cellules de type *arêtes* et *sommets*. Comme le stipule [**?**], il devient alors impossible de traiter une entité topologique (sommet, arête, face) localement, sans modifier les caractéristiques des entités incidentes. Cependant, grâce aux G-Cartes, il est possible d'effectuer des traitements locaux. En effet, les brins d'une G-Carte peuvent être regroupés en sommets, arêtes ou faces.

[**?**] avait déjà mis en place des sortes d'algorithmes locaux sur ces opérations dites *lourdes*.

Le principe de l'algorithme local est le suivant. Au lieu de faire des algorithmes qui touchent un grand nombre de brins (tous les brins d'un triangle, d'une arête, ...), un même mini-algorithme est appliqué pour chaque brin. Les avantages d'un tel processus sont les suivants :

- les algorithmes sont plus courts

```
Procédure SubdivisionCoins(g : 2-G-Carte)
  Pour chaque brin b de la G-Carte g Faire
    bb, b0, b1, b2 : Brin
    bb ← alpha(b,0)
    b0 ← CreerBrin(g)
    b1 ← CreerBrin(g)
    b2 ← CreerBrin(g)
    RelierBrins(g, b0, b, 0)
    RelierBrins(g, b0, b1, 1)
    RelierBrins(g, b1, b2, 2)
    Si( alpha(bb,0) ≠ b )
      RelierBrins( b2, alpha(alpha(alpha(bb,0),1),2), 5)
    FinSi
    Si( est_libre(alpha(alpha(alpha(bb,1),0),1),0) )
      b101, b01 : Brin
      b101 ← alpha(alpha(alpha(b,1),0),1)
      b01 ← alpha(alpha(b,0),1)
      RelierBrins(g, b101, b01,0)
      RelierBrins(g,alpha(b101,2),alpha(b01,2),0)
    FinSi
    Si( !est_libre(b,2) ET est_libre(alpha(alpha(b,2),0)) )
      RelierBrins(g,alpha(alpha(b,2),0),alpha(b,0),2)
    FinSi
  FinPour
FinSubdivisionCoins
```

Figure 2.47 *Algorithme local de subdivision des coins*

- le code est plus simple à écrire et à maintenir.

Exemple

L'algorithme qui a été mis en place pour la subdivision des coins est un algorithme dit "local" (cf figure 2.46). Cela signifie que pour chaque brin de la surface d'origine, un léger traitement unique est exécuté.

Le traitement s'effectue le plus souvent en deux phases. La première phase consiste à créer de nouveaux brins (figure 2.46-B) tandis que la seconde phase va relier les nouveaux brins entre eux et avec les brins déjà existants (figure 2.46-C).

La subdivision des coins consiste à introduire une arête pour chaque noeud (ou coin) du polygone étudié comme le montre les figures 2.42-A. Les extrémités de cette arête correspondent aux milieux des deux arêtes qui constituent le coin.

Chaque nouvelle arête divise le polygone en deux séparant le coin du reste du polygone.

Donc, pour chaque coin (2 brins existants), on crée une arête (4 nouveaux brins) que l'on rattache aux deux arêtes du coin (via 2 nouveaux brins). Finalement, nous venons de prouver que pour chaque brin existant, 3 nouveaux brins sont nécessaires pour satisfaire notre algorithme.

L'implantation de l'algorithme est donné figure 2.47. Cet algorithme est à mettre en parallèle avec celui de la subdivisions des coins utilisant des cellules en page 92. La différence de code entre la version locale et cellulaire n'est pas flagrante en dimension 2. Par contre, en dimension 3, l'algorithme cellulaire requiert une fonction CreerPolyèdre qui n'est pas évidente à mettre en place tandis que l'algorithme local équivalent est proche de l'algorithme local surfacique 2.47. Les seuls changements à apporter sont la création d'un brin supplémentaire ainsi que la gestion des liens α_3 pour tous les brins créés. Il est évident qu'en accroissant encore la dimension des objets, l'algorithme local est beaucoup plus avantageux à écrire et à maintenir que son homologue cellulaire.

2.6 Interface bas et haut niveau

Les noyaux topologiques classiques sont plus intuitifs, ils utilisent les notions de cellules (noeud, arête, face, ...) en tant qu'élément de la structure de données. Par exemple, la structure de Weiler [**?**] possèdent l'objet *Noeud*, *Côté*, ... qui sont plus intuitifs. Cependant, tenir à jour un modèle cohérent est plus compliqué. En effet, une structure non mathématique qui utilise en même temps plusieurs objets de nature différentes (11 pour Weiler) est plus difficile à écrire et maintenir. Ces noyaux sont intuitifs mais peu robustes.

L'utilisation des G-Cartes pour un utilisateur non initié peut s'avérer assez fastidieuse. Il est nécessaire de comprendre son fonctionnement mathématique avant de pouvoir travailler avec. Par contre, sa nature mathématique va permettre à ce noyau d'être robuste. De même, tous les algorithmes agissant sur les G-Cartes peuvent être prouvés. Ce genre de noyau est robuste mais peu intuitif. Nous proposons de cacher l'interface de bas niveau des G-Cartes par une couche *cellulaire*. Cette couche de haut niveau intègrera les objets *Noeud*, *Arête* et *Face*. La figure 2.48 illustre cette interface. L'objet de base (cellule) connaît sa dimension ainsi qu'un brin qui appartient à la cellule choisie. Par exemple, pour un triangle, la dimension est 2 et le brin est l'un des 6 brins qui constituent ce triangle.

Une batterie de fonctions standard sera donnée à chacune de ces cellules.

Une k-cellule peut accéder :

- à ses k-cellules voisines. Pour un noeud d'une surface ou d'une arête, les 0-cellules voisines correspondent à l'ensemble des noeuds que l'on peut atteindre en "suivant" les arêtes qui "partent" de ce noeud (voir figure 2.49-

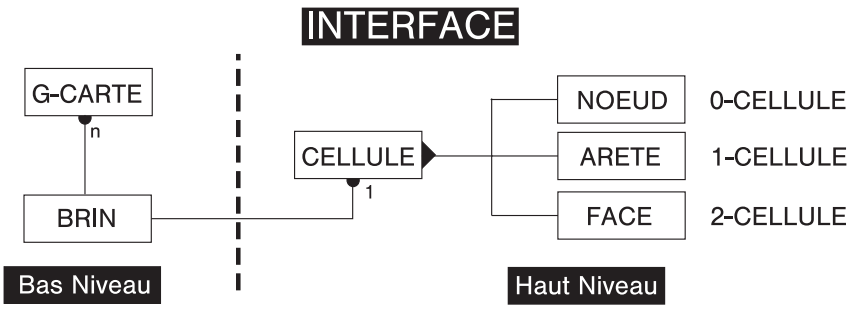


Figure 2.48 *Interface double : bas et haut niveau. A gauche, l'interface de bas niveau est constituée des deux objets que nous avons défini au début de ce chapitre : la* **G-Carte** *qui contient un ensemble de* **Brin**. *A droite, l'interface de haut niveau se base sur la notion de* **Cellule**. *Chaque cellule connaît sa dimension et un* **Brin** *qui la constitue. Pour l'utilisateur, le brin est* caché, *il ne manipule que des objets* **Noeud**, **Arête**, **Face**, ...

A). Pour une arête, il y a deux ensembles de 1-cellules voisines. Celles qui se trouvent directement *à gauche* et *à droite* de chaque extrémité de l'arête considérée et celles qui possèdent l'une des deux extrémités de l'arête considérée (voir figure 2.49-B). Pour les 2-cellules, on distingue à nouveau deux ensembles. Celui qui regroupe les polygones qui ont une arête en commun avec le polygone de départ et celui qui ont un sommet en commun (voir figure 2.49-C).

- à ses sous-cellules. Ainsi, un cellule peut accéder à tous ses sommets, ses arêtes, ... comme le suggère la figure 2.51.

- à son k-étoile d'incidence, c'est-à-dire l'ensemble des cellules de dimension k à la cellule considerée c qui l'entourent avec k supérieure à la dimension de c. Par exemple, la 2-étoile d'incidence d'un noeud d'une surface est l'ensemble des polygones qui ont en commun ce noeud (figure 2.49-D) tandis que la 1-étoile d'incidence du même noeud est l'ensemble des arêtes qui ont en commun ce noeud (figure 2.49-E).

Les possibilités d'interaction entre les cellules augmentent au fur et à mesure que la dimension des cellules grandit. Ainsi, pour une surface il y a trois relations de voisinages (une pour chaque cellule : noeud, arête, face), trois relations de sous-cellules (un sommet et une arête sont des sous-cellules d'une face et un sommet est une sous-cellule d'une arête) et trois relations d'étoile d'incidence (une arête possède une 2-étoile d'incidence et un sommet possède une 2-étoile et une 1-étoile d'incidence.). Ce qui signifie qu'il faut définir 9 fonctions en tout. Ce nombre atteint 16 pour un volume $(4 + (3 + 2 + 1) + (1 + 2 + 3))$ et bien plus pour un objet de dimension n : $n + 1 + (2.\sum_{i=1}^{n} i) = n.(n + 2) + 1$.

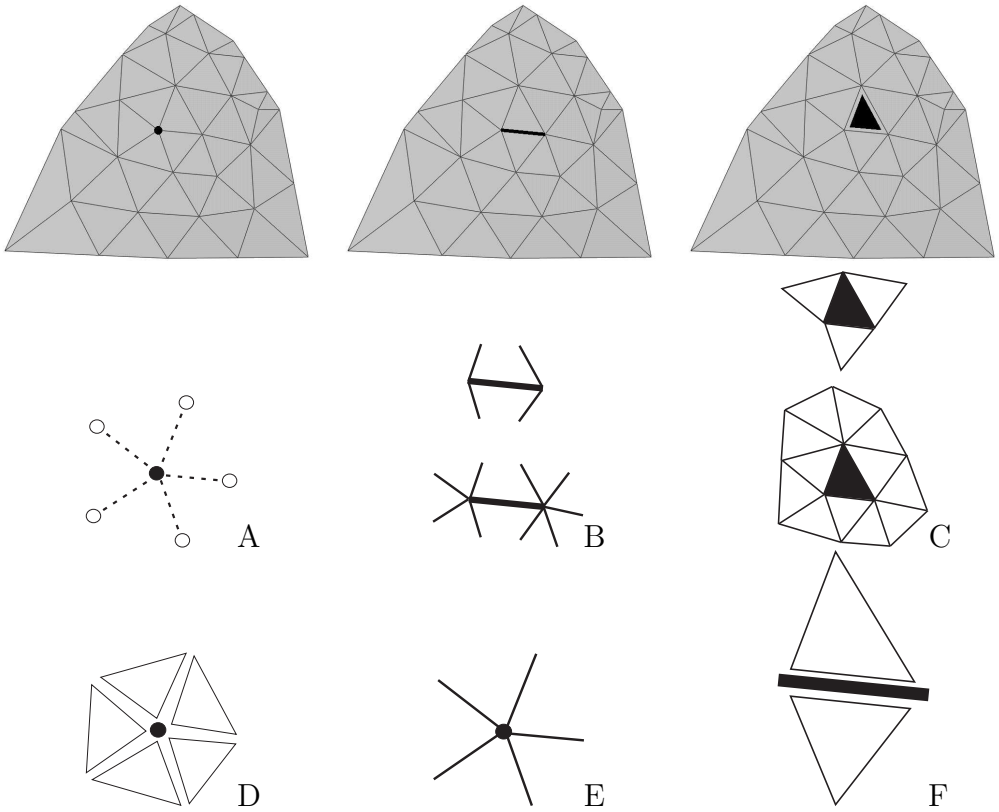


Figure 2.49 *Voisinage et étoile d'incidence d'une cellule. (A) voisinage à un noeud d'une surface. (B) les deux ensembles de voisinage possible pour une arête de surface. (C) les deux ensembles de voisinage d'un triangle. (D) 2-étoile d'incidence d'un noeud. (E) 1-étoile d'incidence d'un noeud. (F) 2-étoile d'incidence d'une arête.*

Pour la programmation de bas niveau, les fonctions sont écrites une fois pour la dimension n. La fonction d'étoile d'adjacence a pour prototype :

EtoileAdjacenceCellule <dimadjacence : Entier, dimcellule : Entier, dimgcarte : Entier > (cellule : Brin).

Pour un noyau classique utilisant des cellules, il faudrait définir autant de fonctions nécessaires que nommées. Pour notre interface de haut niveau, il faut rediriger chaque fonction sur la fonction générale de bas niveau. Par exemple, la fonction qui retourne l'ensemble des cellules correspondant à la 2-étoile adjacence d'une arête d'une surface se nomme 2EtoileAreteSurface(arete : Arete) en haut niveau et fait appel à EtoileAdjacenceCellule<2,1,2>(brin(Arete)). La série de fonctions de haut niveau qu'il faut écrire pour faire appel à celles de bas niveau peut même être générée par un programme.

Pour des objets simples, le nombre de voisins est connu. Le triangle possède trois arêtes et trois sommets. Pour ce cas, il est possible d'accéder à ces sous-

```
Liste<Brin> Voisinage(B : Brin, m : Entier, i : Entier)
   liste : Liste<Brin>
   Pour chaque brin b de la m-cellule de B Faire
      Si b n'a pas déjà été marqué Faire
         Marquer tous les brins de la i-cellule contenant b
         liste ← liste ∪ b
      FinSi
   FinPour
   Retourner liste
FinVoisinage
```

Figure 2.50 *Voisinage : renvoie un brin par i-cellule de la m-cellule donnée*

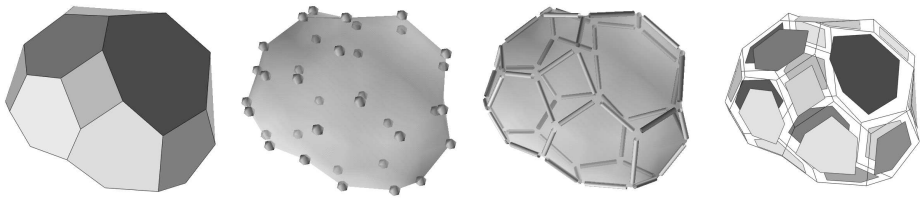


Figure 2.51 *De gauche à droite. Cellule d'origine. Ses noeuds. Ses arêtes. Ses faces.*

cellules en passant par un index. Pour obtenir les sommets d'un triangle, nous utiliserons les index allant de 0 à 2. Cependant, pour certains objets, nous ne pouvons connaître à l'avance le nombre de cellules qui le constitue ou qui l'entoure. Dans ce cas, il existe des itérateurs qui ont été définis et qui permettent d'accéder à ces cellules (de façon analogue à la STL C++ [**?**]).

En bas niveau, pour connaître toutes les i-cellules d'une m-cellule d'un n-maillage (figure 2.51), il faut parcourir tous les brins de la m-cellule en parcourant les involutions $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}, \alpha_{m+1}, \ldots, \alpha_{n-1}, \alpha_n$ et prendre un brin pour chaque nouvelle i-cellule trouvée en parcourant les involutions $\alpha_0, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_{n-1}, \alpha_n$ (cf program 2.50).

Les opérations légères sur les G-Cartes section 2.3 page 66 sont traduites trivialement dans cette nouvelle interface. Chaque fonction est dupliquée pour chaque cellule (noeud, arête, face, ...) et fait appel en interne à la fonction de bas niveau lui correspondant. Ainsi, pour détruire une arête A, la fonction $DetruireArete(A)$ fait appel à $DetruireCellule(G, b, dim)$ avec G la G-Carte, b le brin de l'arête A et dim la dimension de cette cellule, ici 1.

Cette interface a le mérite d'être intuitive car elle utilise des objets de base (noeud, arête, face, ...) qui permettent à l'utilisateur d'apprendre rapidement à

```
V : Volume
P : Polyedre
S : Sommet
Pour IterateurPolyedreVolume poly(V) Faire
   P ← CelluleCourante(poly)
   polyvaleur : Float
   polyvaleur ← 0
   Pour IterateurSommetsPolyedre somm(P) Faire
      S ← CelluleCourante(somm)
      polyvaleur ← polyvaleur + propriete(S)
   FinPour
   propriete(P) ← polyvaleur
FinPour
```

Figure 2.52 *Algorithme cellulaire*

se servir de tels outils. Le noyau topologique des G-Cartes qui se trouve caché derrière ces objets permet de nous assurer de la robustesse de ces fonctions.

Il faut aussi ajouter d'un point de vue programmation, que les objets de haut niveau ne sont pas stockés en permanence. Ils sont construits à la demande et détruits dès qu'ils ne sont plus utilisés. Les fonctions des objets sont *inline* pour la plupart, ce qui nous garantit de ne pas perdre beaucoup de temps en programmant directement avec ces objets. Cependant, il est à signaler que si le developpeur veut avoir un algorithme optimal, il est conseillé de connaître la théorie des G-Cartes et de travailler avec les fonctionnalités de bas niveau directement.

Nous donnons maintenant un exemple d'algorithme intuitif utilisant des cellules ainsi que son homologue de bas niveau pour pouvoir les comparer. Supposons que l'on veuille rattacher des informations à tous les polyèdres d'un volume en fonction d'une propriété stockés dans ses sommets. Cette opération peut être implantée en utilisant la notion de cellules comme le montre l'algorithme de la figure 2.52 ou en version bas niveau à base de G-Carte comme le montre la figure 2.53.

La structure générale du programme ne change pas beaucoup entre les deux interfaces. Cependant, le terme IterateurPolyedreVolume est plus parlant que Iterateur$< 3, 3 >$ pour un utilisateur qui n'est pas habitué à la programmation générique des G-Cartes. De plus, les brins b_1 et b_2 ne montrent pas quelles cellules ils symbolisent tandis que les variable P pour Polyedre et S pour Sommet permettent de mieux guider l'utilisateur pour comprendre l'algorithme.

Utiliser ce niveau d'interface ne sera pas beaucoup plus lent que d'utiliser

```
G : 3-G-Carte
b1 : Brin
b2 : Brin
Pour Iterateur< 3,3 > poly(G) Faire
   b1 ← BrinCourant(poly)
   polyvaleur : Float
   polyvaleur ← 0
   Pour Iterateur< 0,3,3 > somm(G,b1) Faire
      b2 ← BrinCourant(somm)
      polyvaleur ← polyvaleur + propriete(G,0,b2)
   FinPour
   propriete(G,3,b1) ← polyvaleur
FinPour
```

Figure 2.53 *Algorithme à base de G-Cartes*

directement les fonctionnalités de bas niveau grâce à la programmation générique et aux fonctions *inline*[2]. De plus, l'avantage de la généricité est que le temps de travail nécéssaire à cette réalisation se fait lors de la compilation et non pas lors de l'exécution contrairement à l'héritage par exemple.

[2]en C++ [**?**], une fonction inline est une fonction qui sera directement réécrite avant la compilation lors de ses appels par le préprocesseur. Il n'y aura pas d'empilement de fonctions : l'exécution sera plus rapide.

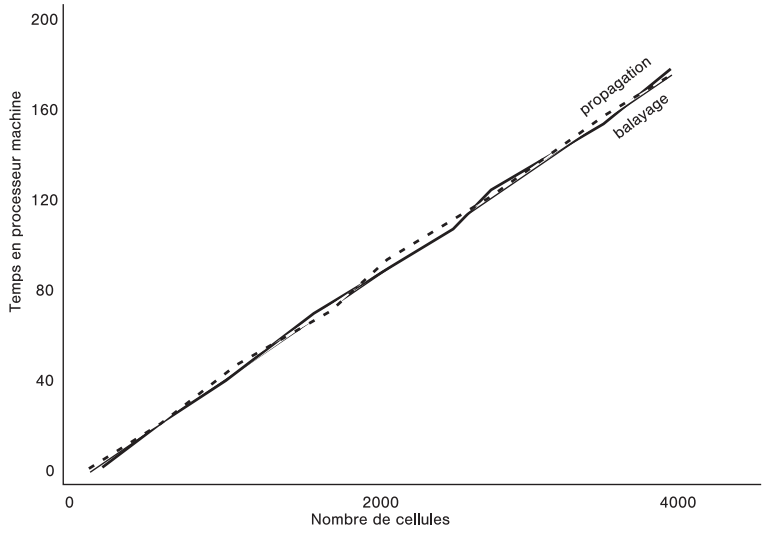


Figure 2.54 *Comparatif entre l'algorithme de subdivision des coins avec une méthode cellulaire (en pointillé) et une méthode locale (trait épais) sur une 2-G-Carte : l'efficacité en temps processeur est équivalente.*

2.7 Conclusion

Dans ce chapitre, nous avons présenté la manière dont nous avons implanté les G-Cartes. Grâce à ce noyau, nous avons la possiblité de construire des objets de dimension 0 à 3 (voir figure 2.55). Ces différents objets sont couramment utilisés en géologie lors de la construction de modèle volumique par exemple. Les éditions de maillage permettent de manipuler les objets. Par exemple, il est possible de supprimer à la main un noeud, une arête ou une face d'une surface suivant le bon vouloir de l'utilisateur. Des opérations de plus haut niveau permettent de supprimer tout un ensemble de cellule, par exemple toutes les cellules qui ont une sous-cellule dégénérées.

D'après [**?**], l'expérience montre que les algorithmes locaux sont au moins aussi efficaces (souvent plus) que les algorithmes classiques. Le tableau comparatif de la figure 2.54 prouve que pour la subdivision des coins, les algorithmes se valent en temps processeur. Par contre, d'un point de vue algorithmique, l'écriture et la maintenance du code "local" est plus simple que le code "cellulaire".

La représentation générique et efficace des objets de dimension n est basée sur la topologie des G-Cartes. Comme cette topologie est peu intuitive, nous avons émulé un noyau traditionnel qui se base sur la notion de cellules. Ce qui permet d'écrire du code intuitif pour des algorithmes simples. Par exemple, au lieu de manipuler des ensembles de brins qui représentent une cellule, on peut directement manipuler cette cellule de façon virtuelle. Il est alors possible de connaître toutes les sous-cellules qui la constituent ou toutes les cellules qui

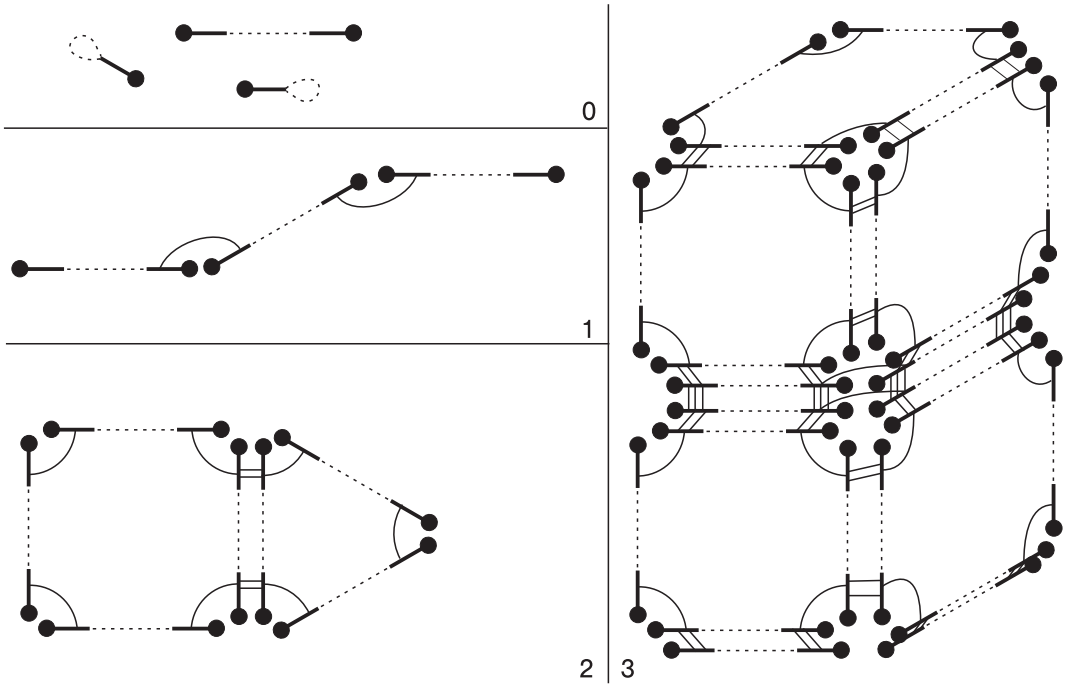


Figure 2.55 *Différentes n-G-Cartes avec n allant de* 0 à 3.

l'entourent.

On peut se demander quel est l'utilité d'une définition générique de bas niveau alors qu'une interface de haut niveau permet d'utiliser les notions intuitives de cellules. D'une part, la programmation générique permet d'avoir un code plus réduit. En effet, au lieu de définir une fonction pour chaque type de cellules avec le risque de se tromper en recopiant ou modifiant le code, ce dernier n'est écrit qu'une seule fois. D'autre part, ceci implique également que la maintenance d'un tel code est allégée du fait de sa petite taille.

Chapitre 3

Opération de coraffinement

Dans ce chapitre, nous étudions l'opération lourde de coraffinement implantée en deux et en trois dimensions pour des objets surfaciques et volumiques. Cette opération est un véritable *couteau suisse* qui s'applique à plusieurs domaines de la géologie.

Dans la section 3.1, nous rappelons les méthodes classiques de construction d'opérations booléennes en insistant plus particulièrement sur la méthode par raffinement. La section 3.2 décrit les différentes méthodes qui permettent d'effectuer un coraffinement 2D. La section **??** décrit l'extension de ces méthodes en 3D. Pour pallier aux problèmes numériques dû aux calculs d'intersection du coraffinement, la section **??** traite des prédicats exacts. Nous détaillons dans la section **??** comment construire les opérations booléennes à partir du coraffinement. Finalement, la dernière section **??** se rapportera aux applications géologiques des opérations de coraffinement.

3.1 Principes des opérations booléennes

Les opérations booléennes sont les opérations logiques connues dans le domaine des mathématiques. Soient deux ensembles E_1 et E_2, les opérations booléennes que l'on peut appliquer sur ces ensembles peuvent être l'union, l'intersection ou la différence. La figure 3.1-A illustre nos propos. Il est possible d'appliquer ces opérations sur des objets géométriques : c'est la géométrie des objets qui est utilisée pour obtenir l'objet final. La figure 3.1-B montre comment l'on peut obtenir un marteau à partir de trois primitives de base en appliquant deux opérations booléennes. On soustrait au parallélépipède le cylindre afin d'obtenir la *masse* du marteau. Ensuite, on rajoute son manche (tube).

Les opérations booléennes sont connues dans le domaine des mathématiques mais aussi de l'industrie. En effet, si elles sont utilisées dans le premier domaine sur des ensembles, elles le sont en industrie sur des objets simples (parallélépipèdes, cylindres, ...) pour fabriquer des pièces mécaniques.

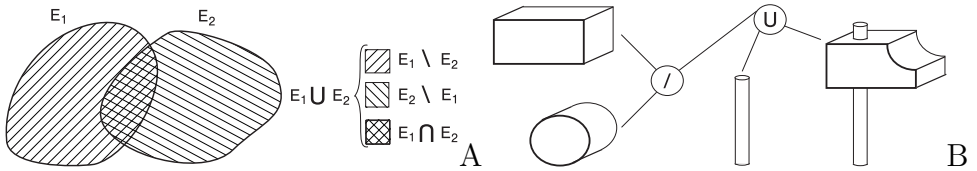


Figure 3.1 *A. Opérations booléennes avec des ensembles mathématiques. B. Opérations booléennes pour construire des objets industriels.*

Il existe deux méthodes de construction d'opérations booléennes [**?**, **?**, **?**, **?**]. La première découpe les objets initiaux en différents sous-objets. Ces sous-objets sont ensuite assemblés pour former le résultat. La deuxième ne construit qu'un objet qui contient les éléments nécessaires à la construction de l'opération booléenne : le *raffinement*.

3.1.1 Opération booléenne par découpe du bord

Historiquement, comme nous l'avons déjà vu dans la section 1.3.2 page 32, certains modèles représentent leurs objets par leurs bords.

La construction d'un objet C correspondant à une opération booléenne par découpe du bord entre deux objets initiaux A et B se divise en trois étapes (cf figure 3.2-A) :

- calcul d'intersection entre les bords de A et B,
- découpage des objets en sous-objets le long des intersections (voir figure 3.2-B),
- assemblage des sous-objets appartenant à l'opération booléenne recherchée (figures 3.2-D-E-F représentant respectivement l'union, l'intersection et les différences des deux triangles superposés).

Pendant la phase de découpage, nous obtenons des objets qui n'ont pas forcément de bord ce qui implique que le modèle utilisé doit être plus large que le modèle initial. D'autre part, Hoffmann *et al.* ont montré [**?**] que les approximations numériques lors de la phase de subdivision peuvent conduire à des bords incompatibles entre les parties de A et B. La construction n'est pas correcte car des bords ne peuvent plus être collés ensemble.

3.1.2 Opération booléenne par raffinement

Cette deuxième méthode se divise en deux étapes :

- coraffinement par découpage des cellules des deux objets et reconstruction des relations d'adjacence (figure 3.2-C),

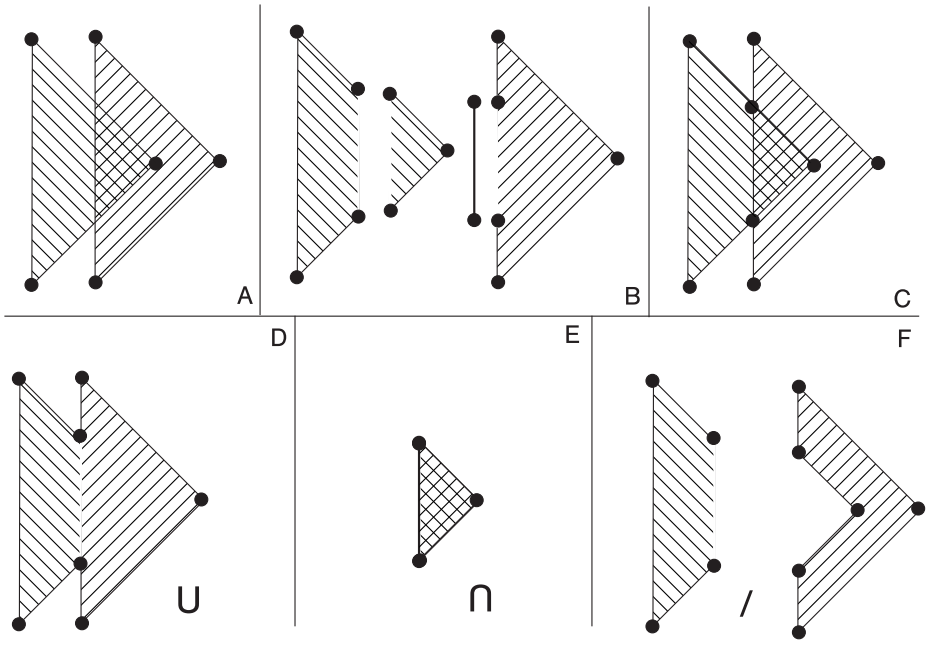


Figure 3.2 *A. Deux triangles superposés. B. Opération booléenne par découpe de bord. C. Opération booléenne par raffinement. D. Union. E. Intersection. F. Différences.*

- sélection des cellules du coraffinement appartenant à l'opération booléenne recherchée (figures 3.2-D-E-F représentant respectivement l'union, l'intersection et les différences de deux triangles superposés),

Soit O_1 un objet de dimension n ayant un ensemble N_1 de noeuds. Soit O_2 un objet de dimension n ayant un ensemble N_2 de noeuds. Le raffinement de ces objets est l'objet O de même dimension ayant un ensemble N de noeuds tel que $N = N_1 \cup N_2 \cup$ cellules provenant des intersections entre les n-cellules de O_1 et O_2.

Le coraffinement d'objets produit un nouvel objet dont les cellules sont les cellules des objets de départ ainsi que les intersections existantes entre ces cellules. Pour obtenir le résultat d'une opération booléenne, il faut sélectionner les parties utiles.

Si certains objets se trouvent entièrement englobés par d'autres sans intersections, diverses techniques permettent de construire et gérer ces relations d'inclusions :

- arbres d'inclusions [**?**, **?**], cette technique nécessite une structure supplémentaire qui est extérieure au modèle,

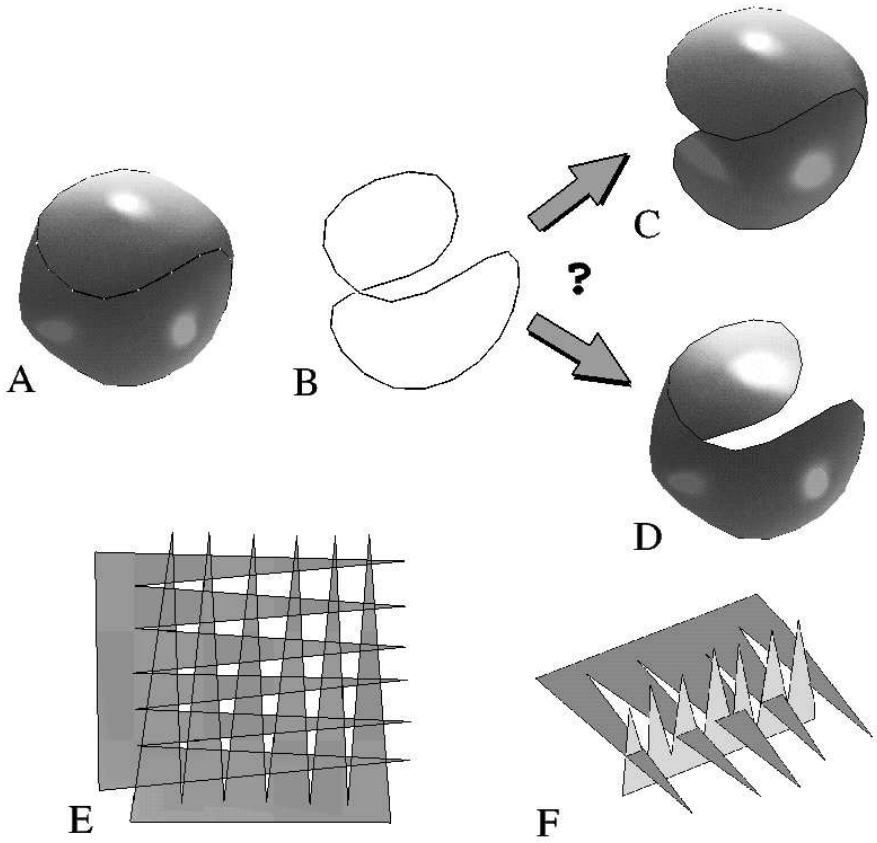


Figure 3.3 *A. Balle de tennis. B. Couture de la balle de tennis. C. Moitié de balle. D. Autre moitié de balle. E. Polygones concaves : grand nombre d'intersections. F. Polygones concaves dans l'espace 3D : intersection relativement compliquée.*

- rajout de cellules virtuelles [**?**, **?**], comme nous en avons déjà parlé dans la section 1.4.5 page 46, les cadres des objets peuvent être reliés entre eux par des brins virtuels simulant une relation d'inclusion, nous réutiliserons cette méthode pour le raffinement.

Nous préférons utiliser cette technique de raffinement plutôt que la précédente car elle permet de mieux contrôler la topologie des objets.

3.2 Coraffinement 2D

Nous proposons de coraffiner des surfaces planes composées de triangles. L'avantage des triangles est leur planarité et le fait que ce soient des polygones convexes. L'utilisation de polygones non plats entraine certaines configurations qui peuvent s'avérer ambigües. Ainsi, par exemple, la figure 3.3-A nous montre une balle de

tennis : si l'on considère la couture de cette balle comme on peut le voir figure 3.3-B en tant que bord d'un polygone fermé, deux polygones différents peuvent être envisagés ; les deux moitiés de la balle de tennis (figures 3.3-C.D). Par ailleurs, utiliser des polygones concaves peuvent générer un grand nombre d'intersections : la figure 3.3-E nous montre un exemple où le nombre d'intersections atteint l'ordre du carré du nombre de côtés des polygones. Plongées dans $\mathbb{R}^3$, les relations topologiques résultant de l'intersection de tels polygones peuvent devenir très compliquées. De plus, la plupart des algorithmes géométriques utilisent des prédicats exacts dans le plan pour réduire les problèmes de précisions numériques. Nous détaillons ces prédicats dans la section **??**.

Nous choisissons donc de coraffiner des surfaces triangulées d'une part pour les raisons que nous venons de citer et d'autres part car c'est cette représentation de surfaces dicrètes qui est la plus utilisée dans la majorité des produits de modélisation. Cependant, l'extension à des surfaces constituées de polygones convexes ne demande pas beaucoup de changement comme nous le verrons dans les sections **??** et **??**.

Soient S_1 et S_2 deux surfaces triangulées et plates ; le coraffinement de ces deux surfaces revient à *combiner* leurs maillages. Le résultat de cette combinaison peut être retriangulé de façon à n'obtenir que des triangles comme le montre la figure **??**-D.

Trois principales méthodes de coraffinement sont à distinguer :

- arête par arête : (cf figure **??**-A) cette méthode consiste à itérer sur chaque arête de la première surface. Pour chacune de ces arêtes, l'algorithme calcule l'intersection avec les arêtes de la seconde surface. Cette méthode est en $\mathcal{O}(m.n)$ avec m, le nombre d'arêtes de la première surface et n celui de la seconde : pour chaque arête de la seconde surface, il est nécessaire de déterminer s'il y a une intersection avec l'arête courante de la première surface. Pour optimiser cette méthode, un *quadtree* [**?**, **?**] peut être construit sur la deuxième surface. Le quadtree est un arbre de recherche dont chaque sommet porte quatre fils et qui va permettre de donner tous les triangles qui se trouvent dans une *boîte* de petite taille. De cette manière, l'arête courante de la première surface n'est comparée qu'avec un petit nombre d'arêtes de la seconde.

- par balayage : (cf figure **??**-B) cette méthode consiste à simuler le balayage des deux surfaces par une droite. Une direction est fixée, par exemple la direction de l'axe des ordonnées et une droite parallèle à cette direction balaye les surfaces de gauche à droite. Bentley et Ottmann [**?**] ont été les premiers à mettre en place un tel algorithme. Un algorithme en $\mathcal{O}(nlogn + a)$, avec a le nombre de paires de segments s'intersectant et n le nombre total de segments, a été donné par Chazelle et Edelsbrunner [**?**]. Nous expliquons plus en détail l'algorithme que nous utilisons dans la section **??**.

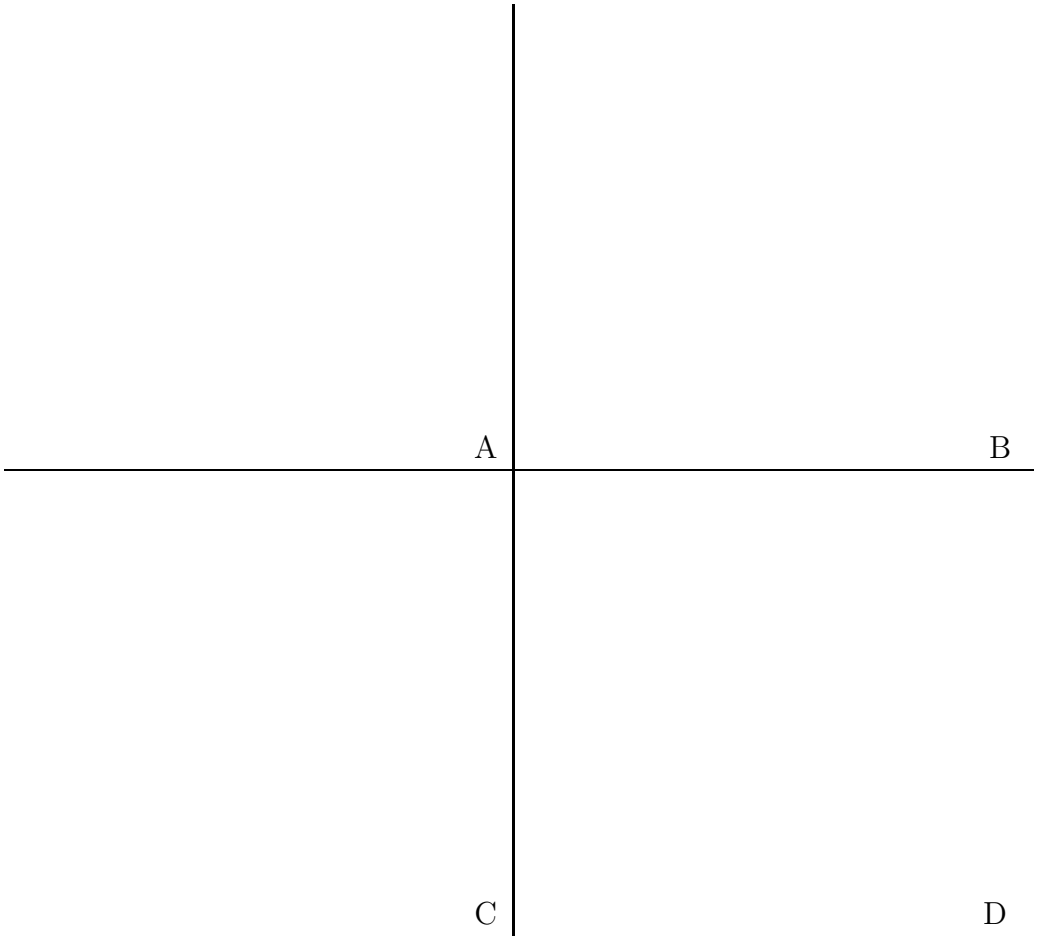


Figure 3.4 *Principales méthodes de coraffinement. Le calcul des nouveaux points est représenté par des cercles blancs. Les arêtes déjà utilisées sont grises tandis que celles qui n'ont pas encore subi de traitement sont blanches. A. Méthode arête par arête B. Méthode avec balayage C. Méthode par propagation D. Résultat du coraffinement : les deux surfaces sont combinées entre elles puis retriangulées.*

- par propagation : (cf figure **??**-C) à partir d'un état initial, cette méthode consiste à propager l'algorithme de triangle en triangle. Dans le cadre de la géoscience, si l'on considère que les surfaces que l'on veut coraffiner sont de tailles identiques et surtout ont un maillage homogène équivalent (c'est-à-dire des triangles de même tailles), cet algorithme est en $\mathcal{O}(n)$ avec n étant le nombre total de segments. Cependant, dans le pire des cas, cet algorithme est en $\mathcal{O}(n^2)$. Nous détaillons la méthode dans la section **??**.

Nous allons dans un premier temps détailler la méthode par propagation puis par balayage.

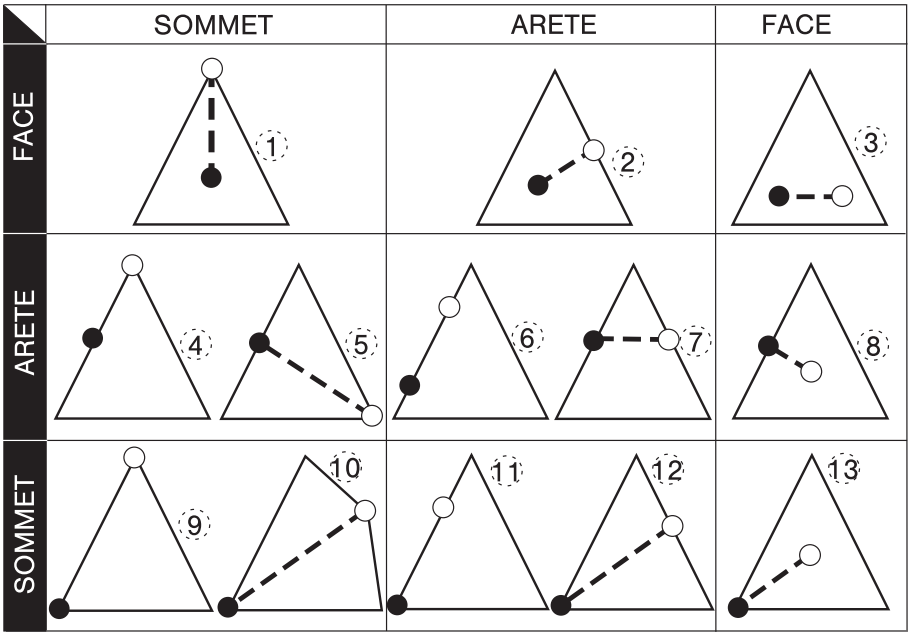


Figure 3.5 *Les 13 configurations de coraffinement par propagation en 2D. L'arête (ou partie d'arête) considérée part du point noir pour finir au point blanc.*

3.2.1 Coraffinement 2d par propagation

L'idée de base du coraffinement 2d par propagation est de se propager de triangle en triangle dans un maillage comme l'illustre la figure **??**-C. Soient S_1 et S_2 les deux surfaces à coraffiner. La première étape consiste à copier la surface S_2 dans le résultat nommé S_3. Ensuite, à partir d'une configuration de départ, par exemple une extrémité d'une arête de S_1 qui se trouve dans un triangle de S_3, l'algorithme insère les arêtes de S_1 dans S_3 en se propageant d'arêtes en arêtes. Nous étudions maintenant les cas d'intersection d'un triangle avec un segment.

Cas d'intersection d'un triangle avec un segment dans le plan

Hypothèse de départ : le segment a déjà touché une cellule du triangle et se prolonge dans ce dernier.

Nous subdivisons nos cas en trois grandes catégories qui sont :

- le segment débute dans le triangle,
- le segment débute sur une arête et
- le segment débute sur un sommet.

Pour chacune de ces trois catégories, différentes propositions peuvent être faites. Le numéro qui se trouve devant chaque proposition correspond aux numéros des cas d'intersection de la figure **??**.

Segment débutant dans un triangle

- Première proposition : (1) le segment ressort par l'un des trois sommets du triangle ; ces trois possibilités sont équivalentes.
- Deuxième proposition : (2) le segment ressort par l'une des trois arêtes du triangle ; ces trois possibilités sont équivalentes.
- Troisième proposition : (3) le segment reste dans le triangle.

Segment débutant sur une arête

- Première proposition : le segment ressort par l'un des trois sommets du triangle. Deux cas sont envisageables : (4) le sommet est celui qui se trouve *en face* de notre arête ou bien (5) le sommet est incident à l'arête.
- Deuxième proposition : le segment ressort par l'une des trois arêtes du triangle. Deux possibilités s'ouvrent alors à nous : (6) l'arête est la même que la première arête ou bien (7) l'arête est une autre arête.
- Troisième proposition : (8) le segment s'arrête dans le triangle.

Segment débutant sur un sommet

- Première proposition : (9) pour un triangle donné, le segment ressort par l'un des deux autres sommets (ou (10) par un autre sommet dans le cas plus général de surfaces polygonales).
- Deuxième proposition : le segment ressort par l'une des trois arêtes du triangle. Deux possibilités s'ouvrent alors : (11) l'arête est celle qui se trouve *en face* de notre sommet ou bien (12) l'arête est une arête incidente au sommet.
- Troisième proposition : (13) le segment s'arrête dans le triangle.

Automate d'états finis

Au vu de ces trois catégories et des différentes propositions qui les accompagnent, nous comptabilisons 13 cas différents (cf figure **??**).

Pour traiter ces cas, nous avons construit un automate d'états fini qui les englobe (figure **??**-A). Cet automate possède trois états qui se nomment SOMMET, ARETE et FACE. Ils correspondent à la cellule dans laquelle commence un

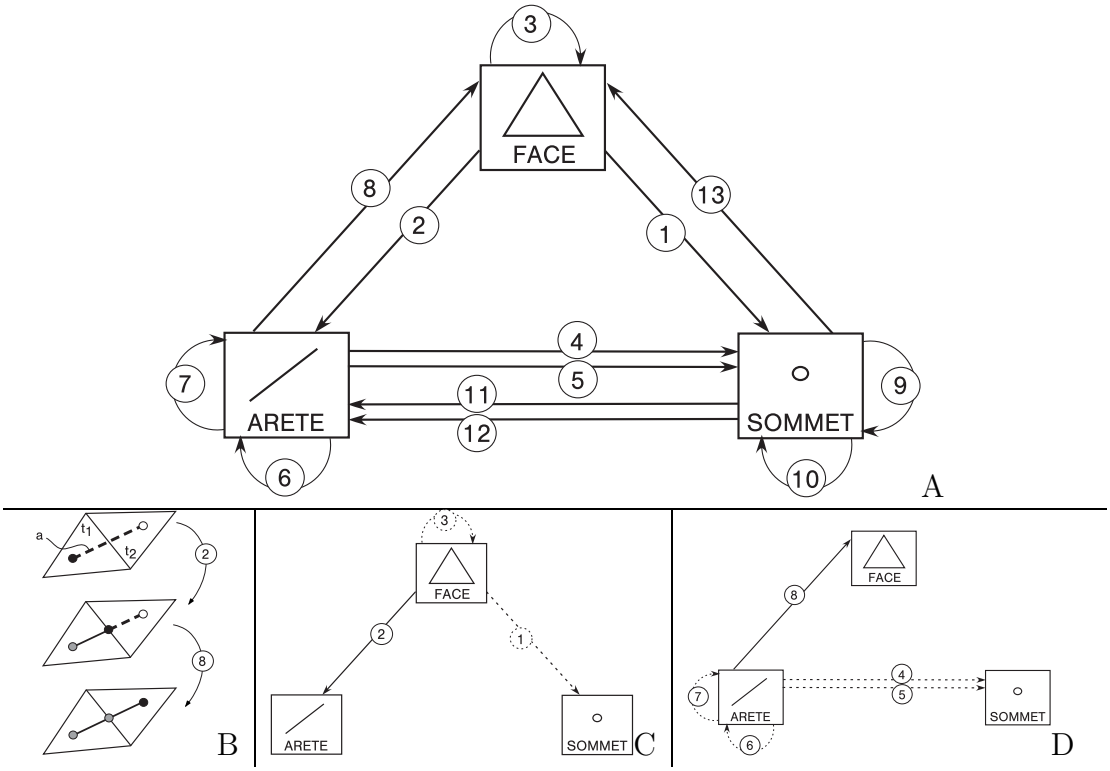


Figure 3.6 *A. Automate d'état fini : à partir d'un état* SOMET, ARETE *ou* FACE, *on cherche à trouver le prochain état en suivant les flèches de l'état dans l'ordre. Les numéros des flèches correspondent également aux numéros de la figure* **??**. *B. Exemple avec une arête et deux triangles. L'arête débute dans une* FACE *et traverse d'abord une arête (C) puis finit sa course dans un triangle (D). Les flèches en pointillées représentent les différentes possibilités autre que celle choisie (trait plein).*

segment. Une flèche reliant deux états représente un des 13 cas d'intersection que nous avons déterminés préalablement. Ces flèches nous emmènent d'une cellule à une autre. Elles sont à parcourir dans l'ordre pour effectuer les différents tests. L'exemple qui suit permet de comprendre comment est utilisé l'automate d'état fini lors de la construction du coraffinement.

Supposons que l'on veuille traiter le cas d'une arête a dont les extrémités se trouvent dans deux triangles t_1 et t_2 ayant une arête commune (voir figure **??**-B). D'après notre automate d'état fini, nous sommes dans l'état FACE étant donné que a *commence* dans t_1. A partir de cet état, il faut déterminer si a se prolonge vers un sommet (1), une arête (2) de t_1 ou reste dans le même triangle (3). Dans notre cas, a se prolonge sur l'arête commune à t_1 et t_2. Nous allons donc prendre la flèche numérotée (2) dans notre automate (voir figure **??**-C) pour atteindre l'état ARETE. Puis, nous testons les possibilités suivantes : est-ce que a se prolonge sur un sommet (4-5) ou une arête (6-7) de t_2 ? Dans notre cas,

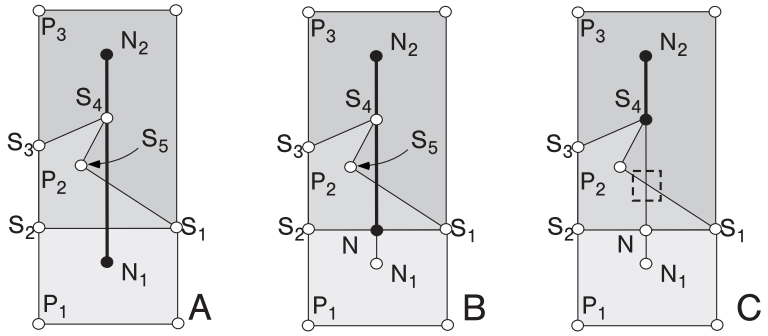


Figure 3.7 *Problème des polygones concaves. L'arête épaisse est l'árête ou bout d'arête qui reste à insérer. (A) L'arête N_1N_2 débute dans le polygone P_1. (B) D'après l'ordre des tests à effectuer selon l'automate d'état fini, la première intersection trouvée est l'arête S_1S_2. (C) La deuxième intersection trouvée est le sommet S_4 d'après l'automate d'état fini. L'intersection avec l'arête S_1S_5 est oubliée (zone en pointillé). Pour résoudre ce problème il aurait fallu tester toutes les intersections puis les trier selon leur ordre d'apparition sur l'arête N_1N_2.*

aucune de ces cellules n'est intersectée. Cela signifie que a termine sa course dans t_2. Nous allons donc prendre la flèche numérotée (8) (voir figure **??**-D) pour nous retrouver dans l'état FACE. Il reste ensuite à traiter l'arête suivante rattachée à a.

Implantation

Le coraffinement se fait entièrement dans la surface finale. Il n'est pas nécessaire de construire une structure annexe pour stocker l'état courant du coraffinement. Cela signifie qu'au cours du traitement, les triangles de la surface S_3 sont modifiés et deviennent des polygones. En fait, l'hypothèse restrictive de départ stipulant que les surfaces devaient être triangulées s'élargit aux polygones concaves car on ne traite pas l'intersection d'un triangle mais celle d'arêtes.

Traiter des polygones concaves aurait pour effets d'augmenter les erreurs liées à la précision numérique. Etudions l'exemple de la figure **??**-A où une arête N_1N_2 intersecte une surface constituée de polygones pouvant être concaves. L'arête débute dans le polygone P_1. En utilisant l'automate d'état fini de la figure **??**, l'algorithme teste si l'arête passe par un sommet de P_1. Comme ce n'est pas le cas, l'étape suivante consiste à tester si l'arête coupe une arête de P_1. Nous constatons que l'arête S_1S_2 commune à P_1 et P_2 répond à ce test (voir figure **??**-B). L'algorithme exécute le traitement de la flèche numéro (2) et l'état courant devient l'état ARETE. Maintenant, l'algorithme teste si l'arête passe par un noeud de P_2. On constate que le sommet S_4 est le candidat de ce test (voir figure **??**-C). Mais, si l'on traite ce cas, on omet le fait que l'arête passait d'abord par l'arête S_1S_5. Ceci est dû au fait que le polygone P_2 est concave. Pour résoudre ce

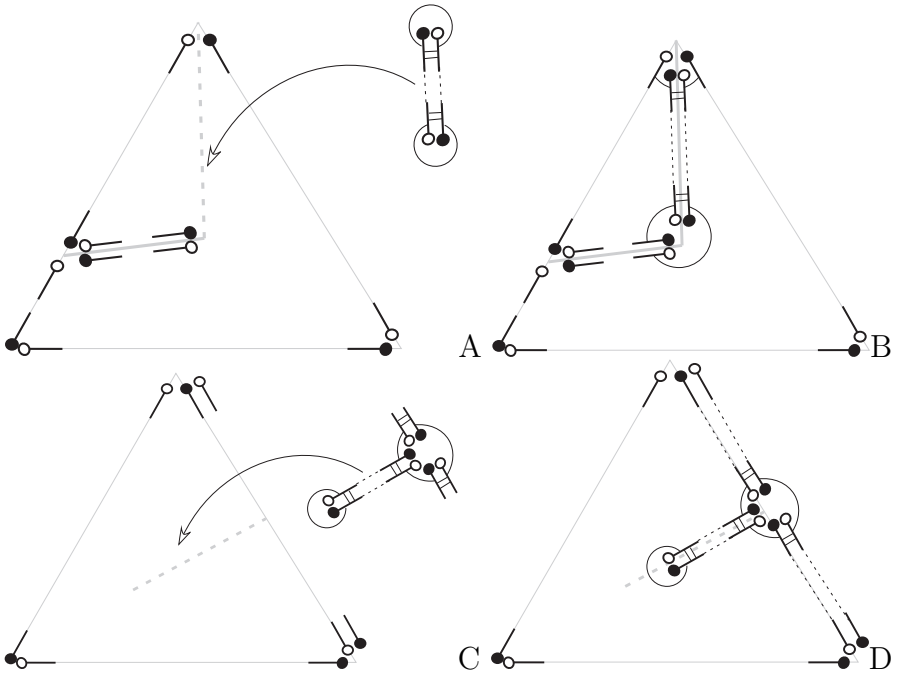


Figure 3.8 *Etude de deux des 13 cas d'intersections. A. Cas 1 : une arête partant d'une face et allant vers un sommet est insérée. Il est à noter qu'il faut faire une distinction entre les sommets qui correspondent à ceux de la surface support de ceux de la surface à insérer. B. En insérant l'arête, on colle les extrémités aux brins existants. C. Cas 2 : une arête partant d'une face et allant vers une arête est insérée. D. Un nouveau sommet coupe l'arête d'arrivée en deux.*

problème, il aurait fallu traiter tous les cas d'intersection possible de l'arête avec le polygone puis de prendre l'intersection la plus proche : ici, l'algorithme aurait d'abord traité l'intersection arête-arête puis arête-sommet. Cependant, le test qui consiste à trier des points dans l'espace est un test géométrique susceptible d'avoir des erreurs à cause de la précision numérique des machines. Par ailleurs, l'utilisation de polygones concaves ralentit l'opération de coraffinement. Afin de se préserver de ces erreurs, nous préferons restreindre les polygones à des polygones convexes.

Seules les opérations topologiques sont modifiées dans l'objet final. En ce qui concerne l'information géométrique, nous gardons une information permanente sur les sommets initiaux. De cette manière, si l'on coupe une même arête plusieurs fois de suite, les erreurs numériques causées par l'arrivée de nouveaux sommets ne se propagent pas (erreurs en cascade).

L'implantation de l'algorithme de coraffinement 2d par propagation consiste à écrire les 13 cas que nous avons observé. Pour expliquer l'utilisation des G-Cartes, nous allons développer quelques cas.

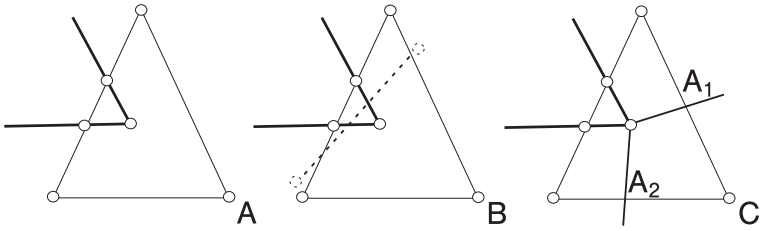


Figure 3.9 *(A) En cours de coraffinement, des polygones deviennent concaves. Dans cet exemple, le triangle initial (trait fin) a été intersecté par deux segments (trait épais). Il en résulte un polygone concave au niveau des segments à trait épais dans le triangle. (B) Insérer l'arête en pointillé demande de traiter les intersections dans l'ordre. Cependant, cette arête ne peut pas exister. Elle ne peut pas appartenir à la surface à trait épais car cela signifierait que cette surface de départ s'auto-intersectaient. (C) Les seules arêtes qui peuvent exister sont des arêtes qui vont joindre un sommet intial de la surface à trait épais comme par exemple les arêtes A_1 et A_2. En aucun cas, ces arêtes ne peuvent couper les arêtes à trait épais.*

Le cas numéro 1 est une arête qui part d'une face et qui passe par un de ses sommets. Il faut alors insérer une arête dans une face. Cette opération nécessite 4 brins correspondant à une arête double (figure **??**-A). L'arête est fermée à ses extrémités (les brins sont cousus entre eux par α_1) ce qui permet d'avoir une G-Carte consistente tout au long de l'opération. De plus, cette fermeture permet d'utiliser l'opération de collage étudiée dans la section 1.4.4 page 43 correspondant au collage de cadre de dimension 2. De cette manière, il ne reste plus qu'à coller les deux extrémités de l'arête aux brins déjà existants (figure **??**-B).

Le cas numéro 2 est une arête qui part d'une face et qui passe par une de ses arêtes. Cette opération nécessite 4 brins correspondant à une arête double ainsi que de 4 autres brins pour créer le point d'intersection sur l'arête traversée (figure **??**-C). En insérant le nouveau point, on colle les 4 brins aux brins existants par α_0.

Il est à signaler qu'au cours du coraffinement, certains polygones deviennent concaves ce qui contredit notre hypothèse de départ rappelée au début de cette section. Cependant, ces polygones concaves ne sont pas gênants. En effet, si l'on considère la figure **??**-A où le triangle à trait fin de la surface de support a été raffiné par deux arêtes à trait épais de la surface à insérer, on constate que l'on obtient un polyèdre concave limité par les six sommets blancs. Dans la figure **??**-B, si l'on introduit l'arête dessinée en pointillé, il faudrait en toute rigueur calculer tous les points d'intersection de cette arête avec les arêtes déjà existantes pour savoir dans quel ordre il faudra insérer les nouveaux points d'intersection. Cependant, cette arête ne peut pas être une arête de la surface à trait épais car cela signifierait que cette surface s'auto-intersecterait. En revanche, les seules arêtes qui peuvent exister sont par exemple les arêtes données dans la figure **??**-C. Ces arêtes ne coupent pas les arêtes à trait épais qui forment la concavité du poly-

gone. C'est pourquoi nous pouvons affirmer que les polygones concaves obtenus lors du coraffinement n'entraînent pas une modification de l'algorithme que nous avons expliqué : il n'est pas nécessaire de calculer tous les points d'intersection d'une arête puis de trier ces points. L'opération de coraffinement s'effectue dans l'ordre donné par l'automate d'état fini.

Nous allons maintenant parler du coraffinement 2D par balayage qui est plus aisé à programmer car il comporte moins de cas à traiter.

3.2.2 Coraffinement 2d par balayage

Le coraffinement appliqué à un algorithme de balayage ([**?**] page 37) consiste à traiter les intersections de *gauche à droite*.

En général, deux structures de données sont utilisées pour cette méthode. L'une gère l'état de balayage ($\mathcal{B}$) tandis que l'autre contient la suite des évènements à traiter ($\mathcal{E}$).

La première structure insère ou retire les segments qui intersectent la droite de balayage. Lorsque cette droite rencontre l'extrémité gauche d'un segment, celui-ci est inséré dans la structure $\mathcal{B}$. Au contraire, si l'extrémité droite du segment est rencontré par la droite de balayage, alors ce segment est retiré de la structure. Au début de l'algorithme, cette structure est vide.

La structure des évènements $\mathcal{E}$ contient les extrémités des segments ainsi que les points d'intersection à déterminer. Au départ, elle est initialisée avec les extrémités des segments à intersecter, triées par ordres d'abscisses croissantes. Les points d'intersection seront insérés au fur et à mesure dans la structure de manière ordonnée. L'algorithme de balayage va consister à traiter dans l'ordre les évènements stockés dans $\mathcal{E}$:

- Si l'évènement est l'extrémité gauche d'un segment, celui-ci est inséré dans la structure $\mathcal{B}$. Si ce segment intersecte les segments déjà inséré dans cette structure, le point d'intersection est inséré dans la suite des évènements $\mathcal{E}$.
- Si l'évènement est l'extrémité droite d'un segment, celui-ci est retiré de $\mathcal{B}$.
- Si l'évènement est un point d'intersection, celui-ci est inséré et un traitement est appliqué aux segments qui ont engendré l'intersection.

Dans la suite, nous allons donner des noms aux deux surfaces à coraffiner. L'une se nomme *surface support* : c'est la surface qui va être intersectée par les arêtes de l'autre. L'autre se nomme *surface à insérer*.

Cas d'intersections

Contrairement au coraffinement par propagation, nous ne noterons que trois cas d'intersections possibles (la figure **??** illustre ces trois cas).

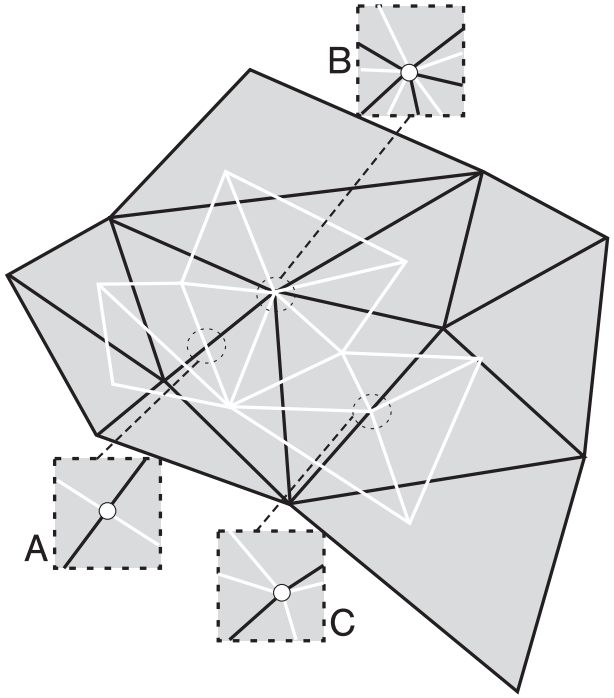


Figure 3.10 *Les 3 configurations du coraffinement par balayage. (A) Intersection arête-arête. (B) Intersection sommet-sommet. (C) Intersection sommet-arête.*

Intersection arête-arête

Le cas d'intersection le plus fréquent est le cas où deux arêtes s'intersectent en leur *milieu* (c'est-à-dire extrémités exclues) comme le montre la figure **??**-A. La résolution de cette configuration est très simple, chaque arête est subdivisée topologiquement en deux par l'insertion d'un nouveau point, les deux noeuds ainsi créés doivent être collés ensemble.

Dans la figure **??**-A, deux arêtes de surfaces symbolisées par des brins s'intersectent en leur milieu. La structure nécessaire pour rajouter un point entre deux segments se trouve sur la figure **??**-B. Cette structure correspond à 4 demi-arêtes qui partent d'un noeud commun. Les figures **??**-C et **??**-D sont les représentations des deux manières d'accrocher la structure aux deux arêtes ou en d'autres termes de relier ces deux arêtes. Pour obtenir une intersection correcte, il faut déterminer, suivant l'ordre trigonométrique, lequel de P_2 ou Q_2 se trouve après P_1.

Intersection sommet-sommet

Le cas d'intersection suivant est le cas où deux noeuds se superposent (voir la figure **??**-B). Cette configuration nécessite de trier de manière trigonométrique toutes les arêtes qui partent de ces deux noeuds. Ensuite, ces arêtes doivent être

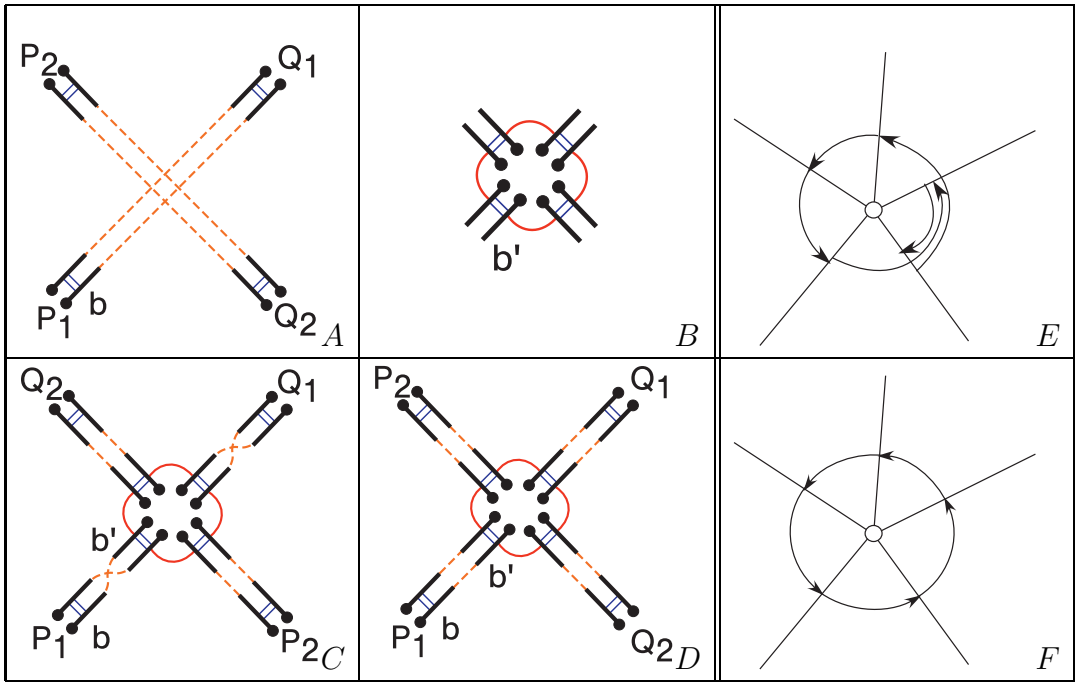


Figure 3.11 *Intersection arête-arête. A. Les deux arêtes s'intersectent. B. Structure nécessaire à l'insertion du nouveau point. C. Mauvaise insertion. D. Bonne insertion. E. Arêtes mal classées autour d'un noeud. F. Arêtes correctement classées.*

collées les unes aux autres dans l'ordre défini comme le montre la figure**??**-E-F.

Pour trier un ensemble d'arêtes donné ayant une extrémité commune, il suffit d'utiliser la fonction quicksort due à C.A.R. Hoare en 1960. Une implantation de cette fonction se trouve dans [**?**].

Intersection sommet-arête

Le dernier cas d'intersection est le cas où un noeud coupe une arête en deux comme dans l'exemple de la figure **??**-C. Pour résoudre ce cas, il suffit de subdiviser topologiquement l'arête coupée en deux puis de traiter cette configuration comme la configuration où deux noeuds se superposent (cf paragraphe ci-avant).

Implantation

L'implantation de l'algorithme de coraffinement par balayage se fait en cinq étapes. Les trois premières étapes consistent à calculer les points d'intersection entre sommet-sommet, sommet-arête et arête-arête. La quatrième étape trie les points d'intersection. Finalement la dernière étape insère les nouveaux noeuds dans l'ordre.

Pour le calcul des points d'intersection, nous faisons appel à des prédicats exacts qui permettent de s'affranchir des approximations faites par la machine.

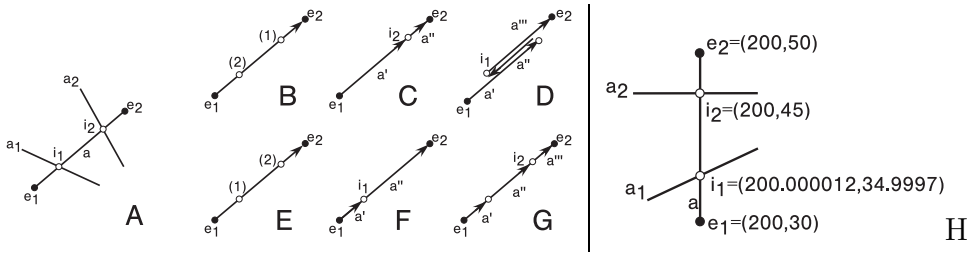


Figure 3.12 *(A) un segment orienté a est coupé par deux autres segments a_1 et a_2. (B-C-D) les noeuds ont éte insérés dans un ordre quelconque : il en résulte de* mauvais *segments. (E-F-G) les noeuds ont été insérés dans l'ordre d'apparition sur l'arête a : les segments sont* corrects. *H. Erreur numérique : le point d'intersection est une approximation. Les noeuds ne seront pas insérés dans l'ordre d'apparition sur l'arête a.*

Ces prédicats sont détaillés dans la section **??**.

Il est très important de trier les points d'intersection avant de les insérer afin d'obtenir un résultat correct. Supposons que nous ayons une fonction qui nous permette de savoir si deux segments s'intersectent et en quel point. Ces segments sont les arêtes de nos deux surfaces à coraffiner. Soit a une arête de la surface support et a_1 et a_2 deux arêtes de la surface à insérer qui intersectent a comme le montre l'exemple de la figure **??**-A. Nous allons couper l'arête a par les deux arêtes sans nous soucier de l'ordre. L'intersection entre a et a_2 coupe l'arête a en a' et a'' (figure **??**-C). En traitant l'intersection entre a et a_1, on réalise qu'en découpant a'' en deux la nouvelle arête a'' *retourne en arrière* (figure **??**-D). C'est pour cette raison que le coraffinement par balayage nécessite d'ordonner les points d'intersection de gauche à droite. Dans l'exemple de la figure **??**-E, on commence par couper a par a_1 (figure **??**-F) puis par a_2 (figure **??**-G).

Ordonner des points d'intersections qui se situent sur un même segment peut paraître trivial. Cependant, l'exemple de la figure **??**-H montre que le calcul du point d'intersection peut s'avérer incorrect. Cette erreur entraîne une erreur d'ordre. En effet, la fonction qui trie les points de gauche à droite retournera d'abord le point i_2 puis le point i_1 car l'abscisse de i_2 est plus petite que celle de i_1. Dans la section **??**, nous utilisons des prédicats exacts pour pallier ces problèmes numériques.

3.2.3 Gestion du bord

Avant de passer au coraffinement 3d, nous allons nous intéresser aux cas d'intersection à traiter sur le bord des surfaces. Les objets utilisés pour le coraffinement 2D sont généralement des objets ouverts. En effet, les géologues ne modélisent qu'une partie du sous-sol (de quelques centaines de mètres à quelques dizaines de kilomètres).

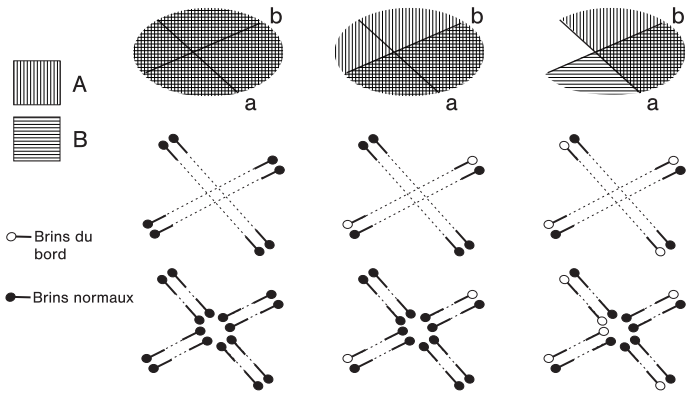


Figure 3.13 *Traitement du bord : (1) Les deux arêtes a et b se trouvent à l'intérieur des surfaces A et B, il n'y a pas de nouveaux brins du bord. (2) L'une des deux arêtes se trouve au bord, il n'y a pas de nouveaux brins du bord. (3) Les deux arêtes se trouvent au bord, les nouveaux brins qui se trouvent à l'extérieur sont marqués en tant que brins du bord.*

Dans les 13 cas du coraffinement par propagation ou les 3 cas du coraffinement par balayage, nous n'avons traité que des cas où l'intersection se trouve à l'intérieur des deux surfaces. Le cas particulier où une arête "sort" d'un triangle qui se trouve au bord est différent. Dans ce cas, des brins sont créés alors qu'ils ne sont pas nécessaires. Il serait nécessaire de traiter ces cas séparément. Cependant, pour ne pas dupliquer les cas d'intersections, nous avons décidé de fermer les surfaces étudiées avant de faire le coraffinement. De cette manière, aucune variante due au bord n'est à programmer. Un post-traitement fera le nécessaire pour supprimer le ou les polygones résultant de cette fermeture.

Pour cela, il est nécessaire de connaître les brins qui sont des brins du bord avant et après le coraffinement. Au cours du traitement de l'opération, il va falloir marquer les nouveaux brins qui sont au bord pour permettre au post-traitement de supprimer les cellules qui ne sont constituées que par des brins du bord.

Dans une première étape, les surfaces à coraffiner seront fermées. Tous les brins qui sont créés par cette fermeture sont marqués en tant que brins du bord. La deuxième étape va consister à marquer les nouveaux brins résultant d'un cas d'intersection si nécessaire. Pour faire cela, la règle est simple. Nous allons nous intéresser à un cas d'intersection qui met en jeu deux arêtes se coupant en leur milieu. Plusieurs cas sont à prendre en compte :

- aucun brin des deux arêtes n'est au bord ; puisque ces arêtes se trouvent à l'intérieur des surfaces, il ne faut pas marquer de nouveaux brins en tant que brins du bord (figure **??**-1).

- l'une des deux arêtes contient des brins du bord mais pas la seconde. Cela signifie que l'arête en question se trouve à l'intérieur de la seconde surface.

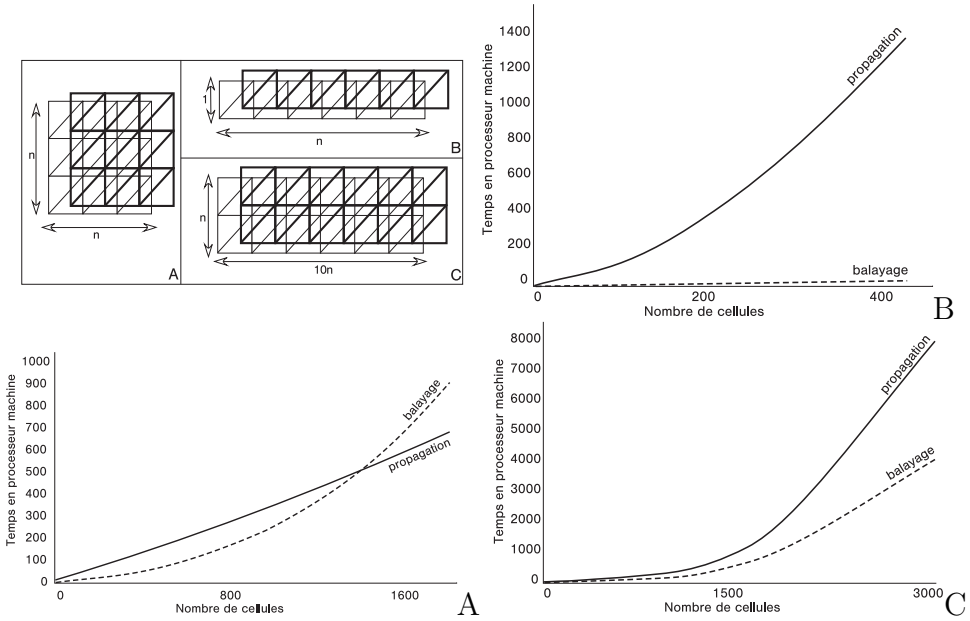


Figure 3.14 *Surfaces de tests. (A) Chaque surface contient* $2n^2$ *triangles dont* $2n$ *arêtes se trouvant au bord. (B) Chaque surface contient* $2n$ *triangles, tous au bord. (C) Chaque surface contient* $20n^2$ *triangles dont* $11n$ *arêtes au bord.*

A nouveau, il ne faut pas marquer les nouveaux brins (figure **??**-2).

- finalement, les deux arêtes contiennent des brins du bord. Certains brins sont alors à marquer : ceux qui se trouvent vers l'extérieur. (figure **??**-3).

Le traitement final consiste à supprimer les 2-cellules qui ne contiennent que des brins du bord. Ce traitement ouvre à nouveau la surface.

3.2.4 Comparatif et résultats

Dans cette section, nous comparons les deux méthodes de coraffinement 2d que nous avons implantées.

Dans les tableaux de la figure **??**, le temps de calcul du coraffinement par rapport aux nombres de cellules permet de visualiser la différence entre nos deux méthodes. Ces comparaisons ont été faites sur une cinquantaine de cas.

Les surfaces utilisées dans le premier tableau (figure **??**-A) représentent des grilles régulières triangulées décalées l'une par rapport à l'autre. Avec un nombre réduit de triangles, l'algorithme de balayage est plus rapide que celui de propagation. Mais à partir de 1000 triangles par surface, l'algorithme de propagation devient plus rapide que celui de balayage. Nous avons fait des tests avec 10000 triangles et avons constaté un facteur de $2,5$ entre les deux algorithmes. Le

coraffinement par balayage est en $\mathcal{O}(nlogn)$ comme le montre sa courbe sur cette figure. Pour le coraffinement par propagation, sa courbe est linéaire dans ce cas.

Dans le deuxième tableau (figure **??**-B), nous mettons en évidence l'effet des arêtes du bord des surfaces. En effet, le traitement particulier, qui a dû être mis en place afin de traiter ces arêtes, ralentit l'algorithme de propagation. Dans notre exemple, tous les triangles sont au bord. Tandis que le temps nécessaire au calcul du coraffinement par l'algorithme de balayage évolue très lentement, celui de l'algorithme de propagation augmente très rapidement.

Finalement, dans le troisième tableau (figure **??**-C), le nombre de triangles au bord est moins important que le nombre de triangles à l'intérieur de la surface. Cependant, l'effet des bords se fait ressentir sur l'algorithme de propagation.

Au vu de ces résultats, il est intéressant d'utiliser le coraffinement par propagation si une surface est entièrement incluse dans une autre mais il est préférable d'utiliser la deuxième méthode dès lors que les surfaces ne s'intersectent que partiellement.

3.3 Coraffinement 3D

Pour les mêmes raisons que nous avons données pour le coraffinement 2D, nous nous restreignons aux tétraèdres ou du moins à des polyèdres convexes à face triangulaires. Mais il sera fort intéressant de modifier ce coraffinement pour utiliser des hexaèdres afin de pouvoir coraffiner des objets tétraédrisés avec des grilles régulières. La section **??** des applications géologiques page **??** montre l'intérêt d'un tel coraffinement.

3.3.1 Cas d'intersection de deux triangles

Etant donné qu'en deux dimensions nous avions traité tous les cas d'intersections entre des segments, nous avons commencé par étudier le cas d'intersections de triangles entre eux. L'intersection entre deux triangles plongés dans $\mathbb{R}^3$ peut être :

- un point,
- un segment,
- un polygone

En 2D, il était possible de traiter tous les cas d'intersections de deux segments entre eux : nous en comptions 5 différents. En 3D, il s'avère impensable de vouloir traiter tous les cas d'intersections de deux triangles dans l'espace. En effet, ce nombre est beaucoup trop grand, la figure **??** nous montre un petit échantillon des cas d'intersections de deux triangles. D'ailleurs, rien que dans le plan, le nombre de cas d'intersection de deux triangles est énorme : nous en avons décompté plus

de 95 tout en sachant que nous ne les avions pas tous répertoriés : nous n'avons pas trouvé de critère fiable qui nous permette de tous les déterminer.

Le coraffinement 3D peut se ramener très facilement au coraffinement 2D, lorsque le résultat d'une intersection est :

- un polygone : dans ce cas, les deux triangles sont dans un même plan, on peut se rapporter au coraffinement 2D (voir les exemples de la figure **??** en bas).

- un segment : dans ce cas, ce segment est défini par deux points, on peut se rapporter à l'un des 13 cas du coraffinement 2D (voir figure **??**).

- un point : un sommet d'un triangle touche un sommet, une arête ou la face du deuxième triangle ou bien deux arêtes se touchent en un point (voir les 4 exemples de la figure **??** en haut). Si ce sont deux sommets qui se touchent, il n'y a rien à faire. Si un sommet touche un segment, on peut diviser ce segment en deux. Si un sommet touche un triangle, on peut choisir de ne pas insérer ce point dans ce triangle. En effet, rajouter un point dans un triangle nécéssiterait d'avoir une structure d'arbres d'inclusions pour permettre ce genre de configuration. Nous proposons plutôt d'insérer une arête entre un noeud du triangle et le nouveau noeud ou bien de ne pas insérer le nouveau noeud dans le triangle. Finalement, si deux arêtes se croisent, il faut diviser les deux arêtes pour rajouter le nouveau noeud.

Cependant, la difficulté se situe dans le raccordement des intersections déjà calculées.

Nous expliquons dans un premier temps la syntaxe que nous utilisons pour exprimer l'implantation faite pour le coraffinement 3D. Rappelons que nous voulons coraffiner deux volumes, que nous appellerons respectivement V_1 et V_2. Le résultat du coraffinement se fait dans un autre volume appelé V_3. Pour le coraffinement 2D, nous copions simplement l'un des deux objets à coraffiner dans un troisième, puis nous inserions le deuxième objet en effectuant les calculs d'intersections des segments à partir de cet objet final. Un grand changement est nécessaire pour le coraffinement 3D. Toutefois, la première étape consiste à copier les deux objets à coraffiner dans un troisième mais les triangles à intersecter seront les triangles des volumes V_1 et V_2. Seules les intersections seront réalisées dans V_3. En effet, pour le coraffinement 2D, l'élément de base était le segment qui, même coupé en deux, reste un segment. Alors que pour le coraffinement 3D, si l'on coupe entièrement ou partiellement un triangle (élément de base), nous n'obtenons plus forcément des triangles. C'est pourquoi, nous avons décidé de garder les objets de départ et de faire les calculs d'intersection ainsi que les parcours des triangles sur ces deux objets. Ainsi, V_3 n'est pas parcouru en cours de traitement mais il est modifié localement coup par coup. Cette méthode nécessite de posséder une syntaxe claire pour ne pas perdre le lecteur ou le programmeur dans certaines ambiguités. Par

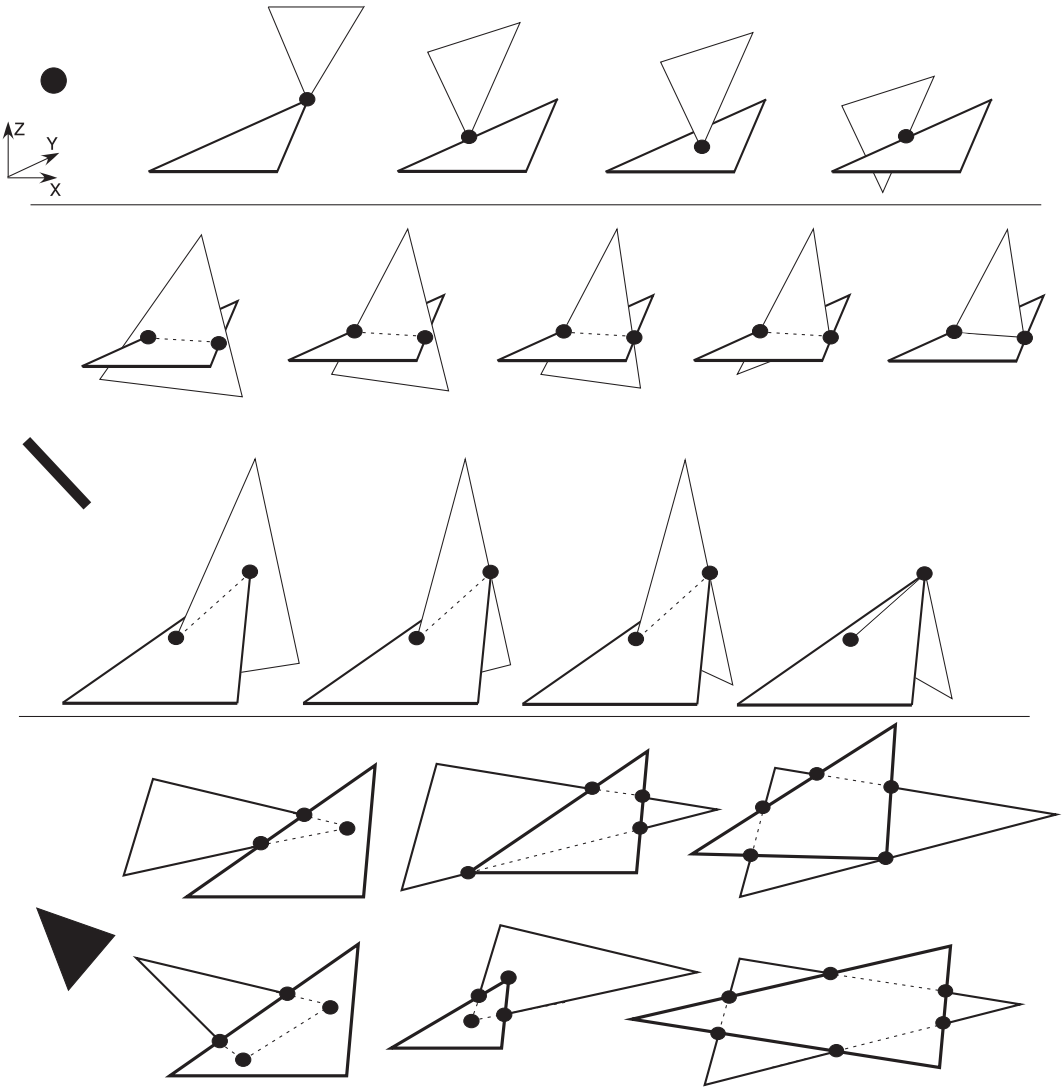


Figure 3.15 *Quelques cas d'intersections de deux triangles dans $\mathbb{R}^3$. De haut en bas, l'intersection est un point, une arête ou un polygone. Parmi les intersections dont le résultat est un point, il existe quatre cas possibles : sommet-sommet, sommet-arête, sommet-face et arête-arête. La première série d'intersection dont le résultat est une arête montre tous les cas possibles d'intersection entre deux triangles où le résultat correspond à une arête dont les extrémités se trouvent sur des arêtes opposées du triangle horizontal. La deuxième série montre des cas d'intersection où l'arête résultante a ses extrémités sur un sommet et dans la face du triangle horizontal. Les 6 dernières figures représentent un petit éventail des possibilités d'intersection dont le résultat est un polygone. Les polygones résultant peuvent avoir de 3 à 6 cotés pour des triangles.*

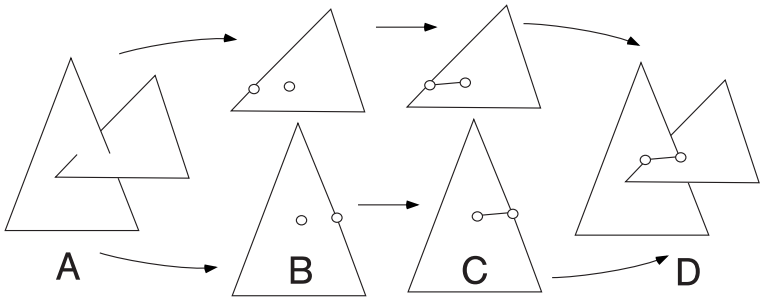


Figure 3.16 *Exemple de coraffinement 3D où l'intersection est un segment. (A) Calcul d'intersection entre deux triangles (B) Détermination des points d'intersection sur chaque triangle. (C) Insertion de la cicatrice. (D) Fusion des deux triangles modifiés.*

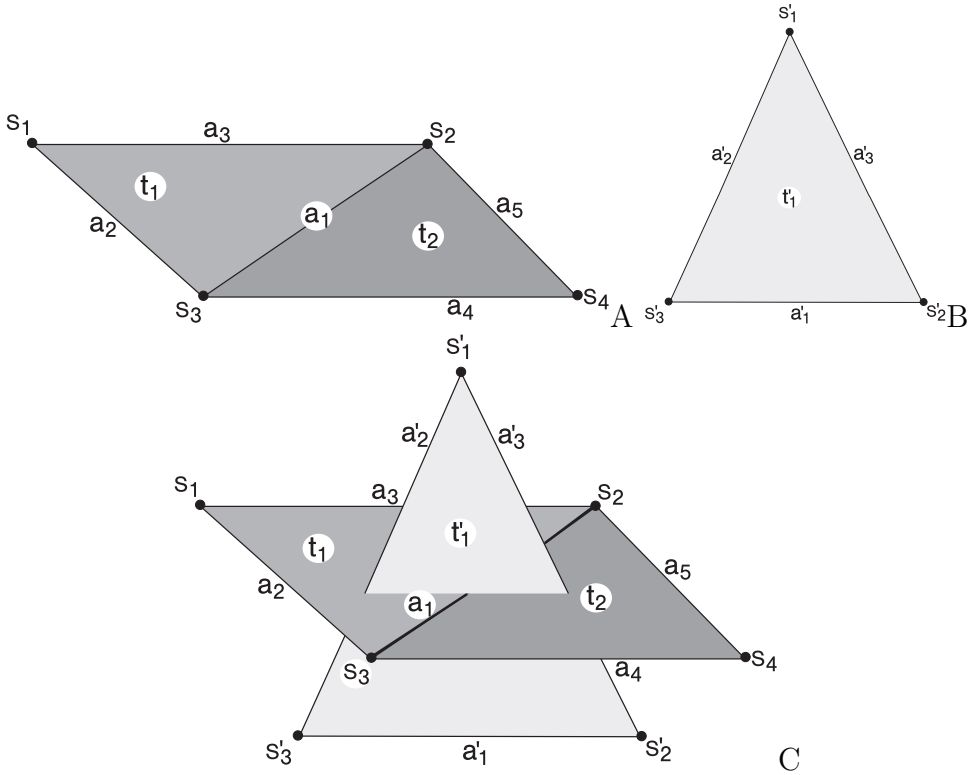


Figure 3.17 *Syntaxe (A) deux triangles du volume V_1 (B) un triangle du volume V_2 (C) copie des trois triangles dans V_3*

exemple, soit t_1 un triangle de V_1, lorsque nous parlerons du triangle qui est une copie de t_1 dans le volume final nous parlerons du triangle t_1 de V_3 et cela, même si ce triangle a été modifié en cours de route par le coraffinement.

Une structure annexe est nécessaire afin d'éviter de calculer plusieurs fois de suite la même intersection. En effet, prenons deux triangles t_1 et t_2 reliés entre eux le long d'une arête a_1 d'un volume V_1. Supposons que cette arête coupe un triangle t'_1 d'un volume V_2. En procédant comme nous l'avons expliqué plus haut, nous allons couper le triangle de V_2 par les deux triangles de V_1. Si aucun pré-traitement n'est mis en action, l'arête commune va couper deux fois le même triangle. Deux possibilités s'ouvrent alors : ou nous obtenons deux points différents du fait de la précision numérique de la machine ou nous obtenons effectivement deux points identiques. Dans les deux cas, il ne faut pas *insérer* ces points dans le triangle V_2 : il ne faut en insérer qu'un seul. C'est pour cette raison qu'il est nécessaire de savoir si un point d'intersection a déjà été calculé ou non.

La structure qui permet de stocker cette information est simple :

Structure de stockage d'intersection
Cellule : cellule du volume 1 Cellule : cellule du volume 2 Brin : brin du point d'intersection du volume 1 dans le volume 3 Brin : brin du point d'intersection du volume 2 dans le volume 3

Ainsi, pour notre exemple précédent, lors du premier calcul d'intersection entre le triangle t_1 de V_2 et le triangle t'_1 de V_2, nous stockons l'information suivante :

Struct {
  Cellule : arête a_1
  Cellule : triangle t'_1
  Brin : brin b_1
  Brin : brin b_2
}

avec b_1 un brin de l'arête a_1 du volume V_3 que l'on a dû diviser en deux pour insérer un nouveau point et b_2 un brin du point à insérer dans la triangle t'_1 de V_3.

Ensuite, lors du calcul de l'intersection de t_2 avec t'_1. Nous consulterons d'abord si une intersection n'a pas déjà été calculée. Dans notre cas, nous saurons que l'arête a_1 a déjà coupé le triangle t'_1. Il ne faut donc pas créer un point dans le triangle t'_1 ni couper a_1 en deux, mais il faudra utiliser les brins b_1 et b_2 pour réaliser la cicatrice du couple (t_2,t'_1).

3.4 Prédicats exacts

La géométrie algorithmique a commencé dans les années 70 et depuis a été développée dans différents domaines scientifiques. La recherche dans ce domaine est très active [?, ?, ?, ?, ?].

En informatique, l'utilisation d'algorithmes géométriques peut engendrer des erreurs numériques. En effet, une machine n'est pas capable, par défaut, de faire des calculs exacts, lors de division par exemple. Afin d'éviter ces problèmes, il est possible d'utiliser des prédicats.

Pour expliquer l'intérêt des prédicats, nous allons traiter un exemple simple. La question que l'on se pose est la suivante :

Est ce que deux segments [AB] et [CD] s'intersectent ?

Il y a deux méthodes pour résoudre ce problème :

- la première consiste à résoudre le système d'équations suivant :

$$\begin{array}{l} Y = a.X + b \\ Y = a'.X + b' \\ \text{avec } a = \frac{B_y - A_y}{B_x - A_x}, b = A_y - A_x.\frac{B_y - A_y}{B_x - A_x}, a' = \frac{D_y - C_y}{D_x - C_x} \text{ et } b' = C_y - C_x.\frac{D_y - C_y}{D_x - C_x}, \end{array} \quad (3.1)$$

  Au cours de la résolution de ce système d'équation, il faut tenir compte du cas où le dénominateur de ces fractions s'annule. C'est pourquoi nous préférerons utiliser la deuxième méthode.

- la deuxième méthode représente les équations sous leur forme paramétrique :

$$\begin{array}{l} M = A + t.\overrightarrow{AB} \\ M = C + t'.\overrightarrow{CD} \\ \text{où } M \text{ est le point d'intersection recherché} \end{array} \quad (3.2)$$

  En résolvant cette équation, on obtient des valeurs pour t et t'. Si ces valeurs se situent entre 0 et 1, il existe un point d'intersection M.

Quelque soit la méthode utilisée, son principal inconvénient est son temps en calcul car elle requièrt de nombreuses opérations d'additions, de multiplications et de divisions. De plus l'opération de division est une opération qui manque de précision en informatique, ce qui complique l'obtention d'un résultat fiable. Il existe une solution plus simple, d'autant plus que la question posée ne concernait pas le point d'intersection mais l'existence ou l'absence d'une intersection.

C'est à ce niveau que les prédicats entrent en jeu. Ce sont des fonctions qui prennent en argument des objets simples (points de l'espace, segments, ...) et qui retournent *vrai* ou *faux*. Prenons l'exemple de l'orientation d'un point R par rapport à un segment $[PQ]$. La fonction *orientation(*P, Q, R*)* retourne un résultat négatif si R se trouve à *droite* de $[PQ]$, positif s'il est à gauche et nul si les trois points sont alignés.

Ainsi, pour répondre à la question de départ, nous allons nous servir de la fonction d'orientation. En effet, si le segment $[AB]$ intersecte le segment $[CD]$, cela signifie que les points A et B se trouvent de part et d'autre de $[CD]$. En utilisant les prédicats, la troisième méthode s'écrit de la manière suivante :

Si orientation$(C, D, A) \neq$ orientation(C, D, B)
Alors
les points A et B se trouvent de part et d'autre de $[CD]$.
Sinon
les points A et B se trouvent d'un même côté
par rapport au segment $[CD]$.
FinSi

Cependant, cette condition n'est pas suffisante. Il faut également que les points C et D se trouvent de part et d'autre du segment $[AB]$.

Par contre, si l'on veut effectivement connaître les coordonnées du point d'intersection, il va falloir réaliser des calculs numériques en prenant le risque d'avoir des problèmes de précision. Une manière efficace de s'affranchir de ces problèmes consiste à utiliser le calcul exact. Dans le cas de l'évaluation d'un prédicat géométrique, un recours systématique à du calcul exact onéreux peut souvent être évité par l'utilisation de filtres. Le prédicat est évalué en flottant et un calcul d'erreur permet de certifier la validité du résultat [**?**].

Une autre méthode consiste à perturber le problème et l'univers : les coordonnées des points sont légèrement modifiées de manière à ne jamais obtenir des points alignés ou co-cycliques. Les cas dégénérées sont ainsi évités, mais les problèmes de robustesse ne sont pas résolus. Devillers *et al* [**?**] montrent que ces approches permettent de concevoir simplement et efficacement une perturbation des points. En effet, soit quatre points p_1, p_2, p_3 et p_4, si un algorithme dit que p_4 se trouve sur le cercle formé par les trois premiers points, il ne dira peut être pas que p_1 se trouve sur le cercle formé par les trois derniers points. Ceci est dû à la précision numérique des machines. En perturbant l'univers, ces quatre points ne seront jamais considérés comme co-cycliques.

Nous n'avons pas traité le calcul exact dans notre étude car l'utilisation des prédicats était déjà suffisante pour les tests que nous avons effectué. Cependant, il sera nécessaire d'implanter de tels calculs pour avoir un coraffinement correct dans tous les cas.

Un prédicat est une fonction qui prend en argument des éléments simples (i.e. des points, segments) et retourne une valeur discrète (i.e. booléen ou type énuméré). Il effectue la passerelle entre les deux parties logiques d'un algorithme : le numérique et le combinatoire [**?**, **?**, **?**]. Dans la suite de cette section, nous allons définir les prédicats que nous avons utilisé pour le coraffinement.

3.4.1 Prédicat de base

Le prédicat de base détermine si le réel passé en argument est positif, négatif ou nul.

$$sign(a) = \begin{cases} 1 & \text{si } a > 0 \\ 0 & \text{si } a = 0 \\ -1 & \text{si } a < 0 \end{cases}$$

Propriété

$$sign(ab) = sign(a/b) \text{ avec } b \neq 0$$

$$sign(ab) = sign(a).sign(b)$$

Cette propriété triviale permet de réduire le degré des prédicats.

3.4.2 Prédicats de comparaison

A partir du prédicat de base, il est facile de déterminer le prédicat de comparaison.

$$compare(a, b) = sign(a - b) = \begin{cases} 1 & \text{si } a > b \\ 0 & \text{si } a = b \\ -1 & \text{si } a < b \end{cases}$$

Les trois prédicats suivants comparent des points ou intersections de points : P, Q et R sont des points, S_i sont des segments. Ces prédicats sont utilisés pour trier les points de gauche à droite.

- Comparaison entre deux points (Figure **??**-A)

  On définit la fonction $compare_x(P, Q)$ comme la fonction qui compare les coordonnées x de ces deux points.

  $$compare_x(P, Q) = compare(P_x, Q_x)$$

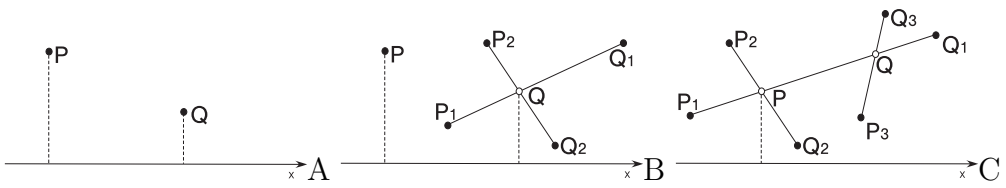


Figure 3.18 *A. Comparaison entre deux points : compare les abscisses des points P et Q. La fonction associée qui se nomme $compare_x$ retournera une valeur positive si $P_x > Q_x$, négative si $P_x < Q_x$ ou nulle si $P_x = Q_x$. B. Comparaison entre un point et deux segments : compare les abscisses des points P et Q où Q est le points d'intersection des segments (P_1Q_1) et (P_2Q_2) C. Comparaison entre trois segments : compare les abscisses des points P et Q qui sont respectivement les points d'intersections des segments $(P_1Q_1), (P_2Q_2)$ et $(P_1Q_1), (P_3Q_3)$.*

- Comparaison entre un point et deux segments (Figure **??**-B)

  Cette opération consiste à comparer un point avec le point résultant de l'intersection de deux segments. Remarquez que le point résultant n'est pas directement précalculé, il est défini par les quatre points des deux segments qui le forment.

$$compare_x(P, S_1, S_2) \text{ avec } \begin{cases} S_1 & : (P_1, Q_1) \\ S_2 & : (P_2, Q_2) \end{cases}$$

  L'équation cartésienne des droites passant par les segments S_1 et S_2 est de la forme :

$$ax + by + c = 0 \text{ avec } a = Q_y - P_y, b = P_x - Q_x \text{ et } c = Q_xP_x - Q_yP_y.$$

  L'abscisse du point d'intersection est donnée par :

$$Q_x = \frac{\begin{vmatrix} b_1 & c_1 \\ b_2 & c_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}} = \frac{\begin{vmatrix} P_{1x} - Q_{1x} & Q_{1x}P_{1y} - Q_{1y}P_{1x} \\ P_{2x} - Q_{2x} & Q_{2x}P_{2y} - Q_{2y}P_{2x} \end{vmatrix}}{\begin{vmatrix} Q_{1y} - P_{1y} & P_{1x} - Q_{1x} \\ Q_{2y} - P_{2y} & P_{2x} - Q_{2x} \end{vmatrix}} = \frac{A}{B}$$

  Ainsi, l'expression de ce prédicat est la suivante :

$$\begin{aligned} compare_x(P, S_1, S_2) &= compare(P_x, Q_x) \\ &= compare(P_x, \frac{A}{B}) \\ &= sign(P_x - \frac{A}{B}) \end{aligned}$$

$$\begin{aligned} &= sign(\frac{P_x.B - A}{B}) \\ &= sign(P_x.B - A).sign(B) \\ &= compare(P_x.B, A).sign(B) \end{aligned}$$

- Comparaison entre trois segments (Figure **??**-C)

  Cette opération consiste à comparer les points d'intersection P et Q du premier segment avec les deuxième et troisième segments.

$$P_x = \frac{\begin{vmatrix} b_1 & c_1 \\ b_2 & c_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}} = \frac{\begin{vmatrix} P_{1x} - Q_{1x} & Q_{1x}P_{1y} - Q_{1y}P_{1x} \\ P_{2x} - Q_{2x} & Q_{2x}P_{2y} - Q_{2y}P_{2x} \end{vmatrix}}{\begin{vmatrix} Q_{1y} - P_{1y} & P_{1x} - Q_{1x} \\ Q_{2y} - P_{2y} & P_{2x} - Q_{2x} \end{vmatrix}} = \frac{A}{B}$$

$$Q_x = \frac{\begin{vmatrix} b_1 & c_1 \\ b_3 & c_3 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_3 & b_3 \end{vmatrix}} = \frac{\begin{vmatrix} P_{1x} - Q_{1x} & Q_{1x}P_{1y} - Q_{1y}P_{1x} \\ P_{3x} - Q_{3x} & Q_{3x}P_{3y} - Q_{3y}P_{3x} \end{vmatrix}}{\begin{vmatrix} Q_{1y} - P_{1y} & P_{1x} - Q_{1x} \\ Q_{3y} - P_{3y} & P_{3x} - Q_{3x} \end{vmatrix}} = \frac{C}{D}$$

$$\begin{aligned} compare_x(S_1, S_2, S_3) &= compare(P_x, Q_x) \\ &= compare(\frac{A}{B}, \frac{C}{D}) \\ &= sign(\frac{A}{B} - \frac{C}{D}) \\ &= sign(\frac{A.D - C.B}{B.D}) \\ &= sign(A.D - C.B).sign(B.D) \\ &= compare(A.D, C.B).sign(B).sign(D) \end{aligned}$$

3.4.3 Prédicat d'orientation

Le prédicat d'orientation est un prédicat de base très utile. Il détermine de quel côté de la droite orientée se situe un point (cf figure **??**).

$$orient(P, Q, R) = sign\left(\begin{vmatrix} P_x & P_y & 1 \\ Q_x & Q_y & 1 \\ R_x & R_y & 1 \end{vmatrix}\right) = sign\left(\begin{vmatrix} P_x - R_x & P_y - R_y \\ Q_x - R_x & Q_y - R_y \end{vmatrix}\right)$$

Grâce à ce prédicat, des fonctions de test d'intersections vont pouvoir être élaborées.

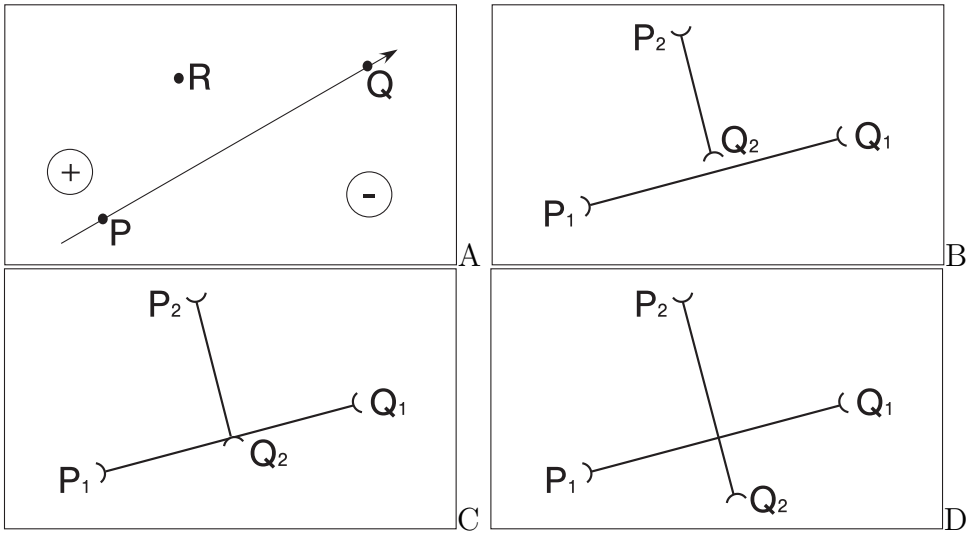


Figure 3.19 *A. Prédicat d'orientation : détermine si le point R se trouve* à gauche, sur *ou* à droite *de la droite* (PQ). *Intersection de deux segments ouverts : B. Les deux segments ne s'intersectent pas car les extrémités* P_2 *et* Q_2 *ne se trouvent pas de part et d'autre du segment* $[P_1, Q_1]$. *C. Les deux segments ouverts ne s'intersectent pas même si* Q_2 *se trouve sur le segment* $[P_1, P_2]$. *D. Les deux segments s'intersectent.*

3.4.4 Fonction de test d'intersections

En combinant les prédicats exacts cités ci-avant, nous allons pouvoir construire des prédicats évolués sur des éléments simples (point et segment).

La fonction d'intersection de deux segments se base sur un postulat simple, deux segments S_1 et S_2 s'intersectent si et seulement si les extremités de S_1 se trouvent de part et d'autre du segment S_2 et réciproquement (cf début de ce chapitre).

Deux types d'intersections sont à distinguer :

- entre segments ouverts cf table **??**
- entre segments fermés cf table **??**

La figure **??**-B montre deux segments ouverts dans un plan. Les extrémités du segment $[P_1Q_1]$ se trouvent de part et d'autres du segment $[P_2Q_2]$ mais pas réciproquement : les segments ne s'intersectent pas. Sur la figure suivante, **??**-C, le point P_2 se trouve à gauche du segment $[P_1Q_1]$ et le point Q_2 sur le segment : il n'y a toujours pas d'intersection car :

$$\begin{aligned} orientation([P_1Q_1], P_2).orientation([P_1Q_1], Q_2) &= 1.0 \\ &\neq -1 \end{aligned}$$

```
Booléen IntersectionSegmentSegmentOuvert(S1 : segment 2D, S2 : segment 2D)
   Si(
      orientation(S1,Point1(S2)).orientation(S1,Point2(S2)) = -1
   ET
      orientation(S2,Point1(S1)).orientation(S2,Point2(S1)) = -1
   )
      retourner VRAI
   sinon
      retourner FAUX
   FinSi
FinIntersectionSegmentSegmentOuvert
```

Table 3.1 *Intersection entre deux segments ouverts*

```
Booléen IntersectionSegmentSegmentFermé(S1 : segment 2D, S2 : segment 2D)
   Si(
      orientation(S1,Point1(S2)) ≠ orientation(S1,Point2(S2))
   ET
      orientation(S2,Point1(S1)) ≠ orientation(S2,Point2(S1))
   )
      retourner VRAI
   sinon
      retourner FAUX
   FinSi
FinIntersectionSegmentSegmentFermé
```

Table 3.2 *Intersection entre deux segments fermés*

Par contre, il y aurait un point d'intersection si les segments avaient été fermés ($1 \neq 0$). Finalement, les segments de la dernière figure **??**-D s'intersectent.

Pour déterminer si un point se trouve sur un segment $[AB]$ en utilisant les prédicats exacts, la première étape consiste à déterminer si ce point se trouve *entre* les deux points en utilisant le prédicat de comparaison lexicographique (voir table **??**). La seconde étape consiste à déterminer si ce point se trouve bel et bien *sur* le segment $[AB]$ en utilisant le prédicat d'orientation.

La comparaison dite *lexicographique* correspond à la comparaison que l'on

```
Booléen IntersectionSegmentPoint(S1 : segment 2D, P : point 2D)
   Si(
      ComparaisonLexicographique(Point1(S1),P) est inférieur à zéro
   ET
      ComparaisonLexicographique(P,Point2(S1)) est inférieur à zéro
   ET
      orientation(S1,P) = 0
   )
      retourne VRAI
   sinon
      retourn FAUX
   FinSi
FinIntersectionSegmentPoint
```

Table 3.3 *Intersection entre un segment et un point*

```
Entier ComparaisonLexicographique(P : point 2D, Q : point 2D)
   Si( Px < Qx ) retourne inférieur à zéro
   Si( Px > Qx ) retourne supérieur à zéro
   Si( Py < Qy ) retourne inférieur à zéro
   Si( Py > Qy ) retourne supérieur à zéro
   retourne 0
FinComparaisonLexicographique
```

Table 3.4 *Comparaison lexicographique sur deux points plongés dans* $\mathbb{R}^2$.

ferait pour comparer deux mots du dictionnaire en utilisant l'ordre alphabétique. La seule différence ici est que l'on va comparer les coordonnées de nos deux points, en commençant d'abord par comparer les abscisses puis les ordonnées si nécessaire.

3.5 Implantation des opérations booléennes

Dans la première section de ce chapitre, page 103, nous avons vu que pour évaluer une opération booléenne, il est possible de passer d'abord par un raffinement puis de sélectionner les parties nécessaires à la réalisation de l'opération.

Nous allons donc expliquer comment identifier les différentes parties du raf-

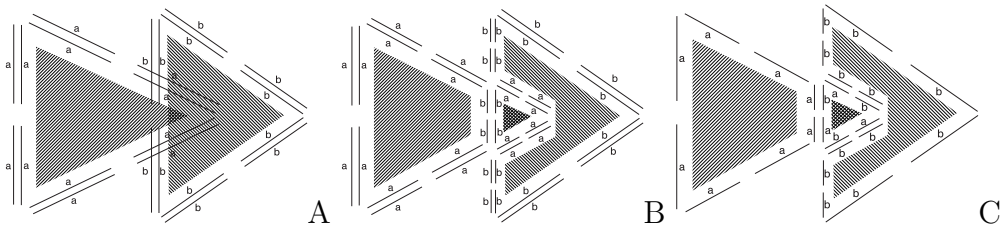


Figure 3.20 *Labelisation : A. Labelisation de chaque brin des deux objets, après fermeture. B. Coraffinement avant le changement de labelisation. C. Résultat final : les différentes parties du raffinement.*

finement de deux objets A et B. Il y a trois parties à distinguer :

- celle qui concerne l'ensemble des polygones venant uniquement de l'objet A ($A \backslash B$),
- celle qui concerne l'ensemble des polygones venant uniquement de l'objet B ($B \backslash A$),
- celle qui concerne l'ensemble des polygones qui proviennent de l'intersection des objets A et B ($A \cap B$).

Pour identifier ces différents parties, nous allons *labeliser* [**?**] (c'est-à-dire nommer) les brins. Ainsi, les brins des polygones appartenant à $A \backslash B$ seront labelisés par a, ceux appartenant à $B \backslash A$ par b et ceux appartenant à $A \cap B$ successivement par a et b.

La première étape de labelisation consiste à nommer tous les brins de l'objet A par a et ceux de B par b (figure **??**-A).

La deuxième étape s'effectue au cours du coraffinement. Lorsque de nouveaux brins sont créés, c'est-à-dire quand on divise une arête en deux, les nouveaux brins sont labelisés identiquement aux brins de l'arête divisée (figure **??**-B).

La dernière étape s'effectue à la fin de l'opération. Selon la position de certains brins, leur label change. La figure **??**-B nous montre les différentes parties du raffinement. Le polygone de gauche qui ne contient que des brins labelisés par a appartient à $A \backslash B$, celui de droite à $B \backslash A$ et enfin, celui du milieu qui possède des brins labelisés par a et b appartient à $A \cap B$.

Pour savoir quel brin doit changer de label, nous étudions l'exemple de la figure **??**. La première image représente deux polygones ou régions qui s'intersectent. Il faut distinguer les brins qui sont à l'intérieur des brins qui sont à l'extérieur des régions. Les brins intérieurs représentent le bord des régions tandis que les brins extérieurs (brins barrés) sont ceux qui ont été ajoutés pour fermer les surfaces (cf section 2.1.5). Les brins de l'objet A sont noirs tandis que ceux de B sont blancs.

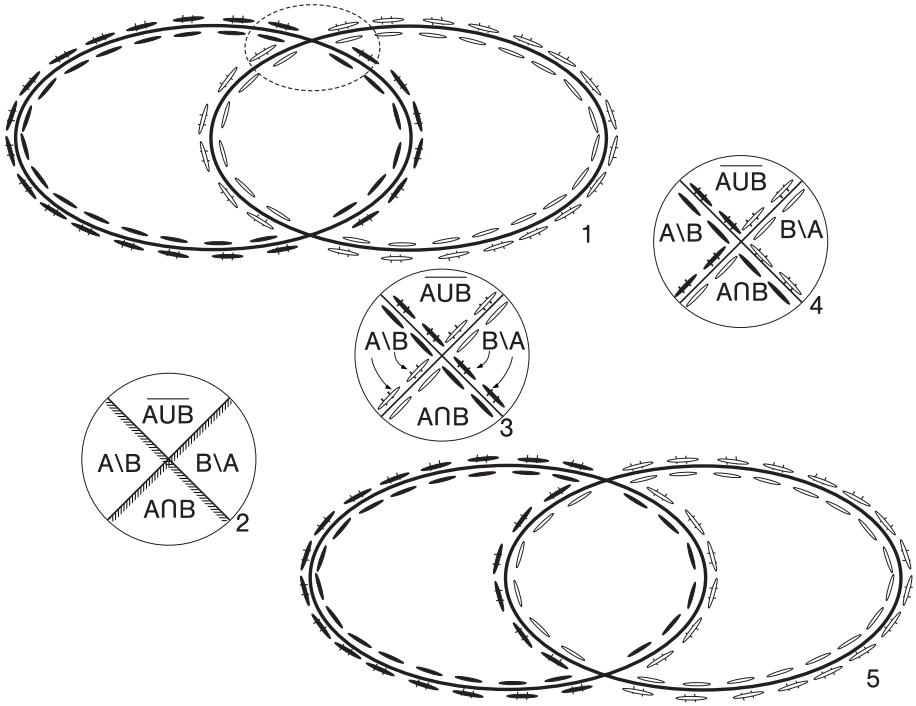


Figure 3.21 *Opérations booléennes sur les brins. (1) Labelisation après coraffinement. Les brins de l'objet A sont noirs, ceux de B sont blancs. (2) Quatre ensembles se trouvent autour de l'intersection de la zone en pointillé de la figure 1. (3) Superposition des brins actuels. (4) Remise à jour des couleurs des brins suivants l'ensemble dans lequel ils se trouvent. (5) Résultat final.*

Si l'on se focalise sur le point d'intersection de la zone en pointillé qui est agrandie sur la figure 2, on constate que cette intersection est commune à 4 ensembles : le quart de gauche correspond à $A\backslash B$, celui de droite à $B\backslash A$, en haut à $\overline{A \cup B}$ et en bas à $A \cap B$. Ceci correspond à la définition même des opérations booléennes.

Si l'on superpose autour de cette intersection les brins correspondants avec leurs couleurs actuelles, on obtient la figure 3. Certains brins n'ont pas la bonne couleur de l'ensemble dans lequel ils se trouvent. Ces brins sont désignés par des flèches. Signalons que ce sont uniquement des brins qui ont été ajoutés pour fermer les surfaces. Plus particulièrement, lorsqu'un brin de fermeture d'une certaine couleur est relié par α_1 à un brin d'une autre couleur qui n'est pas un brin de fermeture, ceci signifie que tous les brins du polygone corespondant à ces deux brins doivent être de la couleur opposée à la couleur du brin de fermeture. C'est ce qui est fait en figure 4 et rapporté à l'ensemble des brins de la figure 5.

Ainsi, les polygones de la région qui ne possède que des brins noirs (ici, la région qui se trouve la plus à gauche) sont des polygones de $A\backslash B$. Ceux de la

région qui ne possède que des brins blancs (à droite) sont des polygones de $B \backslash A$. Finalement, ceux de la région qui possèdent à la fois des brins noirs et des brins blancs (région du centre) sont des polygones qui appartiennent à $A \cap B$. Par ailleurs, les brins de la région qui englobe les trois régions citées ci-avant sont les brins qui devront être supprimés pour réouvrir la surface coraffinée.

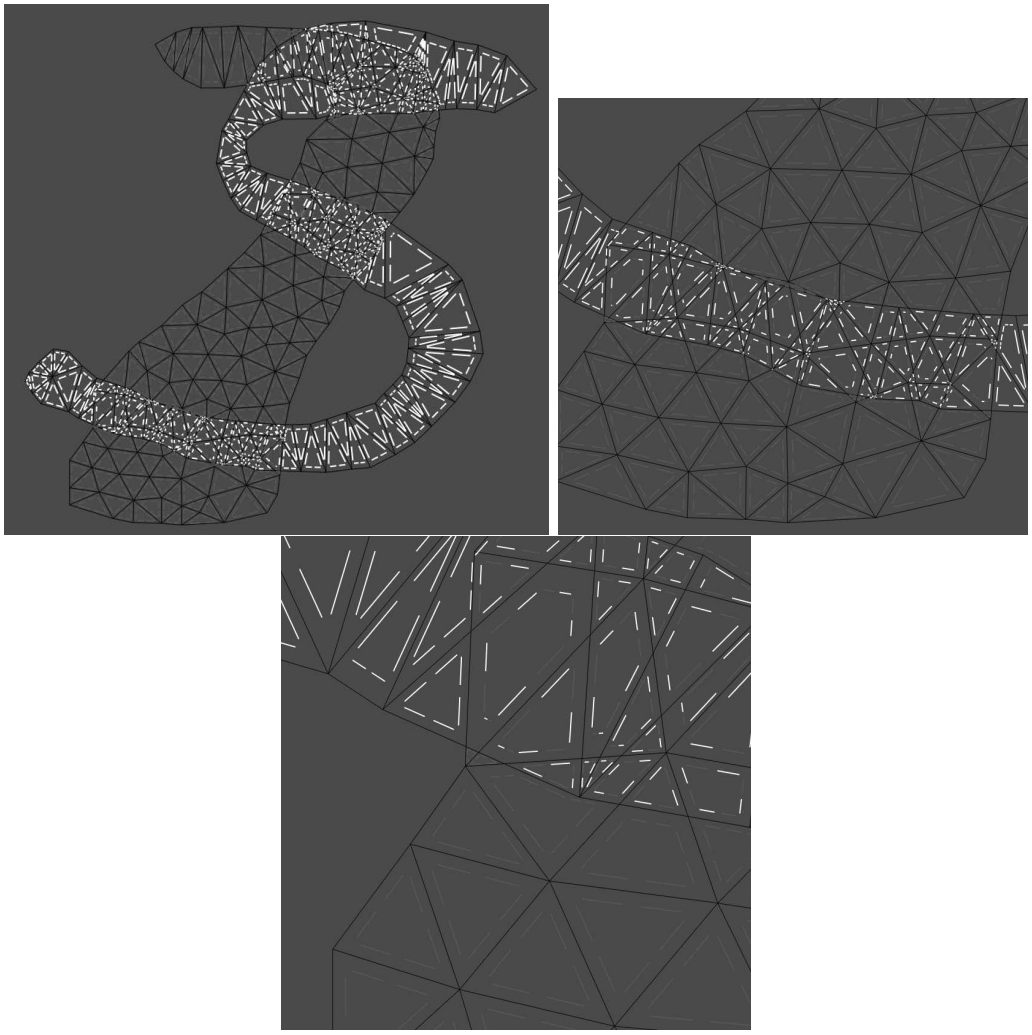


Figure 3.22 *Résultat d'opérations booléennes. Certains polygones ont des brins uniquement clairs, d'autres uniquement foncés et enfin les derniers polgones des brins mixtes.*

3.6 Application à la géologie

En géologie, divers travaux demandent des intersections de maillages 2D ou 3D. Un horizon peut intersecter une faille (ces deux surfaces sont triangulées). Cette opération calcule des intersections entre les triangles et retriangule les surfaces par la suite.

Deux chenaux (des volumes tétraédrisés) peuvent se chevaucher. Le calcul de l'union des deux chenaux permet de construire un volume unique tétraédrisé. Les tétraèdres qui se trouvent à l'intersection de deux chenaux ne sont plus dupliqués mais sont partagés ; ceci évite d'obtenir un calcul de volume faussé.

Dans les simulateurs de flux, les cellules qui se trouvent de part et d'autres d'une faille d'une couche faillée ne se correspondent pas. Cependant, en remaillant ces cellules, il est possible de mettre en correspondance chaque cellule qui se trouve d'un côté de la faille avec une cellule qui se trouve de l'autre côté. L'opération de coraffinement permet d'implanter toutes les opérations citées ci-dessus.

3.6.1 Coraffinement le long des failles

Certaines couches géologiques sont faillées. De part et d'autre de la faille se trouvent des *blocs de faille* comme le montre la figure **??**-A.

Des grilles 3D sont utilisées pour faire des calculs de flux. Les géologues veulent connaître pour toute cellule, celles qui se trouvent à leur proximité. Pour une cellule interne, son voisinage est direct. Cependant, pour les cellules qui se trouvent le long d'une faille, il faut déterminer celles qui se trouvent de l'autre côté et qui touchent notre cellule (figure **??**-B). Pour cela, nous allons utiliser la méthode de coraffinement. En effet, nous pouvons assimiler les deux bords des blocs qui se trouvent de part et d'autres de la faille à des surfaces. Une fois ces surfaces *plongées* dans un espace 2D, il nous est possible de construire le coraffinement (figure **??**-C). Par la suite, il reste à déterminer quelles cellules du bloc A se trouvent en contact avec des cellules du bloc B et réciproquement (figure **??**-D).

3.6.2 Coraffinement de bord

Le coraffinement de bord est utilisé en géologie pour introduire un objet (par exemple un chenal) dans une grille (voir figure **??**). Le maillage interne du chenal ne sera pas modifié. Par contre, le bord de ce dernier sera coraffiné avec les cellules de la grille de manière à ce qu'il y ait une parfaite intersection et une parfaite communication entre les deux objets. Ensuite, il ne reste plus qu'à supprimer les cellules de la grille qui se trouvent *dans* le chenal pour éviter la duplication des cellules (voir figure **??** en bas à droite).

Figure 3.23 *Contacts de cellules le long d'une faille. (A) Couches faillées. (B) Grilles des blocs A et B. (C) Coraffinement 2d des faces de cellules en contact avec la faille. (D) La face en trait épais du bloc B est en contact avec ces quatre cellules du bloc A.*

3.6.3 Erosion

Le phénomène d'érosion entre un élément principal B et un élément secondaire A consiste à *creuser* A par B (voir figure **??**). Les cellules de A qui s'intersectent avec celles de B sont supprimées. Seules les cellules de B qui sont en contact avec les cellules du bord de A sont coraffinées.

3.6.4 Calcul de lignes d'intersection

Le calcul de lignes d'intersection est largement utilisé dans la construction de modèles 3D. Les modeleurs en géologie ont à traiter des données qui sont des horizons et/ou des failles. Ces objets sont représentés par des surfaces (triangulées pour la plupart). Ces surfaces partitionnent l'espace 3D en *régions*. Pour obtenir

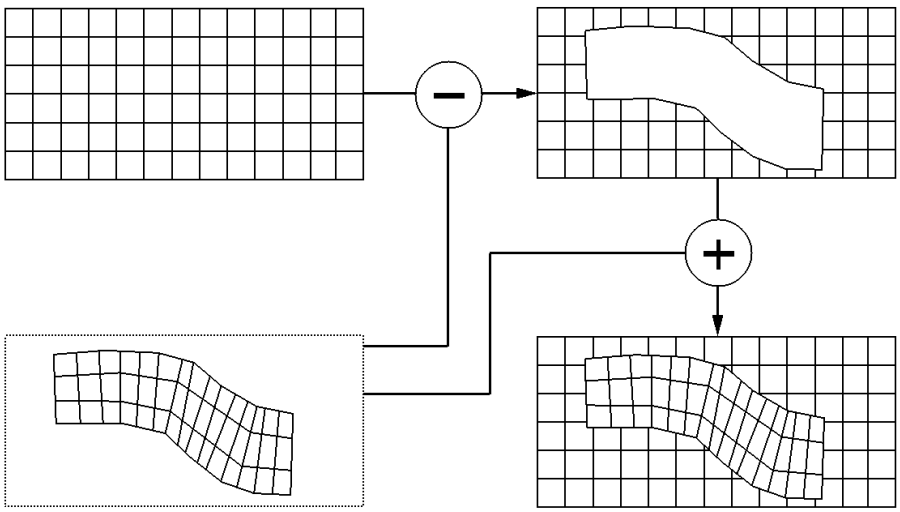


Figure 3.24 *Chenal dans une grille. En haut à gauche, la grille de départ. En bas à gauche, le chenal à insérer dans la grille. En haut à droite, opération booléenne (différence entre la grille et le chenal. En bas à droite, le résultat.*

ces régions, il est nécessaire de couper ces surfaces les unes par rapport aux autres de manière à pouvoir regrouper les parties de surfaces qui limitent une région (voir figure **??**).

3.7 Conclusion

Nous avons vu dans ce chapitre un outil de modélisation qui englobe plusieurs applications qui peuvent être utilisées en géologie. Le coraffinement permet :

- d'évaluer toutes les opérations booléennes entre deux objets,
- de construire la ligne d'intersection de deux surfaces qui se coupent,
- de déterminer le voisinage de cellules de part et d'autre d'une faille

Pour les surfaces, nous avons implanté deux algorithmes. L'un se base sur la propagation, il est rapide lorsque la surface à insérer dans la surface support se trouve entièrement englobée par cette dernière, mais possède beaucoup de cas d'intersection ce qui allourdit sa programmation. Le deuxième algorithme se sert d'une ligne de balayage. L'opération ne possède que trois cas d'intersection à traiter. Elle permet aussi bien de traiter des surfaces avec des polygones convexes que concaves.

Pour les solides, une seule méthode a été implantée. Cette méthode est à un stade de prototype avancé et permet de traiter les cas d'intersections les plus courants.

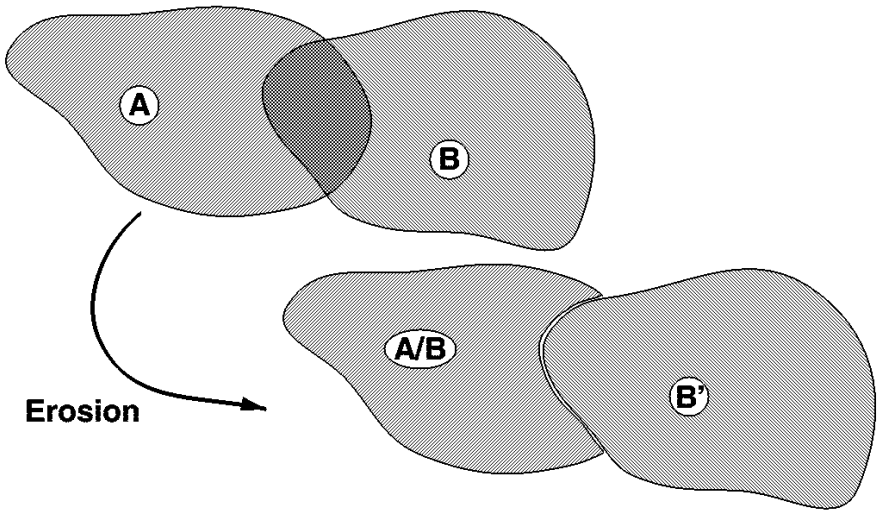


Figure 3.25 *Erosion de A par B. Seules les arêtes du bord de B sont coraffinées avec les arêtes de A. Les cellules qui s'intersectent entièrement avec B ont été supprimées. Les cellules du bord de B qui sont en contact avec A ont été coraffinées pour obtenir l'objet B'.*

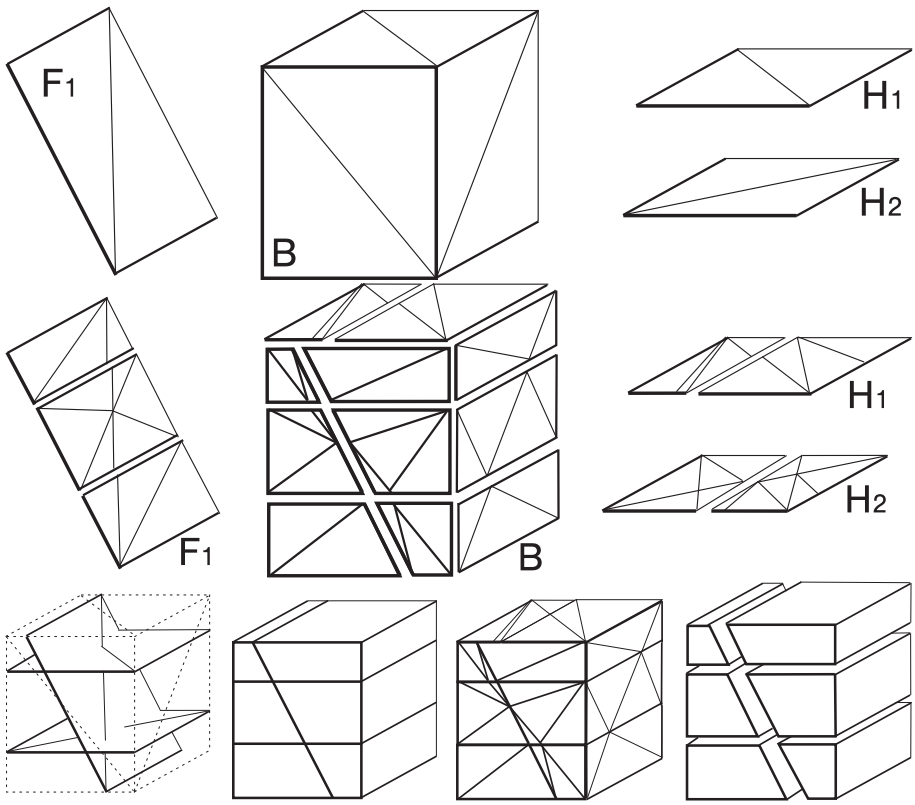


Figure 3.26 *(En haut) faille, boîte englobante et deux horizons. (Au milieu) après le calcul d'intersection. (En bas) modèle compact, triangulé et explosé pour une meilleur visibilité.*

Conclusion

Dans cette thèse, nous avons dans un premier temps expliqué le cadre géologique qui a stimulé cette étude.

Nous avons pleinement utilisé les cartes généralisées hiérarchiques afin de représenter des modèles géologiques qui partitionnent l'espace en régions.

Le chapitre 2 nous a montré comment utiliser l'aspect générique des G-Cartes dans certains algorithmes topologiques comme :

- la destruction de cellules de dimension i d'une n-G-Carte,
- la résorption de cellules en un point,
- la scission de cellules ou
- la fusion de n-cellules le long d'une i-cellule commune.

La suite de ce chapitre regroupe différentes opérations telles que le calcul du dual qui est trivial avec les G-Cartes. Nous y avons cependant ajouté une méthode pour fermer les duaux obtenus de manière à obtenir un bord lisse relativement proche du bord du primal. L'extrusion de surfaces polygonales nous permet de construire des grilles polyèdrales utilisées par les géostatisticiens. Enfin, nous avons énuméré plusieurs méthodes de subdivisions de maillages qui nous permettent d'obtenir des surfaces ou volumes ayant des caractéristiques intéressantes suivant l'utilisation que l'on veut en faire (degré des sommets croissants, accroissements du nombre de sommets tout en gardant des triangles homogènes, ...).

L'étude des algorithmes locaux nous a permis de construire des algorithmes simples à écrire et à maintenir.

Finalement, la création d'une interface de haut niveau permet aux clients d'écrire des algorithmes simples de manière intuitive sans perdre l'aspect mathématique du noyau topologique de l'interface de bas niveau.

Nous avons également implanté un coraffinement 2d à l'origine sur des surfaces triangulées et qui s'étend facilement à des cellules convexes pour la méthode de coraffinement par propagation. Pour utiliser des cellules concaves, il faudrait prendre en compte toutes les intersections entre le segment courant et la cellule considérée puis determiner quelle intersection est la plus proche. C'est deja ce que

nous faisons lorsque le segment courant se trouve dans le polygone qui correspond à l'extérieur de la surface. En ce qui concerne la méthode de coraffinement par balayage, les cellules considérées peuvent être concaves. Cette méthode ne traite que des cas d'intersections entre des segments. La seule étape importante est le tri de ces segments dans un ordre croissant.

Pour le coraffinement 3d, nous n'avons pas un algorithme général mais un prototype assez avancé qui nous permet de traiter certains cas d'intersections. La première étape consiste à insérer les arêtes d'intersection d'un volume sur l'autre et réciproquement. La seconde étape fusionne ces arêtes entre elles de manière à ce que les deux volumes n'en forment plus qu'un.

Perspectives

Nous avons utilisé des prédicats exacts pour pallier aux problèmes d'approximations et d'arrondis. L'utilisation de ces prédicats nous a permis d'éliminer rapidement plusieurs erreurs que nous avons pu obtenir lors des premiers essais de coraffinement 2D. Cependant, le calcul des points d'intersection ne se fait pas à l'aide de réels exacts. Cela signifie par exemple qu'en insérant un point sur un segment, il en résulte 2 segments qui ne sont pas correctement alignés.

Pour nos tests, il s'est avéré que les prédicats exacts étaient suffisants. Pour une industrialisation plus poussée, il faut apporter d'autres solutions et notamment dans le passage des coordonnées des points en calculs exacts vers des réels informatiques. Ce passage peut créer des intersections supplémentaires. En effet, un point qui se trouve à gauche d'un segment avec des coordonnées exactes peut passer à sa droite en coordonnées réelles.

Grâce à la généralité des G-Cartes, il est tout à fait envisageable de construire des objets de dimension 4. En géologie, un tel objet peut représenter un modèle tridimensionnel dans le temps. Ainsi, à partir d'un modèle contemporain et son état initial (plusiers milliers d'années en arrière), il est possible de simuler son activité tectonique pour voir apparaitre les failles finales au fur et à mesure que l'on avance sur l'axe des temps.

Pour visualiser un tel modèle, il est nécessaire de faire une coupe dans l'objet 4D au temps t désiré. En étandant l'opération de coraffinement, il est possible d'obtenir la coupe voulue. Dans [**?**], Skapin et Lienhardt nous montrent comment utiliser des objets 4D pour faire de l'animation par exemple (création d'objets par extrusion, ...).

Avenir

Pour les mois à venir, le noyau topologique utilisé va être industrialisé au sein de l'entreprise T-Surf. Ce noyau va permettre aux utilisateurs de manipuler de

nouveaux objets et notamment les grilles polyédrales. Déjà plusieurs demandes de clients vont dans cette direction.

Bibliographie

[ABD+95] F. Avnaim, J.-D. Boissonnat, O. Devillers, F. Preparata, and M. Yvinec. Evaluation of a new method to compute signs of determinants. In *Communication at the 11th Annual ACM Symposium on Computational Geometry*, pages C16–C17, 1995.

[ADS98] P. Alliez, O. Devillers, and J. Snoeyink. Removing degeneracies by perturbing the problem or the world. In *Proc. 10th Canad. Conf. Comput. Geom*, 1998.

[AFF85a] S. Ansaldi, L. De Floriani, and B. Falcidieno. An Edge-Face Relational Scheme for Boundary Representations. *Computer Graphics Forum*, 4:319–332, 1985.

[AFF85b] S. Ansaldi, L. De Floriani, and B. Falcidieno. Geometric Modeling of Solid Objects by Using a Face Adjacency Graph Representation. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, volume 19 of *3*, pages 131–139. ACM, July 1985.

[Ago76] M. Agoston. *Algebraic Topology : A First Course - Pure and Applied Mathematics*. Marcel Dekker, 1976.

[AK89] D. Arquès and P. Koch. Pavages tridimensionnels. In *Bigre*, volume 61-62, pages 5–15, Avril 1989.

[Arb90] F. Arbad. Set models and boolean operations for solids and assemblies. *IEEE Computer Graphics and Applications*, pages 76–89, Novembre 1990.

[Bar81] B.A. Barsky. *The β-Spline : A Local Representation Based on Shape Parameters and Fundamental Geoemtric Measures*. PhD thesis, University of Utah, 1981.

[Bar88] B.A. Barsky. *Computer Graphics and Geometric Modeling using β-Splines*. Springer Verlag, 1988.

[Bau75] B. Baumgart. A Polyhedron Representation for Computer Vision. In *AFIPS Nat. Conf. Proc.*, volume 44, pages 589–596, June 1975.

[BBB87] R.H. Bartels, J.C. Beatty, and B.A. Barsky. *An introduction to Splines for use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1987.

[BBT97] D. Bechmann, Y. Bertrand, and S. Thery. N-dimensional gregory-bézier for multidimensional cellular complexes. In *Proceedings of Mathematical Methods for Curves and Surfaces IV*, 1997.

[BDFL93] Y. Bertrand, J.F. Dufourd, J. Françon, and P. Lienhardt. Algebraic Specification and Development in Geometric Modeling. In *Proc. of TAPSOFT'93*, April 1993.

[Ber92] Y. Bertrand. *Spécification Algébrique et Réalisation d'un Modeleur Interactif d'Objets Géométriques Volumiques*. PhD thesis, Université Louis Pasteur de Strasbourg - Département d'Informatique, 1992.

[Ber97] Y. Bertrand. Modeleurs géométriques à base topologique: spécification, performances et applications. Université Louis-Pasteur de Strasbourg, Décembre 1997. Habilitation à diriger des recherches, spécilaité informatique.

[Bez70] P. Bezier. *Emploi des machines à commande numérique*. masson, 1970.

[Bez86] P. Bezier. *Courbes et Surfaces*. Hermès, 1986.

[BFH95] H. Bendels, D.W. Fellner, and S. Havemann. *Erweiterbare datenstrukturen zur modellierung und visualisierung polygonaler welten*, pages 149–157. D.W. Fellner, November 1995.

[BH88] D. Badouel and G. Hegron. Opérations booléennes sur les polyèdres : évaluation d'arbres CSG. Technical report, INRIA-Lorraine, Avril 1988.

[BO79] J.L. Bentley and T.A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28:643–647, 1979.

[Boi88] J.D. Boissonnat. Shape reconstruction from planar cross sections. In *Computer Vision, Graphic and Image Processing*, volume 144, pages 1–29, 1988.

[Bor91] P. Borianne. *Conception d'un Modeleur de Subdivisions de Surfaces Orientables ou non Orientables, avec ou sans bords*. PhD thesis, Université Louis Pasteur de Strasbourg, 1991.

[BP92] L. Bardis and N.M. Patrikalakis. Topological structures for generalized boundary representations. Geometric Algorithms in CS368, 1992. Handout 21.

[Bri89] E. Brisson. Representing Geometric Structures in D Dimensions: Topology and Order. In *Symposium on Computational Geometry*, pages 218–227. ACM, June 1989.

[Bri90] E. Brisson. *Representation of d-dimensional geometric objects*. PhD thesis, Department of Computer Science, University of Washington, August 1990.

[BY95] J.D. Boissonnat and M. Yvinec. *Géométrie Algorithmique*. Ediscience international, 1995.

[BY97] H. Brönnimann and M. Yvinec. Efficient exact evaluation of signs of determinants. In *Proc. 13th ACM Symposium on Computational Geometry*, 1997.

[Caz97] D. Cazier. *Construction de systèmes de réécriture pour les opérations booléennes en modélisation géométrique*. PhD thesis, Université Louis Pasteur de Strasbourg - Département d'Informatique, 1997.

[CC78] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, 1978.

[CE92] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *ACM*, 39:1–54, 1992.

[CG91] G. Celniker and D. Gossard. Deformable Curve and Surface Finite Elements for Free-Form Shape Design. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 257–265. ACM, 1991.

[Chi88] H. Chiyokura. *Solid Modeling with DESIGNBASE, Theory and Implementation*. Addison Wesley, 1988.

[Cla81] J.H. Clark. Parametric curves, surfaces and volumes in computer graphics and computer-aided geometric design. Computer Systems Laboratory 221, Departments of Electrical Engineering and Computer Science, Stanford University, 1981.

[Con97] J. Conraud. *Génération de maillages de simplexes pour la modélisation d'objets naturels*. PhD thesis, INPL, 1997.

[Cor75] R. Cori. *Un code pour les graphes planaires et ses applications*, volume 27 of *Soc. Math. de France*. Astérisque, Paris, 1975.

[CP89] C. Cattani and A. Paoluzzi. Solid modeling in any dimensions. Technical report, Genoa, 1989.

[dB78] C. de Boor. *A Practical Guide to Splines*. Springer, 1978.

[dB87] C. de Boor. *B-Form basics*, pages 131–148. SIAM, 1987.

[dC59] P.F. de Casteljau. Outillage méthodes calculs. Technical report, André Citroën Automobiles SA, 1959.

[DGS89] J.F. Dufourd, C. Gross, and J.C. Spehner. A digitisation algorithm for the entry of planar maps. In *Proc. Computer Graphics International'89*, pages 649–662, 1989.

[DL87] D.P. Dobkin and M.J. Laszlo. Primitives for the manipulations of three-dimensional subdivisions. In *Proc. of the* 3_{rd} *Symposium of Computational Geometry*, June 1987.

[DP99] O. Devillers and F.P. Preparata. Further results on arithmetic filters for geometric predicates. *Computational Geometry : Theory and Applications*, 13:141–148, 1999.

[DS78] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10(6):356–360, 1978.

[Du88] W.H. Du. *Etude sur la représentation de surfaces complexes: Application à la reconstruction de surfaces échantillonées*. PhD thesis, Ecole Nationale Supérieure des Télécommunications, 1988.

[Ede87] H. EdelsBrunner. *Algorithms in Combinatorial Geometry*, chapter 10. Springer-Verlag, 1987.

[EL92] H. Elter and P. Lienhardt. Extensions of the Notion of Map for the Representation of the Topology of Cellular Complexes. In *Proc. of* 4^{th} *Canad. Conference on Computational Geometry*, 1992.

[Elt90] H. Elter. Etude d'un modeleur de subdivisions d'espaces 3d basé sur les 3-g-cartes. Technical report, Université Louis Pasteur, 1990.

[Elt94] H. Elter. *Etude de structures combinatoires pour la représentation de complexes cellulaires*. PhD thesis, Université Louis Pasteur de Strasbourg, 1994.

[EPW90] H. Edelsbrunner, F.P. Preparata, and D.B. West. Tetrahedralizing point sets in three dimensions. *J. Symbolic Comput.*, 10(3-4):335–347, 1990.

[Far86] G. Farin. Triangular Bernstein Bézier Patches. *Computer Aided Geometric Design*, 3:83–130, 1986.

[Far87] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*, pages 235–245. SIAM, 1987.

[Far90] G. Farin. Surfaces over Dirichlet tesselations. *Computer Aided Geometric Design*, 7:281–292, 1990.

[Far92] G. Farin. *Curves and Surfaces for Computer AAided Geometric DDesign (3rd Edition)*. Academic Press, 1992.

[FF88] L. De Floriani and B. Falcidieno. A Hierarchical Boundary Model for Solid Object Representation. *ACM-Transactions on Graphics*, 7(1):42–60, January 1988.

[FG92] L. Fang and D.C. Gossard. Reconstruction of smooth parametric surfaces from unorganized data points. In J. Warren, editor, *Curves and Surfaces in Computer Vision and Graphics 3*, volume 1830, pages 226–236, 1992. SPIE.

[FG99] Pascal Jean Frey and Paul-Louis George. *Maillages : applications aux éléments finis*. Hermes, 1999.

[Fio95] C. Fiorio. *Approche interpixel en analyse d'images, une topologie et des algorithmes de segmentation*. PhD thesis, Université Montpellier II, 1995.

[FP91] V. Ferrucci and A. Paoluzzi. Extrusion and boundary evaluation for multidimensional polyhedra. *Computer-Aided Design*, 23:40–50, February 1991.

[FP95] L. De Floriani and E. Puppo. Hierarchical Triangulation for Multiresolution Surface Description. *ACM-Transactions on Graphics*, 14(4):363–411, October 1995.

[FR80] A. Foucault and J.F. Raoult. *Dictionnaire de géologie, (4ème édition)*. Masson, 1980.

[Fuc97] L. Fuchs. *Une Spécification Formelle des Modèles de Courbes et de Surfaces de Forme Libre*. PhD thesis, Université Louis Pasteur /UPRES-A CNRS 7005, November 1997.

[FW92] T.R. Fisher and R.Q. Wales. Three dimensional solid modelling in geo-objects using non-uniform rational b-splines (nurbs). In A.K. Turner, editor, *Three dimensional modeling with geoscientific information systems*, volume 354 of *C : Math. and Phys. Sciences*, pages 85–105, 1992.

[GGJ+97] S. Garrett, S. Griesbach, D. Johnson, D. Jones, M. Lo, W. Orr, and C. Sword. Earth model synthesis. *First Break*, 15(1):13–20, 1997.

[GHPT89] M. Gangnet, J.C. Hervé, T. Pudet, and J.M. Van Thong. Incremental Computation of Planar Maps. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, volume 23 of *3*, pages 345–354, July 1989.

[GLH93] J.A. Gregory, V.K.H. Lau, and J.M. Hahn. *High order continuous polynomial patches*, pages 117–132. Springer Verlag, 1993.

[Gre80] J.A. Gregory. Triangular Interpolation Patch for CAGD. *Computer Graphics and Image Processing*, 13:80–87, 1980.

[GS85] L. Guibas and J. Stolfi. Primitives for the Manipulation of General Subdivisions and the Computation of Voronoï Diagrams. *ACM-Transactions on Graphics*, 4(2):74–123, 1985.

[HC93] H.O. Hansen and N.J. Christensen. A model for n-dimensional boundary topology. In 2_{nd} *ACM Solid Modeling*, pages 65–73, 1993.

[HHK89] C.M. Hoffmann, J.E. Hopcroft, and M.S. Karasick. Robust set operations on polyhedral solids. *IEEE Computer Graphics and Applications*, 9(6):50–59, 1989.

[Hop96] H. Hoppe. Progressive Meshes. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 99–108. ACM, August 1996.

[Jac70] A. Jacques. Constellations et graphes topologiques. *Combinatorial Theory and Applications*, pages 657–673, 1970.

[Ket98] L. Kettner. Designing a data structure for polyhedral surfaces. In *Proc 14th Annu. ACM Symposium on Computational Geometry*, pages 146–154, 1998.

[KKM95] Y. Kallinderis, A. Khawaja, and H. McMorris. Hybri prismatic/tetrahedral grid generation for complex geometries. 1995. AIAA, paper 95-0211.

[KMK95] A. Khawaja, H. McMorris, and Y. Kallinderis. Hybrid grids for viscous flows around complex 3d geometries including multiple bodies. In *Proc. 12th AIAA CFD Conf.*, 1995. AIAA, paper-1685.

[Lan96] V. Lang. *Une étude de l'utilisation des ensembles simpliciaux en modélisation géométrique interactive.* PhD thesis, Université Louis Pasteur - Strasbourg, 1996.

[Lav86] M. Lavergne. *Méthodes sismiques.* Editions Technip, 1986.

[LE92] P. Lienhardt and H. Elter. Chains of maps : a combinatorial model for the representation of cellular complexes defined as structured simplicial complexes. Technical report, Centre de Recherche en Informatique, Université Louis Pasteur, September 1992.

[Lev99] B. Levy. *Topologie Algorithmique, Combinatoire et Plongement*. PhD thesis, INPL, 1999.

[Lie88] P. Lienhardt. Extension of the notion of map and subdivisions of a three-dimensional space. In *Proc. of* 5^th *Symposium on Theoritical Aspects in Computer Science*, february 1988.

[Lie89a] P. Lienhardt. Subdivisions of N-Dimensional Spaces and N-Dimensional Generalized Maps. In *Proc. Annual A.C.M. Symposium on Computational Geometry, Saarbrucken, R.F.A.*, pages 228–236, 1989.

[Lie89b] P. Lienhardt. Subdivisions of Surfaces and Generalized Maps. In *Proc. Eurographics'89, Hamburg, Germany*, pages 439–452, September 1989.

[Lie91a] P. Lienhardt. Subdivisions de surfaces et cartes généralisées de dimension 2. *Informatique théorique et applications*, 25(2):171–202, 1991.

[Lie91b] P. Lienhardt. Topological models for boundary representation : a comparison with n-dimensional generalized maps. *C.A.D.*, 23(1):59–82, 1991.

[Lie92] P. Lienhardt. Extensions de la notion de carte et modélisation géométrique à base topologique. Habilitation à diriger des recherches, Université Louis Pasteur, Département Informatique, 1992.

[Lie94] P. Lienhardt. N-Dimensional Generalized Combinatorial Maps and Cellular Quasi-Manifolds. *Journal on Computational Geometry and Applications*, 4(3):275–324, 1994.

[LL95] V. Lang and P. Lienhardt. Geometric modeling with simplicial sets. Technical Report 95/06, Université Louis Pasteur, LSIIT - URA CNRS 356, 1995.

[Mal88] J.L. Mallet. Three dimensional graphic display of disconnected bodies. In *Mathematical geology*, volume 122 of *8*, pages 977–990, 1988.

[Mal92] J.L. Mallet. Discrete Smooth Interpolation. *Computer Aided Design Journal*, 24(4):263–270, 1992.

[Mal01] J.L. Mallet. *Geomodeling*. (sous presse), 2001.

[Män88] N. Mäntylä. *An Introduction to Solid Modeling*. Computer Science Press, 1988.

[May67] J.P. May. *Simplicial objects in algebraic topology*. PhD thesis, Van Nostrand, Princeton, 1967.

[Meh84] K. Mehlhorn. *Data Structures and Algorithms 3 : Multi-dimensional Searching and Computational Geometry*, chapter 3. Springer-Verlag, 1984.

[MFC97] J-P. Michel, R.W. Fairbridge, and M.S.N. Carpenter. *Dictionnaire des sciences de la terre*. Dunod, 3 edition, 1997.

[Mil89] J.R. Miller. Architectural issues in solid modelers. *IEEE Computer Graphics and Applications*, Septembre 1989.

[Mor85] M.E. Mortenson. *Geometric modelling*. John Wileyand and Sons, 1985. New York, 763 pages.

[NM82] R. Sulonen N. Mäntylä. Gwb : A solid modeler with euler operators. *IEEE Computer Graphics and Applications*, 2:17–31, September 1982.

[PBCF93] A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-Independant Modeling with Simplicial Complexes. *ACM Transactions on Graphics*, 12(1):56–102, January 1993.

[Pet85] J.P. Petit. *Le Géometricon - Les Aventures d'Anselme Lanturlu (bande dessinée)*. Belin, 1985.

[Pio99] Sylvain Pion. *De la géométrie algorithmique au calcul géométrique*. Thèse de doctorat en sciences, Université de Nice-Sophia Antipolis, France, 1999.

[PS85] F.P. Preparata and M.I. Shamos. *Computational Geometry : An Introduction*. Springer-Verlag, 1985.

[PS86] L.K. Putnam and P.A. Subrahmanyam. Boolean operations on n-dimensional objects. *IEEE Computer Graphics and Applications*, 6(6):43–51, 1986.

[RH87] J.G. Ramsay and M.I. Huber. *Modern Structural Geology*. Academic Press, 1987.

[RO89] J.R. Rossignac and M.A. O'Connor. Sgc : A dimensional-independent model for pointsets with internal structures and incomplete boundaries. Technical report, IBM, Research Division, T.J. Watson Research Center, Yorktown Heights, N.Y. 10598, January 1989.

[Ros93] J.R. Rossignac. Representations, design and visualization of solids and of geometry structures. In *Proc. Eurographics'93, Tutorial, Barcelona, Spain*, Septembre 1993.

[RR99] J. Rossignac and A. Requicha. Solid modeling. *Encyclopedia of Electrical and Electronics Engineering*, 1999.

[RV80] A.A.G. Requicha and H.B. Voelcker. Constructive solid geometry. In *Production Automation Project Tech. Memo 25, Rochester University*, November 1980.

[RV82] A.A.G. Requicha and H.B. Voelcker. Solid modeling : A historical summary and contemporary assessment. *IEEE Computer Graphics and Applications*, March 1982.

[RV85] A.A.G. Requicha and H.B. Voelcker. Boolean operations in solid modeling : Boundary evaluation and merging algorithms. *IEEE*, 73(1):30–44, January 1985.

[Sam84] H. Samet. The quadtree and related hierarchical data structures. In *ACM Computing Surveys 16*, pages 187–260, 1984.

[SBD86] F.J.M Schmitt, B.A. Barsky, and Wen-Hui Du. An Adaptive Subdivision Method for Surface Fitting from Sampled Data. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, pages 179–188. ACM, 1986.

[Sch81] L.L. Schumaker. *Spline Function: Basic Theory*. Wiley, 1981.

[Sed88] B. Sedgewick. *Algorithms*. Addison-Wesley, 2nd edition, 1988.

[Seg98] D. Segonds. *Intégration de l'approche paramétrique dans le géomodeleur Gocad*. PhD thesis, INPL, 1998.

[Sei98] H. P. Seidel. 3D Geometry Compression with Splines. In *SIGGRAPH course notes: 3D Geometry Compression (course 21)*. ACM, July 1998.

[SG95] R.E. Sheriff and L.P. Geldart. *Exploration Seismology*. Cambridge University Press, 1995.

[She96] J.R. Shewchuk. Robust Adaptative Floating-Point Geometric Predicates. In *Symposium on Computational Geometry*, pages 141–150. ACM, 1996.

[SL99] X. Skapin and P. Lienhardt. Utilisatoin du produit cartésien pour l'animation. *12èmes journées AFIG 99*, 1(1):283–298, 1999.

[Spe91] J.C. Spehner. Merging in maps and pavings. *Theoretical Computer Science*, 2(86):205–232, 1991.

[Str97] B. Stroustrup. *The C++ Programming Language, 3rd. ed.* Addison Wesley, 1997.

[Tut84] W.T. Tutte. Graph theory. In *Encyclopedia of Mathematics and its Applications*, 1984.

[War69] J. Warnock. *A Hidden Surface Algorithm for Computer Generated Halftone Pictures*. PhD thesis, Computer Science Dept., University of Utah, TR 4-15, june 1969.

[WD96] P. Weimer and T.L. Davis. *Application of 3D seismic data to exploration and production*. P. Weimer and T.L. Davis eds., 1996.

[Wea88] N.P. Weatherill. *A strategy for the use of hybrid structured-unstructured meshes in CFD*. Oxford Univerity Press, 1988.

[Wei85] K. Weiler. Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments. *IEEE Computer Graphics and Applications*, 5(1):21–40, 1985.

[Wei86a] K. Weiler. The Radial Edge Structure: a Topological Representation for Non-Manifold Geometric Boundary Modeling. In *Proc. of the IFIG WG 5.2*, may 1986.

[Wei86b] K. Weiler. *Topological Structures for Geometric Modeling*. PhD thesis, Rensselear Polytechnic Institute, 1986.

[Wei88] K. Weiler. Boundary graph operators for non-manifold geometric modeling topology representations. In *Geometric Modeling for CAD applications*. Elsevier Science Publishers, 1988.

[Zor97] D. Zorin. *Stationary Subdivision and Multiresolution Surfaces Representations*. PhD thesis, Caltech, Pasadena, 1997.

[Zor98] D. Zorin. Subdivision Surfaces. In *SIGGRAPH course notes: Subdivision for Modeling and Animation (course 36)*, pages 45–81. ACM, July 1998.

[ZS98] D. Zorin and P. Schröder. Foundations I: Basic Ideas. In *SIGGRAPH course notes: Subdivision for Modeling and Animation (course 36)*, pages 17–44. ACM, July 1998.

[ZSS96] D. Zorin, P. Schröder, and W. Sweldens. Interpolating Subdivision for Meshes with Arbitrary Topology. In *Computer Graphics (SIGGRAPH Conf. Proc.)*, Annual Conference Series, pages 189–192. ACM Siggraph, August 1996.

[ZSS97] D. Zorin, P. Schröder, and W. Sweldens. Interactive Multiresolution Mesh Editing. In *Computer Graphics (SIGGRAPH Conf. Proc.)*. ACM, 1997.