energie atomique • energies alternatives

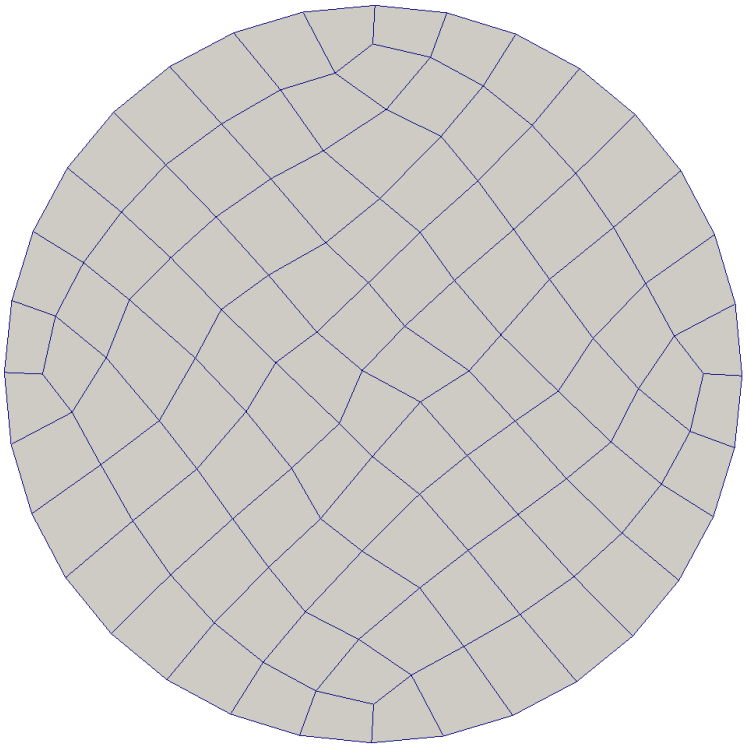
Commissariat à l'Energie Atomique
et aux Energies Alternatives
DAM, DIF
F-91297, Arpajon, France

Université de Technologie de
Compiègne
Rue du Dr Schweitzer
60200 Compiègne, France

Stage Assistant Ingénieur - TN09 - Génie Informatique

Génération de maillages quadrangulaires pour un code hydrodynamique Lagrangien

05 septembre 2011 - 17 février 2012



GAUMONT Noé
noe.gaumont@etu.utc.fr

Maître de stage : M. Franck LEDOUX
Suiveur UTC : M. Djalil KATEB

Remerciements

Je tiens à remercier tout particulièrement les personnes suivantes pour tout ce qu'elles m'ont apporté durant les 24 semaines de stage au CEA de Bruyères-Le-Châtel. Je remercie tout d'abord le chef du service pour son accueil dans son équipe et sans qui je n'aurais pas pu effectuer mon stage.

Je remercie Franck Ledoux mon maître de stage au CEA qui m'a aidé tout au long de ce stage et avec qui j'ai découvert le domaine des maillages. Il m'a permis de laisser libre cours à mes solutions tout en aidant à leurs réalisations. Ce fut un plaisir de collaborer et d'apprendre avec lui.

Mes remerciements vont également à Thao Le pour son accueil au bâtiment Ocre et à toute l'aide qu'il m'a apporté au long de ce stage. Plus généralement, je remercie l'ensemble du personnel travaillant au CEA, ainsi que les autres stagiaires universitaires et doctorants, avec qui j'ai beaucoup échangé et appris.

Sommaire

Partie 1: Introduction

Pour étudier un phénomène physique, la simulation numérique est une solution alternative à l'expérimentation. En comparaison avec cette dernière, la simulation offre de nombreux avantages notamment en matière de sécurité et en terme de coût. Elle permet aussi d'extrapoler dans des domaines non accessibles à l'expérimentation. D'un point de vue informatique, elle nécessite d'avoir une représentation discrète du phénomène étudié. Ainsi pour réaliser une simulation, la première étape est la création d'une représentation numérique simplifiée de l'objet d'étude. Cette représentation est un maillage composé de mailles qui sont des éléments géométriques simples. Par exemple, en dimension 2, il s'agit des faces (triangle, quadrangle ...). Chaque élément peut alors posséder des quantités physiques (température, pression ...) représentant l'état du matériau à une position donnée de l'espace.

Lors d'une simulation numérique Lagrangienne, le maillage utilisé se déforme et ainsi suit l'évolution spatiale des différents matériaux présents. Ce fonctionnement permet de bien cerner les mouvements présents. Cependant il peut aussi mener à un maillage invalide si les déformations sont trop fortes. Or la qualité de la simulation est directement liée à la qualité du maillage. C'est pourquoi, une attention particulière est portée sur la qualité du maillage que ce soit, de manière statique lors de la création du maillage avant le lancement de la simulation, ou de manière dynamique en modifiant le maillage (topologie et géométrie) en cours de simulation. On parle alors d'adaptation de maillage.

C'est dans ce contexte que se déroule ce stage dont l'objectif est de développer un algorithme permettant de mailler avec des quadrangles un domaine 2D à un seul bord en tenant compte d'un champ de directions défini sur le bord. Cet algorithme sera par la suite intégré dans un code d'étude simulant des problèmes d'hydrodynamique compressible.

L'idée est qu'il puisse servir statiquement pour générer le maillage initial mais aussi dynamiquement (en cours de simulation) pour remailler une zone donnée en suivant par exemple comme champ de directions, celui des vitesses nodales à l'itération précédente.

Dans ce rapport, sera présenté dans un premier temps, le CEA, le centre ainsi que le service qui m'ont accueillis. Puis dans un second temps, le but du stage ainsi que les travaux utilisés sont expliqués. Enfin le travail effectué durant le stage est détaillé avant de revenir sur l'organisation générale du stage.

Partie 2: Présentation du CEA

2.1 Le CEA

Acteur majeur de la recherche, du développement et de l'innovation, le Commissariat à l'énergie atomique et aux énergies alternatives (CEA) intervient dans trois grands domaines : les énergies décarbonées, la Défense et la sécurité globale, les technologies pour l'information et la santé.

Pour être au plus haut niveau de la recherche, le CEA compte plusieurs atouts :
- une culture croisée entre ingénieurs et chercheurs, propice aux synergies entre recherche fondamentale et innovation technologique ;
- des installations exceptionnelles (supercalculateurs, réacteurs de recherche, lasers de puissance...) ;
- une forte implication dans le tissu industriel et économique.

Le CEA est structuré autour de cinq pôles opérationnels (défense, énergie nucléaire, recherche technologique, sciences de la matière et sciences du vivant) et quatre pôles fonctionnels (maîtrise des risques, ressources humaines et formation, stratégie et relations extérieures, gestion des systèmes d'information), implantés dans 10 centres répartis dans toute la France.

Il développe de nombreux partenariats avec les autres organismes de recherche, les collectivités locales et les universités. Reconnu comme un expert dans ses domaines de compétence, le CEA est pleinement inséré dans l'espace européen de la recherche et exerce une présence croissante au niveau international. Le CEA compte actuellement 15 700 personnes pour un budget de 4,3 milliards d'euros.

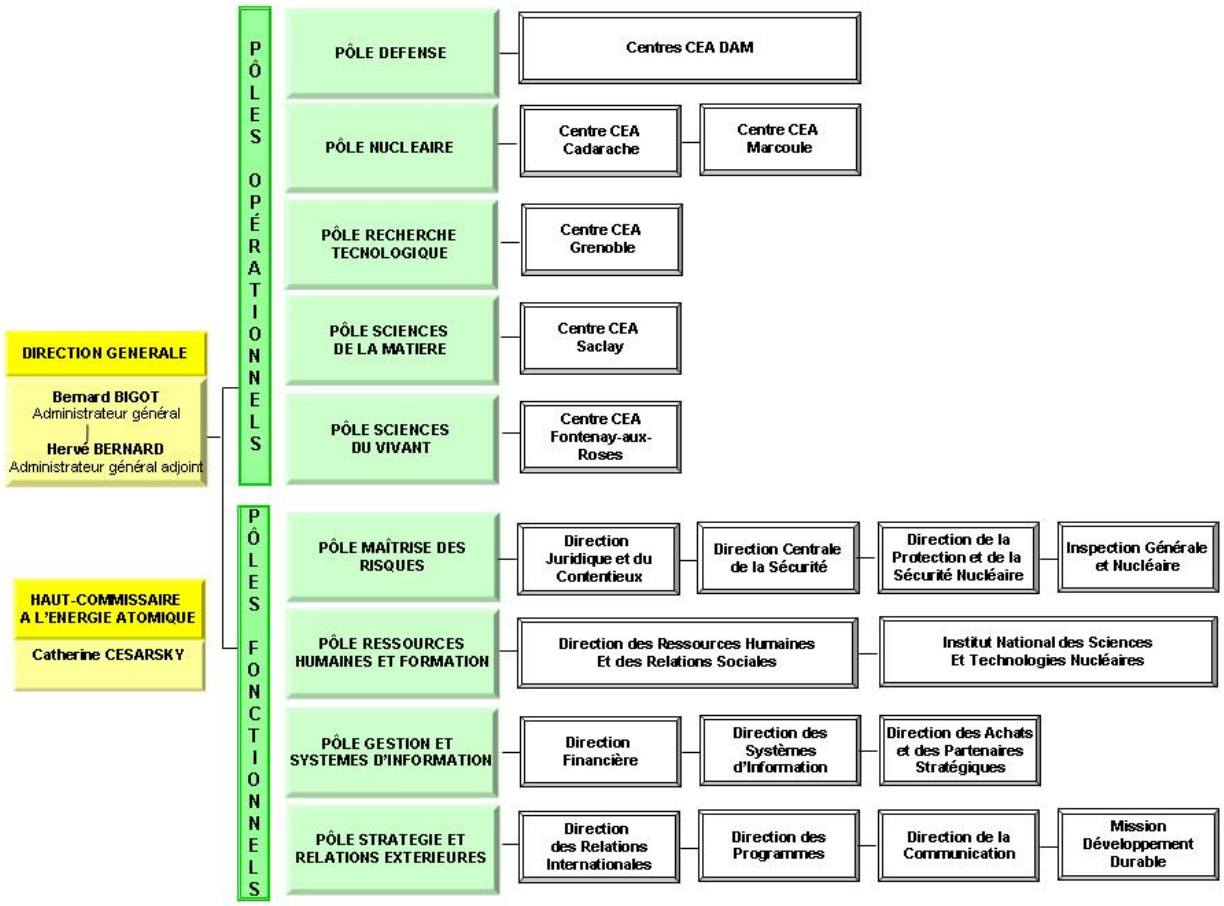


FIGURE 2.1 – Organigramme du CEA

2.2 La Direction des Applications Militaires

La Direction des Applications Militaires (DAM) constitue le pôle Défense du CEA. Ce pôle a pour mission principale de concevoir, fabriquer, maintenir en condition opérationnelle puis démanteler les têtes nucléaires qui équipent les forces océaniques et aéroportées.

Apportant aux pouvoirs publics la garantie que ces têtes sont fiables et sûres, il est l'un des principaux artisans de la crédibilité de la dissuasion nucléaire française.

Aujourd'hui, l'objectif du Pôle Défense, est de continuer à assurer sur le long terme cette capacité de dissuasion sans recourir aux essais nucléaires, définitivement arrêtés depuis 1996.

Ses missions portent également sur :
- l'approvisionnement en matières nucléaires pour les besoins de la Défense ;
- la propulsion nucléaire navale ;
- la lutte contre la prolifération et le terrorisme et la sécurité globale.

Le Pôle Défense publie bon nombre de ses résultats scientifiques et développe des collaborations avec d'autres acteurs de la Recherche. Il valorise ses activités auprès des industriels par le transfert de technologies et le dépôt de brevets. Il s'est également engagé dans une importante démarche de certification qualité de l'ensemble de ses activités liées aux armes nucléaires.

La DAM compte aujourd'hui 4 750 collaborateurs, menant des activités réparties entre la recherche de base, le développement et la fabrication. Son budget est d'environ 1,8 milliards d'euros.

Elle comprend :
- Un échelon Direction ;
- Quatre directions d'objectifs (DOB), qui ont pour mission la définition et la gestion des programmes et le pilotage des projets ;
- Cinq directions opérationnelles (DOP), qui ont pour mission la réalisation des produits conformément aux directions d'objectifs. Les directeurs opérationnels de la DAM sont les directeurs de centre ;
- Quatre directions fonctionnelles (DF), qui ont des missions de directive, de soutien et de contrôle, pour le compte du DAM.

La DAM est implantée sur cinq centres :
- Valduc, en Bourgogne
- Le Ripault, en Touraine
- le Cesta, en Aquitaine
- DAM - île de France (DIF)
- Gramat (CEG) en Midi-Pyrénées

2.3 Le centre DAM Ile de France

Le CEA/DAM - île de France (DIF) est l'une des directions opérationnelles de la DAM.
La DIF compte 2000 salariés CEA et accueille quotidiennement environ 600 salariés d'entreprises ex-

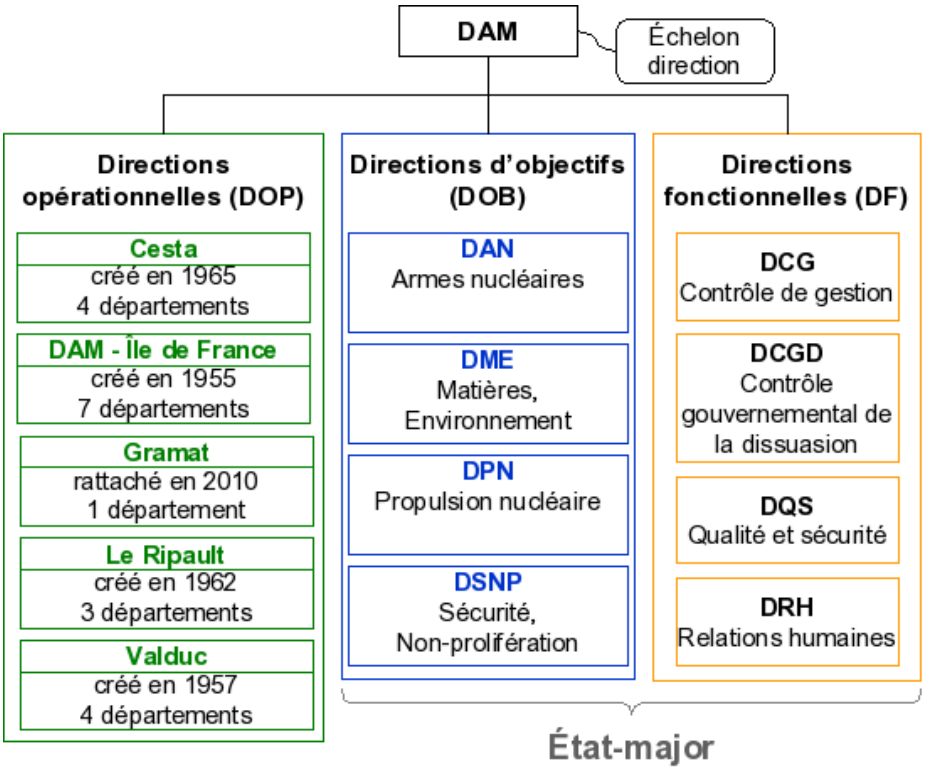


FIGURE 2.2 – Organisation de la DAM

térieures.

Elle comprend deux sites :

- Le site de Bruyères le Châtel, situé à environ 40 km au sud de Paris, dans l'Essonne ;
- Le site du Polygone d'expérimentation de Moronvilliers (PEM), situé à 20 km de Reims, dans la Marne.

Les missions de la DIF comprennent :

- La conception et garantie des armes nucléaires, grâce au programme Simulation. L'enjeu consiste à reproduire par le calcul les différentes phases du fonctionnement d'une arme nucléaire et à confronter ces résultats aux mesures des tirs nucléaires passés et aux résultats expérimentaux obtenus sur les installations actuelles (machine radiographique Airix, lasers de puissance, accélérateurs de particules) ;
- La lutte contre la prolifération et le terrorisme, en contribuant notamment au programme de garantie du Traité de Non Prolifération et en assurant l'expertise technique française pour la mise en œuvre du Traité d'Interdiction Complète des Essais Nucléaires (TICE) ;
- L'expertise scientifique et technique, dans le cadre de la construction et du démantèlement d'ouvrages complexes ainsi que pour la surveillance de l'environnement et les sciences de la terre ;
- L'alerte des autorités, mission opérationnelle assurée 24h sur 24, 365 jours par an, en cas d'essai nucléaire, de séisme en France ou à l'étranger, et de tsunami dans la zone Euro-méditerranéenne à l'horizon 2012. La DIF fournit aux autorités les analyses et synthèses techniques associées.

Depuis 2003, le centre DAM-île-de-France héberge le complexe de calcul scientifique du CEA, qui regroupe l'ensemble des supercalculateurs du CEA. Il comprend l'ordinateur TERA-100 pour les besoins Défense, d'une puissance de 1 petaflops crête, c'est à dire capable d'effectuer un million de milliards d'opérations par seconde, ainsi que les ordinateurs du Centre de Calculs pour la Recherche et la Technologie (CCRT), ouverts à la communauté civile de la recherche et de l'industrie, pour une puissance globale conventionnelle de 200 teraflops, augmentée d'une puissance de 200 teraflops en simple précision sur processeurs GPU.

Dans le cadre de la recherche européenne, le CEA contribue à la participation de la France au projet PRACE (Partnership for Advanced Computing in Europe) au travers de l'implantation sur le site de la DIF du TGCC (Très Grand Centre de Calcul), destiné à héberger un supercalculateur de classe pétaflopique, composant de la future grille de calcul destinée à la communauté scientifique européenne.

Partie 3: Sujet du stage : Solution permettant la poursuite d'une simulation Lagrangienne

On s'intéresse aux problèmes rencontrés lors d'une simulation Lagrangienne sur un maillage constitué exclusivement de quadrangles [1]. Dans ce contexte, le maillage peut subir de fortes déformations lors de la simulation numérique. Ces déformations peuvent alors conduire à l'invalidation du maillage.

Une solution possible pour pallier ce problème est d'imposer plus de contraintes lors de la création du maillage. Ainsi on peut imposer aux quadrangles de suivre une métrique qui contraigne les quadrangles dans une certaine direction et à une certaine taille. Cependant cette technique a des limites car il n'est pas toujours possible de prévoir *a priori* les déformations que vont subir les mailles. En outre même dans un cas idéal où il serait possible de connaître les mouvements des mailles, il n'est pas toujours possible de créer un maillage idéal.

C'est pourquoi nous allons essayer de fournir une brique logicielle qui permette de modifier le maillage localement au cours de la simulation. Si il est possible de détecter des zones à risques avant que le maillage ne devienne inextricable, alors il est possible de créer un nouveau maillage dans ces zones qui soit plus adapté à la poursuite de la simulation que le maillage initial. La détection d'une zone à risques pourra se faire en fonction de la qualité des mailles. Ainsi, on peut intuitivement examiner la validité d'une maille. En effet, il est clair que l'on peut qualifier les mailles de la figure 3.1 comme non valide. Il est donc possible de définir une partie du maillage qui devra être remaillée.

FIGURE 3.1 – Exemples de mailles non conformes.

Le but du travail réalisé est de recréer un maillage dans la partie qu'il est jugé utile de remailler. La détection de la zone et la projection des quantités physiques ne sont pas abordées dans ce stage. Comparée à l'étape initiale de génération de maillage, lors du re-maillage, il existe des contraintes et des informations supplémentaires que l'on peut prendre en compte. C'est pourquoi cette étape nécessite un traitement autre que celui effectué lors de la création du maillage. En l'occurrence, nous tenons compte de la connaissance d'un champ de directions à suivre dans la zone à re-mailler.

1. Les quadrangles sont utilisés car plus performants que les triangles dans le cadre de la méthode des éléments finis [5] .

3.1 Spécificités du problème

Lors d'un remaillage, on peut s'assurer de sélectionner une zone qui ne contienne aucun trou. Cela permet d'éviter les problèmes de bordure intérieure et simplifie le travail. Cependant le "patch" que l'on doit construire doit s'adapter à un maillage déjà existant. Cette configuration implique plusieurs contraintes :

- Une contrainte sur le placement des nœuds. Les nœuds sur le bord sont déjà existants et ne seront pas forcément répartis uniformément. Il est aussi impossible de bouger ces nœuds pour faciliter la projection des nouvelles valeurs physiques. Ne pas bouger les nœuds permet aussi de remailler sans connaître le maillage alentour ;

- Une contrainte de continuité. Il est important pour la simulation d'avoir une certaine continuité dans l'orientation et la taille des mailles. Il en découle un besoin d'avoir un maillage qui soit bien orienté au bord de la forme à mailler [2] ;

- Une contrainte de direction. Le maillage résultat doit pouvoir subir les déformations durant la simulation. Il doit donc suivre une carte de direction passée en paramètre ;

- Une contrainte sur la forme en elle-même. Bien que la sélection de la zone à remailler permette d'avoir une marge de manœuvre sur la forme générale de la zone. Il y a cependant peu de chance que cette zone soit constituée uniquement de formes simples.

Bien évidement le travail effectué n'a pas la prétention de créer une nouvelle méthode. Il s'agit d'utiliser les nombreux travaux de recherche déjà existants dans le domaine de la génération de maillages quadrangulaires et d'adapter une de ces méthodes aux contraintes qui se posent.

3.2 Algorithmes de génération de maillages quadrangulaires

Lorsque l'on souhaite générer un maillage, il existe fréquemment deux étapes principales. D'une part la création effective du maillage et d'autre part l'amélioration de ce maillage. Ces deux parties peuvent être traitées séparément. Pour ces deux étapes, il existe différentes techniques que nous présentons succinctement ci-dessous.

3.2.1 Algorithmes de création de maillage quadrangulaire

Il est possible de séparer les algorithmes en deux grandes catégories : les méthodes structurées et non structurées. Pour fonctionner les méthodes structurées ont besoin de pouvoir séparer un domaine en plusieurs sous domaines simples. Elles produisent des maillages structurés, c'est-à-dire où chaque nœud [3] est relié à quatre faces voisines (voir la figure 3.2). Cependant tous les domaines ne permettent pas ce genre de traitement et, comme explicité dans la section 3.1, ce n'est *a priori* pas le cas des domaines qui devront être traités dans le cadre de ce stage. C'est pourquoi, ces solutions ne sont pas décrites plus amplement.

2. On souhaite généralement avoir des quadrangles au bord qui soient géométriquement proches de carrés ou de rectangles.

3. Un nœud correspond à un sommet d'une face.

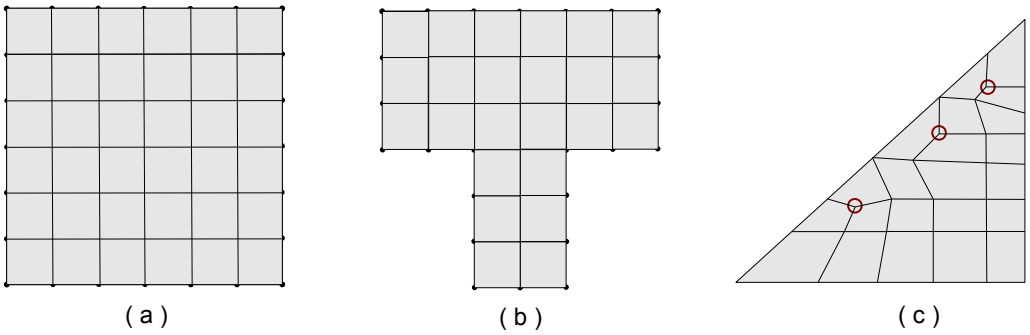


FIGURE 3.2 – Exemples de maillages structurés (a et b) et non structurés (c).

Dans les méthodes non structurées, il existe encore deux catégories : les méthodes directes et les méthodes indirectes. Avec les méthodes indirectes, le domaine est préalablement maillé avec des triangles, puis les triangles sont appareillés deux à deux pour former des quadrangles. Dans les méthodes directes, les quadrangles sont construits directement sans s'aider de triangles. L'utilisation d'un maillage triangulaire est possible car leur génération est relativement simple [4]. Malgré cette distinction entre méthodes directes et indirectes, certaines approches sont communes aux deux méthodes. C'est le cas de l'approche par avancée de fronts.

Lors du maillage par avancée de fronts, les quadrangles sont créés le long de la bordure en tournant dans un sens déterminé (voir figure 3.3). Étape par étape, des rangées de quadrangles sont ajoutées à l'intérieur du domaine. La position de chaque quadrangle est déterminée par des règles locales ce qui permet d'avoir des éléments alignés avec la bordure. L'alignement avec la bordure est une des spécificités que l'on souhaite. C'est pourquoi les algorithmes mettant en œuvre cette technique sont examinés, bien que cette technique ne soit pas exempte de défauts. En effet, chaque irrégularité lors de la création d'un quadrangle peut avoir de graves conséquences sur les prochaines rangées qui seront ajoutées. Ceci est dû aux règles locales. Les effets néfastes des irrégularités peuvent être limités via un lissage local ou global durant le processus de maillage.

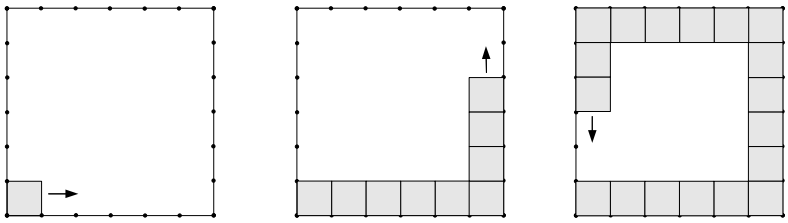


FIGURE 3.3 – Exemple d'avancées de fronts.

La technique par avancée de fronts nécessite de pouvoir déterminer un nœud de départ et un nœud d'arrivée pour la création d'une rangée. Elle nécessite aussi de gérer l'ensemble des nœuds constituant la bordure interne de manière ordonnée. Ainsi on connait tout les nœuds qu'il reste à traiter. On appelle cet ensemble de nœuds un front. Au cours du processus de maillage, il arrive parfois qu'il existe plusieurs bordures internes (voir la figure 3.4). C'est pourquoi les algorithmes utilisant cette technique gèrent un ensemble de fronts.

4. Il existe de nombreux travaux traitant de ce problème utilisant différentes approches [3, 11].

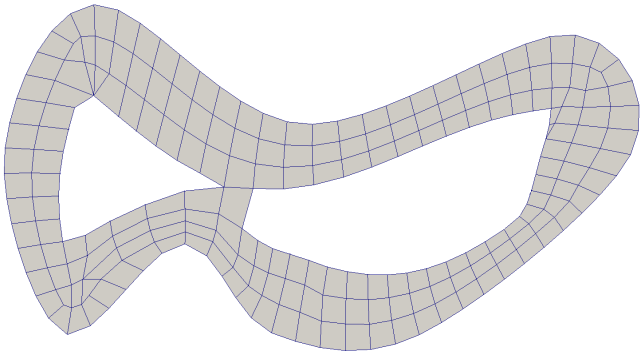


FIGURE 3.4 – Existence de plusieurs bordures dans le maillage au cours des avancées de fronts.

Un des travaux les plus connus utilisant cette technique parmi les méthodes indirectes est l'algorithme *Q-morph* [10] présenté par S. OWEN et ses collaborateurs. Il utilise les nœuds et les arêtes des triangles existants pour créer un nouveau quadrangle (voir figure 3.5). Le maillage triangulaire permet aussi de détecter et de prévenir les intersections qui pourraient avoir lieu lors de la création d'un quadrangle. Pour fonctionner *Q-morph* a besoin d'un maillage triangulaire de bonne qualité. La taille des triangles

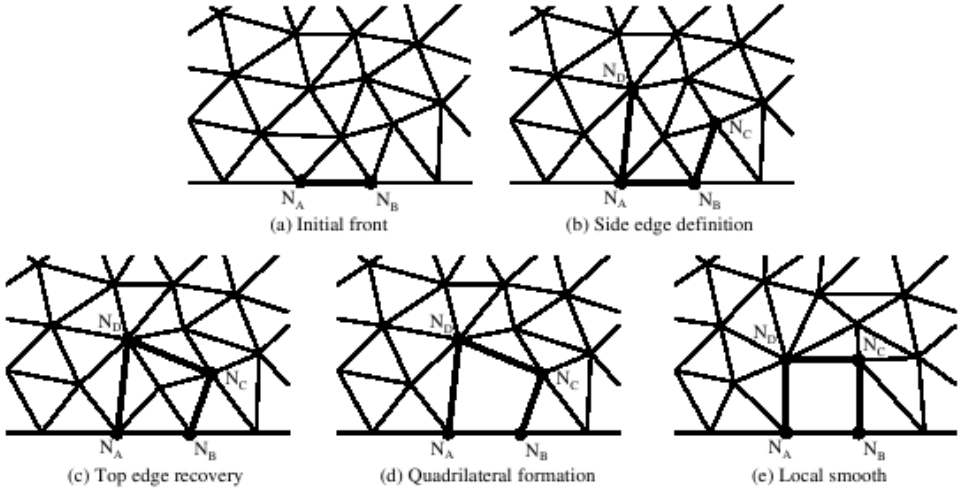


FIGURE 3.5 – Génération d'un quadrangle depuis $N_A - N_B$ avec Q-moprh. Figure provenant de [10].

doit correspondre à la taille des quadrangles voulus et lors du lissage (étape e sur la figure) la taille des triangles varie. De même, pour créer le quadrangle il est parfois nécessaire de "casser" des arêtes existantes pour créer les arêtes du quadrangle. C'est pourquoi la technique présentée par S. OWEN et ses collaborateurs demande d'apporter beaucoup de soin au maillage de fond.

À l'inverse l'algorithme de *paving* présenté par T. BLACKER et M. STEPHENSON est une approche directe [1]. Un maillage triangulaire n'est donc pas nécessaire. Les facilités offertes par le maillage triangulaire ne sont plus accessibles. Ainsi la détection des intersections et de recouvrement devient plus compliquée. En effet lorsque l'on place un nouveau point, il faut aussi vérifier qu'aucune arête du quadrangle n'intersecte un autre quadrangle existant.

Les deux algorithmes *paving* et *Q-morph* sont la base du travail réalisé lors de ce stage. L'algorithme mis en place lors de ce stage tente de concilier ces deux approches pour garder les avantages de chacune. Le but est de garder un maillage triangulaire pour détecter les intersections comme dans *Q-morph* mais avoir une structure proche de *paving*. La mise en place de cet algorithme est explicitée dans la partie 4 page 18.

3.2.2 Algorithmes de lissage

Une fois le maillage créé, il est possible d'améliorer sa qualité grâce à deux étapes successives de lissage. Le lissage topologique vise à réduire le nombre de nœuds irréguliers [5]. En réduisant le nombre de nœuds irréguliers, la qualité du maillage se trouve améliorée puisqu'on se rapproche d'un maillage structuré. Pour fonctionner, les lisseurs topologiques, comme celui de P. KINNEY [7] ou celui de S. CANNAN[2], identifient localement différents cas de figure et appliquent des solutions simples. Il s'agit d'enlever ou d'ajouter une arête ou bien d'ajouter ou de supprimer un nœud.

Plus récemment les travaux de V. CHAMAN SINGH et T. TAUTGES proposent un autre axe de recherche [4]. Des zones entières sont examinées pour tenter d'appliquer des patchs en plus des règles locales. Cependant la détection reste encore très coûteuse en terme de temps.

Une fois le lissage topologique effectué, on peut tenter d'améliorer la forme des quadrangles avec un lissage géométrique. Le lissage géométrique le plus courant est le lissage Laplacien qui consiste à déplacer chaque nœud au barycentre de l'ensemble des nœuds voisins.

3.3 Outil utilisé : GMDS

Pour répondre à ses missions le DSSI [6] a développé de nombreux outils. GMDS est l'un de ces outils. Il s'agit d'une bibliothèque C++ permettant la gestion complète d'un maillage. L'utilisation de GMDS est donc indispensable pour la réalisation du travail demandé.

Introduction à GMDS

La bibliothèque GMDS, pour *Generic Mesh Data Structure*, fournit un canevas permettant d'instancier une structure de données à partir de la déclaration des cellules et des connectivités dont le développeur souhaite disposer. Cette déclaration est le *modèle* du maillage. Par exemple, on pourra définir un maillage 2D comme composé de cellules 2D (faces) et 0D (nœuds) et des connectivités $2\rightarrow0$, c'est-à-dire que les faces connaissent les sommets adjacents, et $0\rightarrow2$, c'est à dire que les sommets connaissent les faces adjacentes. Ceci peut se résumer sous la forme $\{0, 2, 0\rightarrow2, 2\rightarrow0\}$. Pour un modèle donné, GMDS fournit alors les services nécessaires pour créer, supprimer et parcourir les cellules du maillage. Durant le stage, le travail a été effectué sur un maillage ayant pour modèle : $\{0, 2, 0\rightarrow2, 2\rightarrow0\}$.

La bibliothèque GMDS est écrite en C++ et utilise la programmation générique (template C++) entre autre, pour la déclaration des modèles de maillages. L'originalité de l'utilisation de la programmation générique dans GMDS est qu'elle a pour but premier d'optimiser l'occupation mémoire en fonction du modèle de maillage spécifié et des types de cellules.

5. Un nœuds est dit régulier si il est connecté à quatre faces.
6. Département Sciences de la Simulation et de l'Information.

Toutefois, même si le besoin de minimiser l'occupation mémoire est primordial, l'utilisation de classes abstraites et d'interfaces et donc de polymorphisme est conservée pour faciliter l'écriture d'algorithmes [7].

Structure de la bibliothèque

Le noyau de la bibliothèque GMDS est distribué sous la forme de fichiers en-tête C++ d'extension h et t.h. Les fichiers d'extension .h contiennent la déclaration des classes GMDS tandis que les fichiers t.h contiennent leur définitions (ou implantations). Aucune bibliothèque compilée n'est fournie, l'ensemble du noyau étant composé de classes génériques (comme la STL).

Utilisation de GMDS

Définition du modèle du maillage

Pour travailler sur un maillage avec GMDS, la première chose à faire est de définir le modèle de maillage sur lequel on veut travailler et alors d'instancier la classe générique *gmds* :: *Mesh* avec ce modèle. En considérant le modèle décrit en préambule, c'est à dire {0, 2, 0→2, 2→0}, on instanciera la classe suivante :

gmds : :Mesh $< DIM2|N|F|F2N|N2F >$

Le masque définissant le modèle peut être composé des balises suivantes :

- $DIM2$ ou $DIM3$ sont des balises spécifiant la dimension du maillage au sens topologique (un maillage surfacique formé de triangles plongés dans un espace 3D est donc un maillage de dimension 2) ;
- Les balises N, E, F et R correspondent respectivement à la présence dans les maillages de nœuds (cellules de dimension 0), des arêtes (cellules de dimension 1), des faces (cellules de dimension 2) et des régions (cellules de dimension 3) ;
- Les balises de la forme $X2Y$ avec (X,Y) à valeur dans $\{N,E,F,R\}^2$ indiquent la présence de la connectivité des cellules de type X vers les cellules de type Y.

Construction et destruction

La classe Mesh fournit des méthodes pour créer les cellules qui lui correspondent. Les méthodes *new...* servent à la création des cellules, les méthodes *delete...* suppriment des cellules. Les cellules sont construites à l'aide de pointeurs vers d'autres cellules de dimensions inférieures ou égales. Dans l'exemple 3.6 page 16, un maillage est instancié et quatre nœuds, deux triangles et un quadrilatère sont créés.

Attention cependant à la gestion des connectivités. Si le modèle impose que les nœuds connaissent [8] les faces ce qui est le cas dans l'exemple page 16, il faut mettre à jour la connectivité des nœuds manuellement. De manière générale, il faut ajouter les connectivités manuellement pour toutes les connectivités dite montantes [9].

7. Avec l'inconvénient d'utilisation des tables d'indirection pour les méthodes virtuelles qui pénalisent à la fois en temps de calcul et en occupation mémoire.

8. Une cellule a connait une cellule b si il est possible d'accéder à b à partir de a.

9. Il 'agit pour les cellules composant une autre cellule de connaître la cellule qu'elle compose.

```
1   int main(){
2       Mesh<DIM2|N|F|F2N|N2F> m;
3       Node* n[4];
4
5       n[0] = m.newNode(0,0);
6       n[1] = m.newNode(0,1);
7       n[2] = m.newNode(1,1);
8       n[3] = m.newNode(1,0);
9
10      m.newTriangle(n[0],n[1],n[2]);
11      m.newTriangle(n[0],n[2],n[3]);
12      m.newQuad(n[0],n[1],n[2],n[3]);
13  }
```

FIGURE 3.6 – Exemple de code créant diverses cellules.

Parcours d'un maillage

La structure GMDS permet de parcourir l'ensemble des cellules grâce à des itérateurs de manière classique. Bien souvent on préféra accéder aux cellules via leurs voisines et ainsi avancer de proche en proche pour accéder à l'élément voulu. Dans l'exemple page 33, on parcourt le maillage en examinant les faces voisines successives. Les connectivités permettent ce genre de parcours, c'est pourquoi une gestion correcte est très importante.

Marques booléennes et variables

Afin de faciliter l'écriture d'algorithmes, des marques booléennes peuvent être réservées et utilisées pour distinguer des cellules déjà traitées. Dans l'exemple page 33, le but est de récupérer tous les triangles dont le cercle circonscrit contient un point. Dans l'exemple il s'agit du point *pnt*. La liste des faces est initialisée avec une face contenant le points puis les faces voisines sont examinées. De cette manière on crée une partie du maillage qui est étoilée par rapport au point *pnt*.

Comme pour les marques, il est également possible de définir des variables sur les cellules du maillage, ces variables sont ensuite consultables directement. Une variable est associée à un type de cellule et à une valeur pour chaque cellule de ce type. Si une face est supprimée, sa valeur l'est aussi. Dans l'exemple 6.1, une variable de type `Color` est associée aux faces du maillage.

Visualisation d'un maillage

Lors de la manipulation d'un maillage, il est souvent utile de pouvoir visualiser le maillage obtenu. C'est pourquoi il est possible de lire et d'écrire les informations d'un maillage dans plusieurs formats dont le format VTK [9]. Pour les utiliser, il faut créer une instance du lecteur/écrivain associé à un maillage et ensuite on écrit ou lit dans un fichier les informations stipulées par un masque. Les fichiers VTK sont visualisables grâce au logiciel *paraview*[8] (voir la figure 3.7).

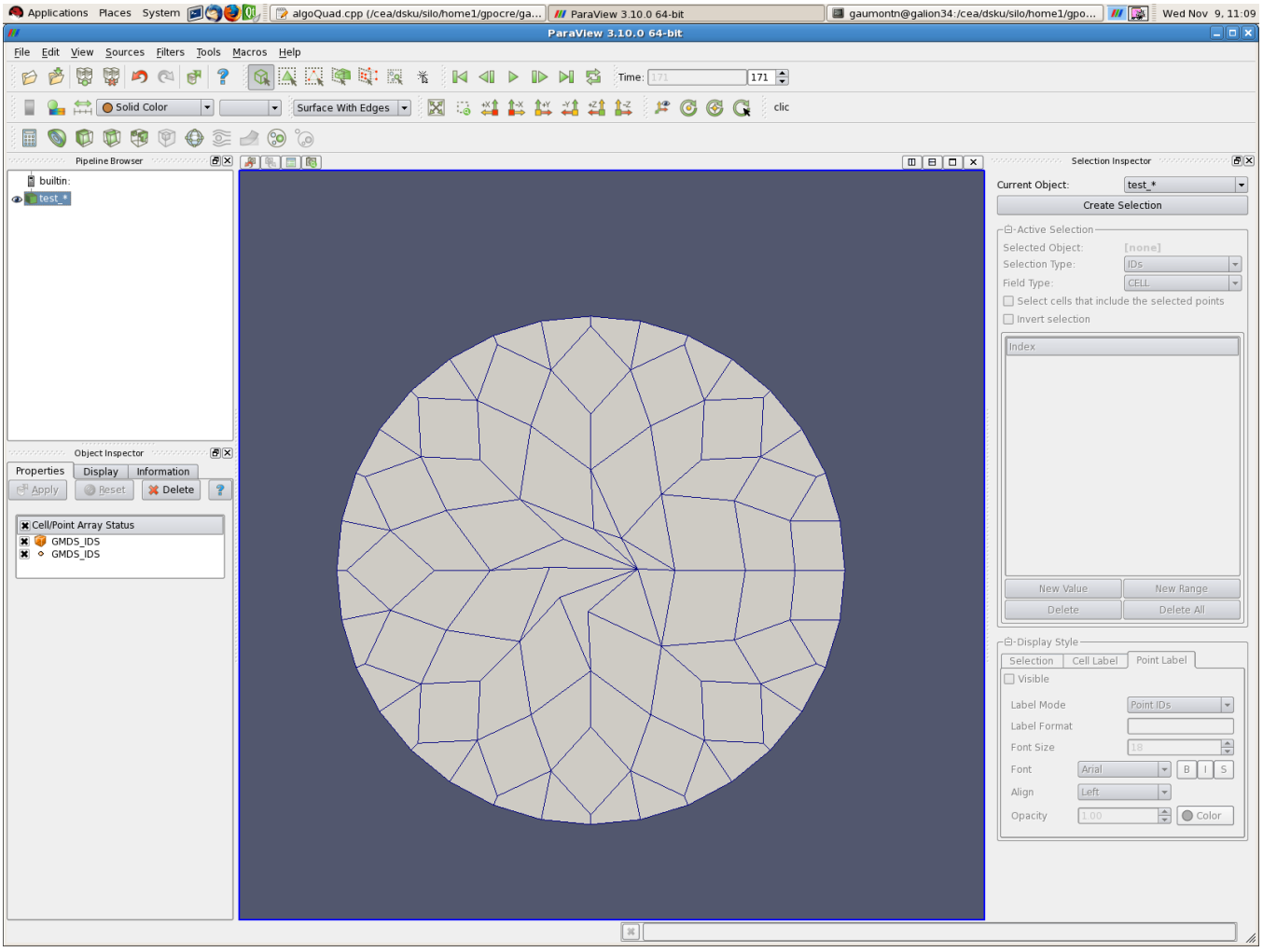


FIGURE 3.7 – Capture d'écran du logiciel paraview.

Partie 4: Mise en place d'un algorithme d'avancée de front

L'algorithme choisi n'est pas une simple implémentation d'algorithme existant en utilisant GMDS. Un algorithme hybride entre les deux méthodes présentées page 13 a été choisi. On utilise un maillage triangulaire qui n'utilise que les nœuds de la bordure du domaine. Ainsi il n'existe aucun nœud interne qui n'appartienne pas à la bordure contrairement à ce qui se fait avec l'algorithme q-morph. C'est pourquoi les quadrangles sont générés de la même manière que dans l'algorithme paving car il n'est plus possible d'utiliser totalement le maillage triangulaire.

Tout d'abord, les données d'entrées nécessaires à l'algorithme et les résultats attendus sont présentés. Puis le fonctionnement général de l'algorithme est expliqué étape par étape à l'aide d'un exemple (de la page 34 à la page 38). Enfin les étapes clefs de l'algorithme sont expliquées plus en détail, avant de conclure sur les résultats obtenus et les améliorations possibles de l'algorithme.

4.1 Données d'entrée et résultats attendus

L'objectif de l'algorithme est de créer un maillage quadrangulaire d'un domaine donné. Ce maillage ne doit comporter que des quadrangles et le bord initial du domaine est fixe. Les quadrangles doivent suivre un champs de directions (qui peut être constant). Le champ de directions doit être continu et pouvoir retourner une direction qui devra être suivie. Le maillage doit aussi suivre la bordure du domaine ce qui se traduit par la création de quadrangles dont les arêtes sur les cotés forment un angle proche de 90° avec le bord. Ces deux contraintes sont fortes et peuvent mener à des contradictions. En effet si le champ de direction impose une direction à 45°, il est impossible de suivre également la bordure du domaine (voir figure 4.1).

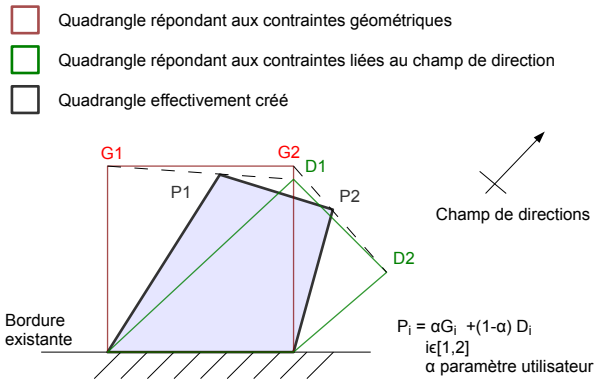


FIGURE 4.1 – Différence entre les contraintes liées à la géométrie et celles liées au champ de directions.

C'est pourquoi la conciliation de ces deux objectifs est un point sensible lors de l'implémentation de l'algorithme. On parlera en général de maillage géométrique et de maillage directionnel suivant si le maillage suit les contraintes dues à la géométrie du domaine ou celles dues au champ de directions. La taille des quadrangles n'est pour le moment pas contrainte.

Pour parvenir à la création du maillage, l'algorithme prend en entrée une liste ordonnée de points géométriques. Il faut également fournir le champ de directions que le maillage devra suivre. Il existe aussi de nombreux paramètres et stratégies par défauts qui peuvent être modifiés à l'aide de mutateurs. Par exemple, il est possible de modifier les critères d'acceptabilité d'un quadrangle. Pour des raisons de commodité, la liste de points est ordonnée dans le sens direct afin que le domaine soit défini sur la gauche de deux points consécutifs. Le diagramme de classe simplifié de l'algorithme est fourni page 24.

4.2 Déroulement général de la solution proposée

Pour utiliser le travail effectué, il suffit d'instancier la classe *Algo* et d'appeler la méthode *execute*. Cette méthode n'a besoin que d'une liste de points géométriques ordonnée qui correspond au bords du domaine. À la sortie de la méthode tout le traitement a été effectué et le domaine est maillé intégralement en quadrangle, sinon une exception est levée. Le maillage résultat est retourné à la sortie de la méthode *execute*. Si l'on souhaite modifier certains paramètres, il faut évidement utiliser les mutateurs avant l'appel de la méthode *execute*. Par exemple, il est possible de modifier le champ de directions que l'on souhaite suivre. Il doit hérité de l'interface *IMetric* et redéfinir la méthode : $getVector(Numeric < double > x, Numeric < double > y)$. La stratégie par défaut est un champ de directions constant.

Création du maillage triangulaire et du premier front

Lors de l'appel de *execute*, on vérifie tout d'abord que la liste de points contient un nombre pair de nœuds [1]. Puis le maillage triangulaire est créé en ne créant aucun nœud interne grâce une fonction déjà existante (figure 6.3 page 34). Le premier Front contenant l'ensemble des nœuds de la bordure est alors ajouté au vecteur de pointeurs de Fronts, VFront. C'est pourquoi la classe *Front* a été implémentée. Un Front est constitué d'une liste de nœuds et de tous les paramètres et stratégies de la classe *Algo*. C'est la classe *Front* qui possède toutes les méthodes permettant la création d'une rangée. L'ensemble des nœuds est parcouru afin de marquer les nœuds qui peuvent être utilisés comme nœud de départ ou d'arrivée d'une bordure. Tous les nœuds peuvent être rangés en cinq catégories ordonnées selon l'angle issu du nœud dont trois peuvent être utilisés comme nœud de départ ou d'arrivée (voir figure 4.2).

Création de rangées successives

Une fois le maillage de fond et le premier front correspondant aux nœuds de la frontière initiale créé, on rentre dans la boucle principale (voir figure 4.3). C'est ici que l'on créé les rangées successives grâce à la méthode $offsetBorder$ (voir figure 6.4 à la figure 6.10). Lors du traitement d'un front, des rangées de quadrangles sont ajoutées tant que le front possède plus de six nœuds. Quand il ne reste que six nœuds ou moins, des motifs de résolutions sont appliqués (voir figure 38). Il arrive qu'un nouveau front soit créé lors du traitement du front courant. Cela arrive si la création d'un nouveau quadrangle de la rangée entraine une intersection avec un front existant. Une fois l'intersection traitée, le front est scindé en deux au niveau de l'intersection. C'est pourquoi on gère un vecteur de pointeurs de fronts.

1. La parité de la bordure est une condition nécessaire et suffisante pour mailler le domaine uniquement à l'aide de quadrangles.

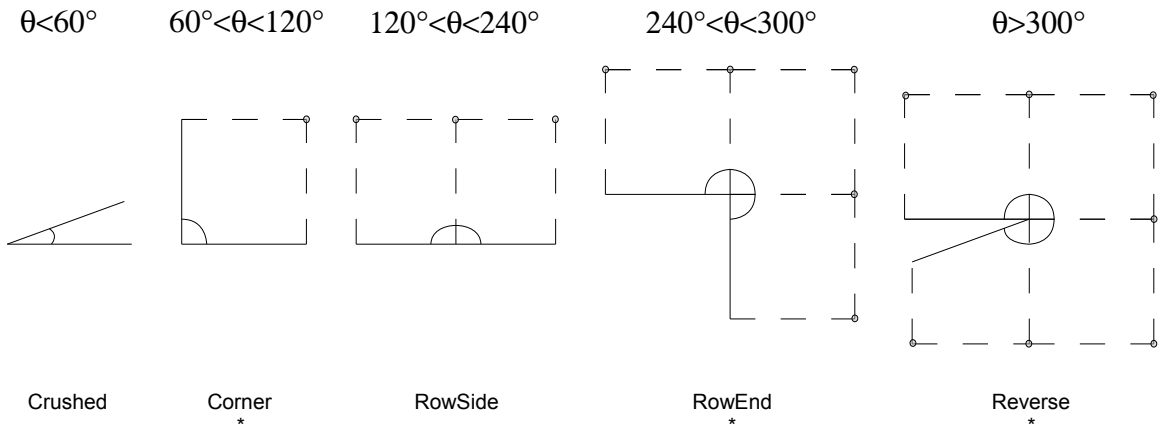


FIGURE 4.2 – Les différents types de nœud existants. Ceux marqués d'une étoile peuvent être utilisés comme nœud de départ ou d'arrivée.

```
1   std::pair<Front *,bool> pair;
2   Front F;
3   while(not(Vfront.empty())){
4           F = Vfront.front();
5           Vfront.erase(Vfront.begin());
6           do{
7                   pair = F->offsetBorder(false);
8                   if(pair.first!= static_cast<Front *>(NULL)){
9                           Vfront.push_back(pair.first);
10                          pair.first=static_cast<Front *>(NULL);
11                  }
12          }while(pair.second==false);
13          F->sixNodeEnding();
14  }
```

FIGURE 4.3 – Boucle principale de création du maillage.

Lissage topologique et lissage géométrique

Aucun lissage topologique n'a été pour l'instant implémenté au cours de ce stage. Cependant il existe des lisseurs topologiques internes au CEA qui pourront être intégrés. Pour le lissage géométrique, un simple laplacien est effectué plusieurs fois d'affilés (voir figure 6.12). Il permet d'améliorer globalement la taille des quadrangles même si il le champ de directions peut être moins respecté. Un autre problème avec les lissages laplaciens apparait lors de l'exemple 6.12. Lors du lissage laplacien, un point peut être déplacé à l'extérieur du domaine. Le problème peut être résolu en utilisant un lissage laplacien contenu ou un autre lissage.

4.3 Étapes clefs

Dans la section précédente, le fonctionnement général de l'algorithme a été présenté. Maintenant la création d'une rangée et la gestion des intersections sont expliquées en détail ainsi que les stratégies de création des quadrangles.

Stratégies de création d'un quadrangle

Comme indiqué précédemment, il existe cinq types de nœud et pour chaque type il existe un traitement différent. Cependant il s'agit toujours de créer un ou plusieurs quadrangles, à part pour les nœuds de types *crushed*. Il est donc possible de découper le traitement d'un nœud en deux parties : dans un premier temps, création d'un quadrangle et dans un second temps, appel de la stratégie d'un nœud du type inférieur. Ainsi pour traiter un nœud de type *RowEnd*, un premier quadrangle est créé sur le bord gauche puis la stratégie *RowSide* est utilisé pour créer les deux autres quadrangles nécessaires. Pour le calcul des coordonnées du ou des nouveaux nœuds, les stratégies utilisent des règles locales. Il est également important d'examiner les nœuds précédents et suivants. En effet, si ils sont de type *Corner* ou *Crushed* il est utile de les traiter en premier. De manière générale, il est possible de résumer le traitement d'un nœud avec le graphique de la figure 4.4.

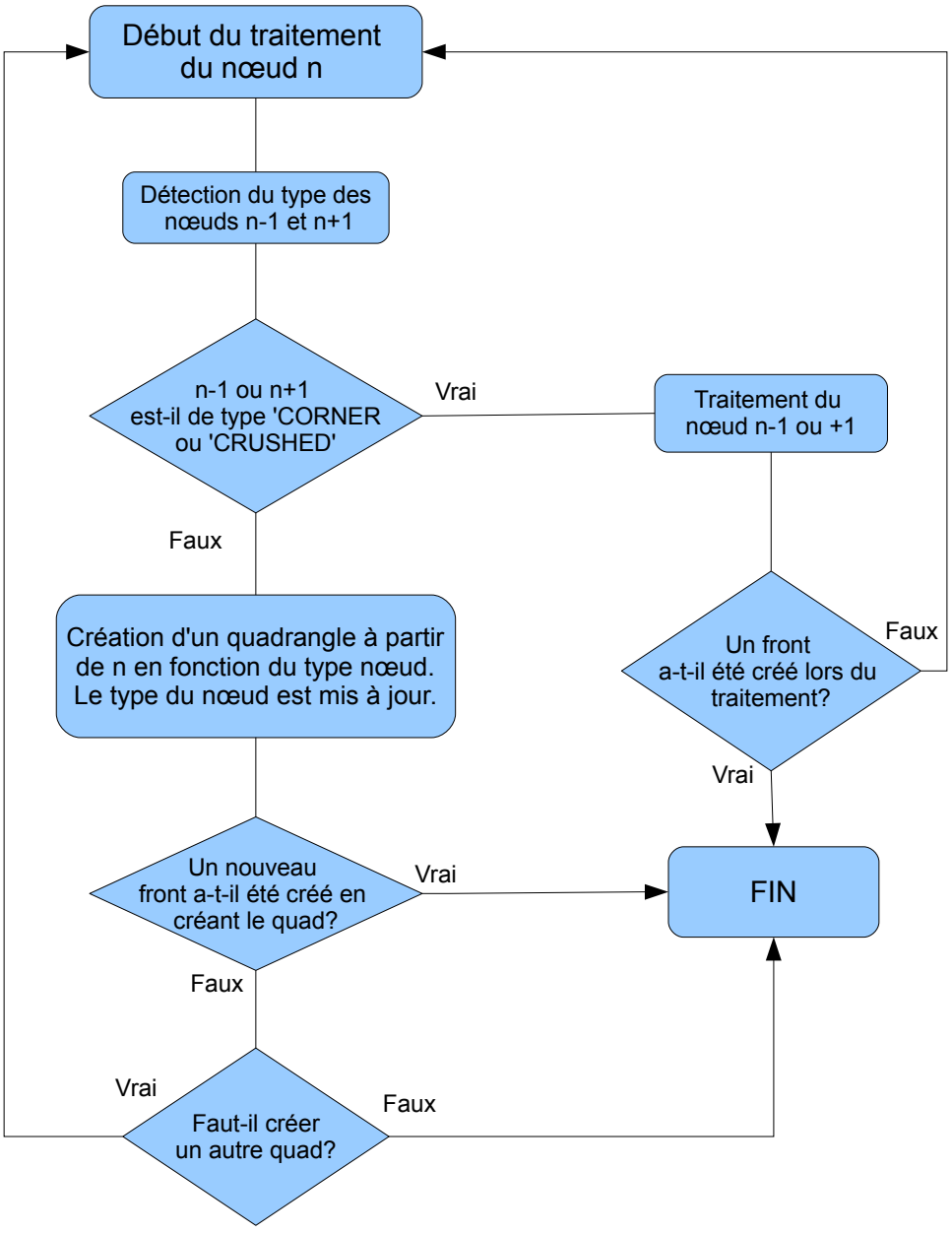


FIGURE 4.4 – Algorithme de traitement d'un nœud.

Les nœuds de type *Crushed* ont un traitement à part car les quadrangles issus de ces nœuds n'auront que rarement une forme acceptable. Il sera souvent préférable de fusionner les deux nœuds adjacents au nœud de type *Crushed*, au lieu de créer un quadrangle. C'est pourquoi, lors du traitement d'un nœud de ce type, une étape préliminaire est ajoutée. Cela revient sur la figure 4.4 à ajouter un test de fusion avant la création du quadrangle.

Détection des intersections lors de la création d'un quadrangle

Lors du traitement d'un nœud, la création d'un quadrangle nécessite beaucoup d'attention. Il ne suffit pas de créer un nœud aux coordonnées données par la stratégie puis de former le quadrangle. Il est nécessaire d'effectuer plusieurs vérifications. C'est ici que l'utilisation du maillage triangulaire prend tout son sens. En effet, à partir d'un triangle, il est possible de repérer dans quelle zone se trouve un point par rapport à ce triangle. Cela se fait en examinant la position du point par rapport à chaque arête du triangle. Ainsi, on distingue six zones différentes, visibles sur la figure 4.5.

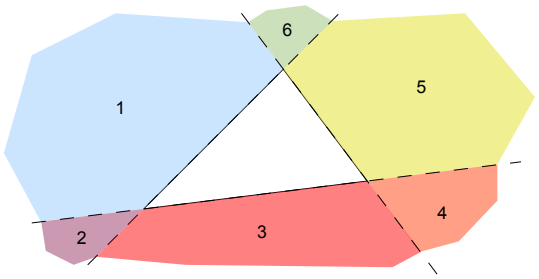


FIGURE 4.5 – Découpage en zones autour d'un triangle pour repérer un point.

Une fois la zone qui contient le point repérée, on peut choisir la face voisine du triangle de départ et ainsi se rapprocher du point. Ainsi de suite, la face qui contient le point est trouvée en parcourant relativement peu de faces du maillage. C'est le principe du parcours barycentrique. Si la face d'arrivée est un triangle, il n'y a pas eu d'intersection. Si c'est un quadrangle alors il y a une intersection. La face d'arrivée permet aussi de déterminer les nœuds proches du nœud que l'on veut créer. Pour s'assurer de l'intégrité du nouveau quadrangle, des parcours barycentriques sont effectués le long de chaque future arête avant la création du quadrangle. Ainsi les intersections sont détectées juste avant la création des quadrangles.

Gestion des intersections lors de la création d'un quadrangle

Il existe deux sortes d'intersections. Soit le nœud qui devait être créé est dans un quadrangle, soit un quadrangle est entre le nœud et les autres nœuds du quadrangle qui devait être créé. Ce dernier cas n'est pour l'instant pas traité et une exception est levée. Dans le premier cas, il faut réutiliser un nœud existant à la place du nœud que l'on voulait créer. D'ailleurs, même si il n'y a pas d'intersection, les nœuds proches sont examinés pour potentiellement les réutiliser. Ces nœuds sont facilement localisables car il s'agit des nœuds de la face d'arrivée du parcours barycentrique. Une fois le candidat trouvé, il faut encore examiner la parité du front car si on réutilise un nœud déjà existant alors le front courant est séparé en deux fronts. Si les deux fronts ont un nombre pair de nœuds alors le quadrangle est créé ; si les fronts ont un nombre impair de nœuds, le front courant est séparé le long de la diagonale du quadrangle qui aurait dû être créé. Tout le déroulement de la création d'un quadrangle peut donc être résumé par la figure 4.6.

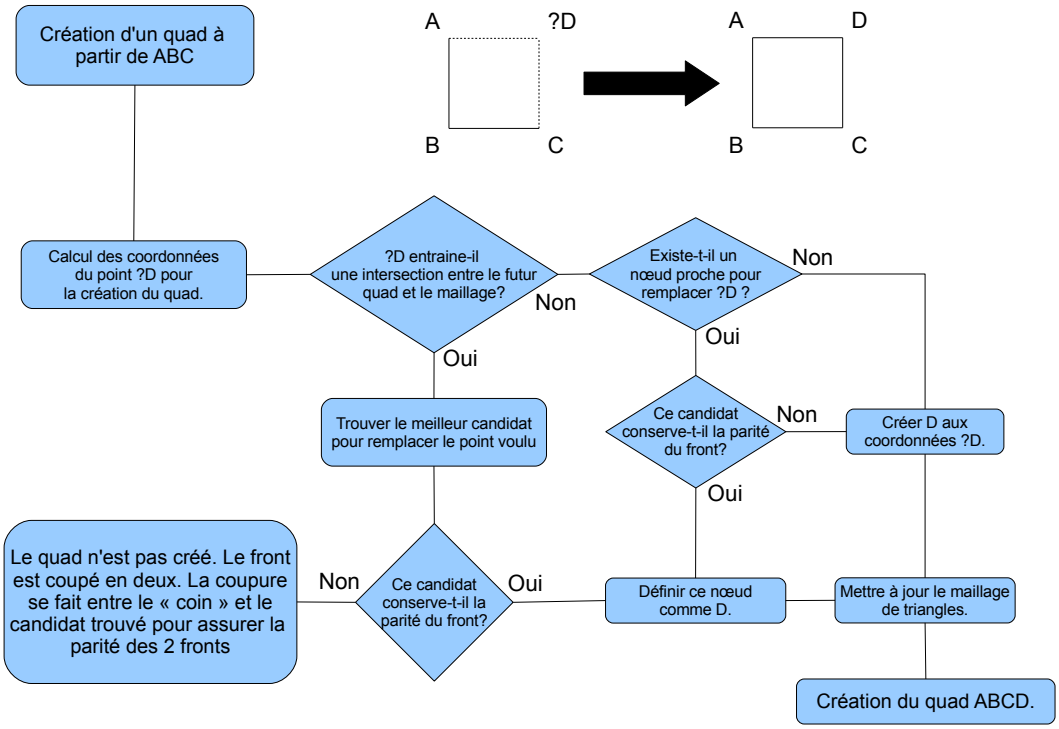


FIGURE 4.6 – Algorithme de création d'un quadrangle à partir de 3 nœuds.

Mise à jour du maillage triangulaire

Le quadrangle est donc créé ou à défaut le front a été séparé en deux. Il faut maintenant mettre à jour le maillage triangulaire. Grâce au parcours barycentrique, la liste des triangles intersectant le nouveau quadrangle est connue. Ces triangles seront donc détruits mais cela ne suffit pas. Il faut aussi intégrer le nouveau nœud au maillage triangulaire. Pour faciliter la tâche, on sélectionne également les triangles autour du nouveau nœud afin d'avoir une cavité étoilé autour de ce nœud comme vu dans l'exemple page 33. Un *set* des nœuds composants ces triangles est récupéré lors de leurs destructions. Ainsi, en ordonnant le *set* dans le sens horaire, la cavité est remaillée avec des triangles, chaque triangle ayant comme sommets le nouveau nœud et deux nœuds consécutifs du *set* ordonné. Tout ceci se fait en ayant pris soin de ne pas créer de triangle à l'intérieur du nouveau quadrangle [2]. La figure 4.7 résume les différentes étapes de restauration du maillage triangulaire lors de l'ajout d'un quadrangle.

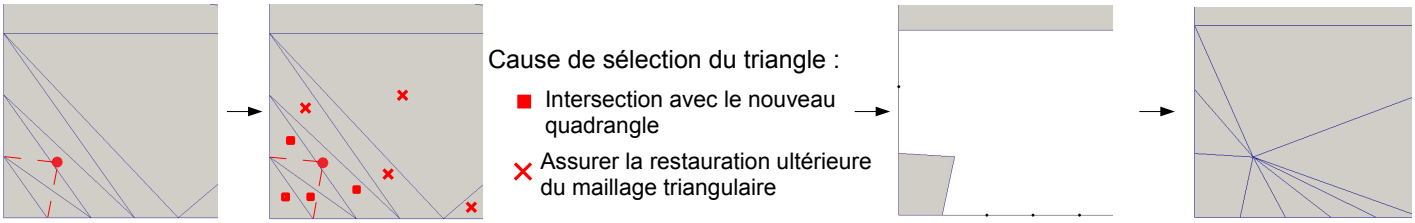


FIGURE 4.7 – Mise à jour du maillage triangulaire lors de l'ajout d'un quadrangle.

2. Pour ce faire il suffit de retirer du set les nœuds composant le quadrangle.

4.4 Résultats obtenus et limites

Le code résultant de ce stage est un code fonctionnant avec des paramètres d'entrées simples : un ensemble de points définissant le bord du domaine et un champ de directions. Pour plus de souplesse dans le fonctionnement, il est possible de modifier différents paramètres internes au code (attributs de classe) tels que les marges d'angle d'acceptabilité d'un quadrangle. Ainsi, le résultat final est paramétrable par l'utilisateur.

Structurellement, le code est constitué de sept classes C++ pour environ 4000 lignes de code. Il a été pensé et architecturé de manière modulaire et évolutive avec l'utilisation de stratégies (patron de conception [6]) pour que le code soit étendu par la suite.

La structure du code en question est illustrée sur la figure 4.8. L'ensemble du code s'articule autour de la classe *Algo*. C'est l'unique classe que l'utilisateur doit manipuler, ainsi le fonctionnement reste facilement accessible.

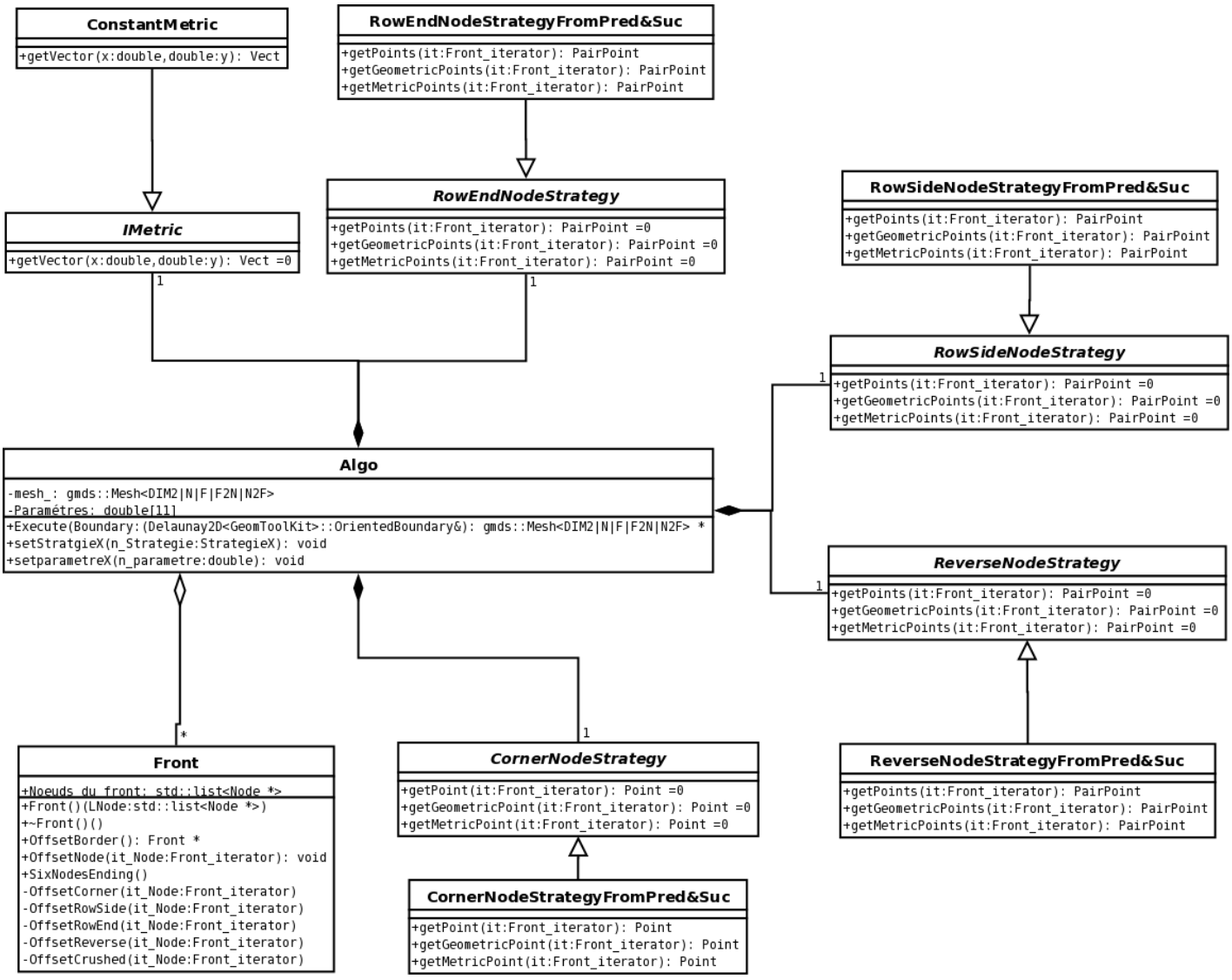
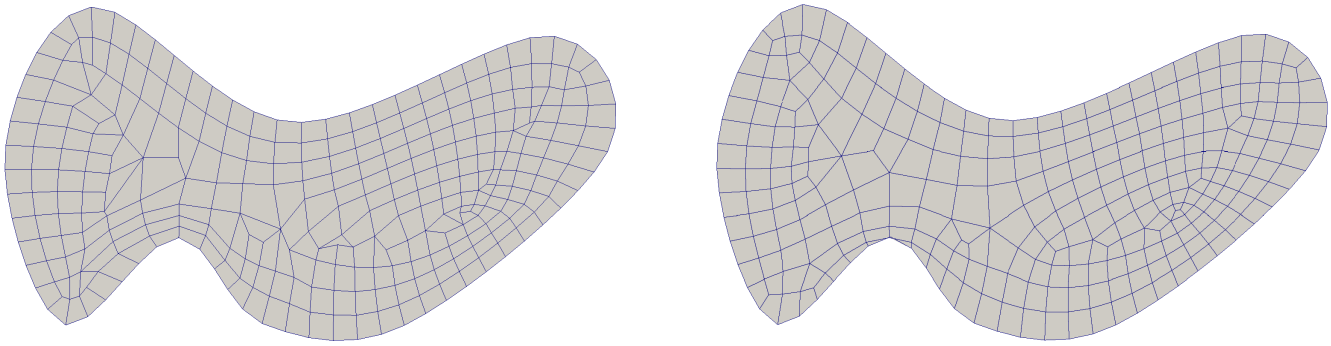


FIGURE 4.8 – Diagramme de classe simplifié du travail réalisé.

D'un point de vue fonctionnel, les figures 4.4 à 4.12 illustrent les résultats obtenus. Elles montrent bien la capacité du code à mailler des formes relativement complexes. Notamment le maillage de formes courbes est possible grâce aux règles locales actuelles ; on observe la formation de rangées qui sont bien orientées par rapport à la bordure. L'intégration du champ de direction a nécessité un travail important mais on observe dans les figures que globalement les quadrangles tendent à suivre le champ de directions. Ceci permet d'obtenir des maillages totalement différents avec différents champ de directions pour la même forme géométrique .

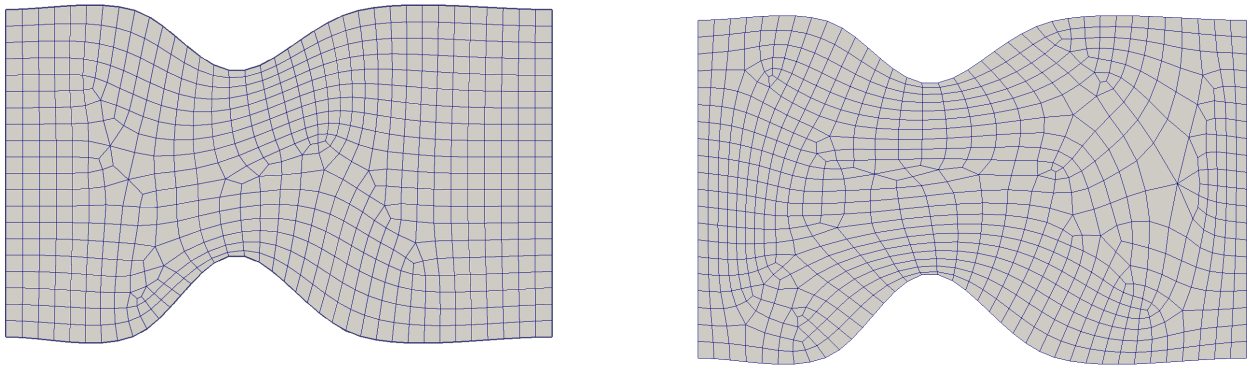
Par contre le champ de directions n'est plus respecté lors de la fusion de fronts. En général, l'alignement avec le champ de directions est perdu et des nœuds irréguliers (valence différent de 4) apparaissent.



(a) (b)

FIGURE 4.9 – Influence du lissage sur l'orientation des quadrangles d'un maillage.

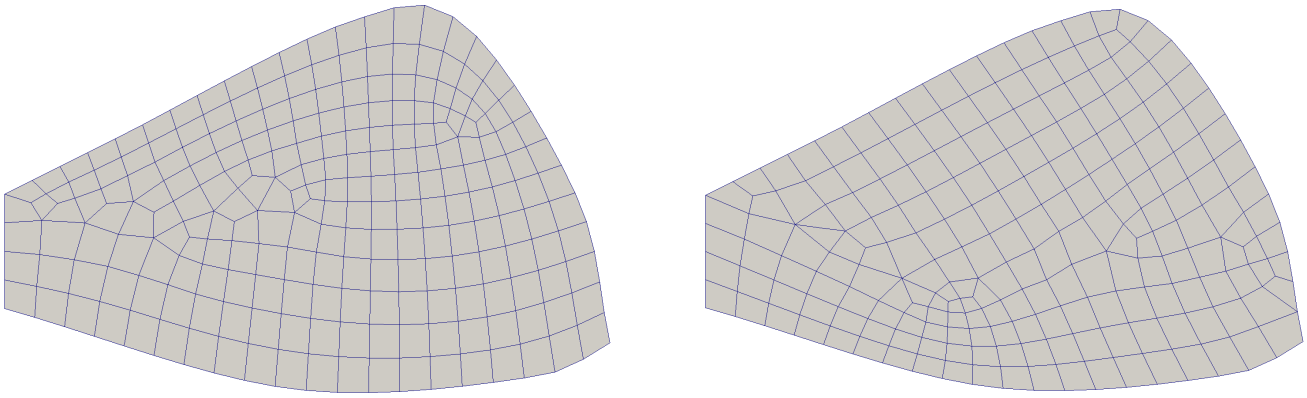
En (a) le maillage obtenu avant l'aplication d'un lissage géométrique de type Laplacien. En (b), le maillage obtenu après lissage. Les quadrangles du maillage (b) suivent moins le champ de directions que ceux du (a).



(a) (b)

FIGURE 4.10 – Influence du champ de directions lors du maillage d'un domaine en forme de "sablier".

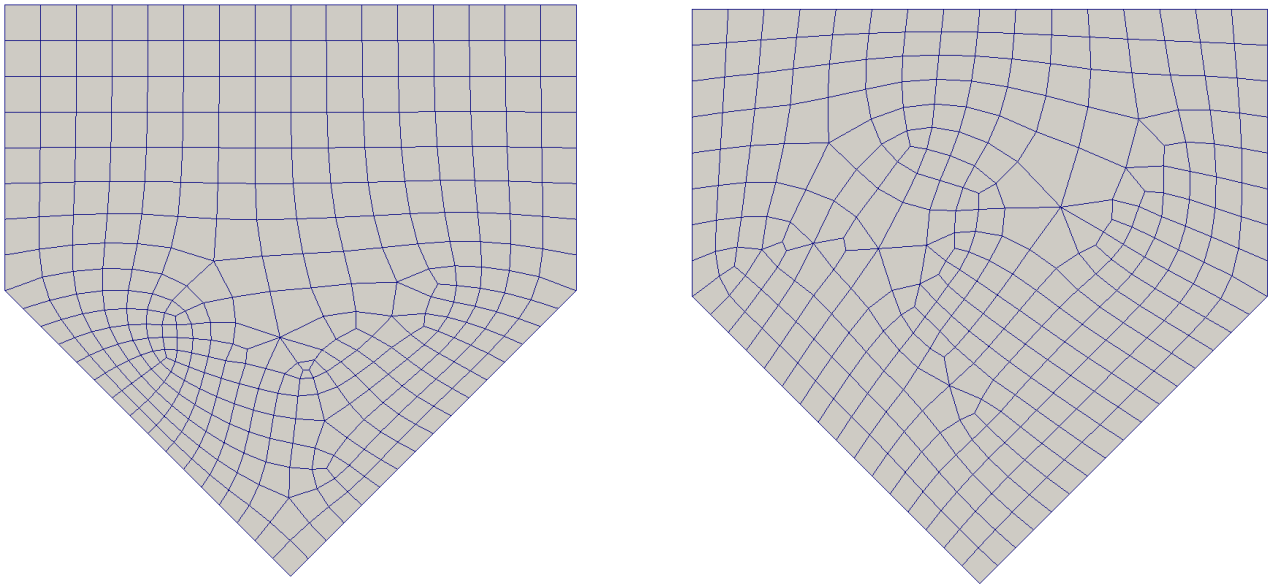
En (a), le maillage est supposé suivre un champ de directions constant d'axe (Oy). Le maillage est fidèle à cette direction lorsque la géométrie est en accord avec (extrémités du sablier). Par contre, la courbure en son centre nécessite l'ajout de nombreux nœuds irréguliers. En (b), la direction suivie est légèrement différente, ce qui induit la création d'un maillage totalement différent.



(a) (b)

FIGURE 4.11 – Influence du champ de directions lors du maillage d'un domaine en forme de "poisson".

En (a), le champ de directions suivi est défini par le vecteur $(0,1)$ en tout point du domaine alors qu'en (b), il est défini par le vecteur $(1,1)$. Dans les deux cas, le maillage suit le champ de directions.



(a) (b)

FIGURE 4.12 – Influence du champ de directions lors du maillage d'un domaine en forme de pentagone non régulier.

Comme dans l'exemple précédent, le champ de directions suivi en (a) est défini par le vecteur $(0,1)$ et par le vecteur $(1,1)$ en (b). Cependant plus de nœuds irréguliers sont créés au centre de du domaine dans le cas (b).

Bien que les résultats obtenus soient très encourageants, le travail n'est pas terminé et il existe des problèmes qui nécessitent plus d'attention. Suite aux tests, deux points principaux sont à améliorer dans l'intégration du champ de directions.

- L'intégration du champ de directions est maitrisée à l'échelle locale mais à l'échelle globale des améliorations sont encore possibles. En effet, près du bord du domaine l'aspect géométrique est plus important mais au centre du domaine le champ de directions est plus important. Il peut être intéressant de ne pas donner la même importance sur tout le domaine au champ de directions par rapport à la géométrie ;

- Le second point à améliorer est la prise en compte du champ de directions lors de la phase d'amélioration du maillage résultat. Les lisseurs de type Laplacien sont certes efficaces mais ils ne tiennent pas compte du champ de directions.
  Il en va de même avec les lissages qui se font au cours du processus de maillage.

Ces améliorations permettraient à terme de mailler un nombre plus large de domaines, car si le champ de directions entre en complète contradiction avec la géométrie alors le processus peut ne pas aboutir et indiquer les causes de cette incapacité. Il existe également un autre point qui devra être amélioré. Il s'agit de la gestion de la taille des quadrangles, bien qu'il s'agisse d'un paramètre non considéré initialement durant ce stage. Cependant considérer la taille des quadrangles et leur variation peut grandement augmenter la qualité du maillage.

Malgré ces pistes qu'il convient d'explorer, la solution actuelle est satisfaisante. De plus, de par la structure adoptée, les modifications qui seront faites ne nécessiteront que peu de travail d'intégration. C'est aussi à cause de cette vocation à être modifié que le code est un code d'étude.

Partie 5: Organisation du stage

Ce statut de code d'étude n'est pas la seul particularité de ce stage. L'organisation du stage a également été particulière de part sa teneur recherche assez importante et de part son environnement. Je reviens donc sur ces points dans cette partie en suivant une approche chronologique du déroulement du stage.

5.1 Environnement de travail

Pour travailler durant ce stage, le CEA a mis à disposition des postes avec un système UNIX, Red Hat Entreprise 6, et l'accés à internet (sous réserve de contraintes de sécurité). Tous les stagiaires et doctorants du service sont dans un bâtiment différent de celui du service. Cette configuration a permis d'avoir de l'autonomie et du recul sur mon travail. Elle a été d'autant plus bénéfique que les autres stagiaires étaient présents pour aider pour les erreurs programmations, et que mon tuteur était également présent ou joignable par téléphone. Les conditions de travail sont plus classiques avec une semaine au 35h.

5.2 Mise en place et prise en main des outils nécessaires

Une fois les formalités administratives d'entrée effectuées, la découverte de l'environnement et le stage ont pu commencer. Il a été nécessaire d'effectuer un travail de recherche important pour connaitre la majorité des possibilités dans le domaine du maillage quadrangulaire. En effet, bien que l'objectif du stage ait été clairement défini, la manière d'y parvenir n'a pas été imposée. C'est pourquoi durant le premiers mois de ce stage, il y eu beaucoup d'échanges entre mon tuteur et moi-même pour définir le ou les algorithmes qui répondent aux problèmes soulevés. À partir de ces échanges, la solution, présentée dans la partie 4, a été conçue au fur et à mesure.

Dans le même temps, il a également été nécessaire de prendre en main les outils tels que GMDS et paraview. La présence d'exemples, a beaucoup aidé lors de la découverte de GMDS. L'utilisation de paraview se limite à la simple visualisation d'un maillage, c'est pourquoi sa prise en main a été plus rapide. Une fois la base acquise durant ce premier mois, le travail sur l'algorithme final a pu commencer. Il s'est articulé en deux phases principales : tout d'abord créer toute la structure de l'algorithme sans contrainte de direction, puis intégrer la prise en compte d'un champ de directions à l'algorithme final.

5.3 Algorithme d'avancée de fronts sans la contrainte du champ de directions

Cette partie du stage a duré environ 2 mois car il faut créer toute la structure et les routines de fonctionnement. Il s'agit de définir la gestion du maillage triangulaire, l'ajout d'une maille quadrangulaire dans différentes configurations et différents cas d'intersections géométriques. Un travail de hiérarchisation du code a également été fait pour assurer la lisibilité et donc la modification. Il a également été nécessaire de créer un ensemble de fonctions permettant le traitement du maillage, notamment le traitement géométrique. En effet, beaucoup des traitements nécessaires sont spécifiques à l'algorithme et n'étaient donc pas implémentés dans la bibliothèques GMDS. C'est durant cette phase que le fonctionnement a été peaufiné notamment avec les stratégies. Une phase de test a été nécessaire pour assurer une certaine robustesse de l'algorithme.

5.4 Intégration du champ de directions

Arrivé à ce stade du stage, l'intégration a été relativement rapide grâce au travail préliminaire. Le champ de directions devait aider le maillage et ne pas ajouter plus de problèmes. En effet, un champ de directions peut aider le maillage en contraignant l'orientation des quadrangles mais cela n'est vrai que si il est bien constitué. Un champ constant peut être efficace dans une forme simple telle un rectangle mais dans une forme quelconque, il entraine des conflits. C'est pourquoi de nombreux problèmes sont apparus durant cette phase.

C'est ainsi que commença une nouvelle phase de recherche et de test. Les opérations qui étaient simples se compliquent rapidement que se soit la création d'un quadrangle ou un lissage local au cours de l'avancée de fronts. C'est pourquoi le reste du stage a été consacré à l'amélioration de la prise en compte des contraintes du champ de directions.

Partie 6: Conclusion

Lors de ce stage, j'ai eu, avec l'aide de mon tuteur, à concevoir puis à implémenter une brique logicielle capable de mailler une cavité en tenant compte d'un champ de directions. Elle devra ensuite être intégrée dans un code d'étude simulant des problèmes d'hydrodynamique compressible. En plus de la réutilisation du code, la robustesse du code rendu était un critère d'évaluation de la solution proposée.

Au début de ce stage, les travaux existants ont été étudiés pour la conception de la solution. Suite à ces recherches, j'étais donc en mesure de pouvoir comprendre les avantages respectifs de chaque algorithme. Afin de répondre aux contraintes spécifiées, un algorithme hybride a été implémenté étape par étape. Durant l'implémentation, des phases de tests ont été conduites et ont révélé des problèmes inattendus. Certains ont pu être résolus à la volée, d'autre nécessiteront une recherche approfondie. Cependant la solution présentée répond aux objectifs d'évolutivité et de robustesse de ce stage. L'algorithme permet déjà de mailler des formes diverses ou à défaut de comprendre ce qui a empêché le fonctionnement normal.

Au moment où ces lignes sont écrites, il reste des solutions qui n'ont pas été implémentées et testées. En outre il reste des domaines d'amélioration non étudiés à notre connaissance, notamment sur les lisseurs géométriques. En effet, il nous faut améliorer en même temps l'aspect d'un quadrangle et en même temps son orientation. Il en va de même avec les motifs de résolution de fermeture de front qui ne tiennent pour l'instant pas compte du champ de directions.

Au final, le stage en lui-même a été une expérience très enrichissante. Tant techniquement avec des problématiques liées au maillage quadrangulaire, domaine que je n'avais pas exploré avant le stage, qu'humainement. En effet, les échanges constants entre mon tuteur et moi-même m'ont permis d'appliquer mes connaissances à un projet de grande envergure. Ce stage au CEA m'a également fait connaitre un environnement de recherche.

Bibliographie

[1] T. D. BLACKER and M. B. STEPHENSON. Paving : A new approach to automated quadrilateral mesh generation. International journal for numerical methods in engineering, 32 :811–847, 1991.

[2] S. CANANN, S. MUTHUKRISHNAN, and R. PHILLIPS. Topological improvement procedures for quadrailateral finite element meshes. Engineering with Computers, 14 :168–177, 1998.

[3] J. C. CAVENDISH. Automatic triangulation of arbitrary planar domains for the finite element method. International journal for numerical methods in engineering, 8 :679–697, 1974.

[4] V CHAMAN SINGH and T TAUTGES. Jaal : Engineering a high quality all-quadrilateral mesh generator. Proceeding 20th International Meshing Roundtable, pages 511–530, 2011.

[5] R. D. COOK, D. S. MALKUS, and M. E. PLESHA. Concepts and Applications of Finite Element Analysis. 1989.

[6] E. GAMMA, E. HELM, R. JOHNSON, and J. VLISSIDES. Design patterns-Catalogue de modèles de conception réutilisable. 1999.

[7] P. KINNEY. Cleanup : improving quadrilateral finite element meshes. Proceeding 6th International Meshing Roundtable, pages 449–461, 1995.

[8] Kitware. Site web de paraview. http ://www.paraview.org.

[9] Kitware. Site web de vtk. http ://www.vtk.org.

[10] S. J. OWEN, M. L. STATEN, S. A. CANANN, and S. SAIGAL. Q-morph : An indirect approach to advancing front quad meshing. International journal for numerical methods in engineering, 44 :137–1340, 1999.

[11] J. PERAIRE, M. VAHDATI, K. MORGAN, and O. C. ZIENKIEWICZ. Adaptive remeshing for compressible flow computations. Journal of Computational Physics, 72 :449–466, 1987.

Table des figures

Annexes

6.1 Exemples d'utilisation de GMDS

```
1        Mesh<DIM2|N|F|F2N> mesh;
2        Variable<Color> *v = mesh.newVariable<Color>(GMDS_FACE, "color" );
3        Mesh<DIM2|N|F|F2N>::faces_iterator it=mesh.faces_begin(), ite=mesh.
             faces_end();
4        int i=1;
5        for(;it!=ite;it++){
6                id k = (*it)->getID();
7                Color c =(*v)[k];
8                (*v)[k].setR(i++);
9                (*v)[k].setV(i++);
10               (*v)[k].setB(i++);
11       }
```

FIGURE 6.1 – Utilisation des variables dans GMDS.

```
1        std::set<Face *>::iterator it1,it2;
2        std::set<Face *> S2 = face_de_depart), Stmp, resultat;
3        it2=S2.begin();
4        while(!S2.empty() ){
5                Face * cur;
6                it1 = S2.end();
7                it1--;
8                cur= *it1;
9                mesh->mark(cur,done);
10   // On marque la face avec la marque done pour ne pas la re-traiter
11               S2.erase(it1);
12   // Si le cercle circonscrit au triangle cur contient le point pnt,
13   // alors on doit egalement traiter tous les triangles non traites adjacents a cur
14               if ( isInCircumCircle( cur , pnt ) ){
15   // le cercle circonscrit a la face cur contient le point pnt, cur est donc ajoute
         dans le set resultat.
16                       resultat.insert(cur);
17   // On recupere ici les faces ayant une arrete en commun avec la face courante.
18                       Stmp=getSuroundingTriangles(cur);
19                       it2=Stmp.begin();
20                       while(it2 != Stmp.end()){
21                               if(not(mesh->isMarked((*it2),done))){
22    // Si la face n'est pas marquee on l'ajoute dans le set de face a traiter.
23                                       S2.insert(*it2);
24                               }
25                               it2++;
26                       }
27               }
28       }
29       mesh->unmarkAll(done);
30       mesh->freeMark(done);
```

FIGURE 6.2 – Code construisant un étoilé autour du point *pnt*.

6.2 Déroulement du processus de maillage d'un domaine

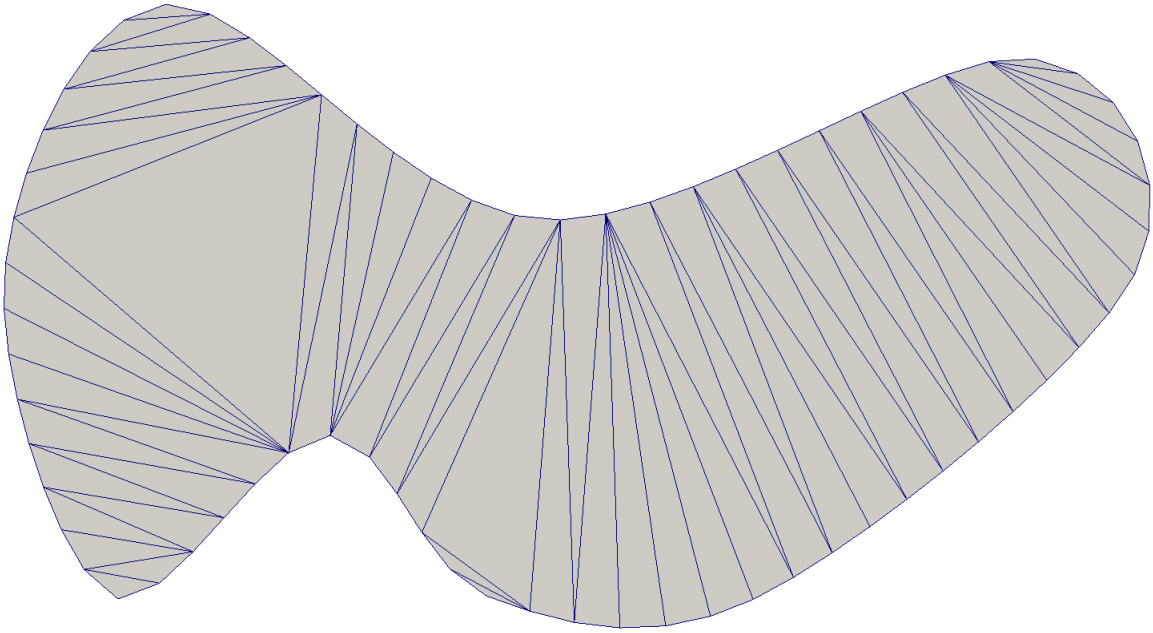


FIGURE 6.3 – Création du maillage triangulaire et création du premier front.

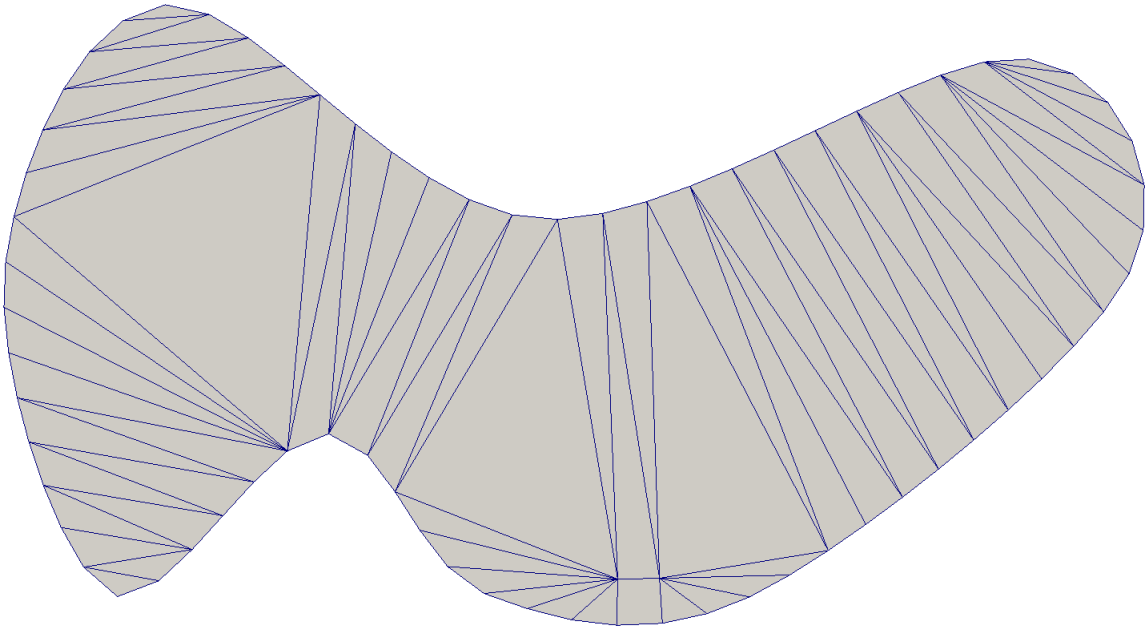


FIGURE 6.4 – Création du premier quadrangle avant la mise à jour du maillage triangulaire.

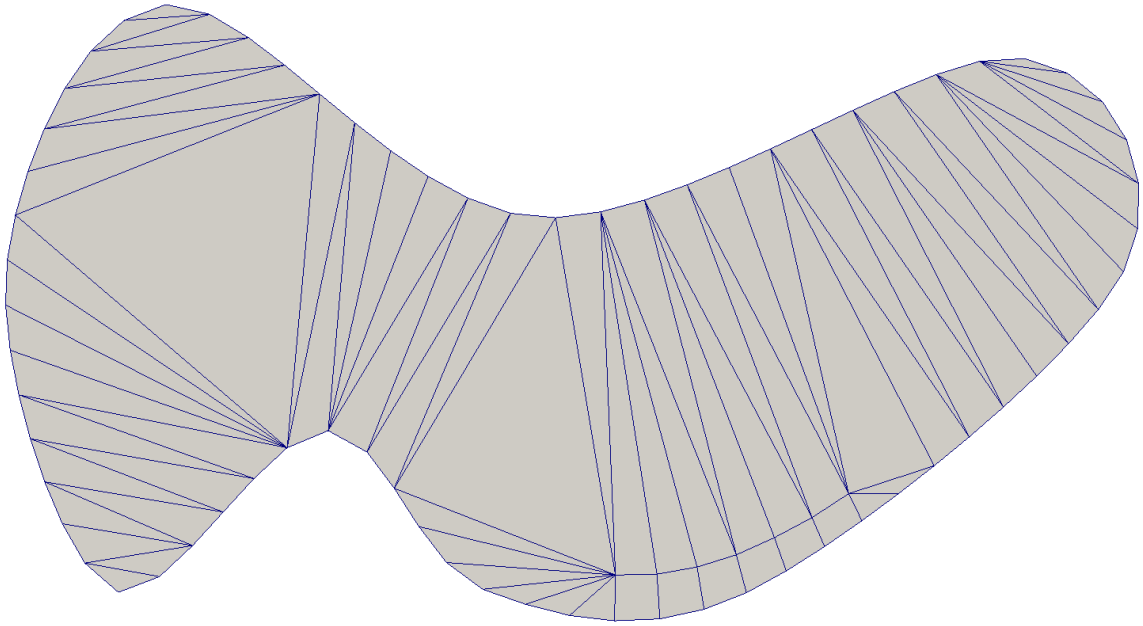


FIGURE 6.5 – Début de création de la première rangée.

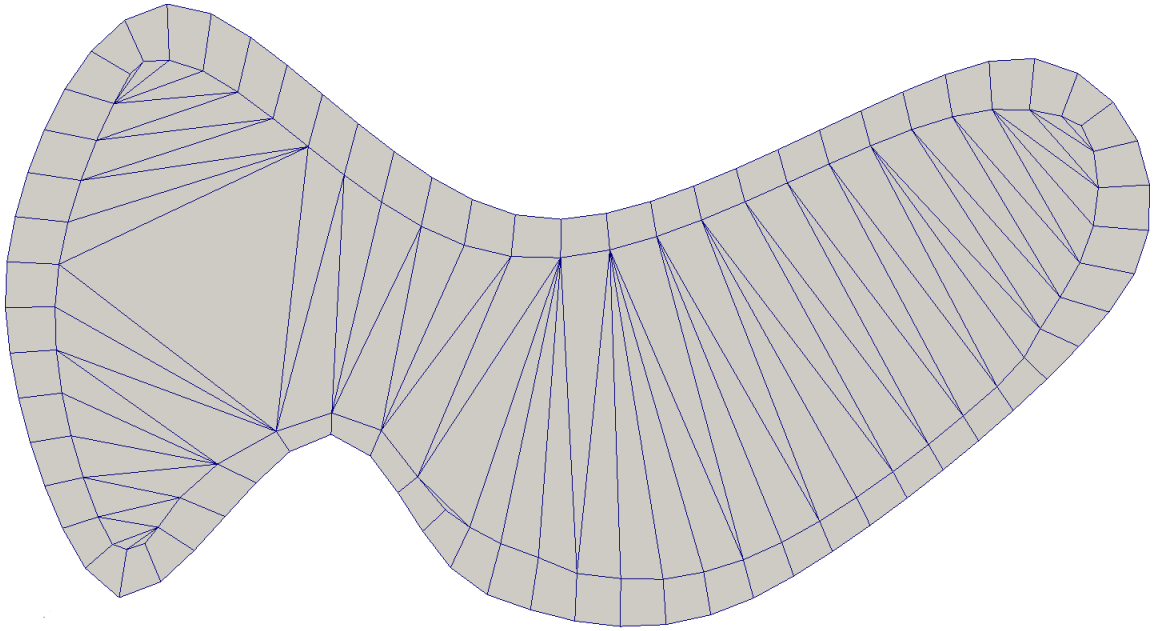


FIGURE 6.6 – La première rangée est créée.

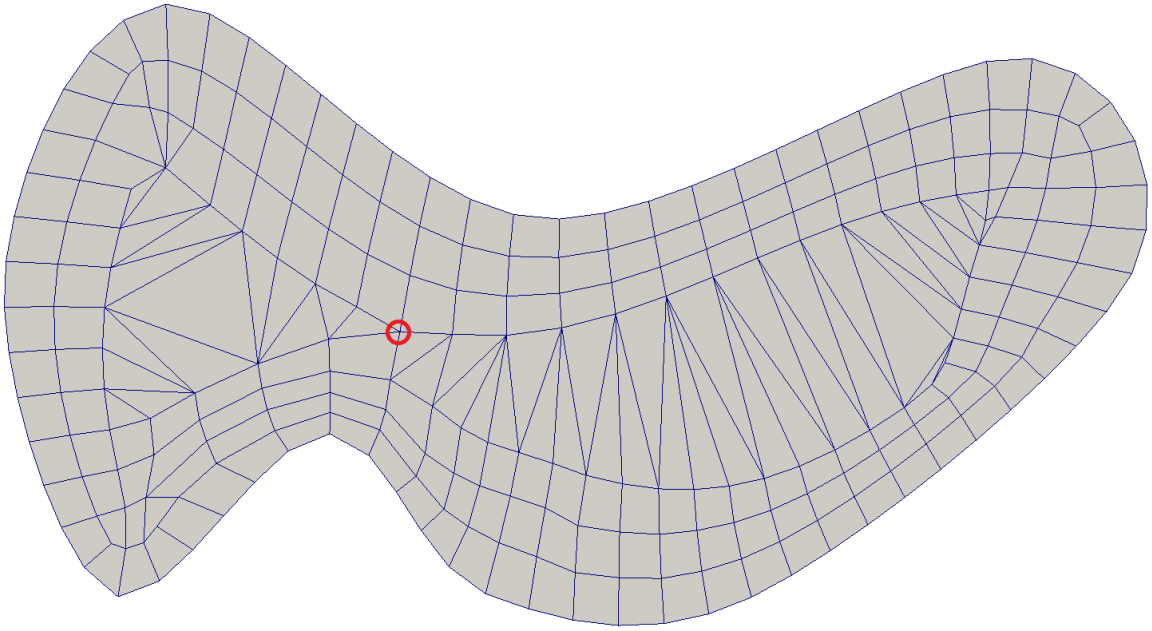


FIGURE 6.7 – Séparation du front courant en deux fronts.

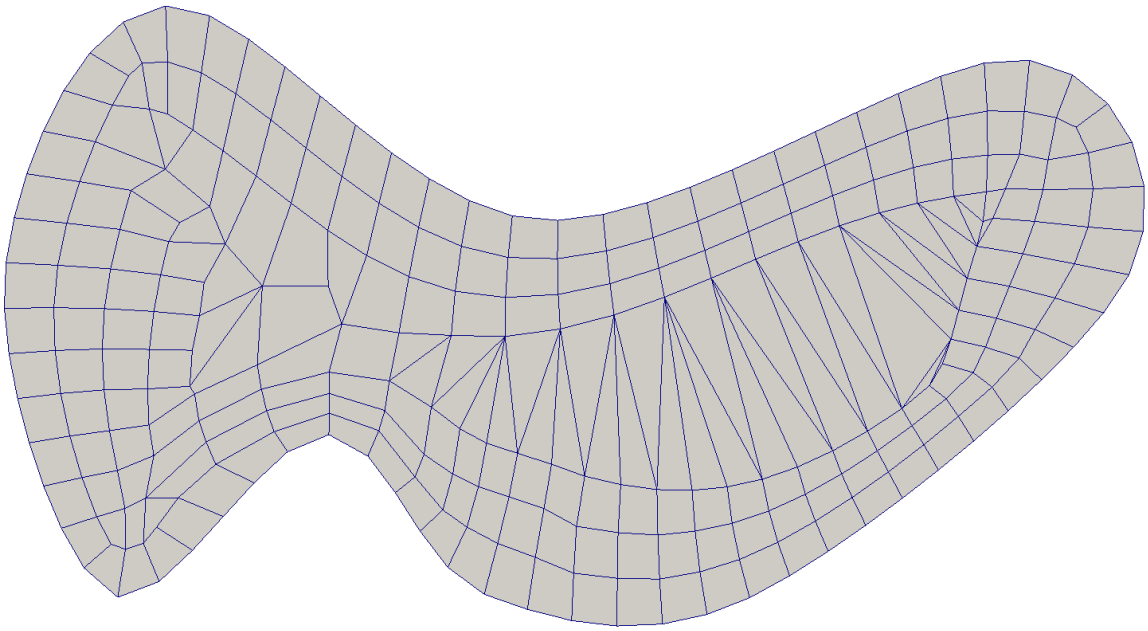


FIGURE 6.8 – Un des fronts a finit d'être traité.

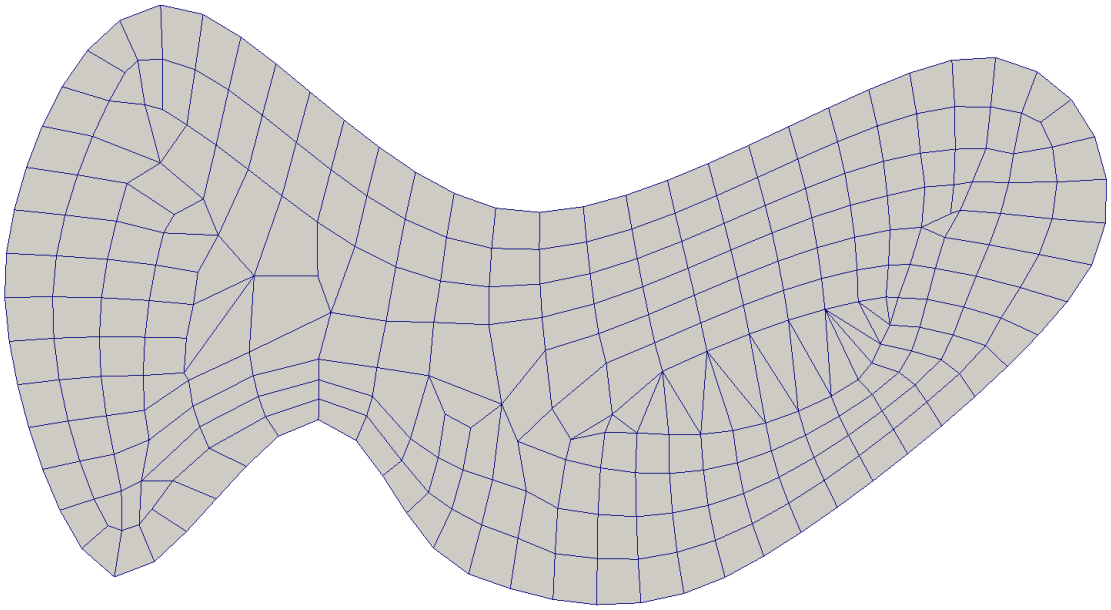


FIGURE 6.9 – Début du traitement du second front.

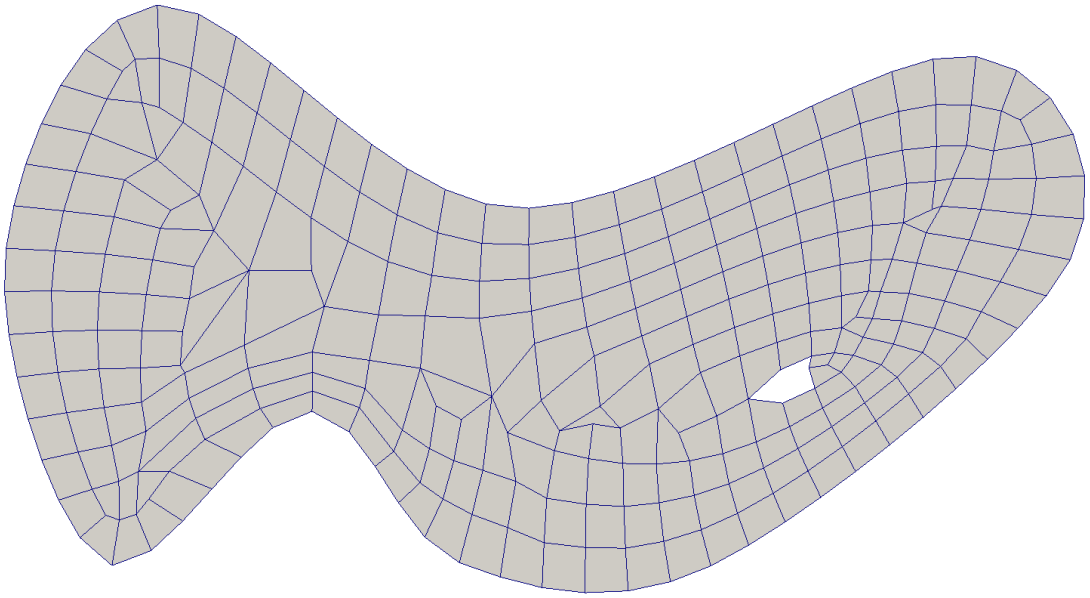


FIGURE 6.10 – Fin de l'ajout de rangées dans le second front.

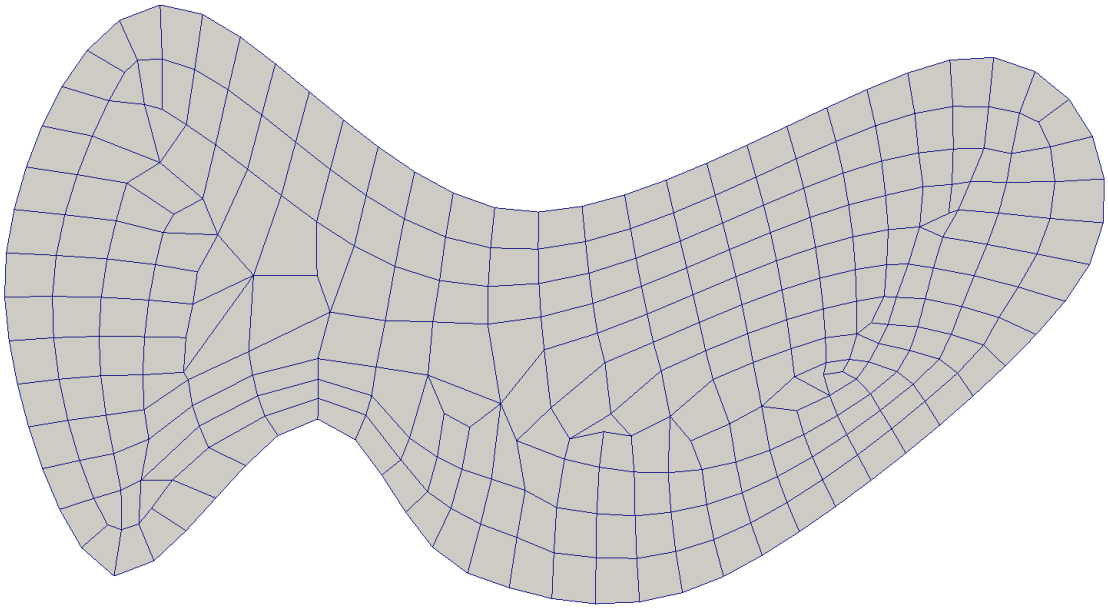


FIGURE 6.11 – Application d'un motif de résolution pour les fronts à 6 nœuds ou moins.

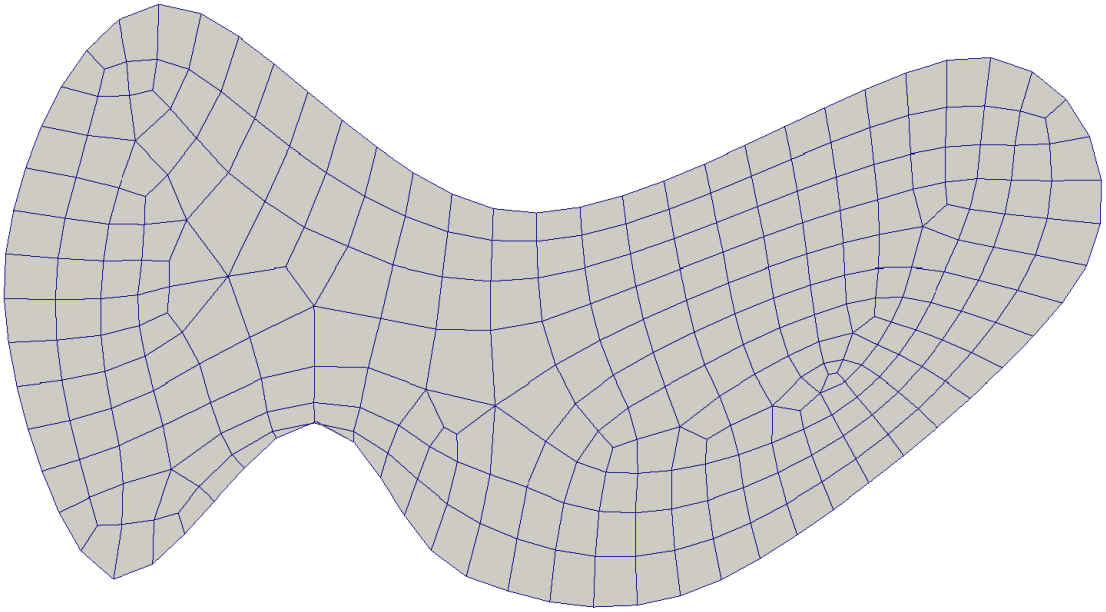


FIGURE 6.12 – Lissage géométrique simple.