Étude d'algorithmes graphiques dans le cadre de la simulation numérique

Robert Fournier, Wihem Kherrati

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Étude d'algorithmes graphiques dans le cadre de la simulation numérique.

Robert Fournier, Wihem Kherrati

N° 2502

mars 1995

PROGRAMME 6

ISSN 0249-6399

INRIA
SOPHIA ANTIPOLIS

Étude d'algorithmes graphiques dans le cadre de la simulation numérique.

Robert Fournier, Wihem Kherrati*

Programme 6 — Calcul scientifique, modélisation et logiciel numérique
Projet SINUS

Rapport de recherche n° 2502 — mars 1995 — 94 pages

Résumé : Une méthode utilisée fréquemment pour effectuer des simulations numériques, est de représenter le volume étudié par un maillage de surface (une peau) et un maillage volumique, aux nœuds desquels sont simulées des propriétés physiques (pression, densité, température, vitesse, ...).
Des outils graphiques interactifs sont nécessaires pour l'analyse des résultats.
Après un aperçu des techniques expérimentales, nous présenterons les méthodes de visualisation apliquées à la simulation numérique.
Nous distinguerons la représentation des données scalaires (propriétés physiques telles que densité ou pression), de celle des données vectorielles (le vecteur vitesse).
Les approches principales pour visualiser les valeurs scalaires définies aux nœuds d'un maillage sont: les plans de coupe, les isosurfaces ou le rendu volumique. Les deux premières techniques sont des représentations locales: seule une partie du volume est prise en compte (coupes) ou la répartition d'une seule valeur est representée (isosurfaces).
Par contre le rendu volumique permet de visualiser la répartition de toutes les valeurs dans le volume, pour une propriété donnée.
La représentation des données vectorielles passe par la construction de lignes et surfaces d'écoulement.
Une étude ainsi qu'une implémentation de ces différentes méthodes est proposée.

Mots-clé : Maillages 3D, Coupes, Isosurfaces, Rendu Volumique, Lignes et Surfaces d'écoulements.

(Abstract: pto)

*. MECALOG Sarl Les Algorithmes Rte des Lucioles BP152 06903 Sophia-Antipolis
Email : wihem@mecalog.fr

Unité de recherche INRIA Sophia-Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex (France)
Téléphone : (33) 93 65 77 77 – Télécopie : (33) 93 65 77 65

Visualization applied to CFD : a study.

Abstract: A method frequently used to compute numerical simulations, is to represent an object by surface (a skin) and volumic meshes. Some physical properties (like pressure, density, temperature, velocity, ...) are computed on these mesh nodes. There are two kinds of meshes :
◇ The structured grids represented as a matrix of points, can be distinguished to :
◇ The unstructured grids, which don't have any regularity. The connectivity of these elements is arbitrary.

Interactive graphics tools are needed to analyze the results.
After a study of experimental methods, we present visualization methods applied to numerical simulation.
We distinguish the representation of scalar data (physical properties such as density or pressure) and vectorial data (velocity field).
The principal approaches to represent scalar data defined on the mesh nodes are : planar cuts, isosurfaces or volume rendering. The first two ones are local visualization methods: just a part of the volume is considered (cut), or the distribution of a single value is represented (isosurface) ; whereas the volume rendering allows to visualize the distribution of all the values in the volume for a given property.
The construction of lines and flow surfaces allow to visualize vectorial data.

A study and implementation of all these kind of visualization methods is proposed.

Key-words: 3D mesh, Cut Planes, Isosurfaces, Volume Renderind, Flow lines and surfaces.

Table des matières

Table des figures

Introduction

La mécanique des fluides est un domaine d'études fondamentales qui permet de faire évoluer la compréhension des phénomènes liés à l'écoulement des fluides. Ce domaine a de nombreuses applications industrielles.
Les écoulements gazeux sont étudiés dans l'industrie automobile ou aéronautique : pour la construction de voitures, d'avions ou d'engins spatiaux, ou plus spécifiquement pour la conception de turbines et de moteurs à explosion.
Les écoulements liquides ont une application directe dans l'industrie navale, pour la conception de bateaux, mais également dans le génie civil avec la construction de ports et la protection des côtes.
Dans l'industrie chimique, la connaissance des déplacements des fluides dans les réacteurs joue un rôle important. En médecine, l'écoulement du sang dans les prothèses cardiaques est également étudié. De nombreuses autres applications peuvent être citées, telles que la météorologie ou l'océanographie.

Dans les vingt dernières années, un domaine de la recherche appelé la CFD (Computational Fluid Dynamics) est apparu. La CFD est l'étude des écoulements fluides par des méthodes numériques. Ce domaine complète les techniques d'expérimentation physique afin de mieux comprendre les écoulements de fluides et leurs effets sur un solide. La visualisation des résultats tant expérimentaux que numériques apparaît nécessaire à la compréhension de ces phénomènes.
La visualisation de l'écoulement d'un fluide autour d'un objet n'est pas immédiate ; spécialement quand le fluide est transparent comme l'air ou l'eau et qu'il se déplace à grande vitesse. La visualisation des écoulements fournit une aide précieuse pour la compréhension des phénomènes de mécanique des fluides.

Après une brève étude des techniques expérimentales de visualisation (inspiration première de la visualisation assistée par ordinateur), nous présenterons les méthodes de représentations infographiques appliquées à la simulation numérique. Enfin nous détaillerons les méthodes implémentées dans le logiciel VIGIE (VIsualisation Générale Interactive d'Écoulements).

Chapitre 1

Techniques expérimentales

1.1 Dispositifs expérimentaux

L'étude de l'écoulement d'un fluide ne peut en général pas s'effectuer en grandeur réelle. Une maquette de l'objet, autour (ou à travers lequel) l'écoulement doit être étudié, est alors construite. Cet objet est placé dans une soufflerie ("wind tunnel") dans laquelle le mouvement du fluide est créé. Le comportement du fluide est étudié aux abords de la maquette. Par exemple pour étudier l'écoulement de l'air autour d'un avion, celui-ci est placé dans une soufflerie où des particules de fumée sont injectées.
Les modèles de soufflerie diffèrent selon les laboratoires d'expérimentation. Schématiquement une soufflerie est composée d'un conduit où circule le fluide étudié et d'une chambre de mesures où sont injectées les particules. Certaines souffleries peuvent être à boucle fermée et à fonctionnement continu.
Une des difficultés de ces techniques réside dans l'observation du mouvement de fluides souvent transparents.

De nombreux articles [PW93, Mer87, Yan89, SS91] présentent ces différentes techniques de visualisation.

1.2 La visualisation expérimentale

La visualisation des écoulements existe depuis aussi longtemps que la recherche sur les écoulements des fluides. Jusque récemmment, seule la visualisation expérimentale des écoulements était possible .
Cette technique expérimentale peut être utilisée dans le but de comparer un phénomène d'écoulement décrit par un modèle, à l'écoulement réel du fluide. La validité du modèle peut être vérifiée par le calcul et la visualisation de l'écoulement, puis comparés avec des résultats expérimentaux.

W. Merzkirch ([Mer87]) définit la visualisation des écoulements de la façon suivante : un phénomène de mécanique des fluides s'interprête en visualisant un fluide s'écoulant à travers un conduit ou autour d'un obstacle solide. En observant ce phénomène, qui peut être stationnaire ou variable au cours du temps, il est possible d'avoir une bonne idée de l'écoulement. Cependant la plupart des fluides gazeux ou liquides sont transparents et leur mouvement est invisible à l'observation directe. Pour pouvoir observer l'écoulement d'un fluide il est nécessaire de trouver des techniques pour le rendre visible. Dans le cadre de la visualisation expérimentale, trois méthodes peuvent être distinguées: la technique des traceurs (par ajout de corps étrangers), les techniques de densité (basée sur l'optique) et la combinaison des deux précédentes (par addition de chaleur et d'énergie).

1.2.1 Technique des traceurs

Cette méthode est réalisée par l'ajout de corps étrangers.
Les courbes représentant des déformations dans le temps ("time lines"), le courant ("streak lines") et la trajectoire des particules ("path lines") jouent un role important dans la visualisation expérimentale.
Les premières sont des lignes qui une fois lachées dans le fluide (en général perpendiculairement à l'écoulement) subissent les transformations et mouvements de l'écoulement. Le mouvement et la courbure de la ligne traduisent l'écoulement. En pratique ces lignes consistent en un lâcher de petites particules dans le fluide ; des bulles d'hydrogènes sont souvent utilisées.
Les "streak lines" se produisent quand une matière colorante telle que de l'encre

est injectée dans le fluide à une position fixée. L'injection pendant une certaine période de temps produit un tracé coloré dans le fluide.
Une dernière méthode représente la trajectoire d'une particule dans le fluide ("path lines") . En imaginant des particules émettant de la lumière, leur trajectoire est obtenue quand une photographie est exposée pendant un temps donné.

Dans les écoulements stationnaires, les "streak lines" et les " path lines" sont confondues ; elles représentent les lignes de courant "stream lines" qui sont tangentes en tout point au vecteur vitesse.
Ces trois types de visualisation peuvent être représentés expérimentalement en injectant un matériau étranger dans le fluide qui est en général transparent.
Nous avons déjà vu que de l'encre pouvait être injectée pour représenter des "streak lines". Dans le cas d'écoulements gazeux, de la fumée peut la remplacer.
Les lignes de trajectoire ("path lines") sont obtenues également en injectant des particules dans le fluide. De la poudre de magnésium est utilisée pour les liquides, des gouttes d'huile pour les gaz.
Malheureusement ces techniques peuvent perturber l'écoulement ; par exemple si la densité et la température des matériaux ne sont pas les mêmes que celles du fluide au point d'injection.
Pour ne pas perturber le fluide, des techniques d'électrolyse et de photo-chimie sont également décrites pour visualiser les lignes de temps. La technique des bulles d'hydrogènes est basée sur l'électrolyse de l'eau. Quand les électrodes sont plongées dans le fluide avec un certain voltage, des bulles d'hydrogènes se forment à la cathode, de l' oxygène à l'anode. En isolant des parties d'électrodes et en faisant varier le voltage, toutes les combinaisons de "time lines" et streak lines" peuvent être obtenues.
Les techniques de photo-chimie appliquées au fluide ne le perturbent pas non plus : elles produisent un traceur visible (matière colorante) en focalisant de la lumière en un point du fluide ou en utilisant un laser pour produire une ligne colorée dans le fluide par réaction photo-chimique.

1.2.2 Techniques de densité

Les techniques de visualisation expérimentale utilisant des propriétés optiques sont basées sur le fait que le changement de densité provoque un changement d'indice de réfraction de la lumière. Ces techniques ne sont utilisées que pour un fluide de densité non constante.
Il faut distinguer trois techniques : l'ombre, la méthode "schlieren" et l'interférométrie. La technique la plus simple est la technique d'ombres : des graphes d'ombres sont produits en faisant passer des rayons lumineux parallèles à travers le fluide en mouvement. La variation de densité provoque la réfraction des rayons lumineux.
La technique de schlieren utilise également des rayons lumineux parallèles mais en rajoutant deux diaphragmes : un pour sélectionner les rayons passant à travers le fluide, le second pour arrêter certains rayons réfractés. Cette technique permet de visualiser les gradients de densité.
L'interférometrie est basée sur le fait que le changement de densité produit non seulement la réfraction de la lumière mais également un changement de phase. Une interférence est provoquée en réunissant les rayons lumineux issus directement de la source et ceux qui ont traversés le fluide. Ces derniers ont subi un décallage de phase révélant les variations de la densité du fluide traversé.

1.2.3 Addition de chaleur et énergie

La dernière technique est utilisée quand les deux précédentes ne le peuvent pas : par exemple pour des fluides de faible densité. De la chaleur ajoutée artificiellement modifie la densité du fluide, (à pression constante, la chaleur transmise à une partie du fluide fait baisser la densité). On se retrouve dans le cas où la technique précédente peut être appliquée.
D'autres méthodes basées sur la ionisation des particules du fluides sont également utilisées.

1.2.4 Bilan des techniques expérimentales

Toutes ces méthodes de visualisation expérimentale présentent cependant quelques inconvénients :

- ◇ Tous les phénomènes d'écoulement ne peuvent être visualisés par une méthode expérimentale.
- ◇ La construction de modèles physiques miniatures, ainsi que l'équipement expérimental coûtent cher.
- ◇ Ces techniques de visualisation souffrent des aléas de l'expérimentation.

Durant les vingt dernières années, la croissance de la puissance de calcul des ordinateurs a conduit à leur utilisation massive pour la simulation numérique. Dans le domaine des écoulements, les ordinateurs sont utilisés pour simuler des quantités physiques. Ceci a contribué à l'émergence d'un nouveau domaine de recherche et d'applications : la CFD (Computational Fluid Dynamics). L'ordinateur a été utilisé pour simuler les écoulements des fluides d'après des modèles physiques. Il est aussi employé pour visualiser les résultats de ces simulations. Cette visualisation assistée par ordinateur s'inspire largement des techniques expérimentales.

Chapitre 2

La visualisation appliquée à la CFD

Définissons tout d'abord le cadre d'études de la simulation numérique d'écoulements, afin de mieux cerner les problèmes liés à la visualisation dans l'espace tridimensionnel.

2.1 Définition du contexte

Tout l'espace ne pouvant être utilisé pour représenter une solution continue (quantité physique), les simulations numériques d'écoulement reposent sur une discrétisation de l'espace. Ceci consiste à ne calculer la solution qu'en un nombre fini de points appelés nœuds, répartis dans l'espace.
Soit $f(x, y, z)$ une fonction définie sur une région Ω de $\mathbb{R}^3$. Dans de nombreuses applications scientifiques (comme par exemple le domaine de l'imagerie médicale), Ω est une région simple (un parallépipède rectangle); dans d'autres applications comme les techniques d'éléments finis, Ω est plus complexe (chambre de combustion d'un moteur diesel.)
Dans les deux cas, la fonction f est l'approximation d'une fonction que l'on cherche à étudier (densité en eau des tissus ou température de combustion). Cette fonction est définie comme un ensemble fini de points de $\mathbb{R}^3$ associés à une fonction d'interpolation remplissant les zones entre les points. Ces points (ou nœuds) sont répartis à l'intérieur du domaine sous forme d'un maillage

volumique.
Les données sont organisées en maillages qui définissent une certaine structure dans les localisations de l'espace et du temps où les équations différentielles ont été résolues numériquement. Ces positions sont visualisées après connection par des segments de droites qui donnent l'apparence du maillage.
Le résultat d'une simulation numérique est constitué d'un ensemble de données contenant plusieurs types d'informations :

- ◇ une information géométrique définissant la localisation des nœuds dans le maillage.
- ◇ une information topologique définissant la connection des nœuds dans le maillage.
- ◇ des quantités scalaires ou vectorielles représentant des quantités physiques.

Les maillages sont classés selon des critères de topologie, c'est-à-dire selon le type de connection entre les nœuds pour former des éléments volumiques. Il faut distinguer principalement les maillages structurés des maillages non structurés.
Les maillages structurés sont formés d'hexaèdres répartis sous forme d'une grille. La topologie est définie implicitement par la numérotation (i, j, k) des nœuds.
Dans les maillages non structurés, au contraire, la topologie est arbitraire et doit être explicite. L'inconvénient principal est le manque d'informations sur les propriétés d'adjacence d'une cellule ; contrairement aux grilles structurées, il n'existe pas de relation entre la numérotation des nœuds et leur position physique dans le maillage (cf figure 2.1).
Les applications dans le cadre de la simulation numérique d'écoulement produisant un nombre important de données, il est commun d'utiliser un maillage contenant plusieurs milliers de cellules. De plus ces données posent un problème de visualisation, les informations intéressantes sont situées à l'intérieur du volume et pas seulement à la surface. Tout le problème consiste alors à transformer les données représentant l'écoulement en affichage de primitives graphiques usuelles.
De nouvelles techniques ont vu le jour pour la génération d'images représentant au mieux l'écoulement.

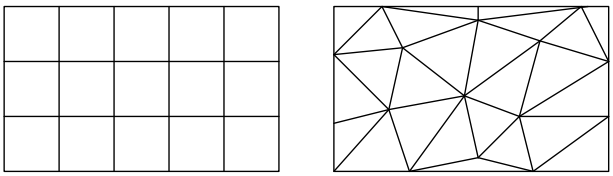


FIG. 2.1 - *Maillages 2D structuré / non structuré couvrant les mêmes domaines.*

Donnons un aperçu des principales techniques permettant de visualiser les données contenues dans un maillage tridimensionnel, issues de simulation numérique d'écoulement ([Pag93, Nee90, HGD91, DH91, HD91, Be87]).

2.2 Visualisation assistée par ordinateur.

Nous avons vu que les données représentées sur un maillage pouvaient être de deux types : scalaires (telles que la densité, la pression ou la température) ou vectorielles (comme la vitesse).
La visualisation des données scalaires a été étudiée en détail ces dernières années. Des techniques telles que le positionnement interactif de plans de coupe, la construction d'isosurfaces ou le rendu volumique sont répandues dans la littérature.
D'autres méthodes permettant de visualiser les quantités vectorielles dans les écoulements ont été moins abordées. Il s'agit de l'étude du tracé de particules, des lignes et surfaces d'écoulement ou des vecteurs.
Après une brève description de ces diverses techniques de représentation, nous nous attarderons plus longuement sur l'implémentation de certaines de ces méthodes.

2.2.1 Données scalaires

Une première technique utilise des contours : lignes (2D) ou surfaces (3D) sont tels que la variable scalaire (quantité physique) a une valeur constante sur cette ligne ou surface. Ces iso-lignes ou iso-surfaces permettent de visualiser

localement un champ scalaire dans le maillage.

Les isolignes peuvent être calculées dans une coupe du maillage volumique. Le plan de coupe est une surface plane qui coupe le domaine de calcul. Les valeurs scalaires connues aux nœuds des cellules intersectées, sont interpolées et affichées sur le plan.
Pour faciliter la visualisation interne du maillage, le choix du plan de coupe doit être interactif et doit pouvoir effectuer toutes les coupes voulues, dans toutes les directions.

Les isosurfaces sont une technique intéressante pour illustrer la structure interne des champs scalaires. Cette technique consiste à construire la surface formée par tous les points de même valeur.

Ces deux techniques sont un mode de représentation locale ; seule une partie du volume est prise en compte (coupe) ou la répartition d'une seule valeur est visualisée (isosurfaces).

Le rendu volumique est par contre un mode de représentation globale. Cette méthode permet de visualiser toutes les valeurs présentes dans le volume pour une propriété donnée. Le volume est considéré comme transparent et chaque point qui le compose émet de la lumière dont l'intensité est liée à la valeur de la propriété étudiée.

Ces trois techniques permettent de représenter des champs scalaires. Elles n'ont pas d'équivalent en visualisation expérimentale et ont été développés principalement dans le but de visualiser les valeurs scalaires présentes à l'intérieur du volume étudié. Ces trois méthodes seront détaillées par la suite.
Décrivons les principales techniques utilisées pour représenter un champ vectoriel.

2.2.2 Données vectorielles

La visualisation des données vectorielles s'inspire directement des techniques expérimentales.

Les vecteurs peuvent être directement représentés par un segment de droite terminé par une flêche donnant le sens de l'écoulement. La longueur du segment peut correspondre à la norme du vecteur. Ce type de représentation efficace en 2D pose des problèmes de perception en 3D (cf figure 2.2, schéma 1 et 3).
Il est par exemple impossible de distinguer si la flêche pointe ou non vers

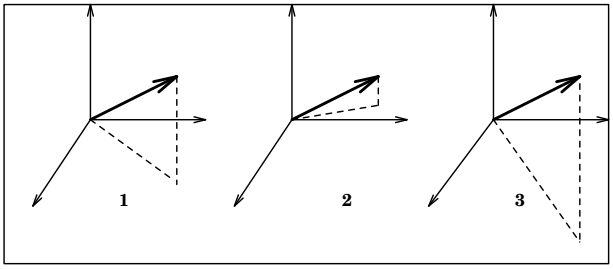


FIG. 2.2 - *Ambiguité de perception d'un vecteur en 3D.*

l'observateur (cf figure 2.2, schéma 1 et 2). De plus une certaine longueur du segment peut être due à la valeur vectorielle mais également à l'orientation du vecteur par rapport à l'observateur.
L'affichage de vecteurs comme des objets polygonaux améliore le rendu ; l'occlusion d'une surface visible améliore la perception visuelle et réduit l'ambiguité sur la direction.
Les vecteurs peuvent être alors représentés par des cônes de taille variable selon la valeur du champ et dirigés dans le sens de l'écoulement.
La représentation peut cependant être difficile à interprêter du fait de l'encombrement dû au nombre important de vecteurs.

Les lignes et surfaces d'écoulement nécessitent pour leur définition un ou plusieurs (dans le cas des surfaces) points de départ (x, y, z).
Dans le cas qui nous intéresse nous nous plaçons dans un mode stationnaire :

définissons alors les lignes de courant ("stream lines") (cf chapitre 7), tangentes en tout point au vecteur vitesse. Une ligne de courant correspond alors à la trajectoire d'une particule lachée dans le fluide à une position donnée.
Comme il est difficile de percevoir une ligne courbe en 3D, des surfaces de courant peuvent être générés à partir de lignes de courant afin de mieux saisir la direction de l'écoulement ainsi que des effets tels que divergence et courbure.

Les particules sont rendues comme des petits objets ou des points. Il faudrait éviter cependant de les représenter par des sphères en 3D. Ce type d'objet ne changeant pas d'aspect après rotation, il est difficile alors d'avoir les positions et orientations dans l'espace. À l'inverse il faut aussi éviter de les représenter par des points pour ne pas rencontrer un problème d'aliasing causé par la limite de résolution des écrans.

Les chapitres suivants sont consacrés à une étude plus approfondie des méthodes de représentation des données scalaires et vectorielles. Ces méthodes ont été implémentées dans VIGIE, logiciel de visualisation, dont le but est plus la compréhension des écoulements des fluides que la génération d'images de rendu réaliste. Les algorithmes sont spécialement adaptés au cas des maillages non structurés 3D (avec tous les problèmes liés à la connectivité que cela implique) mais ils fonctionnent également pour des maillages structurés.
Commençons par étudier la représentation des données scalaires.

Chapitre 3

Interactivité

Tout calcul sur un jeu de données numériques destiné à la visualisation nécessite des moyens d'interaction efficaces pour permettre de positionner et de visualiser un objet en trois dimensions.
Un façon agréable de le positionner consiste à lui appliquer des rotations associées au déplacement de la souris considérée comme plaquée sur une "boule virtuelle".
Un façon efficace de le visualiser consiste à construire des intersections avec le maillage de peau par des plans parallèles qui permettent par un tracé rapide, un compréhension grossière du positionnement de l'objet dans l'espace. Nous appelerons "squelette" un tel jeu d'intersections.

3.1 Interactions avec l'objet

La position de l'objet est modifiée interactivement à la souris. Il subit une rotation autour du centre du domaine de visualisation. L'axe et l'angle de cette rotation sont définis par les interactions de l'utilisateur.
Un déplacement de la souris définit deux points (départ **D** et arrivée **A**) à l'écran. On calcule alors leur projection sur une boule virtuelle centrée sur le domaine de visualisation, soient **D' et A'**. Ces deux nouveaux points, associés au centre **C** de la sphère définissent un plan. L'axe de rotation $\mathbf{\Delta}$ choisi passe par **C**, orthogonalement à ce plan. L'angle de rotation α est naturellement l'angle (**CD', CA'**). (*cf figure 3.1*).

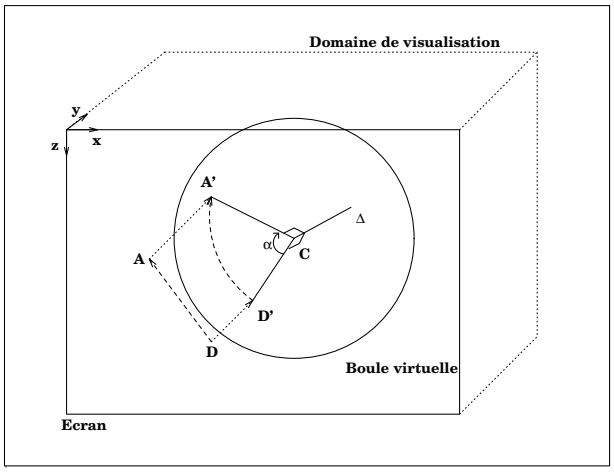


FIG. 3.1 - *Schéma d'explication de la boule virtuelle.*

Le but du logiciel VIGIE est d'effectuer de la visualisation plus ou moins "interactive", sur tous types de stations de travail, et notamment celles ne possédant pas d'accélérateur graphique. Or un maillage volumique est représenté à l'écran par des milliers de facettes (qui constituent l'enveloppe surfacique du volume, soit la peau). Nous avons trouvé une méthode simple qui permet de positionner un objet 3D dans l'espace de façon interactive.

3.2 Squelette

3.2.1 Définition

Pour positionner interactivement un objet dans l'espace, il faut trouver un mode de visualisation rapide. La représentation de l'objet par son squelette est suffisamment simple pour permettre une fréquence d'affichage compatible avec la vitesse de déplacement de la souris.
Le squelette est constitué de segments représentant l'intersection entre des plans de coupe parallèles et le maillage de peau de l'objet (*cf figure 3.2*).
Pour l'affichage d'une vingtaine de plans il suffit de quelques centaines de segments. C'est donc beaucoup plus rapide que l'affichage des milliers de facettes qui constituent le maillage de peau. Cela permet de faire bouger l'objet de

manière interactive, sur des postes de travail ne possédant pas d'accélérateur graphique.

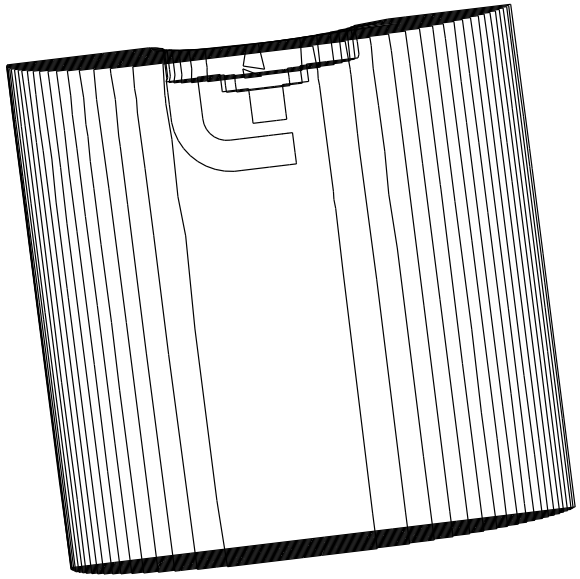


FIG. 3.2 - *Squelette d'une bougie, défini par des plans ZX.*

L'intersection entre plans de coupe parallèles et maillage de peau peut être optimisée, afin d'accélérer la construction du squelette.

3.2.2 Optimisation du calcul d'intersection d'un plan avec la peau

L'optimisation du calcul d'intersection d'un plan avec un ensemble de facettes est basée sur un découpage régulier de l'espace en cubes.
Le cube englobant l'objet est subdivisé en cubes réguliers. A chaque cube est associé la liste des facettes qu'il contient. (Ces listes sont construites une seule fois à la lecture des données).
Avant de calculer l'intersection d'un plan avec l'ensemble des facettes, il faut déterminer quels sont les cubes coupés par ce plan.

Le nombre de cubes coupés est de l'ordre de n^2; n étant le facteur de subdivisions employé (le nombre de mini-cubes le long d'une arête). Le nombre total de cubes est n^3.
Avec m facettes régulièrement réparties on teste donc seulement: $\frac{m*n^2}{n^3} = \frac{m}{n}$ facettes.
Dans la pratique cette valeur varie en fonction de l'objet et du plan choisi, mais elle reste nettement inférieure à m .
Une facette traversant la frontière entre deux cubes, appartient aux deux cubes. Elle est donc représentée dans deux listes. Si le plan traverse ces deux cubes, l'intersection avec cette facette sera testée deux fois. Pour éviter cela, il suffit d'associer un numéro à chaque nouveau plan, de stocker pour chaque facette le numéro indiquant le dernier plan pour lequel elle a été testée, puis de comparer le numéro du plan courant à celui stocké pour une facette.
Finalement ces optimisations ont permis de rendre le calcul de l'intersection suffisament rapide pour être interactif. Il en découle le mode de visualisation du squelette et la possibilité de placer rapidement les plans de coupe dans le maillage.

L'étape suivante après le positionnement de l'objet dans l'espace, puis le choix d'un plan de coupe dans le volume, consiste à effectuer l'intersection entre le plan de coupe et les polyèdres du maillage. Il s'agit enfin d'effectuer des interpolations linéaires, à partir des valeurs connues aux nœuds du maillage, afin de connaître les valeurs aux points d'intersection le long des arêtes.

Chapitre 4

Plans de coupe

Il s'agit d'observer les valeurs placées à l'intersection d'un plan de coupe et du maillage pour une propriété physique donnée.
La méthode qui consiste à tester si toutes les cellules sont coupées par le plan est coûteuse, elle est surtout utilisée en instationnaire quand les données peuvent changer d'un instant à l'autre. Strid et Rizzi [SR89] évitent de faire une recherche complète. Ils testent toutes les cellules de la frontière qui sont coupées, puis se déplacent ensuite de cellules en cellules grâce à une table d'adjacence (qui définit les relations de voisinage entre les cellules).
Le calcul d'une table d'adjacence étant coûteux (décrit au chapitre 7), nous avons préféré effectuer le calcul exhaustif, en évitant cependant de dupliquer les calculs pour les arêtes communes à plusieurs éléments. L'intersection est testée pour chaque volume, avec toutes ses arêtes. Une table d'adressage calculé[1] indexée sur les arêtes, permet de tester si l'intersection n'a pas déjà été calculée.
La méthode avec subdivision de l'espace, appliquée dans le cas de l'intersection avec la peau du maillage, peut être réutilisée ici. Toutefois la structure nécessaire à sa réalisation est coûteuse en mémoire. Des contraintes de temps ne nous ont pas permis de tester si le gain d'interactivité justifiait ce surcoût en consommation mémoire.

Après avoir récupéré les intersections entre plan de coupe et maillage, puis

1. hash table.

effectué une interpolation linéaire pour calculer les valeurs, plusieurs applications peuvent être envisagées. Il est possible de générer le maillage du plan de coupe en créant des facettes au fur et à mesure des intersections ; le maillage, défini dans le repère écran, est ensuite projeté dans le repère du plan, afin d'obtenir un maillage 2D avec des valeurs connues aux nœuds des polygones le constituant.
Une fois déterminée la facette issue de l'intersection du plan de coupe avec un élément de volume, il est possible de visualiser les valeurs qui y sont localisées. Il est par exemple facile d'afficher la facette avec un dégradé de couleurs qui indique la variation de valeur sur la facette ou d'y dessiner des isolignes (cf figure 4.1).

Cette méthode de coupe du maillage volumique permet d'avoir une représentation locale des valeurs. Une propriété physique étant choisie, ses valeurs ne sont visualisées qu'en une certaine localisation dans le maillage.

Définissons une autre méthode de représentation locale d'une propriété physique dans le maillage : les isosurfaces. Contrairement aux coupes, la valeur est visualisée dans tout le volume. Il ne s'agit cependant que d'une seule valeur, pour une propriété physique donnée.

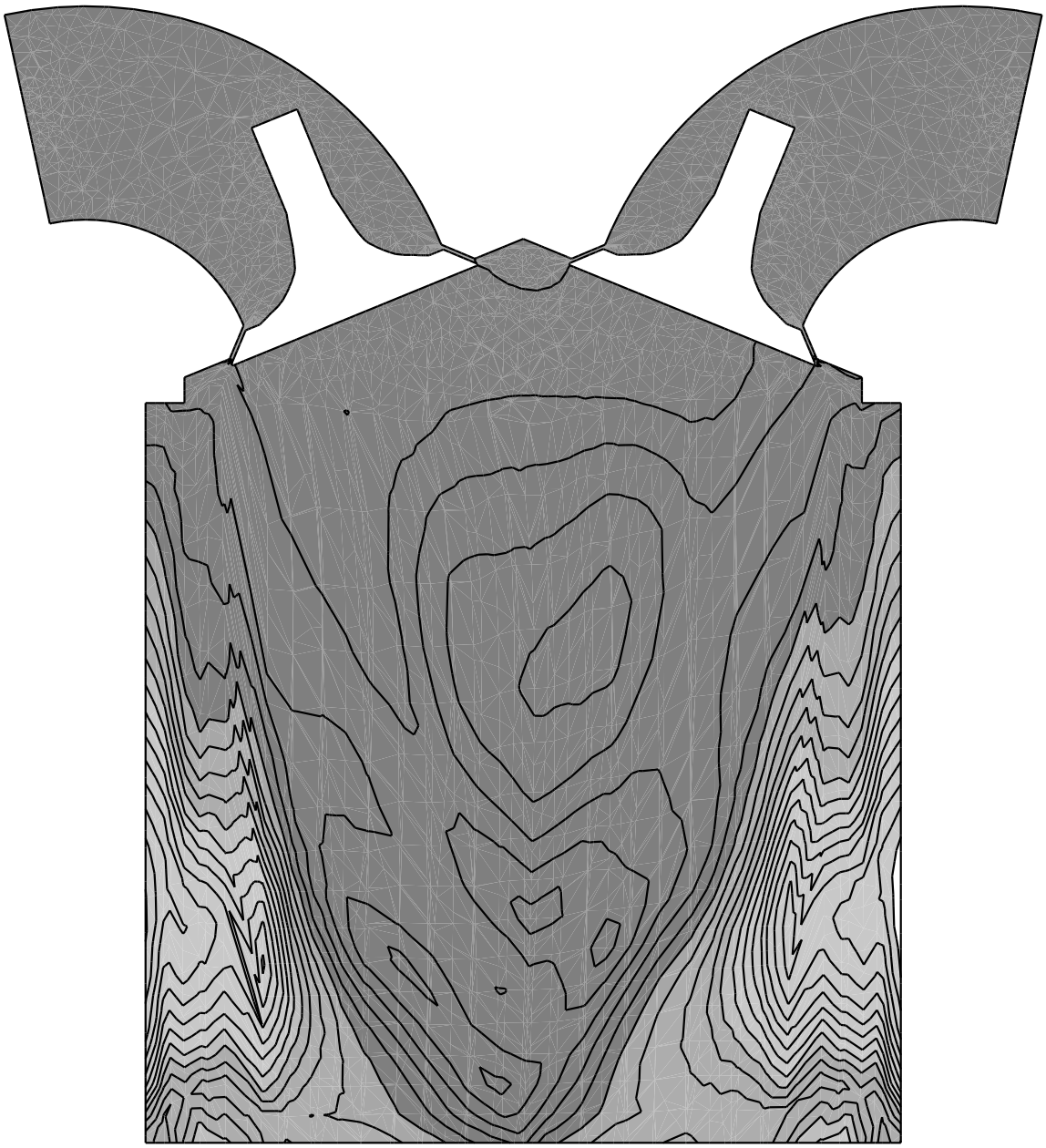


FIG. 4.1 - *Isolignes et Remplissage de facettes d'une coupe de maillage 3D.*

Chapitre 5

Isosurfaces

5.1 Principes

Définissons par $f(x, y, z)$ une fonction définie sur une région Ω de $\mathbb{R}^3$.
Une stratégie pour visualiser f est de choisir une valeur particulière k et d'afficher tous les points où $f(x, y, z) = k$. Si f est continue cet ensemble de points définit une isosurface.
Par exemple en utilisant un maillage volumique aux nœuds duquel une propriété physique est connue, il est possible de construire la surface formée par tous les points de même valeur; (soit l'isosurface des points correspondants à la température 100 degrés par exemple, *cf figure 5.2*).
Nous disposons d'un maillage 3D, représenté par des valeurs scalaires aux nœuds des polyèdres. L'utilisateur choisit une isovaleur pour une propriété donnée (*cf figure 5.1.*)
Le but est alors de calculer la surface engendrée par les points portant cette valeur constante; c'est à dire de déterminer l'intersection de chaque élément de volume avec l'isosurface. Le résultat est un maillage surfacique formé de polygones.
Il est cependant difficile de reconstruire des polygones plans en disposant seulement des points d'intersection sur les arêtes du polyèdre. Par exemple, dans un cas où on trouve huit intersections avec un hexaèdre, (*cf figure 5.3*) comment reconstruire les facettes planes?
Nous nous sommes intéressés à l'algorithme classique des "Marching Cubes" .

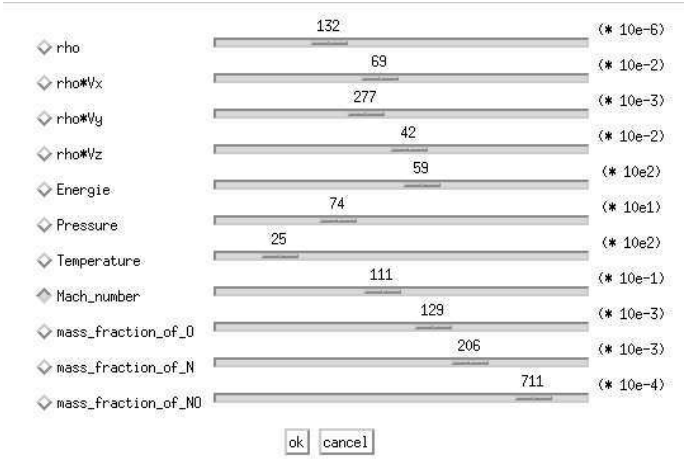


FIG. 5.1 - *Fenêtre de saisie de l'isovaleur.*

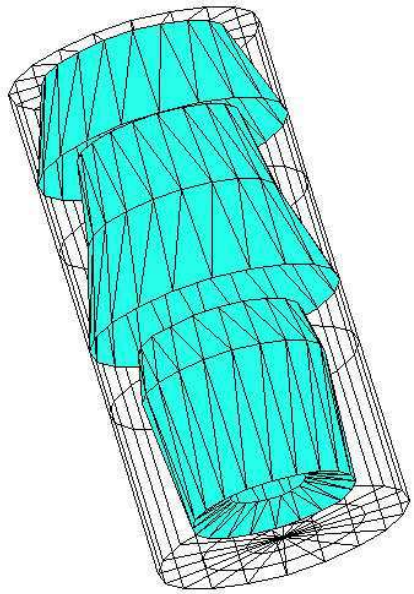


FIG. 5.2 - *Isosurface dans un cylindre plein.*

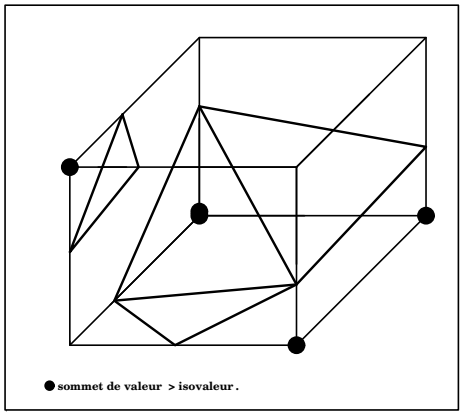


FIG. 5.3 - *Configuration avec 8 intersections.*

5.2 Algorithme des "Marching Cubes".

Cet algorithme a été développé par W.E. Lorensen et H.E. Cline ([LC87]) pour des données issues du domaine de l'imagerie médicale; il s'applique à des maillages structurés uniformes. C'est un algorithme très rapide qui repose sur l'utilisation de table de "lookup" [1]. Le principe est de précalculer les différents aspects (appelés configurations par la suite) que peut prendre l'isosurface dans un élément de volume, selon la répartition des intersections sur les arêtes du volume. Par exemple pour un hexaèdre il y a: 2^8 soit 256 façons pour une surface d'intersecter les arêtes du volume; soit 256 configurations, chacune correspondant à un ensemble de facettes.
La méthode d'indexation de la table de lookup utilise la position des sommets de l'élément de volume par rapport à l'isosurface. Après numérotation conventionnelle pour les huit sommets de l'hexaèdre (*cf figure 5.4*), un octet est génèré, dans lequel chaque bit correspond à la position de la valeur connue au sommet par rapport à l'isovaleur. Par exemple le bit est à un si cette valeur est supérieure (ou égale) à l'isovaleur, zéro sinon.
La configuration 0 étant le cas où tous les sommets de l'hexaèdre sont au dessous de l'isosurface; il n'y a donc pas d'intersection. La configuration suivante, (numéro 1), décrit le cas d'un sommet séparé des sept autres par l'isosurface,

1. table de consultation.

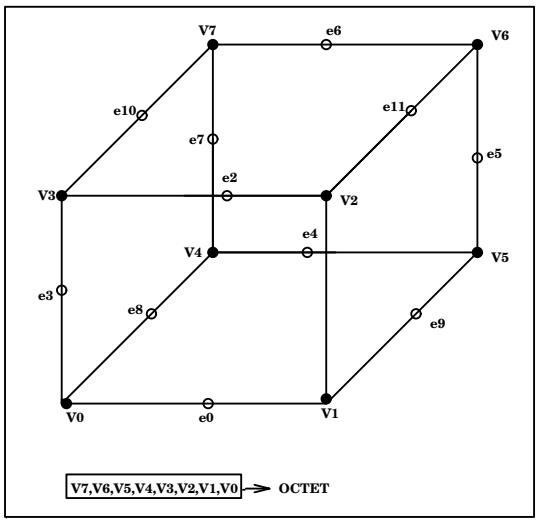


FIG. 5.4 - *Octet correspondant aux positions des sommets par rapport à l'isosurface.*

il en résulte un triangle défini par les intersections des trois arêtes partant de ce sommet.
La figure 5.5, tirée de [LC87] décrit quinze configurations ainsi que les index obtenus pour une certaine convention d'orientation du cube.
Ces configurations sont obtenues à partir des 256 configurations en utilisant les symétries du cube.
Après cette phase de prétraitement pour créer la table de lookup, les éléments de volume du maillage sont parcourus un par un. Pour chaque hexaèdre, les valeurs des huit sommets sont comparées avec l'isovaleur. Un octet est alors généré ; les bits sont définis par les positions de sommets par rapport à l'isovaleur, comme décrit précédemment On récupère grâce à l'octet, la configuration dans la table de lookup. Cette configuration détermine quelles arêtes sont coupées: pour chacune des arêtes, une interpolation linéaire est alors effectuée pour déterminer la position de l'intersection et construire les facettes de l'isosurface.

Cette méthode n'assure cependant pas que les surfaces reconstruites sont continues: en fait elles présentent des trous; M.J.Düurst l'a noté ([Du88]) en pré-

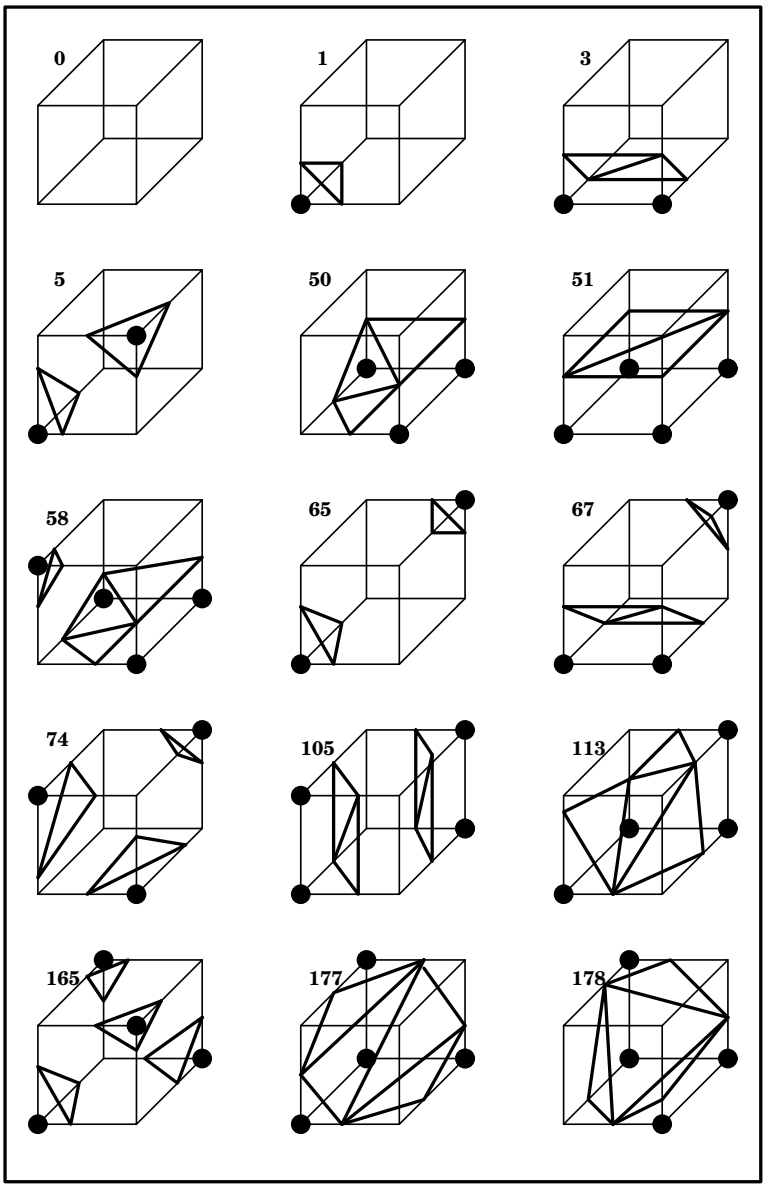


FIG. 5.5 - *Quinze configurations de la table de lookup des hexaèdres.*

sentant un cas erroné. Son exemple est assez compliqué, on peut également montrer un cas ambigu plus simple sur la figure 5.6; présentant deux façons d'orienter les facettes.

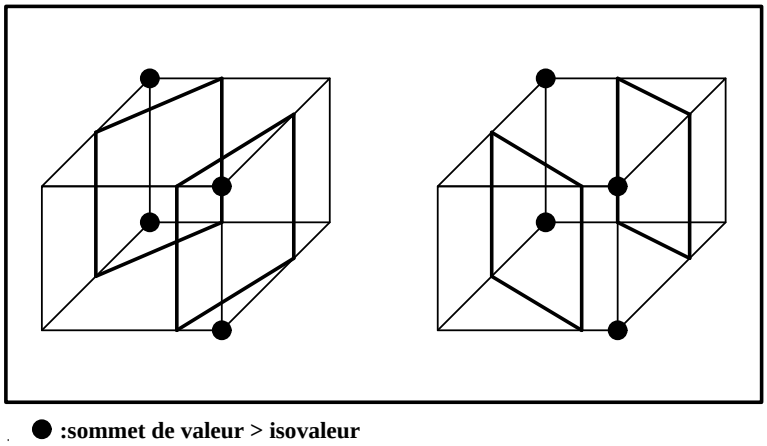


FIG. 5.6 - *Cas ambigu: configurations différentes pour les facettes obtenues à partir des 8 mêmes points d'intersection.*

Toute la difficulté consiste à assurer que l'approximation de l'isosurface calculée a les mêmes propriétés topologiques que l'isosurface réelle.
En fait la table de lookup présente des ambiguïtés pour certaines des configurations. Par exemple dans le cas où l'isosurface coupe plus de trois fois les arêtes sur la face d'un hexaèdre, il y a plusieurs façons d'orienter les facettes. L'algorithme des Marching cubes n'en propose qu'une, d'où la présence de discontinuités dans certains cas (*cf figure 5.6*).
Algorithme des Marching Cubes:

◇ Construction de la table de lookup.
  ⟶ 256 configurations numérotées qui définissent quelles arêtes sont intersectées et décrivent la position des facettes.
◇ Pour chaque élément de volume V:
- Pour tous les sommets S_i de V:
  ⋆ Comparer val_i avec l'isovaleur.
  ⋆ $bit_i \leftarrow 1$ si $val_i \geq$ isovaleur.
  ⋆ $bit_i \leftarrow 0$ sinon.
- Récupérer la configuration $[bit_0..bit_n]$ dans la table de lookup.
- Effectuer le calcul des intersection par interpolation linéaire.
- Construire les facettes de l'isosurface.

5.3 Les "Marching Lines"

J.P. Thirion et A. Gourdon [TG93] résolvent les problèmes de discontinuités présents dans l'algorithme des Marching Cubes. La particularité de leur algorithme baptisé les "Marching Lines", réside dans la preuve qu'ils apportent de la conservation des propriétés topologiques des surfaces reconstruites. Nous ne présentons ici qu'une idée de l'algorithme; les preuves étant dans le rapport [TG93]. Cet algorithme est également adapté au domaine de l'imagerie médicale ; les éléments de volume sont des hexaèdres.
Cet algorithme s'intéresse aux faces des éléments de volume. Chaque face d'un cube est orienté vers l'extérieur; les segments la constituant sont orientés selon la règle de la main gauche. Les faces communes à deux cubes sont alors orientées à l'opposé.
Les valeurs des sommets de la face sont ensuite comparées à l'isovaleur ; le signe + est affecté aux sommets supérieurs (ou égaux) à l'isovaleur, le signe − dans l'autre cas. Une recherche est effectuée pour déterminer quelles arêtes sont intersectées par l'isosurface; ce sont les arêtes dont les deux sommets extrémités n'ont pas le même signe ; soient zéro.
En tenant compte de l'orientation de l'arête, on affecte au point d'intersection le signe du sommet qui le suit. Les points d'intersection sont alors joints pour

former des segments ; soit dans le cas de deux intersections les points de signe − sont reliés à ceux de signe +; pour quatre intersections il y a ambiguïté, (cf figure 5.6, (ambiguïté présente d'ailleurs dans l'algorithme du Marching Cube).
Pour lever cette ambiguïté, la moyenne des valeurs aux quatre sommets de la face du cube est évaluée. Si cette moyenne est supérieure (ou égale) à l'iso-valeur, on se trouve dans le cas 2 (*cf figure 5.7*), sinon c'est le cas 1 qui est retenu.

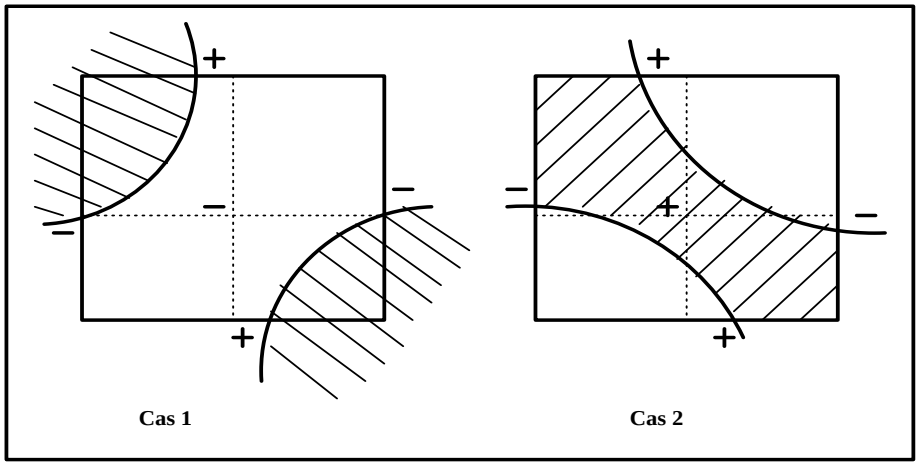


FIG. 5.7 - *Choix de l'orientation des facettes dans un cas ambigu.*

Pour chaque face du cube, un ou deux segments joignant les points d'intersection entre les arêtes et l'isosurface sont retenus. Tous les segments sont alors reliés sur les six faces du cube, en créant des cycles. Tous les segments sont connectés de façon à ce qu'une extrémité négative soit toujours reliée à une extrémité positive, en commençant d'abord à relier les segments sur une même face avant de passer à une face connexe. Ces cycles ainsi créés servent ensuite à construire des facettes polygonales. Il y a au plus douze segments et quatre facettes par hexaèdre.

Cet algorithme bien que rigoureux au sens conservation de la topologie présente des inconvénients:

◇ l'interpolation est réalisée pour chaque arête sans tenir compte des

arêtes communes à plusieurs cubes.

◇ la méthode coûteuse d'orientation est appliquée même pour les cas ne présentant pas d'ambiguïté, (cas fréquents où moins de trois arêtes par face sont intersectées).

◇ l'algorithme est appliqué aux maillages structurés.

En nous inspirant de ces algorithmes, nous avons voulu conserver la rapidité de l'algorithme des Marching cubes,(cf chapitre 5.2) tout en gardant la rigueur des Marching Lines pour la génération de l'isosurface, avec une optimisation pour ne pas dupliquer les calculs pour les points présents sur des arêtes communes à plusieurs cubes.

5.4 Algorithme proposé

1. Pour utiliser la rapidité de l'algorithme des Marching Cubes, nous disposons d'autant de tables de lookup que de types d'éléments de volume. Il est ainsi possible de traiter les nombreux types d'éléments présents dans un maillage non structuré (tétraèdre, hexaèdre, prisme, pyramide). Cet algorithme est donc extensible à tous les types d'éléments de volume; il suffit de déterminer les différentes tables de lookup.

2. Grâce à une technique d'orientation des faces de l'élément de volume, les ambiguïtés générées par les Marching Cubes n'apparaissent plus. Contrairement à l'algorithme des Marching Lines, nous n'orientons pas tous les éléments de volume du maillage; ce qui est inutilement coûteux dans la plupart des cas. Nous effectuons des tests similaires à ceux effectués dans les "Marching Cubes" pour déterminer la configuration des facettes mais en détectant les cas ambigus qui sont alors résolus par une méthode inspirée des "Marching Lines".

3. Enfin la gestion d'une table de hashage[2] permet d'éviter de dupliquer les interpolations et stockages des points d'intersections constituant l'isosurface.

2. adressage calculé.

Décrivons maintenant les grandes lignes de l'algorithme.
Une table de lookup a, par ailleurs, été précalculé: 256 combinaisons pour les hexaèdres et 16 pour les tétraèdres: (soit 2^n cas pour n sommets).
Tout le volume est alors parcouru, en tenant à jour une table de hashage contenant toutes les arêtes intersectées. Les configurations possibles sont évaluées par un octet (pour les hexaèdres, 4 bits pour les tétraèdres), dont chaque bit détermine la position d'un sommet par rapport à l'isovaleur (le bit est à 1 si cette valeur est supérieure ou égale à l'isovaleur, 0 sinon).
La valeur de l'octet, pour un élément de volume, correspond à une configuration dans la table de lookup: soient les arêtes intersectées et la disposition des facettes polygonales. Dans un cas ambigu (par exemple celui décrit sur la figure 5.7), la table de lookup va générer tous les cas de configuration possibles. Il faut alors calculer les moyennes des sommets des faces de l'élément pour déterminer quelle configuration choisir.
Des interpolations linéaires sont alors calculées pour définir les points formant les facettes de l'isosurface, après s'être assuré que les points d'intersection n'étaient pas déjà stockés dans la table de hashage.

Tout le problème réside en fait dans la génération des tables de lookup. Celle des tétraèdres ne pose aucun problème : 16 configurations et aucune ambiguïté possible pour générer des facettes polygonales ; ce qui n'est pas le cas pour les hexaèdres.
Intéressons-nous justement à la génération automatique de la table de lookup pour des hexaèdres. Les 256 configurations sont générées ; il faut leur associer la position des intersections sur les arêtes du volume et la définition des facettes joignant ces intersections.
Pour chaque face de l'hexaèdre, un tableau "interEgde" est mis à jour ; ce tableau conserve, pour chaque face, un index sur les arêtes qu'elle contient avec la valeur $+/-1$ si l'arête est coupé, (le signe est celui de l'extrémité de l'arête orientée) ou 0 sinon.
Un compteur sur le nombre d'arêtes coupées par face est également maintenu. Si pour chaque face, le nombre d'arêtes coupées est de deux, le segment orienté (de $-$ vers $+$) est créé; s'il est de quatre, il y a ambiguïté. Il faut alors générer tous les cas de configuration possibles et laisser le choix se faire dynamiquement lors de l'exécution de l'algorithme.

Une liste de cycles est alors mise à jour pour relier les segments entre eux sur toutes les faces du cube et créer alors des facettes planes. Le tableau interEdge qui détermine les intersections pour chaque face et chaque arête du cube est utilisé pour relier les segments de signe opposés. Il faut remarquer qu'un point commun à deux faces aura des signes opposés sur chacune des faces. On commence par relier les segments sur une même face avant de poursuivre ce segment sur la face voisine.

Algorithme de génération de table de lookup:

Pour chaque configuration entre 0 et 256
- ◇ Pour chaque face F_i de l'hexaèdre
  - • Pour chaque arête A_i orientée :$[s_i, s_{i+1}]$
    - ⋆ Si signe$(s_i) \neq$ signe(s_{i+1})
      - $interEdge[F_i, A_i] \leftarrow \pm 1$ selon signe(s_{i+1})
    - ⋆ Sinon $interEdge[F_i, A_i] \leftarrow 0$
    - ⋆ Incrémenter le nombre d'arêtes coupées par face : $nbac$.
- ◇ Si pour chaque face F_i
  - • $nbac = 0$: rien
  - • $nbac = 2$: segment de $\ominus$ vers $\oplus$
  - • $nbac = 4$: ambiguïté, générer tous les cas possibles
- ◇ gestion des listes de cycles
  - • Tant qu'il reste des valeurs non nulles dans $interEdge$
    - ⋆ a) Préparer un nouveau cycle
    - ⋆ Trouver la première valeur non nulle dans $interEdge$
    - ⋆ b) Rajouter l'arête correspondante à la liste des cycles
    - ⋆ Marquer la valeur traitée (la remettre à 0)
    - ⋆ Chercher sur la même face le signe opposé
    - ⋆ Si trouvé recommencer en b)
    - ⋆ Sinon chercher sur la face voisine le même point
      - Si trouvé recommencer en b)
      - Sinon cycle terminé recommencer en a)

5.5 Résultats

L'implémentation de cet algorithme se trouve en annexe B.
Le tableau suivant donne une petite idée des résultats obtenus, en fournissant quelques temps de calcul d'isosurfaces pour différents maillages. Les temps sont fournis en seconde, le calcul est assez fiable (obtenu grâce à la fonction "getrusage".) Plusieurs isosurfaces ont été calculées pour différentes propriétés (densité, température, nombre de Mach, ...) et différentes valeurs.
Les temps moyens (utilisateur + système) de calcul pour différents maillages sont visualisés dans le tableau suivant. Ils ont été réalisés sur station Sun SS10/30, 32 Mega de RAM.

maillage	isosurface créée	temps (en s.)
4075 nœuds	450 nœuds	0.15
	9864 nœuds	0.67
	11415 nœuds	0.83
38536 nœuds	31386 nœuds	2.58
	48516 nœuds	3.59
	71040 nœuds	4.8
82429 nœuds	1908 nœuds	1.190
	6174 nœuds	1.63
	18678 nœuds	3.15

Il faut remarquer que le temps est une fonction linéaire du nombre de nœuds de l'isosurface construite pour un maillage donné, et, pour un même nombre de nœuds de l'isosurface, une fonction linéaire du nombre de nœuds du maillage.

Quelques images :

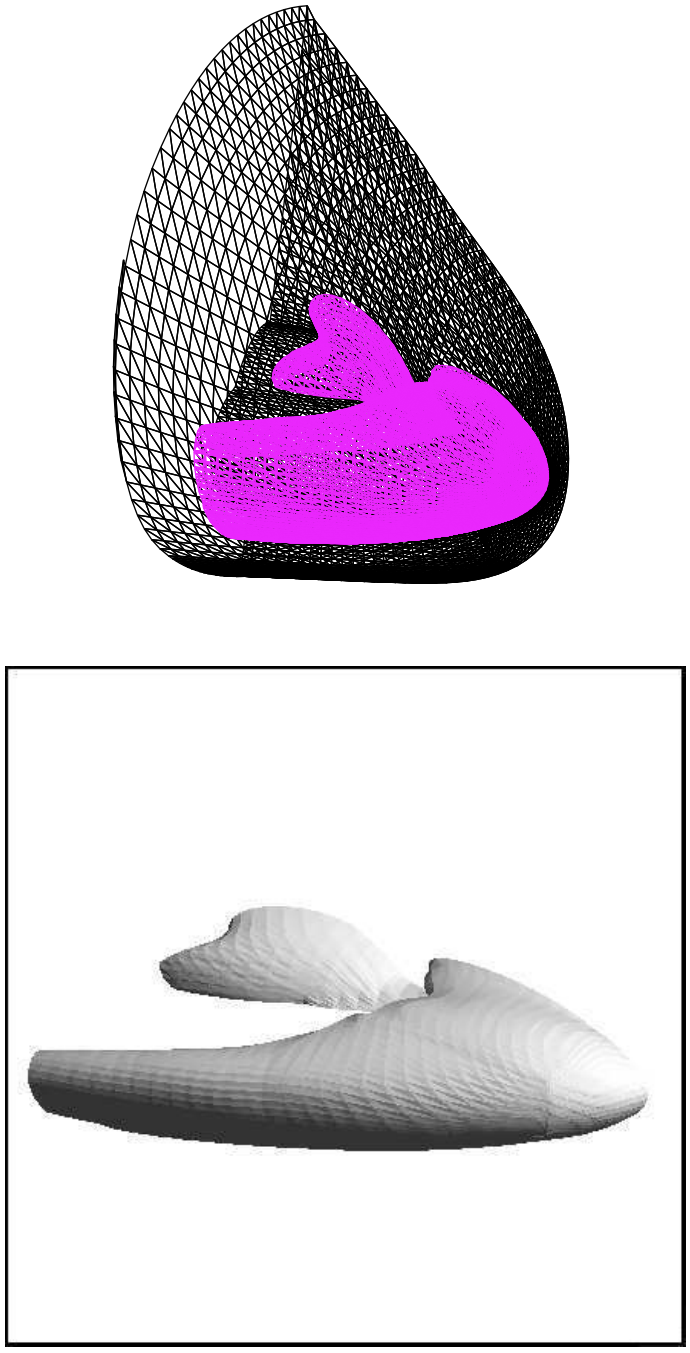
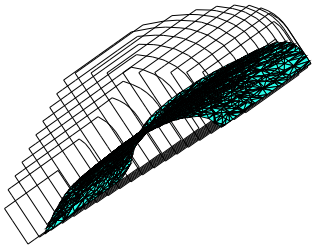
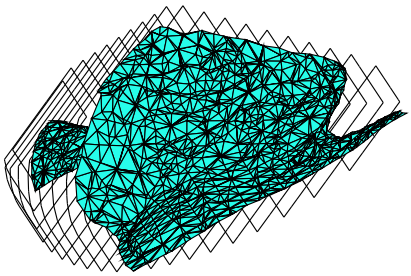


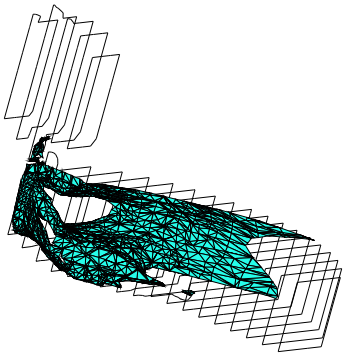
FIG. 5.8 - *Isosurface de la température sur une face avant de navette.*



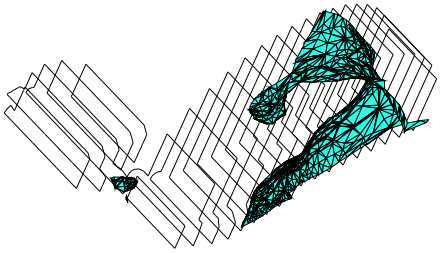
Isosurface dans une chambre de combustion. Rho*u=2.4



Isosurface dans une chambre de combustion. Rho*v=0.29



Isosurface dans une chambre de combustion. Rho=54.42e-4



Isosurface dans une chambre de combustion. Rho*v=0.22

Nous venons de voir deux approches principales pour visualiser les données scalaires connues aux nœuds d'un maillage 3D : les plans de coupe et les isosurfaces.
◇ La méthode de plans de coupe est un mode de représentation locale qui permet de voir plusieurs valeurs localisées en une partie précise du volume. Le résultat visualisé est la variation des valeurs pour une propriété donnée sur une coupe du maillage.
◇ L'extraction d'isosurfaces consiste à générer la surface engendrée par tous les points du volume ayant une certaine valeur. Le mode de représentation est local au sens où seule une valeur est prise en compte.

Une autre technique : le rendu volumique permet, par contre, une représentation globale de toutes les valeurs dans le volume. Toutes les données du maillage contribuent à la construction de l'image visualisée.

Chapitre 6

Rendu Volumique

6.1 Définition

Le principe du "rendu volumique" ou encore "transparence de volume" est de pouvoir visualiser toutes les valeurs présentes dans le volume, pour une propriété donnée. Pour voir simultanément des valeurs réparties dans un volume, il faut utiliser des techniques de transparence; le volume est transparent et chaque point qui le compose émet de la lumière (dont l'intensité est liée à la valeur de la propriété étudiée).

Le but n'est pas d'obtenir des images réalistes mais de fournir un outil permettant d'analyser de manière plus globale les valeurs définies aux nœuds du maillage. Le modèle d'éclairement classique (incluant les composantes de couleur (R, G, B) et l'opacité α) est donc simplifié à l'extrême.
La variation des valeurs est perçue comme un dégradé de couleurs dont l'intensité est fonction de la quantité de valeurs traversées par un rayon partant de l'oeil de l'observateur. Cette méthode est dérivée des techniques de rendu volumique, si ce n'est que la composante d'opacité est omise.

6.2 Intérêts et défauts du rendu volumique

L'intensité lumineuse correspondant à la variation des valeurs est la seule information fournie mais ne suffit pas pour distinguer la variation des valeurs de la profondeur du volume.
Par exemple pour visualiser les variations de température :
- Pour une épaisseur constante du volume, plus la zone apparaît brillante, plus la température est élevée.
- De même pour une température constante, la zone apparaît d'autant plus brillante que l'épaisseur de volume traversée est importante.
Il est donc important de pouvoir lever ces ambiguités. Une bonne solution est de considérer le volume depuis plusieurs points d'observation. Ainsi le calcul du rendu volumique est associé à un outil d'animation ; ce qui nécessite le précalcul de plusieurs images.

Cette méthode de rendu volumique a de nombreux avantages pour les numériciens, ils peuvent ainsi appréhender la répartition des valeurs dans tout le volume, avant de faire des observations plus localisées grâce à des techniques de plans de coupe ou d'isosurfaces. Cela leur permet aussi de déceler des anomalies dans leurs calculs (zones de divergences des valeurs) et de repérer rapidement les particularités de certaines données.

Après une étude d'articles sur le sujet, deux techniques ont été retenues. Il faut noter que la plupart des algorithmes déjà publiés s'appliquent à des maillages structurés.

6.3 Comparaison : lancer de rayon / projection cellule par cellule.

Dans leurs articles R. A. Drebin et ses collègues ainsi que J. Wilhelms ([DCH88, WCAR90, WG91]), comparent les deux méthodes généralement utilisées.
La première est une méthode de lancer de rayons; elle consiste à lancer autant de rayons qu'il y a de pixels sur l'image, et pour chaque rayon, à traverser toutes les cellules rencontrées dans le volume avant de passer au rayon suivant.
La seconde méthode est une méthode de projection; elle s'intéresse à tous les

pixels sur lesquels une cellule (élément de volume) se projette parallèlement, avant de passer à la cellule suivante.
Un principe commun à ces deux méthodes[1] est de déterminer une couleur (et éventuellement une opacité) pour chaque valeur scalaire. La valeur en un pixel est alors l'accumulation des valeurs rencontrées par un rayon partant du pixel et traversant tout le volume. Ces valeurs sont accumulées tant que le volume n'a pas été traversé ou qu'un seuil d'opacité n'a pas été atteint. L'algorithme passe alors à la cellule suivante ou au pixel suivant selon la méthode utilisée.

6.3.1 Lancer de rayon

La méthode du lancer de rayon décrite aussi par [Gar90, KH84, Lev88, MS90, Sab88, ST90, UK88] est accélérée grâce à un critère d'arrêt : le seuil d'opacité. En effet pour chaque pixel un rayon est lancé et pour chaque rayon la contribution de chaque cellule traversée est accumulée, jusqu'à ce que l'opacité atteigne l'unité ou que le volume soit traversé.
Une méthode de base consiste à déterminer l'intersection de chaque rayon avec tous les éléments de volume, soit un algorithme en $O(n * m)$ pour n rayons et m cellules. Dans le cas de maillages structurés, cette complexité excessive peut être diminuée en utilisant des informations sur la répartition spatiale des cellules (structure par couches ou tables d'adjacences).
Cette méthode est cependant difficilement adaptable à des éléments de volume de type polyèdres quelconques. La gestion d'une table d'adjacence est coûteuse dans le cas de maillages non structurés. M. Garrity [Gar90] a cependant décrit un tel algorithme pour des éléments hexaèdriques non réguliers.

Un autre problème se pose: quand un rayon traverse un élément de volume, il faut déterminer l'interpolation des valeurs connues à ses sommets en un point sur la face; or même en choisissant une formule d'interpolation simple, le calcul reste coûteux.
Comme nous le verrons par la suite, la méthode de projection cellule par cellule évite ces problèmes d'interpolation.

1. méthodes de forward et backward

La méthode de lancer de rayon ne s'applique pas facilement à des maillages non structurés. Etudions l'autre méthode, celle de projection cellule par cellule.

6.3.2 Projection cellule par cellule

La méthode de projection (étudiée aussi par [DCH88, MHC90, NSG90, ST90, Wes90, WG91]) consiste à faire l'intersection entre chaque cellule (élément de volume) et tous les plans perpendiculaires à l'écran. Ces plans sont parallèles entre eux et il y en a autant que de lignes écran (*cf figure 6.1*). Le résultat obtenu est un polygone convexe dont les valeurs aux sommets sont calculées par interpolations linéaires des valeurs connues aux sommets de l'élément de volume.
Pour déterminer la couleur de chaque pixel de cette ligne écran, il suffit d'intégrer toutes les valeurs le long d'un rayon traversant le polygone (*cf figure 6.2*) puis de cumuler les contributions de chaque élément de volume.

P. Shirley et A. Tuchman ([ST90]) proposent pour cette méthode de projection un algorithme très efficace qui tire parti de certaines primitives fournies par l'accélérateur graphique. Ils décomposent tous les éléments de volume en tétraèdres et utilisent une technique efficace pour les projeter à l'écran. Leur algorithme est accéléré du fait de l'utilisation des primitives de remplissage de polygones avec dégradé de couleurs fournies par certains accélérateurs graphiques.
Cet algorithme peut déjà s'appliquer à des maillages non structurés, mais dans notre cas, il ne présente pas d'intérêt car notre but est de faire fonctionner les algorithmes sur tous types de stations de travail.

J. Wilhelms et A. Van Gelder ([WG91]) privilégient la méthode de projection. Ils effectuent des intersections par plans de coupe (correspondant à une ligne écran) sur tout le maillage et récupèrent ainsi une coupe du volume tout en créant l'image de plans en plans; ils admettent que cette méthode est coûteuse mais ils conservent ces plans de coupe et les traitent par la suite dans d'autres applications.
Cette technique n'a d'intérêt que si ces informations intermédiaires sont réutilisées par la suite.

6.4 Une première méthode applicable

Un article récent [Gie92] fait le point sur les parutions concernant le rendu volumique. La plupart des algorithmes décrits dans la littérature s'appliquent aux maillages structurés. Seul M. Garrity [Gar90] a proposé un algorithme pour des maillages hexaèdriques non réguliers basé sur une technique coûteuse de lancer de rayon (avec gestion d'une table d'adjacence).
En fait comme le fait remarquer C. Giertsen [Gie92], personne n'a encore décrit d'algorithme général pour des maillages non structurés complexes. Il s'intéresse à des maillages non structurés complexes.
L'idée de l'algorithme est d'utiliser un buffer de "scan-plane conversion" [2] qui contient les informations résultant de l'intersection entre un plan perpendiculaire à une ligne écran et le volume.
L'inconvénient de cet algorithme est qu'il faut pour chaque ligne d'écran déterminer l'intersection avec tout le maillage puis générer les facettes du plan de coupe; ce qui implique le stockage d'une structure intermédiaire assez volumineuse (au moins autant de facettes qu'il y a d'éléments de volume coupés par le plan).

Par rapport aux techniques précédemment décrites, cet algorithme basé sur la méthode de projection, semble déjà mieux adapté au cas de maillages non structurés. L'intersection du plan de coupe avec tout le maillage est cependant assez coûteuse car pour chaque ligne écran, des tests sont effectués avec tout le maillage. Le stockage des facettes intersections est également volumineux.

6.5 Méthode proposée

Nous nous sommes inspirés de la méthode de projection cellule par cellule pour proposer une méthode évitant les tests coûteux d'intersection entre chaque plan associé à une ligne écran et toutes les cellules.
Le principe de l'algorithme est le suivant:

2. balayage d'un volume par autant de plans successifs parallèles qu'il y a de lignes écran.

6.5.1 Un premier algorithme

A chaque ligne de l'image finale est associé un plan perpendiculaire à l'écran. Pour chaque élément de volume, il faut déterminer l'intersection avec ces plans, (*cf figure 6.1*). Pour éviter des calculs inutiles, il suffit de déterminer au préalable les extrema de l'élément de volume selon l'axe y. En effet seuls les plans correspondant à des lignes comprises entre ces valeurs intersecteront l'élément considéré. Cela réduit sensiblement le nombre de tests d'intersections surtout si l'élément de volume est petit et n'est intersecté que par quelques lignes écran.
Le résultat d'une intersection est un polygone, dont les valeurs aux sommets sont obtenues par interpolation linéaire le long des arêtes de l'élément de volume. Sur l'écran, ce polygone (défini dans le plan parallèle au plan (Oxz)) correspond à un segment horizontal.
Pour déterminer la couleur d'un pixel le long de ce segment, on calcule alors l'intersection entre la droite (perpendiculaire à l'écran) partant du pixel et le polygone. Le résultat obtenu est (a, b) : deux intersections qui délimitent un segment. L'intégration de f le long de ce segment fournit la contribution de l'élément de volume au pixel considéré (*cf figure 6.2*).
Soit le segment formé par les points a et b du schéma. La valeur d'intégration le long de ce segment est obtenue par : $valeur = (\frac{f(a)+f(b)}{2}) * (z_a - z_b)$

Algorithme:
Pour chaque élément de volume **V**
- ◇ Calculer les y_{min} et y_{max} de tous les sommets de **V**
- ◇ Pour tous les $y_{ecran} \in [y_{min},\ y_{max}]$
  - • Calculer l'intersection de **V** avec le plan $y = y_{ecran}$
  - • Le résultat est un polygone **P**
  - • Calculer les x_{min} et x_{max} de tous les sommets de **P**.
  - • Pour tous les $x_{pixel} \in [x_{min},\ x_{max}]$
    - * Calculer l'intersection entre **P** et la droite $y = y_{ecran}, x = x_{pixel}$
    - * Récupérer deux points d'intersection a et b.
    - * Intégrer alors les valeurs le long du segment $[a, b]$.
    - * Ajouter cette contribution de **V** au pixel considéré (y_{ecran}, x_{pixel}).

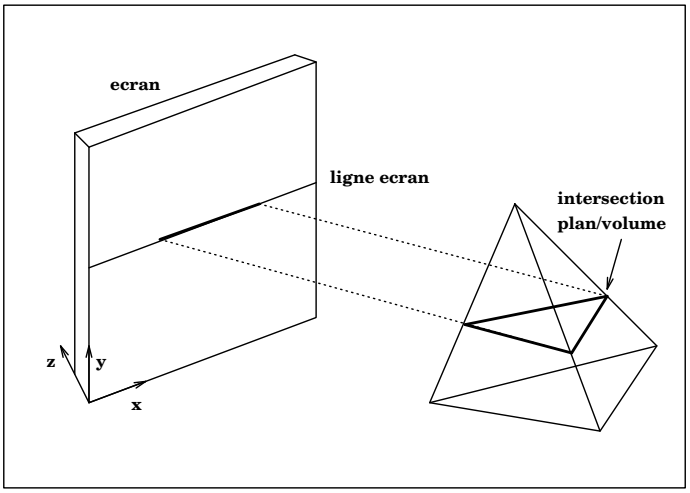


FIG. 6.1 - *Intersection Plan-Volume.*

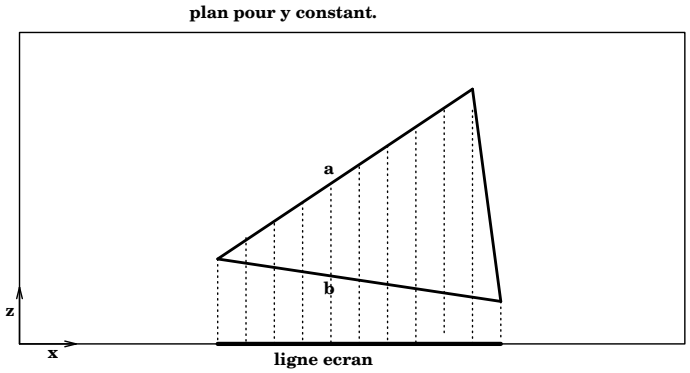


FIG. 6.2 - *Intersection Droite-Facette.*

Pour améliorer cet algorithme il est intéressant de pouvoir rendre incrémentales les procédures coûteuses d'intersection entre volume/plan et facette/droite.

6.5.2 Version incrémentale

Intersection droite/facette

Pour faire l'intersection entre des droites parallèles et une facette (*cf figure 3*), il est inutile de faire toutes les intersections. Une bonne optimisation consiste à calculer l'intersection une première fois, puis de déduire les intersections suivantes de manière incrémentale. Chaque nouveau calcul nécessite seulement deux additions (incrément de la valeur moyenne et de la longueur du segment) et une multiplication (intégration). L'algorithme utilisé pour ces calculs s'inspire de la méthode décrite par P. S. Heckbert [Heck90].
Il est également intéressant de pouvoir généraliser ce code en trois dimensions; inutile en effet de faire toutes les intersections entre les plans parallèles et l'élément de volume.

Intersection plan/volume

Une extension à la méthode incrémentale "droite-facette" a été implémentée pour définir les intersections "plan-volume". Quand un plan franchit un sommet, le nombre d'arêtes intersectées varie (cf figure 6.3)

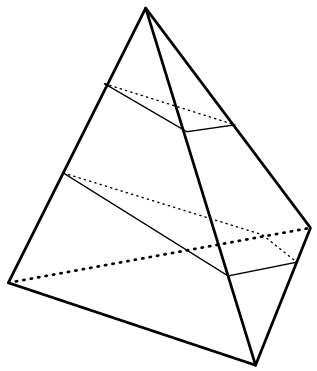


FIG. 6.3 - *Méthode incrémentale de plans de coupe: problème après franchissement d'un sommet.*

La méthode proposée ici s'applique aux polyèdres convexes.
Pour chaque polyèdre convexe, un tableau OE des arêtes orientées est créé. Les plans d'intersection sont définis sont définis par $y = y_{screen}$. Soit i l'arête définie par les deux sommets $v_0^i(x_0^i, y_0^i, z_0^i)$ et $v_1^i(x_1^i, y_1^i, z_1^i)$. Chaque arête est orientée selon les y^i croissants, soit $y_0^i \leq y_1^i$.
Deux tableaux AE, IE contenant les index des arêtes sont maintenus ; AE maintient les index triés des arêtes actives, tandis que IE trie les index des arêtes inactives. Une arête est active quand elle est coupée par le plan courant. A l'initialisation, le tableau des arêtes actives AE est vide, tandis que celui des arêtes inactives IE contient toutes les arêtes triées.

Pour le premier plan de coupe (le plus petit entier y_{ecran}) entre $[y_{min}, y_{max}]$), nous recherchons quelles arêtes inactives sont intersectées ($y_{screen} \in [y_0^i, y_1^i]$) ; ces arêtes deviennent actives et sont ajoutées au tableau AE.
Quand une arête devient active, les coordonnées et la valeur du point d'intersection sont calculées (ainsi que leurs increments pour la coupe suivante). Ces informations sont rajoutées au tableau OE.
Si le nombre d'arêtes ajoutées est supérieur à 3, elles sont triées afin de créer un polygone convexe.
L'algorithme de P.S. Heckbert [Heck90] est alors utilisé pour calculer la contribution au pixel.

Pour les plans suivants (valeur suivante de y_{screen} entre $[y_{min}, y_{max}]$), toutes les arêtes actives de AE sont considérées.
$\diamond$ Si le plan coupe encore l'arête ($y_{screen} \leq y_1^i$), les incréments précalculés des coordonnées et valeur sont ajoutés pour obtenir la nouvelle intersection.
$\diamond$ Si au moins un sommet est franchi, nous recherchons alors les arêtes inactives qui deviennent actives. Les nouveaux points d'intersection sont triés afin de créer un polygone convexe, puis l'algorithme de Heckbert est à nouveau appelé.

Algorithme:

Pour chaque polyèdre convexe :
- ◇ Créer les tableaux :
  OE(arêtes orientées),AE (vide) et IE (toutes les arêtes.)
- ◇ Pour le premier plan (plus petit entier $y_{ecran} \in [y_{min}, y_{max}]$):
  - • Pour chaque arête inactive:
    - ⋆ Si le plan coupe l'arête :
      - Elle devient active
      - Calcul des coordonnées, valeur, incréments de l'intersection.
  - • Si le nombre d'arêtes ajoutées est ≥ 3 :
    - ⋆ Elles sont triées pour former un polygone convexe.
  - • Calcul de l'intersection droite/facette.
- ◇ Pour les plans suivants (entier $y_{ecran} \in [y_{min}, y_{max}]$):
  - • Pour chaque arête active :
    - ⋆ Si le plan coupe toujours l'arête
      - Ajout des incréments précalculés aux coordonnées et valeur.
    - ⋆ Sinon l'arête n'est plus active (enlevée de AE).
  - • Si au moins un arête active est devenue inactive :
    - ⋆ Pour chaque arête inactive, recherche de celles devenant actives.
    - ⋆ Tri des nouveaux points d'intersection pour former un polygone.
  - • Intersection droite/facette.

6.6 Résultats et perspectives

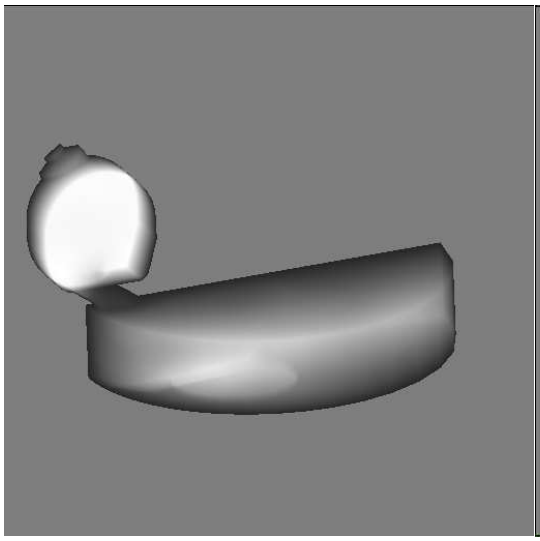
Un système d'animation a été mis en place pour permettre à l' utilisateur de visualiser la répartition des valeurs dans le volume depuis différents points d'observation.
Les temps moyens (utilisateur + système) de calcul pour différents maillages sont visualisés dans le tableau suivant. Ils ont été réalisés sur station Sun SS10/30, 32 Mega de RAM.

L'amélioration par rapport à la version non incrémentale de cet algorithme est de l'ordre de 35%.

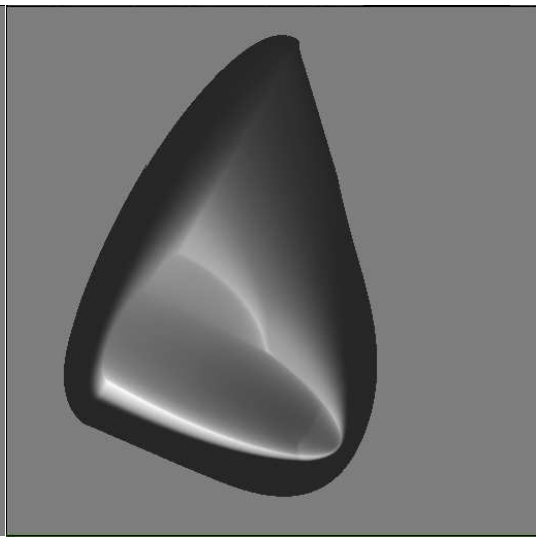
maillage	temps moyens
3290 nœuds	37.48 s
4075 nœuds	40.01 s
12787 nœuds	167.99 s
38536 nœuds	192.05 s
86961 nœuds	206.68 s

Les images produites [Kh93] peuvent être améliorées, notamment en tenant compte du facteur opacité pour représenter des frontières du maillage. Un prétraitement des valeurs avant de les visualiser devrait permettre de mieux déceler certaines de leurs caractéristiques.

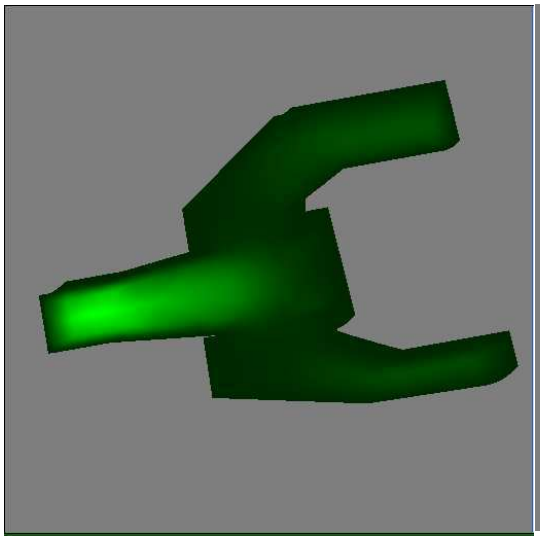
La figure suivante montre quelques images produites.



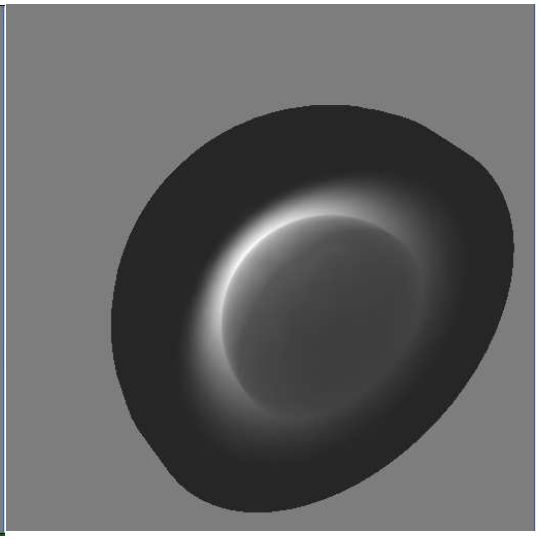
chambre de combustion d'un moteur diesel (Densité)



écoulement autour de la navette Hermès (Température)



tête de chambre de combustion (Vitesse)



écoulement autour d'une sphère (Température)

Chapitre 7

Lignes de courant

Une ligne de courant ("stream line") est une courbe tangente en tous points au champ de vecteur vitesse.
La méthode de calcul est la suivante : à partir d'un point de départ P choisi dans le volume maillé, il faut calculer la valeur du vecteur en ce point puis se déplacer par pas successifs dans la direction du vecteur vitesse pour trouver les points suivants.

Un premier problème consiste à déterminer quel élément de volume contient ce point P puis à estimer le vecteur vitesse au point P par interpolation à partir des valeurs connues aux nœuds de l'élément de volume.

7.1 Localisation d'un point dans un maillage 3D

Soient P un point de $I\!R^3$ et M un maillage tridimensionnel quelconque : structuré ou non, homogène ou formé d'éléments de volume quelconques.
Un premier problème consiste à choisir interactivement à l'écran un point à l'intérieur d'un maillage volumique. En effet pour toutes les visualisations, l'objet 3D est représenté par son maillage de peau (frontière du maillage) et seuls des points en surface peuvent être interactivement choisis à l'écran.
La méthode utilisée dans VIGIE pour choisir un point 3D à l'intérieur d'un

maillage volumique consiste à projeter le volume suivant 3 plans de projection : (XY, YZ, ZX) puis à cliquer un point sur 2 des 3 fenêtres de projection (la première fixant une droite et la seconde un point), (cf figure 7.1).

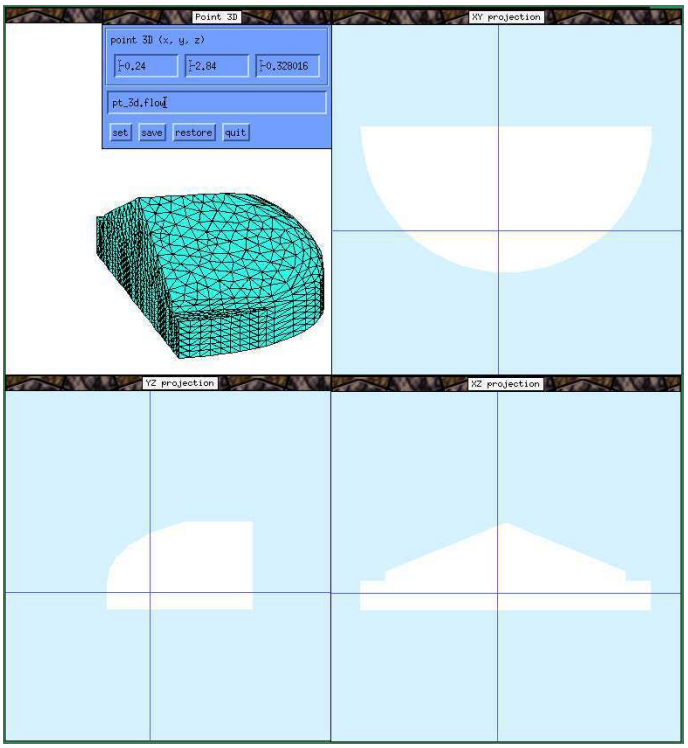


FIG. 7.1 - *Interface pour le choix d'un point dans un maillage 3D.*

Une fois les coordonnées du point récupérées, il faut pouvoir déterminer dans quel élément de volume il se trouve. Si le point P n'est pas dans le premier élément de volume formant le maillage, la recherche doit se poursuivre dans les éléments de volume, tant que le point n'est pas trouvé ou que tout le maillage n'a pas été exploré.

7.1.1 Algorithme de base

La méthode de base, pour rechercher à quel élément le point P appartient, consiste à parcourir tous les éléments de volume et à tester pour chacun d'eux,

s'il contient le point P.
Cette méthode est cependant coûteuse, en $O(n)$, avec n le nombre d'éléments de volume.

7.1.2 Optimisation: propagation par voisinage

Une amélioration consiste a partir d'un élément de départ, puis a tester s'il contient le point; puis dans le cas contraire, a passer à un élément voisin.
Tout élément voisin a une face commune avec le volume courant. On en choisit un, s'il existe, tel que cette face délimite une portion de l'espace dans laquelle se trouve le point P. Les éléments voisins aux autres faces sont alors éliminés. Le nouvel élément peut être considéré comme plus proche du point P du fait qu'il appartient au demi-espace, délimité par la face commune, qui contient le point P, alors que le précédent appartenait à l'autre demi-espace. Il est cependant possible de montrer par un contre-exemple que cela ne garantit pas une meilleure proximité.
La recherche s'effectue en passant d'éléments en éléments en se rapprochant rapidement du volume contenant P. La complexité de cette méthode est inférieure à celle de l'algorithme de base puisque tous les éléments ne sont pas parcourus.

Dans le cas de maillages non structurés, aucune information n'est donnée à priori sur les relations de voisinage entre les cellules ; le passage d'un élément de volume à son voisin n'est donc pas immédiat, contrairement au cas des maillages structurés.
Il est donc nécessaire de construire une table définissant les cellules adjacentes. L'accès à cette table doit être rapide, le passage d'un élément de volume à son voisin étant effectué un grand nombre de fois.

7.1.3 Construction de la table d'adjacence

Principe

L'utilisation d'une table d'adjacence consiste à fournir une face d'un élément et à récupérer l'autre élément de volume partageant la même face.

Les coordonnées des nœuds du maillage sont stockées dans un tableau unique. La description d'une face s'effectue donc en donnant la liste des index des coordonnées des sommets dans ce tableau.
Une face a, en général, une orientation différente selon son appartenance à tel élément de volume ou à tel autre. De plus pour une orientation donnée, le choix du premier sommet reste libre. Il est donc nécessaire d'ordonner les index des sommets formant chaque face, afin d'avoir une représentation canonique de chaque face.
D'autre part, il faut également conserver une trace des permutations ayant permis d'orienter les sommets de la face, afin de conserver la géométrie du problème, et éviter ainsi à l'affichage, d'obtenir des faces croisées ou mal orientées. La construction de la table d'adjacence peut en effet servir au calcul de frontières (faces n'appartenant qu'à un seul élément).
Enfin le but principal de la table d'adjacence est l'accès rapide à un élément voisin. L'utilisation d'une table de hashage[1] facilite l'accès rapide aux informations.

Structure

La table d'adjacence *Adjac* est structurée ainsi:
Considérons le cas d'une face d'au plus 4 sommets (ce qui couvre les cas des tétraèdres, hexaèdres, prismes et pyramides). Choisissons pour fonction de hashage le plus petit index des sommets ; ce choix permet d'éviter de stocker l'index du plus petit sommet [HGD91].
Les index des sommets formant les faces étant triés, par ordre croissant, pour permettre la représentation canonique, l'accès à l'entrée de la table *Adjac* (indexée par le premier index de sommet) est rapide.
Chaque case du tableau *Adjac* (soit la case s_i) pointe sur une cellule contenant les index des sommets ordonnés suivants ($s^1_{i,0}$, $s^2_{i,0}$, $s^3_{i,0}$), formant une face d'élément. Une telle cellule contient également la trace des permutations (*perm*) et les index pointant sur les deux éléments de volume contenant cette face (V_0, V_1) (ou bien ($V_0, -1$) pour une face sur la frontière). Pour finir cette cellule pointe sur les faces suivantes (cellules de même type) dont la représentation canonique

1. adressage calculé.

contient également le sommet s_i (cf figure 7.2) ; la $j$ème cellule correspond à la face de représentation canonique $(s_i, s^1_{i,j}, s^2_{i,j}, s^3_{i,j})$.

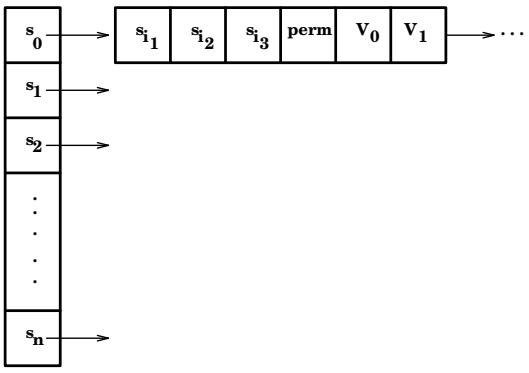


FIG. 7.2 - *Table d'adjacence.*

Génération

Le principe de construction de la table *Adjac* consiste à parcourir tous les éléments de volume. Pour chaque face les constituant, d'ordonner les index des sommets par ordre croissant en conservant une trace de la permutation.
Soit s_i le plus petit index. Les cellules adressées par la case $Adj[s_i]$ sont parcourues jusqu'à l'obtention de la cellule contenant les trois autres sommets, ou jusqu'à la dernière cellule.
Si la cellule n'a pas été trouvée, une nouvelle cellule est rajoutée à la liste. Pour cette cellule, V_0 est positionné à l'index de l'élément de volume courant, V_1 est initialisé à un index d'élément invalide (-1).
Algorithme :

Pour chaque élément de volume **V**
  Pour chaque face **F** de **V**
  ◇ Ordonner sommets; s_i le plus petit.
  ◇ Parcourir la liste des cellules adressées par $Adj[s_i]$.
    • Si la cellule contenant les mêmes sommets existe :
      * Mettre à jour V_1 à l'index de volume courant.
    • Sinon ajouter une nouvelle cellule à la liste.

Trace des permutations

Pour obtenir une table d'adjacence efficace, il est nécessaire d'avoir une représentation canonique des faces. La forme canonique choisie est celle où les index des sommets sont triés par ordre croissant. Pendant le tri, il s'agit de conserver une trace de la permutation pour restituer l'ordre initial et conserver ainsi la géométrie de la face lors d'un réaffichage. Sur la figure 7.3, la face $(1, 9, 2, 4)$ ne peut être affichée correctement en utilisant uniquement la forme canonique $(1, 2, 4, 9)$.

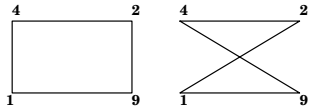


FIG. 7.3 - *Affichage sous forme canonique.*

Pour une face composée de n sommets, il existe $n!$ permutations possibles, soient 24 pour une face de 4 sommets. Soit pour la face (a, b, c, d), la variable *pattern* dont chaque bit représente la comparaison entre index des sommets deux à deux: soient les valeurs: $[a < b, a < c, a < d, b < c, b < d, c < d]$.
Tout le problème consiste à associer la valeur de *pattern* à un index de permutation. Un algorithme de génération automatique a été utilisé pour créer la table des permutations, effectuer le tri et renvoyer un index de pemutation.
Cette méthode génère toutes les combinaisons associées au *pattern* et teste alors la validité des combinaisons. Par exemple, la combinaison 000010 est impossible car elle signifie: $b < a, c < a, d < a, c < b, b < d, d < c$ or cela est contradictoire.

Nous venons de voir comment passer d'un élément de volume à son voisin, grâce à la construction d'une table d'adjacence.
Déterminons maintenant comment choisir un polyèdre de départ afin de localiser le point P recherché.

7.1.4 Choix d'un élément de volume de départ

Pour localiser un point dans un maillage 3D, l'élément de volume de départ est choisi au hasard. Si le point n'est pas localisé dans cet élément, la recherche est réitérée sur l'élément voisin, grâce à la table d'adjacence.
Pour des maillages non convexes, le parcours de la table d'adjacence pour trouver les volumes suivants peut mener à un élément de la frontière. Le point P sera alors jugé inaccessible.
Une méthode utilisée pour éviter ce genre de problème consiste à choisir aléatoirement un nouvel élément de départ et à tester l'appartenance du point à ce volume puis à se déplacer dans les voisins.
Deux techniques ont été implémentées. Un premier algorithme consiste à tirer aléatoirement un élément de volume, puis à se déplacer de voisins en voisins en conservant la trace des éléments testés. Ainsi à chaque nouveau tirage aléatoire, parmi les éléments non encore testés, la recherche du volume contenant le point est poursuivie tant que l'élément n'a pas déjà été testé. Ces tirages aléatoires sont répétés jusqu'à l'obtention de l'élément de volume contenant le point, ou tant que tous les éléments n'ont pas été testés. Cette recherche est exhaustive mais coûteuse.
Une autre méthode consiste à effectuer un nombre très limité de tirages aléatoires sans conserver la trace des éléments testés. Par contre si le point n'est toujours pas localisé, sur demande explicite de l'utilisateur, un parcours exhaustif de tous les éléments est effectué afin de vérifier l'appartenance du point au domaine maillé.
Ces techniques de tirage aléatoire évitent ainsi de trop vite juger un point inaccessible.

Après avoir sélectionné un point à l'intérieur du maillage, départ de la ligne de courant, nous avons choisi un élément de volume de départ dans le maillage. Nous savons également nous déplacer de proche en proche pour localiser le volume contenant le point recherché. Détaillons la méthode pour déterminer si un point est dans un éléments de volume.

7.1.5 Localisation d'un point dans un élément de volume.

Soit un élément de départ, ordonnons les index des sommets de chaque face du volume et déterminons *sop* le sommet opposé à chaque face.
Soit l'équation du plan passant par chacune des faces, il s'agit de tester si le point P recherché et le sommet opposé *sop* à une face sont du même côté du plan. Le point est considéré comme appartenant au volume si cette propriété est vérifiée pour toutes les faces. Sinon on détermine, grâce à la table d'adjacence, l'élément de volume voisin (contenant également la face courante).
Algorithme de localisation :
Données : M maillage, *Adj* table d'adjacence de M et P point cible.
Sortie :V_p élément de volume contenant le point cible, ou erreur indiquant le point inaccessible.

1) Choisir un élément V de M aléatoirement.
◇ 2) Pour toutes les faces f_i de V :
- Ordonner les sommets des faces.
- $sop \leftarrow$ sommet de V opposé à f_i.
- Si P et *sop* sont du même côté de la face f_i, continuer.
- Sinon $V' \leftarrow$ voisin de V de face commune f_i.
  - ⋆ Si V' existe, $V \leftarrow V'$, retour en (2).
  - ⋆ Sinon continuer.

◇ Si P et *sop* sont du même côté pour toutes les faces f_i
- $V_p \leftarrow V$, fin.

◇ Sinon point jugé inaccessible, retour en (1).

7.2 Interpolation dans un élément de volume

Une fois l'élément de volume localisé, il faut effectuer l'interpolation des valeurs vectorielles connues aux nœuds.
Dans le cas des tétraèdres, une interpolation linéaire fournit la solution exacte. Chaque vecteur v au sommet du polyèdre peut être considéré comme une combinaison linéaire des coordonnées du sommet: $v_i = ax_i + by_i + cz_i + d$,

$i \in [0, n]$ avec n: le nombre de sommets.
Pour les autres types d'éléments de volume, deux méthodes sont comparées.

7.2.1 Décomposition en tétraèdres

Après avoir déterminé le barycentre G ($\frac{\sum_{i=1}^{n} S_i}{n}$ avec S_i sommet i) du polyèdre, le volume est décomposé en tétraèdres, contenant tous G comme sommet. Une recherche est à nouveau effectuée pour localiser quel tétraèdre contient le point P. Une interpolation des valeurs connues aux nœuds du tétraèdre est alors réalisée.
Un héxaèdre est décomposé en 12 tétraèdres, un prisme en 8 tétraèdres, une pyramide en 6 tétraèdres. Les tables d'adjacence sont également construites pour ces décompositions.

7.2.2 Interpolation trilinéaire dans un héxaèdre

L'autre méthode consiste à effectuer directement l'interpolation dans un hexaèdre. Elle est la généralisation au 3D de la méthode 2D de D. Seldner et T. Westermann [SW88].
Soit f une application de R^3 dans R. On suppose d'abord la valeur de f connues aux sommets du cube C qui a 3 de ses arêtes confondues avec les vecteurs unitaires du repère orthonormé utilisé. On référencera les sommets de C en utilisant les index de la figure 7.2.2, schéma 1.

Il s'agit de trouver une méthode d'évaluation de la valeur de f en un point P intérieur à C. Désignons par $\alpha_1, \alpha_2, \alpha_3$ les coordonnées du point P.

Une interpolation linéaire des valeurs de f le long des arêtes parallèles à l'axe des x, puis, dans le carré obtenu, le long des arêtes parallèles à l'axe des y, et enfin le long du segment obtenu, comme schématisé sur la figure 7.2.2 (schéma 2), fourni l'évaluation de la valeur de f au point P suivante:

$$\begin{aligned} f_e(P) = (1-\alpha_3)\ &[(1-\alpha_2)((1-\alpha_1)f_{000} + \alpha_1 f_{100}) + \alpha_2(1-\alpha_1)f_{010} + \alpha_1 f_{110})] \\ + \alpha_3\ &[(1-\alpha_2)((1-\alpha_1)f_{001} + \alpha_1 f_{101}) + \alpha_2(1-\alpha_1)f_{011} + \alpha_1 f_{111})] \end{aligned}$$

Cette évaluation de f au point P peut aussi se formuler comme une pondération des valeurs connuees aux sommets par le volume du parallèpipède

rectangle, délimité par les faces du cube et les plans parrallèles aux faces passant par P, opposé au sommet (figure 7.2.2, schéma 3):

$$\begin{aligned} f_e(P) = (1-\alpha_1)(1-\alpha_2)(1-\alpha_3)\ f_{000} \\ +\alpha_1(1-\alpha_2)(1-\alpha_3)\ f_{100} \\ +(1-\alpha_1)\alpha_2(1-\alpha_3)\ f_{010} \\ +\alpha_1\alpha_2(1-\alpha_3)\ f_{110} \\ +(1-\alpha_1)(1-\alpha_2)\alpha_3\ f_{001} \\ +\alpha_1(1-\alpha_2)\alpha_3\ f_{101} \\ +(1-\alpha_1)\alpha_2\alpha_3\ f_{011} \\ +\alpha_1\alpha_2\alpha_3\ f_{111} \end{aligned} \tag{7.1}$$

Par définition de cette méthode d'interpolation, pour toute fonction f linéaire suivant chaque composante de P, la valeur interpolée est la valeur exacte:

Pour $f = ax+by+cz+dxy+eyz+fzx+gxyz+h$, $f_e(P) = f(P)$, $\forall P \in C$

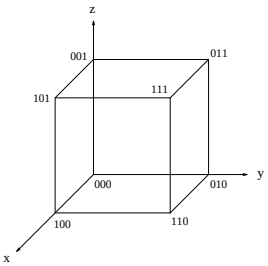


Schéma 1.

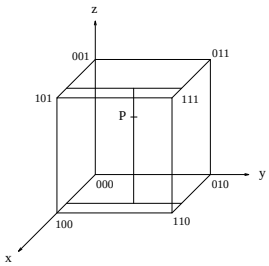


Schéma 2.

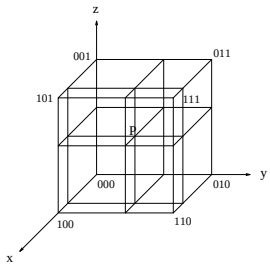


Schéma 3.

On suppose maintenant la fonction f connue aux sommets d'un hexaèdre quelconque H, pour lequel les sommets seront référencés par des index, suivant le même ordonnancement que ceux du cube C pour faciliter une mise en correspondance, comme représenté sur la figure 7.2.2 schéma 4. Soit P_h un point de composantes x, y, z à l'intérieur de H. Supposons que l'on sache associer à P_h un point P dans C de composantes $\alpha_1, \alpha_2, \alpha_3$ par une transformation qui applique les sommets de H sur les sommets de C référencés par les mêmes index, on pourrait alors évaluer la valeur de f au point P_h en utilisant la pon-

dération des valeurs de f connues aux sommets de H suivant la formulation définie par (7.1).

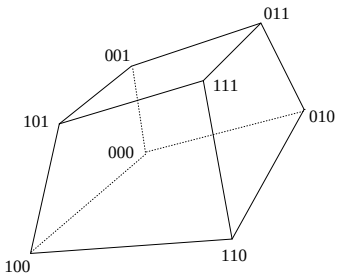


Schéma 4.

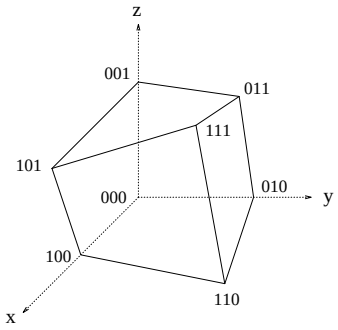


Schéma 5.

On va de plus imposer que la transformation qui permet de passer de H à C soit telle que pour toute fonction f de la forme: $f(P_h) = ax + by + cz + d$ la valeur de f calculée au point P_h par cette méthode d'interpolation soit exacte, ce qui revient à dire que:

$$\begin{aligned} ax + by + cz + d = (1-\alpha_1)(1-\alpha_2)(1-\alpha_3)\ & (ax_{000} + by_{000} + cz_{000} + d) \\ +\alpha_1(1-\alpha_2)(1-\alpha_3)\ & (ax_{100} + by_{100} + cz_{100} + d) \\ +(1-\alpha_1)\alpha_2(1-\alpha_3)\ & (ax_{010} + by_{010} + cz_{010} + d) \\ +\alpha_1\alpha_2(1-\alpha_3)\ & (ax_{110} + by_{110} + cz_{110} + d) \\ +(1-\alpha_1)(1-\alpha_2)\alpha_3\ & (ax_{001} + by_{001} + cz_{001} + d) \\ +\alpha_1(1-\alpha_2)\alpha_3\ & (ax_{101} + by_{101} + cz_{101} + d) \\ +(1-\alpha_1)\alpha_2\alpha_3\ & (ax_{011} + by_{011} + cz_{011} + d) \\ +\alpha_1\alpha_2\alpha_3\ & (ax_{111} + by_{111} + cz_{111} + d) \end{aligned}$$

doit être vérifié, $\forall a, b, c, d$.

L'identification des coefficients de a, b et c permet d'établir la relation entre les composantes x, y, z de P_h et $\alpha_1, \alpha_2, \alpha_3$ de P qui définie la transformation qui permet de passer de H à C.

Pour simplifier la formulation de cette relation, supposons que, par une translation, le point sommet d'index 000 de H soit ramené à l'origine. Désignons par P_{xxx} le vecteur $\begin{pmatrix} x_{xxx} \\ y_{xxx} \\ z_{xxx} \end{pmatrix}$ ayant pour composantes les coordonnées

du sommet d'index xxx de H après avoir subi cette translation, et par P le vecteur ayant pour composantes les coordonnées du point P_h translaté.

La transformation qui permet de passer du H translaté à C se formule alors comme suit:

$$\begin{aligned}
&\alpha_1\alpha_2\alpha_3(P_{100} + P_{010} - P_{110} + P_{001} - P_{101} - P_{011} + P_{111}) \\
&+\alpha_1\alpha_2(-P_{100} - P_{010} + P_{110}) \\
&+\alpha_2\alpha_3(-P_{010} - P_{001} + P_{011}) \\
&+\alpha_3\alpha_1(-P_{100} - P_{001} + P_{101}) \\
&+\alpha_1 P_{100} + \alpha_2 P_{010} + \alpha_3 P_{001} \qquad\qquad = P
\end{aligned}$$

On va réaliser une transformation affine intermédiaire qui permettra d'appliquer les 3 arêtes issues du sommet d'index 000 sur les 3 vecteurs unitaires du repère orthonormé (figure 7.2.2 schéma 5).

Définissons pour cela la matrice M:

$$M = \begin{pmatrix} x_{100} & x_{010} & x_{001} \\ y_{100} & y_{010} & y_{001} \\ z_{100} & z_{010} & z_{001} \end{pmatrix}$$

et désignons par P^s le vecteur solution de $MP^s = P$.

La transformation qui reste à réaliser pour passer au cube C se formule alors comme suit:

$$\begin{aligned}
&\alpha_1\alpha_2\alpha_3(\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - P^s_{110} - P^s_{101} - P^s_{011} + P^s_{111}) \\
&+\alpha_1\alpha_2(P^s_{110} - \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}) + \alpha_2\alpha_3(P^s_{011} - \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}) + \alpha_3\alpha_1(P^s_{101} - \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}) \\
&+\alpha_1 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \alpha_2 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + \alpha_3 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \qquad\qquad = P^s
\end{aligned}$$

Si ce système admet une solution à l'intérieur du cube C, on peut considérer que le point P_h était effectivement à l'intérieur de H. Ceci pourrait être considéré comme une définition de l'intérieur de H.

Une méthode efficace de résolution de ce système consiste, en partant d'une valeur initiale au centre du cube, à effectuer des itérations en utilisant une méthode de Newton pour la première équation, relativement à la première inconnue, puis pour la deuxième équation, relativement à la deuxième inconnue et enfin pour la dernière. Dans le cas d'un hexaèdre assez régulier, une seule itération suffit pour une précision largement suffisante pour réaliser l'interpolation des valeurs de f connues aux sommets de H.

Cette méthode peut être explicitée en posant:

$$\begin{aligned}
\alpha^i &= \begin{pmatrix} \alpha_1^i \\ \alpha_2^i \\ \alpha_3^i \end{pmatrix} \\
a &= \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - P^s_{110} - P^s_{101} - P^s_{011} + P^s_{111} \\
b &= \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = P^s_{110} - \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \\
c &= \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = P^s_{011} - \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \\
d &= \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} = P^s_{101} - \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}
\end{aligned}$$

après l'initialisation au centre de C:

$$\alpha^0 = \begin{pmatrix} .5 \\ .5 \\ .5 \end{pmatrix}$$

Le calcul de α^{i+1} pourra se faire à partir de α^i comme suit:

$$f_1^i = \alpha_1^i\alpha_2^i\alpha_3^i a_1 + \alpha_1^i\alpha_2^i b_1 + \alpha_2^i\alpha_3^i c_1 + \alpha_3^i\alpha_1^i d_1 + \alpha_1^i - x^s$$

$$d_1^i = \alpha_2^i\alpha_3^i a_1 + \alpha_2^i b_1 + \alpha_3^i d_1 + 1$$

$$\alpha_1^{i+1} = \alpha_1^i - f_1^i/d_1^i$$

$$f_2^i = \alpha_1^{i+1}\alpha_2^i\alpha_3^i a_2 + \alpha_1^{i+1}\alpha_2^i b_2 + \alpha_2^i\alpha_3^i c_2 + \alpha_3^i\alpha_1^{i+1} d_2 + \alpha_2^i - y^s$$

$$d_2^i = \alpha_1^{i+1}\alpha_3^i a_2 + \alpha_1^{i+1} b_2 + \alpha_3^i c_2 + 1$$

$$\alpha_2^{i+1} = \alpha_2^i - f_2^i/d_2^i$$

$$f_3^i = \alpha_1^{i+1}\alpha_2^{i+1}\alpha_3^i a_3 + \alpha_1^{i+1}\alpha_2^{i+1} b_3 + \alpha_2^{i+1}\alpha_3^i c_3 + \alpha_3^i\alpha_1^{i+1} d_3 + \alpha_3^i - z^s$$

$$d_3^i = \alpha_1^{i+1}\alpha_2^{i+1} a_3 + \alpha_2^{i+1} c_3 + \alpha_1^{i+1} d_3 + 1$$

$$\alpha_3^{i+1} = \alpha_3^i - f_3^i/d_3^i$$

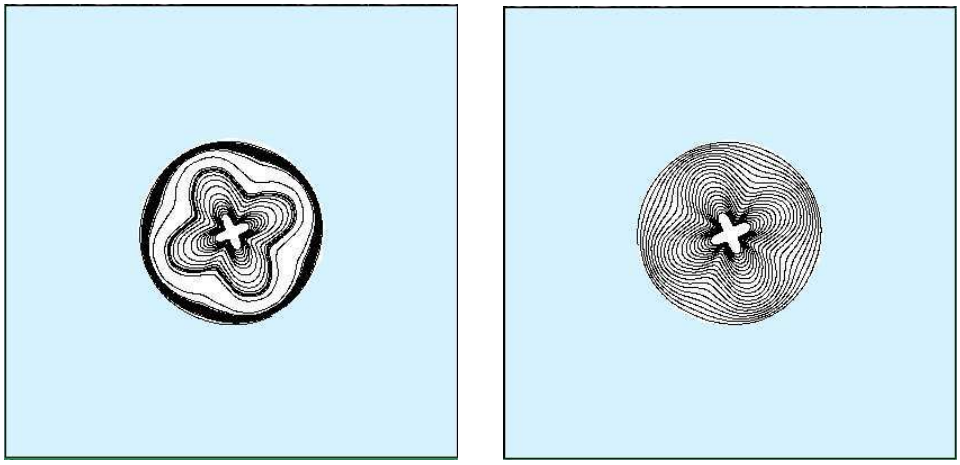
Dans l'implémentation réalisée, le critère d'arrêt de cette itération utilisé porte sur la valeur absolue des fonctions f_1^i, f_2^i et f_3^i. Lorsque chacune d'elle est inférieure à 0.0001 on considère que le α^i obtenu permet de localiser le point P avec une précision suffisante pour l'interpolation trilinéaire de la fonction f. On limite le nombre total d'itérations et on controle l'appartenance du point P obtenu au cube unitaire C, en acceptant également un voisinage du cube du fait que l'algorithme de localisation qui a permit de trouver l'hexaèdre H qui contient le point P_h n'utilise pas le même critère d'intérieur pour un hexaèdre.

Dans le cas où le α^i obtenu ne permet pas de satisfaire au critère d'arrêt des itérations ou défini un point P n'appartenant pas au cube C ou à son voisinage immédiat, on utilise comme valeur estimée de la fonction f au point P la moyenne des valeurs connues aux sommets de l'hexaèdre.

Résultats

La figure suivante 7.2.2 compare une interpolation dans un cylindre grossièrement maillé (formé de prismes et d'héxaèdres) selon les deux méthodes

précedemment décrites. On remarque que l'interpolation trilinéaire dans un héxaèdre est plus régulière que celle de décomposition en tétraèdres.



Figures: Ligne d'écoulement obtenue par décomposition en tétraèdres ou par interpolation trilinéaire d'un héxaèdre. Projection sur le plan Oxy.

Nous avons décrit le moyen de visualiser le premier point de la ligne de courant. Il a fallu tout d'abord choisir l'élément de volume de départ, puis déterminer s'il contenait le point. Dans le cas contraire, la construction d'une table d'adjacence a permis de trouver le volume voisin. Enfin, une fois l'élément localisé, une interpolation des valeurs connues aux nœuds du volume a été nécessaire pour définir la valeur en ce point.
Déterminons maintenant comment calculer les points suivants afin de construire la ligne de courant.

7.3 Calcul du point suivant sur la ligne de courant.

H.J. Neeman ([Nee90]) a comparé trois méthodes pour construire une ligne de courant ; ce sont les méthodes d'Euler, de Runge-Kuntta au 4ème ordre (RK4) et une méthode itérative de correction-prédiction.

Il s'agit en fait d'intégrer l'équation différentielle ordinaire: $dX/dt = \vec{V}$.
Soit $P : (x, y, z)$ la position d'une particule au temps t.
Soit $V : (v_x, v_y, v_z)$ la valeur vectorielle en tout point P notée $V(P)$.
Déterminons $P' : (x', y', z')$ la position de la particule au temps $t + \Delta t$.

méthode d'Euler:
$P' = P + V(P)\Delta t$.

méthode RK4:
$k_1 = P\Delta t$;
$k_2 = V(P + \frac{1}{2}k_1)\Delta t$;
$k_3 = V(P + \frac{1}{2}k_2)\Delta t$;
$k_4 = V(P + k_3)\Delta t$;
$P' = P + \frac{(k_1+2k_2+2k_3+k_4)}{6}$.

méthode itérative:
$P' = P + \frac{1}{2}(V(P) + V(P'))\Delta t$
Il s'agit en fait de la résolution du schéma implicite ce Crank-Nicholson.

La méthode d'Euler est simple et efficace. Elle ne nécessite qu'une seule interpolation par position de particule mais c'est celle qui provoque la plus grande erreur sur un intervalle donné.
La méthode de RK4 fournit la meilleure précison (de l'ordre de δt^4) mais nécessite quatre interpolations.
La méthode itérative fournit une précision de l'ordre de δt^2 mais nécessite une bonne estimation du critère de convergence pour éviter des itérations superflues.
Nous avons choisi d'implémenter les deux premières méthodes : Euler et RK4.

7.4 Résultats

Quelques images représentant des lignes de courant (figure 7.4):

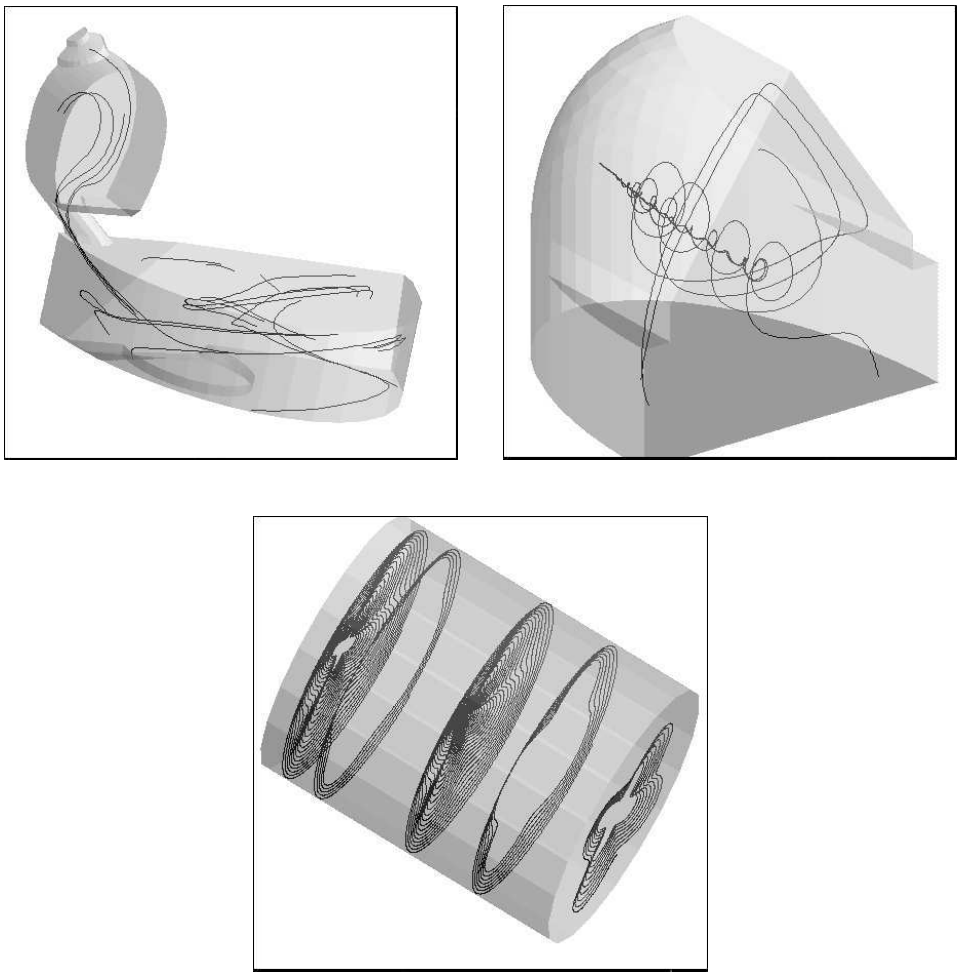


FIG. 7.4 - *Lignes de courant.*

La représentation de courbes en 3D pose des problèmes d'interprétation. Les lignes de courant sont regroupées pour former des surfaces permettant une meilleure perception de l'écoulement. Le chapitre suivant (8) présente différentes méthodes de calcul de surface d'écoulement.

Chapitre 8

Surfaces de courant

Une ligne de courant est une courbe tangente en tous points au vecteur vitesse. Une surface de courant est une surface contenant plusieurs lignes de courant. Différentes méthodes ont été décrites pour constituer un pavage qui représente au mieux une surface de courant.

8.1 Études de différents algorithmes

La méthode classique consiste à paver la surface de façon régulière. Des polygones sont construits avec autant de points pris sur chacune des deux lignes adjacentes, [Be87]. Ainsi M.W. Krueger [Kr92] construit des surfaces formées de paires de lignes adjacentes, avec un pavage régulier, mais en tronquant les polygones excédant une certaine largeur.
Dans certains types d'écoulement, les champs de vecteur divergent beaucoup. Cette déformation complique l'approximation polygonale des surfaces. La divergence des écoulements provoque la séparation des lignes adjacentes.
Dans de tels cas, le pavage régulier n'est pas une bonne solution ; en effet les polygones peuvent être très larges et ils représentent alors mal la courbure de la surface, (cf figure 8.1). La méthode de troncature de Krueger [Kr92] produit une meilleure distribution des points sur la surface.
Un autre problème est posé par les écoulements cisaillés : les lignes de courant adjacentes avancent à des vitesses différentes. La plus lente nécessite plus de pas de temps. Dans ce genre d'écoulements, la méthode de pavage régulier n'est

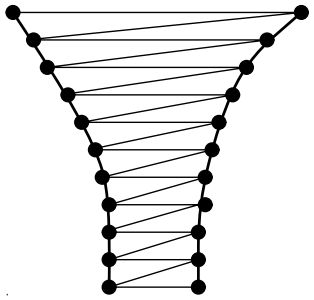


FIG. 8.1 - *Pavage régulier dans un écoulement divergent.*

pas non plus adaptée, car la connection entre les points répartis régulièrement sur deux lignes adjacentes provoque de longs triangles pointus (cf figure 8.2); l'idéal étant bien sûr des triangles isocèles, voire équilatéraux.

M. Siclari [Si89] tente de régler ce problème en persistant avec un pavage

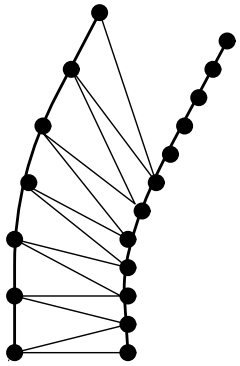


FIG. 8.2 - *Pavage régulier dans un écoulement cisaillé.*

régulier. Il propose de normaliser les courbes par une longueur d'arc imposée. Mais un nouveau problème surgit dans le cas d'écoulements hélicoïdaux : la courbe la plus proche de l'axe sera plus droite que sa partenaire et les triangles aussi pointus.

Le pavage régulier semble donc mal adapté à la construction de telles surfaces de courant.

Un pavage de type irrégulier résoud les problèmes décrits précedémment. La construction de la surface est plus coûteuse mais les triangles du pavage se

rapprochent de triangles isocèles.
La surface qui est globalement minimale peut être obtenue par la méthode de Fuchs, Kedem et Uselton [FKU77]. L'algorithme construit un pavage minimal à partir de deux ensembles de points. Helman et Hesslink [HH89, HH91] s'en servent également pour construire des surfaces d'aire totale minimale.
L'algorithme classique de pavage minimal construit toutes les courbes avant de pouvoir créer le pavage. Hultquist [Hu92] propose un algorithme permettant de construire la surface au fur et à mesure. Les résultats sont identiques mais de plus faible coût. Tandis que l'algorithme classique stocke une courbe entière, tant que ses voisins ne sont pas connus avant de construire la surface, la méthode de Hultquist conserve seulement quelques points le long de chaque courbe. Le pavage est alors progressif le long de chaque courbe. La surface est bien construite et l'accès aux données est local, donc plus rapide. De plus de nouveaux points sont facilement insérés quand les points sur la courbe sont trop éparpillés, ou encore des points peuvent être enlevés dans le cas d'un échantillonnage trop dense.
Le maintien d'un front de particules est une meilleure méthode pour générer la surface, plutôt que le calcul à priori des courbes puis le pavage.

Nous nous proposons de comparer les deux méthodes de construction d'une surface de courant : par un pavage régulier ou par un pavage minimal.
Donnons tout d'abord la définition géométrique de la surface.

8.2 Définition de la surface

La construction d'une surface peut être réalisée par une génération de points bien échantillonnés, puis par un pavage polygonal de ces points.
Une surface d'écoulement idéale est la combinaison d'une infinité de lignes de courant. Une surface est définie à la base par le déplacement d'un point le long d'une ligne initiale, ($s \in [0..n]$)au temps $t = 0$. Chaque ligne de courant s_i est paramétrée par $t \in [0..m]$. Un point de la surface est alors référencé par ses deux paramètres (s, t), (cf figure 8.3).

Pour chaque point (s_i, t_j), la première coordonnée définit un déplacement le

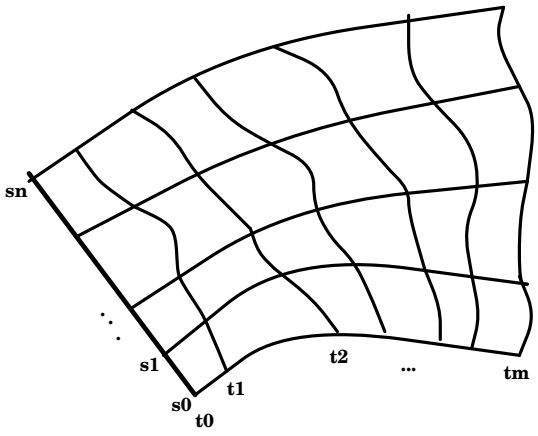


FIG. 8.3 - *Une surface d'écoulement.*

long de la ligne au temps t_j, tandis que la seconde coordonnée définit l'intégration accumulée le long de la courbe (ligne de courant).
La surface peut être approximée numériquement par des polygones. L'implémentation la plus naturelle consiste à créer un maillage rectangulaire de points (s, t) pris à intervalles réguliers.
Le choix du nombre de lignes de courant constituant un maillage rectangulaire régulier peut être satisfaisant dans le voisinage des points initiaux, mais devenir insuffisant ou superflu au cours de la construction de la surface. Une représentation plus efficace de la surface nécessite un échantillonage adaptatif des points. La densité d'échantillonage le long de chaque ligne de courant peut être contrôlée par un ajustement du pas d'intégration ; de même les lignes de temps (t_j fixé) peuvent être paramétrables afin de mieux représenter les courbures de la surface.
Le découpage de surfaces augmente également la densité d'échantillonage le long de la surface polygonale. Les méthodes définies par Helman, Krueger et Hultquist sont basées sur ce principe de découpage.

La surface est donc construite à partir de points choisis le long de deux courbes adjacentes, en produisant un nouveau triangle à chaque itération.

Avant toute chose il faut définir la méthode de construction de la ligne initiale : départ de toutes les lignes d'écoulement qui vont constituer la surface.

8.3 Choix de la ligne de départ

La plupart des modules de visualisation permettent à l'utilisateur de spécifier des points initiaux, départ des courbes. Cet ensemble de points porte le nom de "rake" par analogie aux particules de fumée introduites dans les souffleries lors de la visualisation expérimentale.
Une attention particulière doit être apportée au positionnement de la ligne de départ afin de suivre au mieux le comportement du fluide.
Nous avons choisi de définir une ligne initiale paramétrable pour correspondre aux différents types d'écoulement.
Une ligne de départ L est constituée de n points, distants chacun d'un pas e sur un droite dont on a défini la position dans l'espace préalablement. Nous n'avons cependant pas trouvé de méthode satisfaisante pour permettre à l'utilisateur de positionner interactivement cette ligne de départ.

Une méthode consiste à choisir un point de départ P_i (premier point de la première ligne de courant), à le localiser et à récupérer la valeur d'interpolée du vecteur vitesse dans le maillage tri-dimensionnel (comme décrit précédemment 7).
La ligne de départ choisie est perpendiculaire à la ligne de courant issue de P_i.

Présentons tout d'abord la méthode de pavage régulier.

8.4 Pavage régulier

Dans une surface constituée d'un pavage régulier, autant de points sont choisis sur chacune des lignes d'écoulement. Cette surface est constituée des n points de la ligne de départ et de m points sur chaque ligne d'écoulement.
La surface, initialisée tout d'abord à la ligne de départ, progresse par "bandes":
Pour chaque point de la ligne de départ (s_i, t_0), un calcul du point suivant

(s_i, t_1) sur la ligne d'écoulement correspondante est effectué selon la méthode décrite en 7, et ceci pour m points sur les lignes d'écoulement (cf figure 8.3).
Algorithme:

Pour tous les points t_j, j de 1 à m
- ◇ Pour chaque point s_i, i de 1 à n
  - • Calcul du point s_i, t_j, de la ligne d'écoulement i.
- ◇ Affichage de quadrangles par bandes t_{j-1}, t_j

8.5 Pavage minimal

8.5.1 Méthode de base

Cette technique s'inspire de l'algorithme de Hultquist [Hu92]. La surface est formée à partir de plusieurs lignes de courant ; les éléments de la surface sont construits à partir de courbes adjacentes prises deux par deux. Deux lignes adjacentes pavées en triangles forment un ruban.
Le pavage commence à la ligne de départ. Pour construire le premier ruban, à partir de deux points sur la ligne de départ, deux nouveaux points sont calculés sur les lignes de courant respectives selon la méthode classique (chapitre 7). La plus courte diagonale du quadrilatère ainsi formé constitue l'arête conductrice du triangle conservé (cf figure 8.4).
Ce mécanisme est répété pour construire chaque ruban.
La surface est construite au fur et à mesure : au temps t_k, tous les points (s_i, t_j) ont été calculés ($i \in [0..n], j \in [0..k]$).
L'algorithme étant récursif, il est nécessaire de garder un contrôle sur le nombre de points de chaque ligne de courant, tandis que le front de particules avance. Hultquist a choisi de construire la surface à partir de la ligne de courant la plus à gauche, mais le problème est symétrique. Ainsi le contrôle du nombre de points sur la ligne s'effectue sur la courbe de gauche; la récursion s'effectue sur la courbe de droite. La surface se construit alors au fur et à mesure.
L'appelation "gauche-droite" est purement conventionnelle. L'itération commence avec la ligne la plus à gauche. À chaque niveau de récursivité, pas plus d'un point n'est calculé à gauche afin de garder un contrôle sur le nombre de points des courbes. Ainsi l'avancée récursive à droite ne dépassera pas le point

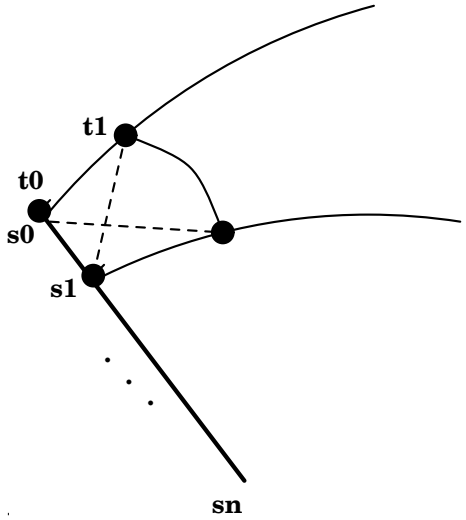


FIG. 8.4 - *Pavage minimal.*

gauche.

Les points constituant le pavage sont stockés dans le tableau *surface*. Le tableau *ruban* définit la connectivité, en indexant les points de $surface$ formant chaque triangle. $front$ est un tableau contenant les index des plus hauts points sur chaque ligne de courant : cela représente le front avançant de particules. $frontSuivant$ contient les index des points suivants (sur la ligne de courant) du front. Ces deux tableaux $front$ et *frontSuivant* permettent d'éviter de recalculer ou de stocker plusieurs fois les points communs à plusieurs rubans. La $surface$ ainsi que le $front$ sont initialisés à la ligne de départ ; le $frontSuivant$ est rempli avec des valeurs invalides.
Une boucle est itérée pour contrôler le nombre m de points sur une ligne de courant. L'algorithme démarre avec la ligne de courant "la plus à gauche", soit la ligne 0 (ln_0) formant le ruban 0.
Notons g_0 l'index du point gauche sur la ligne de courant au temps t_i et calculons g_1 l'index du point gauche au temps t_{i+1}.
La fonction *construireRuban* prend comme arguments les points g_0, g_1, ainsi que les tableaux $front$, $frontSuivant$ et le numéro du ruban courant. Les index

des points "droits" d_0, d_1 sont récupérés ($d_0 \leftarrow front[ln_i]$) et calculés ($d_1 \leftarrow$ point suivant sur la ligne d'écoulement).
Soient g_0, g_1, d_0, d_1, les points servant à la construction d'un nouvel élément de surface (cf figure 8.5).

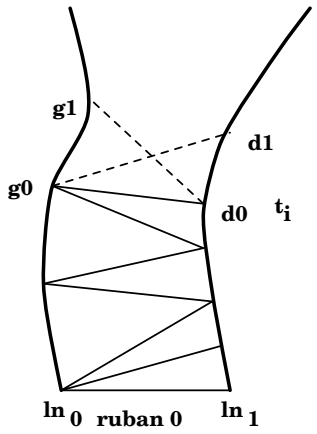


FIG. 8.5 - *Notations pour un pavage minimal.*

Afin de déterminer quel triangle va constituer le pavage du ruban, les diagonales (g_0, d_1) et (d_0, g_1) sont comparées. Le pavage étant minimal, la diagonale la plus courte forme un triangle du ruban. Il s'agit ensuite de progresser dans le calcul des points sur les lignes de courant, et de poursuivre l'avancée du front de particules.

◇ Si la diagonale minimum est celle de gauche : (d_0, g_1), le triangle (g_0, d_0, g_1) est affiché, et la fonction "construireRuban" est stoppée. La boucle sur le nombre de points sur chaque courbe prend alors le relai avec g_1 comme point gauche et une remise à jour de $front$ et $frontSuivant$.

◇ Si, par contre, la diagonale minimum est celle de droite : (g_0, d_1), le triangle (g_0, d_0, d_1) est affiché. d_0 devient d_1 ; d_1 est calculé, $front$ et $frontSuivant$ sont également remis à jour. Le calcul des diagonales minimales avec création d'un nouvel élément est effectué tant qu'une diagonale minimum gauche n'est pas rencontrée. De plus si le ruban courant n'est pas le dernier, la fonction

construireRuban est rappelée avec les points (d_0, d_1) qui forment les points gauches du ruban suivant ; $front$ et $frontSuivant$ sont également remis à jour ; ceci afin de poursuivre l'avancée du front dans les rubans suivants.
Algorithme de construction d'une surface minimale:

- $\diamond$ $front \leftarrow$ index des points de la ligne de départ.
- $\diamond$ $frontSuivant \leftarrow$ index invalides.
- $\diamond$ $surface \leftarrow$ points de la ligne de départ.
- $\diamond$ $g_0 \leftarrow front[ln_0]$.
- $\diamond$ Boucle sur les m points d'une ligne de courant.
  - Si $frontSuivant[ln_0]$ invalide, calcul de g_1.
  - $construireRuban(g_0, g_1, front, frontSuivant, ln_0)$.
  - $g_0 \leftarrow front[ln_0]$.
  - $frontSuivant[ln_0] \leftarrow$ index invalide.

$construireRuban(g_0, g_1, front, frontSuivant, ln_i)$.

- $\diamond$ $d_0 \leftarrow front[ln_{i+1}]$.
- $\diamond$ Si $frontSuivant[ln_{i+1}]$ invalide, calcul de d_1.
- $\diamond$ 1) Itérer :
  - diag-min $\leftarrow$ minimum des diagonales (g_0, d_1) et (d_0, g_1).
  - Si diag-min = gauche
    - $\star$ Écriture du triangle (g_0, d_0, g_1).
    - $\star$ $front[ln_i] \leftarrow g_1$, $frontSuivant[ln_i]$ invalidé.
    - $\star$ arrêt.
  - Sinon (diag-min = droite)
    - $\star$ Si $ln_i \neq$ dernier-ruban
      construireRuban$(d_0, d_1, front, frontSuivant, ln_{i+1})$.
    - $\star$ Écriture du triangle (g_0, d_0, d_1).
    - $\star$ $d_0 \leftarrow d_1$.
    - $\star$ Si $frontSuivant[ln_{i+1}]$ invalide, calcul de d_1.
    - $\star$ Continuer l'itération 1).

Pour maintenir une densité d'échantillonnage raisonnable, des particules doivent être ajoutées ou enlevées. Par exemple dans des écoulements divergents, les

rubans doivent être séparés pour raffiner l'approche polygonale de la surface idéale.

8.5.2 Ajout de particules

Un ruban peut être divisé par l'insertion d'une particule ; un triangle est ainsi rajouté (cf figure 8.6).

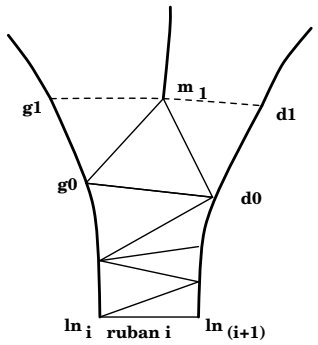


FIG. 8.6 - *Division d'un ruban dans un pavage minimal.*

Le test pour déterminer quand insérer un point doit être efficace car il est évalué à chaque itération. Des méthodes qui mesurent la courbure du front peuvent être employées, mais une méthode simple (employée par Hultquist) qui teste si la largeur du quadrilatère est deux fois supérieure à la hauteur est satisfaisante.
La nouvelle particule m_1 ajoutée doit avoir une position dans l'espace entre les deux particules (g_1, d_1). La méthode la plus simple consiste à effectuer une interpolation au milieu de (g_1, d_1).
Une autre méthode consiste à résoudre le problème de localisation d'un point et interpolation pour $s_i = milieu(g_1, d_1), t_0$. Une solution de compromis consiste à sauver un point $s_i = milieu(g_1, d_1), t_j$ toutes les dix itérations par exemple. Ainsi quand un ruban doit être divisé, une des positions récentes peut être avancée pendant une courte période. Celà permettrait de mieux localiser les zones de divergence.

8.5.3 Retrait de particules

Quand le nombre d'échantillons sur des rubans est trop dense, il est possible de retirer certaines particules non nécessaires à la compréhension de la surface. Deux rubans adjacents sont alors regroupés en un seul ; trois triangles sont alors créés afin d'obtenir un seul ruban (cf figure 8.7).

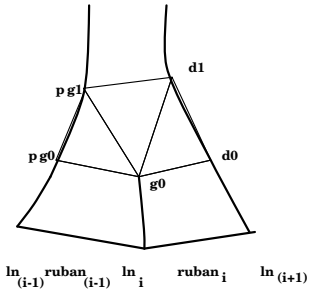


FIG. 8.7 - *Fusion d'un ruban dans un pavage minimal.*

Le test pour déterminer quand regrouper deux lignes doit être efficace car il est évalué à chaque itération. Hultquist propose de décider de retirer des particules quand six points sont coplanaires, et que leur hauteur est plus importante que leur largeur. Ce critère étant coûteux à vérifier, nous avons utilisé le test plus simple qui vérifie si les trois points pg_0, g_0, d_0 sont alignés et que la hauteur est deux fois supérieure à la largeur des deux rubans.

Algorithme général de construction de rubans avec ajout, retrait de particules :

construireRuban$(g_0, g_1, front, frontSuivant, ln_i)$.

- ◇ $d_0 \leftarrow front[ln_{i+1}]$.
- ◇ Si $frontSuivant[ln_{i+1}]$ invalide, calcul de d_1.

Division de rubans.

- ◇ Si $(g_0, d_0) > 2 * max(g_0, g_1), (d_0, d_1))$
  - • Calcul de m_1, écriture du triangle (g_0, d_0, m_1).
  - • Insertion et décallage à droite de $front, frontSuivant[ln_{i+1}]$.
  - • $d_0 \leftarrow front[ln_{i+1}]$.
  - • Si $frontSuivant[ln_{i+1}]$ invalide, calcul de d_1.

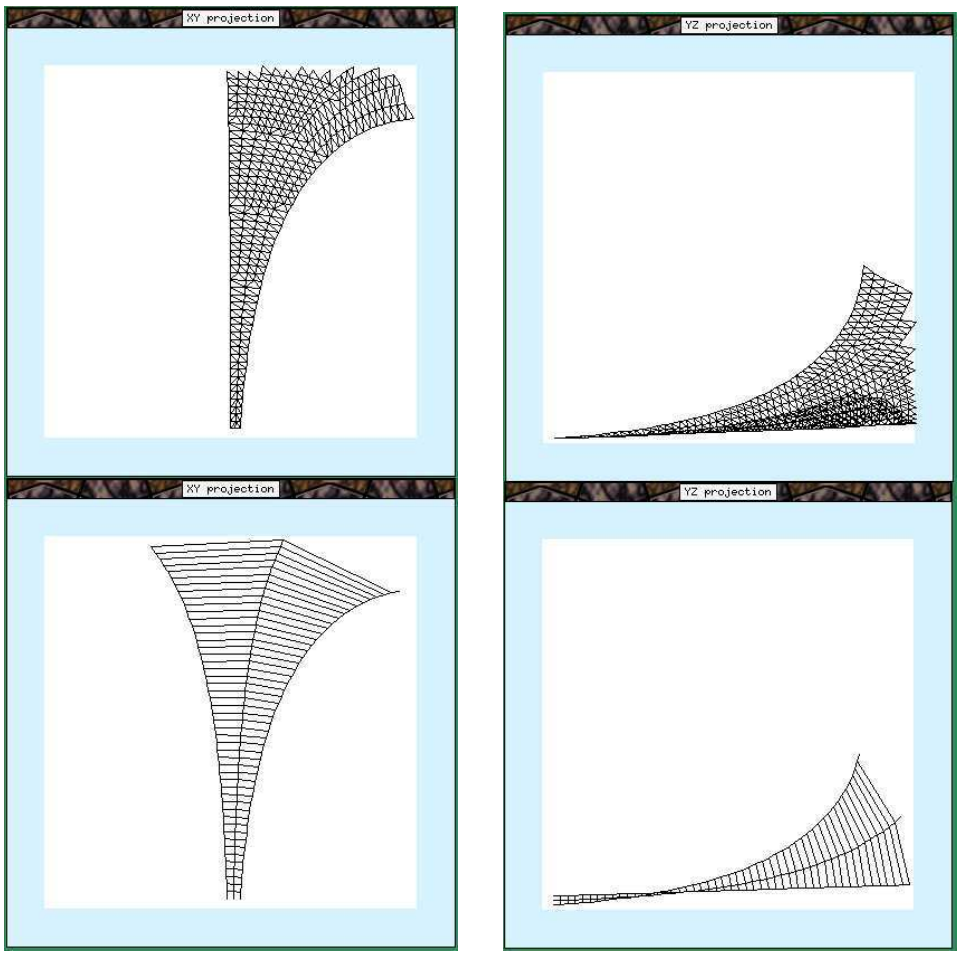
Séparation de rubans.

- ◇ Si ce n'est pas le premier ruban
  - • $pg_0 \leftarrow front[ln_{i-1}]$, $pg_1 \leftarrow frontSuivant[ln_{i-1}]$
  - • Si pg_1, g_1, d_1 alignés et $(g_0, g_1) > 2 * ((pg_0, g_0) + (g_0, d_0))$
    - ⋆ Écriture des triangles (pg_0, g_0, pg_1), (g_0, d_0, d_1), (g_0, pg_1, d_1).
    - ⋆ Retrait et décallage à gauche de $front, frontSuivant[ln_i]$.
    - ⋆ $front[ln_{i-1}] \leftarrow pg_1, front[ln_i] \leftarrow d_1$.
    - ⋆ $frontSuivant[ln_{i-1}], [ln_i]$ invalidés.
    - ⋆ $fusion \leftarrow$ vrai, retour au niveau de récursivité précédent.
- ◇ 1) Itérer :
  - • diag-min ← minimum des diagonales (g_0, d_1) et (d_0, g_1).
  - • Si diag-min = gauche
    - ⋆ Écriture du triangle (g_0, d_0, g_1).
    - ⋆ $front[ln_i] \leftarrow g_1, frontSuivant[ln_i]$ invalidé.
    - ⋆ $fusion \leftarrow$ faux, arrêt.
  - • Sinon (diag-min = droite)
    - ⋆ Si $ln_i \neq$ dernier-ruban
      $construireRuban(d_0, d_1, front, frontSuivant, ln_{i-1})$.
    - ⋆ Si fusion Vrai
      - · $d_0 \leftarrow front[ln_{i+1}]$.
      - · $g_0 \leftarrow front[ln_i]$.
      - · Si $frontSuivant[ln_{i+1}]$ invalide, calcul de d_1.
      - · $fusion \leftarrow$ Faux.
    - ⋆ Sinon
      - · Écriture du triangle (g_0, d_0, d_1).
      - · $d_0 \leftarrow d_1$.
      - · Si $frontSuivant[ln_{i+1}]$ invalide, calcul de d_1.
    - ⋆ Continuer l'itération 1).

8.6 Résultats

Les deux techniques précédemment décrites ont été implémentées. Les résultats montrent que :

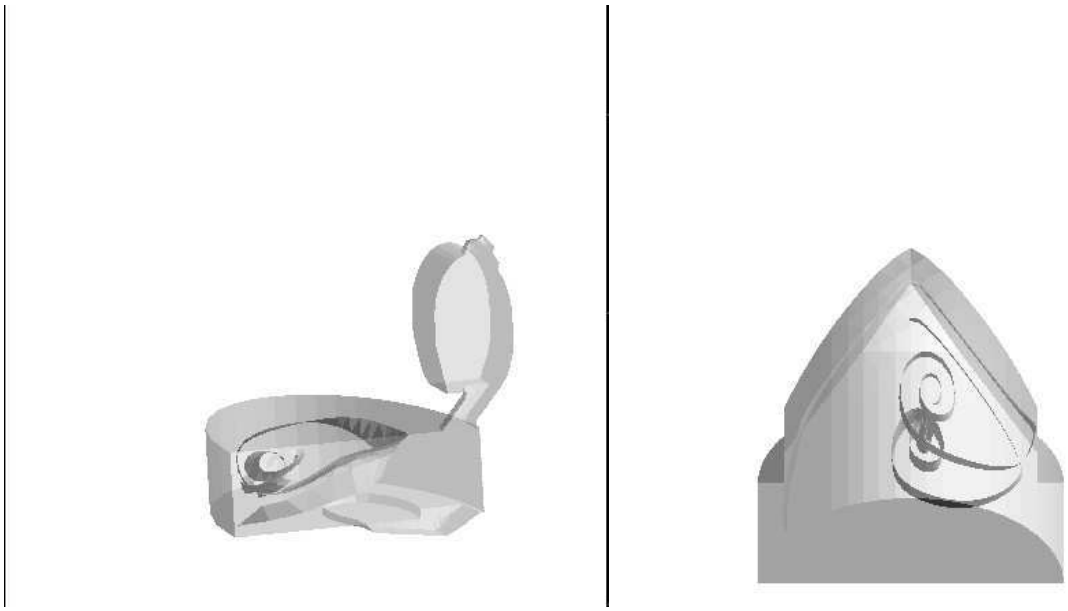
- ◇ C'est avec un pavage minimal, que les polygones de la surface sont plus petits : ce qui conduit à une meilleure approximation de la surface réelle. La différence avec le pavage régulier se remarque bien dans le cas d'écoulements divergents (cf figure 8.6).
- ◇ Dans le cas d'écoulements cisaillés, la méthode minimale choisit plus de points sur la ligne qui avance le plus vite : les éléments sont moins déformés et approximent mieux la surface.
- ◇ Dans la méthode minimale, le pavage de polygones est non structuré ; ce qui facilite l'adaptation du nombre de rubans à la densité d'échantillon optimale.



Comparaison de pavages minimal (haut) et régulier (bas), projection sur les plans XY et YZ.

Tout porte à retenir la méthode de pavage minimal même si les algorithmes impliqués sont plus délicats à mettre en œuvre.

Pour permettre une meilleure interprêtation du phénomène d'écoulement, une telle surface peut être ombrée. Quelques images (8.6)...



Surfaces d'écoulement dans une chambre de combustion et une bougie.

8.7 Autres méthodes de représentation des surfaces de courant

D'autres méthodes ont été définies pour construire des surfaces à partir de lignes de courant.
Du fait des problèmes de divergence ou de cisaillement dans les écoulements, Helman et Hesslink [HH89, HH91] proposent d'étudier la topologie des lignes de courant avant de construire des surfaces ne contenant que des points significatifs.
D'autres auteurs étudient la déformation et le mouvement des particules pendant l'écoulement du fluide.
Schroeder et ses collègues [SVL91] représentent les champs de vecteur en 3D

par des polygones régulier de n côtés, positionné le long de l'écoulement, orienté dans une direction normale au champ de vecteur local. Ces polygones subissent les tensions et rotations locales liées à l'écoulement ; ce qui se traduit par une déformation et une rotation du polygone.
Darmofal et Haimes [HD91, HGD91] s'intéressent à la représentation des composantes divergence et rotationnel du champ de vecteur vitesse.
D'après la décomposition de Helmhotz de l'écoulement, tout mouvement se décompose en une partie déformation $\vec{u_c}$ et une partie rotation $\vec{u_i}$.
$\vec{u} = \vec{u_c} + \vec{u_i}$, avec:
$\nabla \cdot \vec{u_i} = 0$ et $\nabla \times \vec{u_c} = 0$.
Ils définissent les rubans formés à partir d'un ligne de courant d'une certaine épaisseur et d'une courbure fonction de la rotation. Ils distinguent aussi des tubes de rayon variable, fonction de la divergence locale pour illustrer l'expension et la compression du fluide.

Nous nous sommes intéressés à la comparaison de deux méthodes de pavage (régulier et minimal) pour la construction de surfaces de courant.
La technique de pavage minimal semble mieux adaptée au cas d'écoulements divergents ou cisaillés.
L'étude d'autres techniques de représentation des surfaces doit être approfondie en vue d'une implémentation future.

Conclusion

Nous avons présenté un tour d'horizon des techniques de visualisation appliquées à la simulation numérique. Ces algorithmes ont été implémentés dans le logiciel VIGIE.
Nous avons distingué deux types de représentations : celles traitant des données scalaires (telles que densité ou pression), ou la représentation des données vectorielles (vecteur vitesse).

Pour les données scalaires, nous nous sommes intéressés à des modes de représentations locales : les coupes et les isosurfaces, qui permettent l'une de visualiser la répartition des valeurs dans une section du volume, et l'autre la surface formée par tous les points de même valeur pour une propriété donnée. Une autre méthode a été étudiée : le rendu volumique. Il s'agit d'un mode de représentation global qui permet de visualiser la répartition de toutes les valeurs présentes dans le volume.

Dans le cadre de la représentation des données vectorielles, les techniques de visualisation de lignes et sufaces de courant ont été présentées. Les courbes sont tangentes en tous points au champ de vecteur vitesse.

Pour toutes ces méthodes proposées, une étude des techniques existantes a été effectuée avant implémentation. De nombreuses améliorations peuvent encore être apportées. Dans le cadre de la visualisation des données scalaires, il serait intéressant de traiter la composante opacité pour améliorer les images produites par rendu volumique.
D'autres méthodes peuvent également être étudiées pour représenter les vec-

teurs en 3D : comme par exemple la représentation de petites particules solides subissant les transformations et déformations des écoulements.
Enfin une étude des phénomènes instationnaires peut être envisagée.

Bibliographie

[Be87] R.G. Belie. *Some Advances in Digital Flow Visualization.* AIAA Paper 87-1179, Reno, NV, Juillet 1987.

[Bun89] P.G. Buning. *Numerical algorithms in CFD Post-Processing* VKI, Lecture Series 1989-07, Vol. 1, 25-29 Mars 1989.

[Cli90] N. Clinch *Fast volumetric rendering for visualisation in science* Computer Graphics, pages 347–53, Novembre 1990

[DCH88] R. A. Drebin, L. Carpenter, et P. Hanrahan *Volume rendering* Computer Graphics 22(4):65–74, Août 1988

[DH91] D. Darmofal et R. Haimes. *Visual Feature Identification for 3-D Data Sets.* AIAA Paper 91-1583, 1991.

[Du88] M.J.Duurst. *Additional reference to "Marching Cubes".* Computer Graphics, 22(2):72, Avril 1988.

[FDF90] J.D.Foley, A. Van Dam, S.K. Feiner, and J.K. Hughes. *Computer Graphics; principles and practice.* Addison Wesley, 1990.

[FKU77] H. Fuchs, Z.M. Kedem et S.P. Uselton. *Optimal Surface Reconstruction from Planar Contour.* Communications of the ACM, 20(10):693–702, Octobre 1977.

[Gar90] M. Garrity. *Ray tracing irregular volume data.* Computer Graphics, 24(5):35–40, Novembre 1990.

[Gie92] C. Giertsen *Volume visualisation of sparse irregular meshes* IEEE Computer Graphics and Applications, pages 40–48, Mars 1992

[Han90] P. Hanrahan *Three-pass affine transforms for volume rendering* Computer Graphics, 24(5):71–77, Novembre 1990

[HD91] R. Haimes et D. Darmofal. *Visualization in Computational Fluid Dynamics : a Case Study.* IEEE Computer Society, Visualization'91, San Diego (CA), Octobre 91.

[Heck90] P. S. Heckbert *Graphics Gems* chapter 2, pages 84–86, A.S.Glassner, 1990

[HGD91] R. Haimes, M. Giles et D. Darmofal. *Visual3 : a Software Environnement for Flow Visualization.* VKI, Lecture series 1991-07, 16-20 Sept. 91.

[HH89] J.L. Helman et L. Hesslink. *Representation and Display of Vector Field Topology in Fluid Flow Data Sets.* IEEE Computer, 27–36, Août 1989.

[HH91] J.L. Helman et L. Hesslink. *Surface Representation ot two- and three-dimensional Fluid Flow Topology.* IEEE Computer Society, Visualization'91, 6–13, San Diego (CA), Octobre 1991.

[Hu92] J.P.M. Hultquist. *Constructing Stream Surfaces in Steady 3D Vector Fields.* IEEE Computer Society, Visualization'92, 171–178, Boston (MA US), Octobre 1992.

[Kh93] W. Kherrati. *Rendu volumique pour des maillages non structurés complexes.* Premières Journées AFIG-GROPLAN, Bordeaux, 1-3 Décembre 1993.

[KH84] J. Kajiya et B. Von Herzen *Ray tracing volume densities* Computer Graphics, 18(3):165–173, 1984

[Kru90] W. Krueger *Volume rendering and datafeature enhancement* Computer Graphics, 24(5):21–26, Novembre 1990

[Kr92] M.W. Krueger. *Artificial reality.* Addison-Wesley, second edition, 175–176, 1991.

[LC87] W.E.Lorensen et H.E.Cline. *Marching Cubes: a high resolution 3d surface construction algorithm.* Computer Graphics, 21(4):163–169, Juillet 1987.

[Lev88] M. Levoy *Display of surfaces from volume data* IEEE Computer Graphics and Applications, 8(3):29–37, Mai 1988

[Mer87] W. Merzkirch. *Flow Visualization, 2nd edition* Academic Press Inc., 1987.

[MHC90] N. Max, P. Hanrahan et R. Crawfis *Area and volume coherence for efficient visualization of 3D scalar function* Computer Graphics, 24(5):27–33, Novembre 1990

[MS90] C. Montani et R. Scopigno *Rendering volumetric data using the sticks representation scheme* Computer Graphics, 24(5):87–93, Novembre 1990

[Nee90] H.J. Neeman. *Visualization Techniques for 3D Flow Fields.* M.Sc. Thesis, University of Illinois at Urbana-Champaign, 1990.

[NSG90] K. L. Novins, F. X. Sillion, et D. P. Greenberg *An efficient method for volume rendering* Computer graphics, 24(5):95–100, Novembre 1990

[Pag93] H.-G. Pagendarm. *Scientific Visualization in computational fluid dynamics.* Visualization and Intelligent Design in Engineering and Architecture, J.J. Connors, S. Hernandez, T.K.S. Murtly et H. Power (eds). Elsevier Science Publisher, 1993.

[PW93] F.H. Post et T. van Walsum. *Fluid Flow Visualization.* in Focus on Scientific Visualization, H. Hagen, H. Muller et G.M. Nielson (eds), Springer-Verlag,1993.

[Sab88] P. Sabella *A rendering algorithm for visualizing 3d scalar fields* Computer graphics, 22(4):51–55, Août 1988

[Si89] M. Siclari. *Private communication, Science 245(28), Juillet 1989.*

[SR89] T. Strid et A. Rizzi. *Development and use of some flow visualization algorithms.* VKI, Lecture Series 1989-07, Vol.2, 25-29 Mars 1989.

[SS91] P.G. Swann et S.K. Semwal. *Volume rendering of Flow-Visualization Point Data.* IEEE Computer Society, Visualization'91, San Diego (CA), Octobre 91.

[ST90] P. Shirley et A. Tuchman A polygonal approximation to direct scalar volume rendering Computer Graphics, 24(5):63–70, Novembre 1990

[SVL91] W.J. Schroeder, C.R. Volpe et W.E. Lorensen. *The Stream Polygon : A Technique for 3D Vector Fields Visualization.* IEEE Computer Society, Visualization'91, 126–132, San Diego (CA), Octobre 1991.

[SW88] D. Seldner et T. Westermann. *Algorithms for interpolation and localization in irregualr 2D meshes.* Journal of computational physics 79, 1-11 (1988)

[TG93] J.P. Thirion et A. Gourdon. *The Marching Lines algorithm; new results and proofs.* Technical Report no 1881, Inria, Mars 1993.

[Tod92] T. Todd Elvins *A survey of algorithms for volume visualization* Computer Graphics, 26(3):194–201,Août 1992.

[UK88] C. Upson et M. Keeler *V-buffer: Visible volume rendering* Computer Graphics, 22(4):59–64, Août 1988

[WCAR90] J. Wilhelms, J. Challinger, N. Alper et S. Ramamoorthy *Direct volume rendering of curvilinear volumes* Computer Graphics, 24(5):41–47, Novembre 1990

[Wes90] L. Westover *Footprint evaluation for volume rendering* Computer Graphics, 24(4):367–376, Août 1990

[WG91] J. Wilhelms et A. Van Gelder *A coherent projection approach for direct volume rendering* Computer Graphics, 25(4):275–283, Juillet 1991

[Yan89] W.-J. Yang. *Handbook of Flow Visualization.* Hemisphere Publishing Corp., New-York 1989.

[ZS92] N. Zabusky et D. Silver. *Case Study : Visualizing Classical Problems in CFD.* IEEE Computer Society, Visualization'92, 436–440, Boston (MA US), Octobre 92.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399