Développement d'une bibliothèque de structures et algorithmes pour un mailleur polyédrique

Alexandre Marin
Master Mathématiques et Applications Sorbonne Université
Parcours Ingénierie Mathématique
Majeure Ingénierie Mathématique Pour l'Entreprise

24/06 – 30/10/2020
IFP Énergies Nouvelles
Site de Rueil-Malmaison

Encadrants :
Laurent Astart
Alexandra Bac

Remerciements

Je remercie Laurent Astart de m'avoir accueilli à IFP Énergies Nouvelles pour ce stage de fin d'études et je remercie Alexandra Bac de m'avoir co-encadré.

Table des matières

1 Introduction

Ce document est le rapport de mon stage qui a eu lieu à IFP Énergies Nouvelles (IFPEN), et clôture donc ma seconde année de master à Sorbonne Université.

Ma position de stagiaire a été la suivante : j'ai été intégré dans la direction scientifique *Sciences et Technologies du Numérique*, et plus précisément dans le département *Informatique Scientifique* du site de Rueil-Malmaison de IFPEN.

Ce rapport présente d'abord succinctement le déroulement du stage, expose ensuite son contexte en présentant l'entreprise, puis la mission du stage est détaillée et une quatrième partie apporte la conclusion.

Les informations qui concernent IFPEN ont été recueillies sur son site internet et sur l'intranet de l'entreprise.

2 Déroulement du stage

Le premier mois du stage a été consacré à la découverte des locaux, à certaines formations obligatoires pour les nouveaux salariés (comme la formation sécurité) et à l'installation de logiciels nécessaires pour le stage. À la fin du mois de juillet, j'avais pris en main l'environnement de développement intégré QtCreator, j'avais développé en C++ une structure de données en demi-arêtes pour des maillages bidimensionnels et quelques classes outil pour générer des maillages de Delaunay.

Pendant le deuxième mois, le code C++ a été réorganisé dans le but d'ajouter plus facilement d'autres fonctionnalités comme le calcul de triangulations de Delaunay contraintes. De nombreux tests ont aussi été effectués pour vérifier la stabilité des programmes ainsi que la qualité des résultats.

Le rapport de stage a été composé lors du troisième mois. Entretemps, des diagrammes de Voronoï étaient générés et le code était amélioré.

Quant au dernier mois du stage, il a permis une seconde étude bibliographique portant, entre autres, sur des structures de données adaptées à des maillages polyédriques 3D.

3 Contexte et données générales

Cette section a été composée à partir d'informations récupérées sur le site internet de IFPEN et sur son réseau intranet.

IFP Énergies Nouvelles [1] (IFPEN) est une entreprise du groupe du même nom : il s'agit d'un établissement public de recherche et de formation dans les domaines de l'énergie, du transport et de l'environnement, placé sous la tutelle du ministre chargé de l'énergie. Plus précisément, IFPEN est depuis 2006 un établissement public à caractère industriel et commercial (EPIC) [2]. IFPEN est financé environ à moitié par l'État et crée lui-même plus de 50% de son budget, notamment en valorisant le travail issu de la recherche. L'entreprise est présente sur deux sites : l'un à Rueil-Malmaison en Île-de-France et l'autre à Solaize en Auvergne-Rhône-Alpes.

En 2019, IFPEN comptait 1 633 salariés à temps plein, dont 1 136 ingénieurs et techniciens R&I (Recherche et Innovation), ainsi que près de 200 allocataires de recherche, postdoctorants et stagiaires. Son budget atteignait $283,3$ M€. IFPEN déposait aussi 185 brevets et était à l'origine de plus de 600 publications scientifiques et communications à congrès.

IFPEN conçoit et développe des procédés, des équipements et des logiciels qui concernent quatre domaines : la mobilité durable, les énergies renouvelables, les hydrocarbures responsables, et le climat/l'environnement. Dans le domaine climat et environnement, l'une des problématiques est par exemple le captage, le stockage et l'utilisation du CO_2. Il s'agit de proposer aux industries lourdes (sidérurgie, cimenterie, raffinage, chimie, pétrochimie) des technologies pour réduire massivement leurs émissions de CO_2.

IFPEN collabore avec d'autres instituts de recherche et industries dans le cadre de partenariats qui sont de différents types. IFPEN est ses partenaires financent conjointement un projet de recherche et définissent les règles de propriétés des résultats grâce aux contrats de recherches bilatéraux ou aux consortiums dont certains sont des Joint Industry Project (JIP) : IFPEN y opère le programme R&I et conserve la propriété industrielle. IFPEN participe aussi à des projets de recherche collaborative qui bénéficient de soutiens publics.

La plupart des entités de IFPEN, que l'on peut voir sur la figure 3, se répartissent dans deux ensembles : *les directions de recherche*, qui rassemblent

1. L'ancienne dénomination était Institut Français du Pétrole jusqu'en 2010.
2. D'après l'article Wikipédia sur les EPICs, il s'agit « d'une personne morale de droit public ayant pour but la gestion d'une activité de service public de nature industrielle et commerciale ».

les compétences scientifiques, et *les directions fonctionnelles*, comme les ressources humaines, les finances ou le juridique.

Les programmes de recherche sont menés à travers des projets pouvant faire intervenir plusieurs directions de recherche et fonctionnelles. Ces mêmes projets sont pilotés par l'un des cinq centres de résultats qui s'occupent également de leur valorisation industrielle.

On trouve aussi dans l'entreprise une école d'ingénieurs, IFP School, ainsi que la direction générale. Il y a enfin un conseil d'administration, composé de seize membres dont quatre sont des représentants de l'État venant des ministères de l'énergie, de l'industrie, du budget et de la recherche.

4 Mission du stage

4.1 Objectifs

Ce stage sert de prélude à la thèse « *Maillage polyédrique de volumes 3D optimisé pour la simulation en géosciences* » : l'objectif de cette thèse est de proposer et confronter plusieurs stratégies de génération d'un maillage polyédrique du sous-sol à de nouveaux types de schémas numériques. Afin de rendre possibles des simulations concernant les problématiques du stockage du CO_2, la géothermie ou l'hydrogéologie, il faut représenter le plus fidèlement possible les discontinuités et hétérogénéités de la structure des sous-sols qui influencent beaucoup les transferts de masse et d'énergie. Il s'agit donc, étant donnée une description surfacique des éléments en jeu, de mailler un volume en respectant certaines contraintes, comme la conformité ou les rapports de forme des mailles.

Le premier objectif était, d'une part, d'effectuer un travail bibliographique sur la génération de maillages, et d'autre part, de créer une bibliothèque logicielle qui fournisse un maximum d'algorithmes et les structures de données nécessaires au développement du mailleur.

Cependant le véritable but du stage était l'appropriation de concepts liés à la génération de maillages 2D pour ensuite mieux saisir ceux relatifs à la génération de maillages 3D polyédriques.

4.2 Travail effectué

Éléments théoriques pour les maillages

On présente ici les triangulations de Delaunay, les triangulations de Delaunay contraintes pour mailler un domaine non nécessairement convexe, puis le dual des ces triangulations qui sont une piste pour l'obtention d'un maillage polyédrique ayant de bonnes propriétés.

Une *triangulation de Delaunay* d'un ensemble de points $V \subset \mathbb{R}^2$ est un maillage conforme de l'enveloppe convexe de V, constitué de triangles satisfaisant la propriété du cercle vide : le disque ouvert défini par le cercle circonscrit d'un triangle ne doit contenir aucun sommet de V. Ces triangles sont dits alors Delaunay. Parmi toutes les triangulations de V, celles qui sont Delaunay maximisent la valeur du plus petit angle formé par deux arêtes : ce sont donc, pour cet ensemble V et du point de vue de la qualité des mailles, les « meilleures » triangulations. On peut voir un maillage Delaunay sur la figure 4.

La propriété de ces maillages s'exprime aussi localement : un maillage est Delaunay si et seulement si chaque arête est *localement Delaunay*, *i.e.* si chaque arête e est au bord ou bien est la frontière commune de deux triangles τ_1 et τ_2 tels que le cercle circonscrit de τ_1 n'englobe aucun sommet de τ_2. Dans ce dernier cas, cela revient à dire qu'un cercle passant par les extrémités de e n'englobe aucun sommet de τ_1 et de τ_2. Une illustration de la propriété d'être localement Delaunay pour une arête se trouve sur la figure 5. On y aperçoit les mêmes points et presque les mêmes arêtes à gauche et à droite, cependant l'arête e ne relie pas les mêmes sommets. À gauche, un cercle passant par les deux extrémités de $e = \mathrm{AC}$ n'englobe ni B ni D, donc e est localement Delaunay (les arêtes et sommets n'appartenant pas aux triangles ACD et ABC ne doivent pas être pris en compte). Quant au dessin de droite, le cercle circonscrit du triangle ABD englobe C, donc $e = \mathrm{BD}$ n'est pas localement Delaunay.

Lorsque l'on souhaite faire apparaître des arêtes spécifiques d'un ensemble L dans le maillage à obtenir, on peut produire des maillages *Delaunay contraints* : il suffit pour cela de créer des maillages dont chaque arête est soit dans L, soit localement Delaunay. Les éléments de L sont appelés des *segments*. Plutôt que de mailler l'enveloppe convexe d'un nuage de point, on peut alors mailler un domaine qui n'est plus convexe et qui possède potentiellement des trous. Un triangle est dit *Delaunay contraint* s'il satisfait la condition suivante : si son cercle circonscrit englobe un autre sommet du maillage, le triangle est « caché » de ce sommet par un segment.

Le *diagramme de Voronoï* d'un ensemble fini de points $V \subset \mathbb{R}^2$ est constitué d'un ensemble de cellules V_p pour tout $p \in V$, définies par

$$V_p \;:=\; \left\{ x \in \mathbb{R}^2 \,\middle|\, \forall\, q \in V,\ \|x - p\| \leqslant \|x - q\| \right\}.$$

Il s'agit du dual de la triangulation de Delaunay $\mathcal{T}$ de V : le diagramme de Voronoï relie les centres de cercle circonscrit des triangles de $\mathcal{T}$. Il existe toujours cependant des cellules de Voronoï qui sont infinies et délimitées aussi par des bissectrices des arêtes du bord de $\mathcal{T}$. De façon générale, les cellules V_p sont des polygones convexes.

Lorsque l'on définit en plus l'ensemble des segments L qui doivent apparaître dans le maillage $\mathcal{T}$, on s'intéresse plutôt au diagramme de Voronoï *étendu* $\mathcal{V}$, défini dans [1, pages 30-31]. Ici on ne représente que la « feuille primale » : on ne fait pas figurer des arêtes qui relient des centres de cercle circonscrit dans le diagramme $\mathcal{V}$ si elles intersectent un segment de L. Les sommets des cellules de Voronoï peuvent alors être des milieux de segments ou des intersections de bissectrices d'arêtes de $\mathcal{T}$ avec d'autres arêtes de $\mathcal{T}$.

Un exemple de diagramme de Voronoï obtenu avec la bibliothèque développée est visible sur la figure 7.

Structure de données en demi-arêtes

Les maillages non structurés peuvent être représentés dans une structure compacte qui ne contient que les points ainsi que la liste des faces, elles-mêmes vues comme des listes de sommets. Mais une requête concernant des informations topologiques du maillage demande une préparation d'une autre structure de données représentant les adjacences, ce qui constitue un inconvénient majeur lorsque l'on veut des algorithmes efficaces : par exemple, toute modification du maillage demande une mise à jour des informations d'adjacences.

Afin d'obtenir rapidement des données sur la *topologie* du maillage, la structure de données en demi-arêtes a été mise en œuvre : cela consiste à représenter le maillage par un graphe plan orienté. Chaque arête donne naissance à deux demi-arêtes, opposées l'une à l'autre, chacune pointant vers l'une des extrémités de l'arête. Pour se déplacer à l'intérieur de la structure de données, chaque demi-arête he intérieure à une face f pointe vers une autre demi-arête qui part de la destination de he et qui est intérieure à f, et chaque demi-arête correspondant au bord est considérée comme intérieure à une face fictive de « peau » ; de plus chaque demi-arête fait référence à celle qui lui est opposée. Enfin chaque sommet v pointe vers une demi-arête qui part de v.

Ainsi, il est aisé de se déplacer dans le maillage et l'on a accès instantanément à des informations topologiques. Par exemple, on peut facilement parcourir une partie de bord du maillage, connaître les voisins d'un sommet et les arêtes qui lui sont incidentes.

Les algorithmes de génération de maillage ont été écrits en tenant compte des avantages de cette structure.

Structure de données accélératrice

Les fonctions programmées étant censées générer de « grands » maillages, des nuages de points très denses doivent être constitués : la complexité temporelle des algorithmes prend alors toute son importance. Par exemple, pour l'algorithme de Bowyer-Watson présenté en page 8, la complexité dépend aussi d'une procédure de localisation de points dans une face du maillage ; dans le cas d'une recherche exhaustive, la complexité de l'étape de localisation est en $O(t)$, avec t le nombre de triangles du maillage. On décide de remplacer cette procédure coûteuse en s'aidant d'un *quadtree*, *i.e.* un arbre

quaternaire.

Le principe de la recherche est le suivant : au préalable, on part d'une boîte B censée englober le maillage et on convient du nombre maximal d'éléments pouvant être contenus dans une feuille de l'arbre. Chaque nœud de l'arbre représente une région rectangulaire qui est une portion de la boîte englobante B. Lorsque qu'une feuille contient trop d'éléments, on lui crée quatre fils qui correspondent chacun à un quart de la zone représentée par l'ancienne feuille, puis on répartit les éléments dans les nouvelles feuilles selon leur position dans la région. Ainsi, la complexité de la recherche d'un élément selon sa position est en $O(h_{\text{quadtree}})$, avec h_{quadtree} la hauteur de l'arbre construit.

Une illustration d'un quadtree se trouve en figure 6.

Algorithmique

On peut citer quatre grandes familles d'algorithmes de génération de maillage :

les méthodes d'avancée de front : les éléments sont intégrés un par un, en partant du bord du domaine afin de progresser vers son centre ;

les procédures de raffinement : elles insèrent soigneusement de nouveaux sommets afin de respecter des contraintes sur la qualité et la taille des éléments, tout en gardant une propriété du maillage de départ (comme celle d'être Delaunay) ;

les algorithmes de grilles : à l'aide d'une grille, d'un quadtree ou d'un octree, une grille structurée est représentée pour servir de guide lors de la subdivision du domaine à mailler ;

les algorithmes d'amélioration de maillage : ils effectuent des opérations visant à optimiser localement un premier maillage.

Dans la bibliothèque à concevoir, on a ajouté des algorithmes qui construisent des maillages de Delaunay, contraints ou non. On y trouve les deux algorithmes principaux suivants.

Algorithme de Bowyer-Watson : cette procédure transforme un maillage Delaunay en une autre triangulation de Delaunay en insérant un nouveau sommet, par retriangulation locale. Les étapes sont :

1. localisation du nouveau sommet s : trouver un triangle τ dont le cercle circonscrit contient s ;
2. à partir de τ, trouver de proche en proche tous les triangles qui ne satisfont plus la propriété du cercle vide à cause de s ;
3. suppression des triangles trouvés ;

4. retriangulation de la cavité formée : chaque nouveau triangle doit avoir s pour sommet et avoir une arête de la cavité pour côté.

Avec une procédure de localisation efficace, on peut espérer une complexité en $O(n \log n)$, avec n le nombre de sommets du maillage.

Emballage cadeau (*Gift-Wrapping*) : c'est un algorithme issu de la première classe citée qui maille un domaine défini par des polygones, des arêtes et des points, les éléments des deux derniers types devant apparaître dans le résultat. Le principe consiste à ajouter au fur et à mesure des triangles Delaunay contraints. Pour cela, on considère un ensemble $\mathcal{A}$ contenant des arêtes (orientées), initialisé au début avec les arêtes du bord du domaine. L'ensemble $\mathcal{A}$ est mis à jour à chaque insertion de triangle en fonction de ses côtés. L'algorithme prend fin lorsque $\mathcal{A}$ est vide, sinon une autre itération commence avec une arête $e \in \mathcal{A}$ et l'on cherche un sommet en entrée qui va former avec e un triangle Delaunay contraint.

S'il y a n sommets et m segments à considérer en entrée, la complexité de cet algorithme est en $O(n^2 m)$.

Variantes : on peut s'inspirer des algorithmes précédents pour insérer autrement un nouveau sommet dans une triangulation Delaunay (contrainte ou non) tout en conservant la propriété de Delaunay.

Pour insérer un sommet à l'extérieur d'un maillage Delaunay, il suffit d'ajouter certains triangles au bord du maillage à modifier et d'appeler la procédure Bowyer-Watson dans un ancien triangle du bord qui ne satisfait plus la propriété du cercle vide.

Pour ajouter un nouveau sommet dans une triangulation de Delaunay contrainte, il suffit d'invoquer l'algorithme de Bowyer-Watson mais en ne permettant à la procédure de se déplacer uniquement de triangle en triangle adjacent si l'arête frontière à traverser n'est pas un segment. De plus un cas particulier doit être traité : lorsque le nouveau sommet à insérer se situe sur un segment, il faut aussi diviser ce segment en deux autres segments et penser à appeler la procédure de Bowyer-Watson de part et d'autre de l'ancien segment.

Enfin, on peut ajouter un segment l à une triangulation de Delaunay contrainte en supprimant toutes les arêtes dont l'intérieur intersecte l puis en appliquant l'algorithme d'emballage cadeau aux deux cavités formées qui se touchent selon l.

Bibliographie et mise en œuvre informatique

La bibliographie servant de point de départ se trouve à la fin du rapport, en page 20. L'article [2] décrit une procédure qui génère des maillages de volumes séparés par des interfaces, à partir des diagrammes de Voronoï. Puisqu'il faut au préalable déterminer des triangulations de Delaunay, les notes [3] ont été lues pour avoir à disposition des algorithmes de génération de maillages de Delaunay. Le livre [1] complète les deux premières références, en donnant par exemple la définition des diagrammes de Voronoï étendus pour les maillages de Delaunay contraints.

La bibliothèque logicielle à créer a été programmée en C++ à l'aide de l'environnement de développement intégré QtCreator [9], sous Windows. La compilation y est automatisée grâce à des fichiers de type `Makefile`, générés après l'analyse par le programme `QMake` de fichiers projets `.pro` (un par sous-projet et un pour le projet global). Cette bibliothèque est découpée en plusieurs sous-projets que l'on présente ici :

noyau géométrique (GeometricCore) : cela permet d'effectuer des calculs en géométrie. La bibliothèque Eigen [5] écrite en C++ a été incorporée au projet pour faire de l'algèbre linéaire ;

maillages bidimensionnels (MeshCore) : on utilise la structure en demi-arêtes pour représenter les maillages dans le plan. D'autres classes modélisant des concepts ayant traits aux maillages y figurent, comme des propriétés concernant des faces ou des arêtes ;

entrées/sorties (MeshIO) : afin de visualiser ou réutiliser des maillages, l'importation et/ou l'exportation a été codée pour les formats `OBJ`, `PLY` et `XDMF`[3] ;

Delaunay : ce projet fournit des algorithmes pour générer des triangulations de Delaunay et de Delaunay contraintes ;

Voronoï : ce projet permet de calculer des diagrammes de Voronoï ;

Test : dans ce dernier sous-projet, on y prépare des cas tests pour valider les programmes et pour s'assurer que ces derniers sont stables, rapides et/ou suffisamment précis.

Un certain nombre de mécanismes du C++ ont été utilisés pour ajouter de la clarté au projet. Chaque module de la bibliothèque contient plusieurs

3. `XDMF` (eXtensible Data Model and Format) est un dialecte de `XML` permettant de renseigner des polyèdres généraux. Le format est décrit à l'adresse http://www.xdmf.org/index.php/XDMF_Model_and_Format .

fichiers sources et en-têtes où sont définies des classes, les interactions entre ces dernières étant régies par l'ajout de méthodes, le réglage des visibilités des membres de classes, ou encore par une technique de création de clé, nécessaire pour invoquer une méthode, mais que seules certaines classes peuvent se procurer. Dans la mesure du possible, les concepts étaient modélisés par des classes ; les méthodes qui prenaient en argument une fonction étaient codées comme des patrons que l'on utilisait par exemple grâce à des *lambda*.

La classe maillage manipule des indices pour référencer les sommets, demi-arêtes et faces, or la structure de données évolue beaucoup à cause d'ajouts et suppressions répétés d'éléments. Il a fallu donc avoir à disposition une classe « vecteur à trous » qui gère elle-même la modification du contenu d'une zone mémoire contiguë : les cases vides sont réutilisées dès que possible et les indices permettant de retrouver les éléments des vecteurs restent inchangés pendant tout le programme.

Les tests se sont avérés essentiels : un fichier de test par sous-projet a été écrit. On y appelle les méthodes publiques des classes principales pour construire par exemple des maillages simples, mais de temps en temps dans la mesure du possible on essaye de considérer plusieurs cas limites. On prépare aussi des *tests unitaires* pour vérifier le bon fonctionnement d'algorithmes et structures de données, indépendamment des autres éléments du programme. Par exemple, différentes configurations ont été considérées pour vérifier si le programme parvenait à savoir quand deux segments ouverts s'intersectaient.

La gestion de version s'est faite grâce à l'outil Git [6]. L'architecture du projet est visible sur la figure 1 et quelques données sur le projet sont regroupées dans la figure 2.

Seconde étude bibliographique

D'autres références à propos des maillages ont été lues : certaines décrivent une structure de données pour représenter un maillage polyédrique, d'autres proposent des méthodes de génération de maillages 3D destinés à des simulations en géosciences. Cette seconde bibliographie est donnée en page 21.

4.3 Résultats

Des images sont fournies en annexe pour illustrer quelques résultats issus de l'exécution des tests. On montre des triangulations de Delaunay, Delaunay contraintes ainsi que des diagrammes de Voronoï sur les figures 7 et 8. Une

triangulation de Delaunay contenant 100 000 points tirés aléatoirement peut être engendrée en moins de 40 secondes.

4.4 Problèmes rencontrés

La principale difficulté a été la vérification des fonctions programmées par la visualisation des résultats obtenus. Différents logiciels ont été utilisés, comme Paraview, CloudCompare ou MeshLab [8, 4, 7], mais ils n'interprètent pas tous les mêmes formats de fichier et ils peuvent rendre compte des maillages très différemment. Par exemple, le format `PLY` permet de renseigner des couleurs attachées à des points, arêtes et faces ; cependant ces données ne sont pas toujours acceptées et souvent les éléments sont coloriés par interpolation des couleurs des extrémités, ce qui ne correspond pas tout le temps à ce que l'on veut obtenir. Néanmoins ces couleurs se sont révélées pratiques quand il a fallu afficher des propriétés comme celle d'être localement Delaunay pour une arête.

De surcroît, certaines fonctions ne peuvent pas être visualisées simplement. Pour la structure en demi-arêtes, un bogue a été découvert pendant le deuxième mois du stage. En effet, un cas particulier n'avait pas été traité lorsque deux faces doivent être collées, et la connectivité était perdue parce qu'on ne pouvait parfois plus déduire toutes les arêtes incidentes à un sommet à partir de ce dernier.

Une dernière difficulté était l'adaptation d'algorithmes qui étaient exacts en théorie mais qui ne fonctionnaient pas bien en pratique à cause de l'arithmétique des nombres à virgule flottante. Quand un sommet à ajouter avec l'algorithme de Bowyer-Watson se situait très près d'un segment, il fallait prendre une décision : soit ajouter un triangle presque dégénéré, soit diviser le segment en deux autres segments. Or il n'est pas simple de savoir si le nouveau sommet appartient au segment ou pas, et comme des triangles très plats peuvent quand même apparaître dans une triangulation de Delaunay, il n'est pas non plus évident de savoir quand un triangle doit être jugé « trop plat ».

5 Conclusion

Le travail effectué pendant ce stage m'a permis, d'une part, de me perfectionner en programmation en réfléchissant pendant plusieurs mois à l'élaboration d'une bibliothèque, et d'autre part, de découvrir des méthodes pour générer et représenter des maillages 2D et 3D.

Grâce aux difficultés rencontrées, théoriques et pratiques, je peux à présent mieux mesurer la complexité de la mission dans laquelle s'inscrit la thèse correspondante. Cette dernière sera l'occasion de trouver de bonnes propriétés pour des maillages 3D polyédriques, afin de résoudre des équations aux dérivées partielles modélisant des écoulements en milieu poreux.

Enfin, ce stage constitue une expérience professionnelle supplémentaire qui m'a fait connaître une entreprise dont les missions liées au domaines de l'énergie et de l'environnement sont d'actualité et font partie d'un secteur d'avenir.

6 Annexes

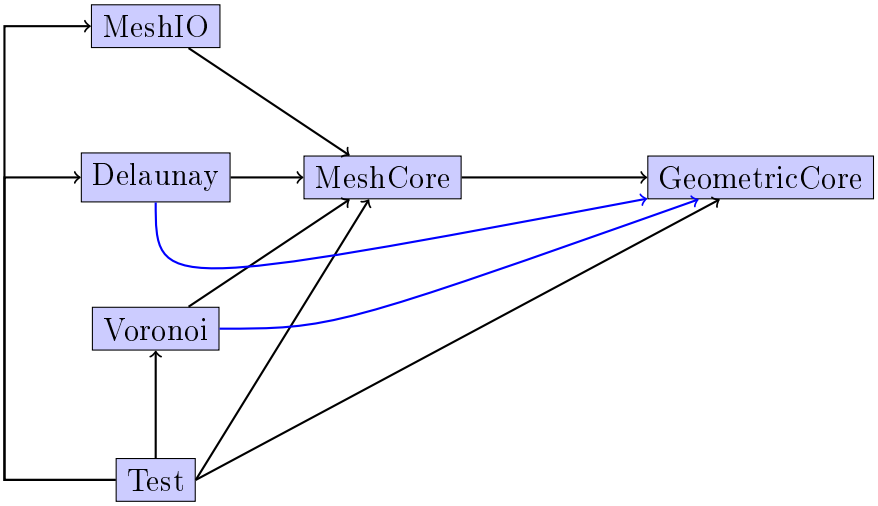

FIGURE 1 – Architecture du projet : les flèches indiquent les dépendances d'un sous-projet à un autre, qu'il faut considérer lors de la compilation. Le sous-projet `Test` permet de construire un exécutable tandis que les autres sous-projets correspondent à des bibliothèques.

Fichiers C++ écrits	29
Lignes	≈ 5000
Classes	41
Nombre moyen de méthodes par classe	$7,8$

FIGURE 2 – Quelques statistiques sur le code produit.

FIGURE 3 – Schéma de l'organisation de IFPEN (source : intranet de IFPEN).

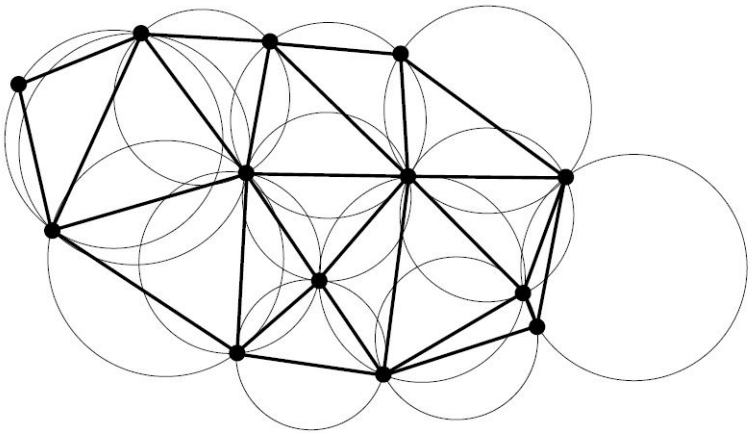

FIGURE 4 – Figure 2.3 de la source [3] : exemple de triangulation de Delaunay. Les cercles circonscrits des triangles sont dessinés et permettent de voir que chaque triangle possède bien la propriété du cercle vide.

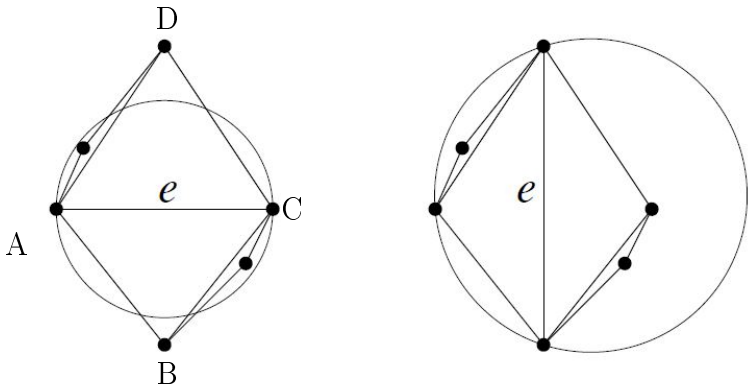

FIGURE 5 – Figure 2.7 de la source [3] : l'arête e est localement Delaunay à gauche mais ne l'est pas à droite.

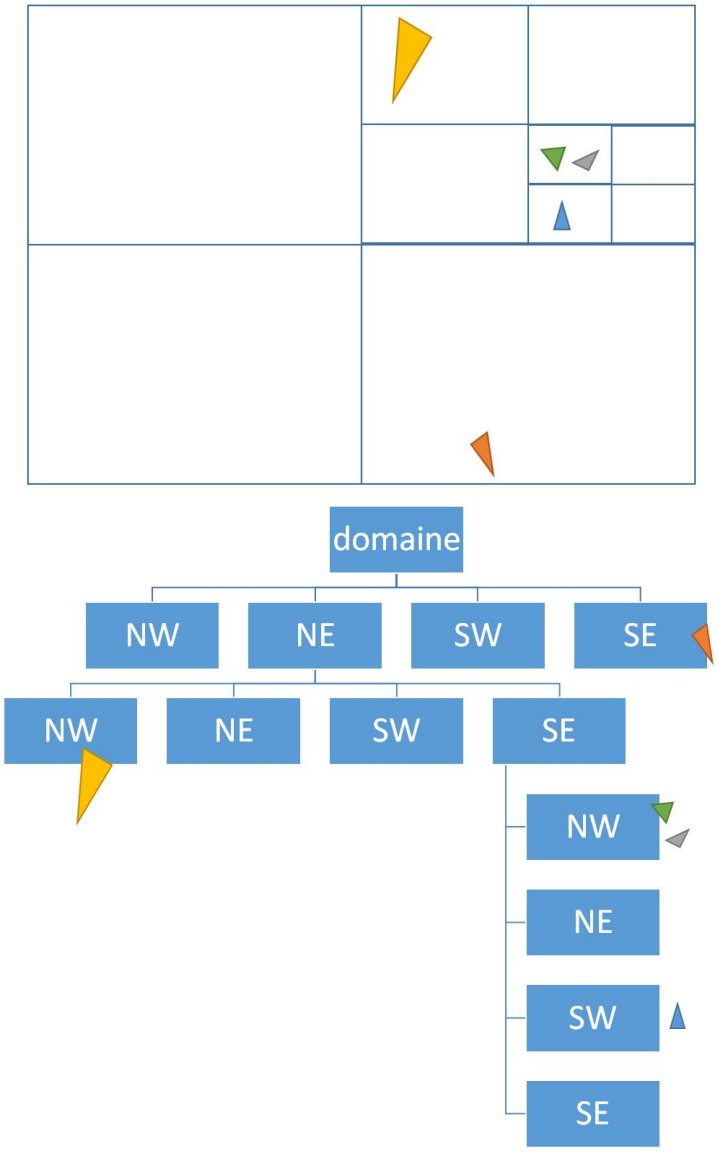

FIGURE 6 – Exemple de quadtree : en haut, des faces avec des boîtes englobantes sont illustrées, et un schéma de l'arbre correspondant se trouve en bas. La racine est associée à la plus grande boîte rectangulaire tandis que les autres nœuds symbolisent des sous-régions. Deux points cardinaux sont utilisés pour identifier le quart de la boîte associée au nœud parent. Chaque triangle est mis à proximité de la feuille dans laquelle il est contenu.

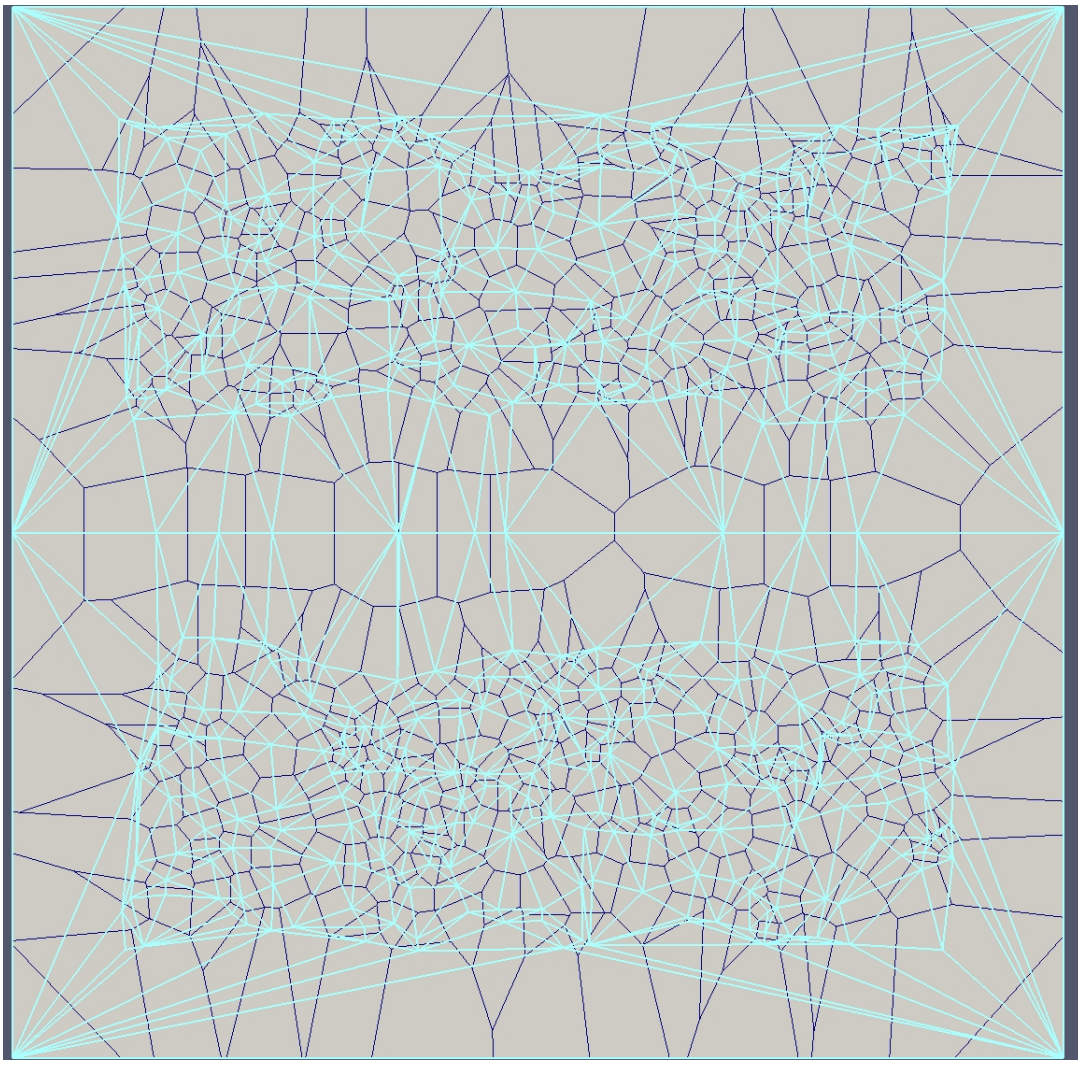

FIGURE 7 – Triangulation de Delaunay d'un carré avec segments disposés à la médiane horizontale (arêtes en bleu clair) et diagramme de Voronoï correspondant (arêtes en bleu foncé) (image obtenue avec Paraview).

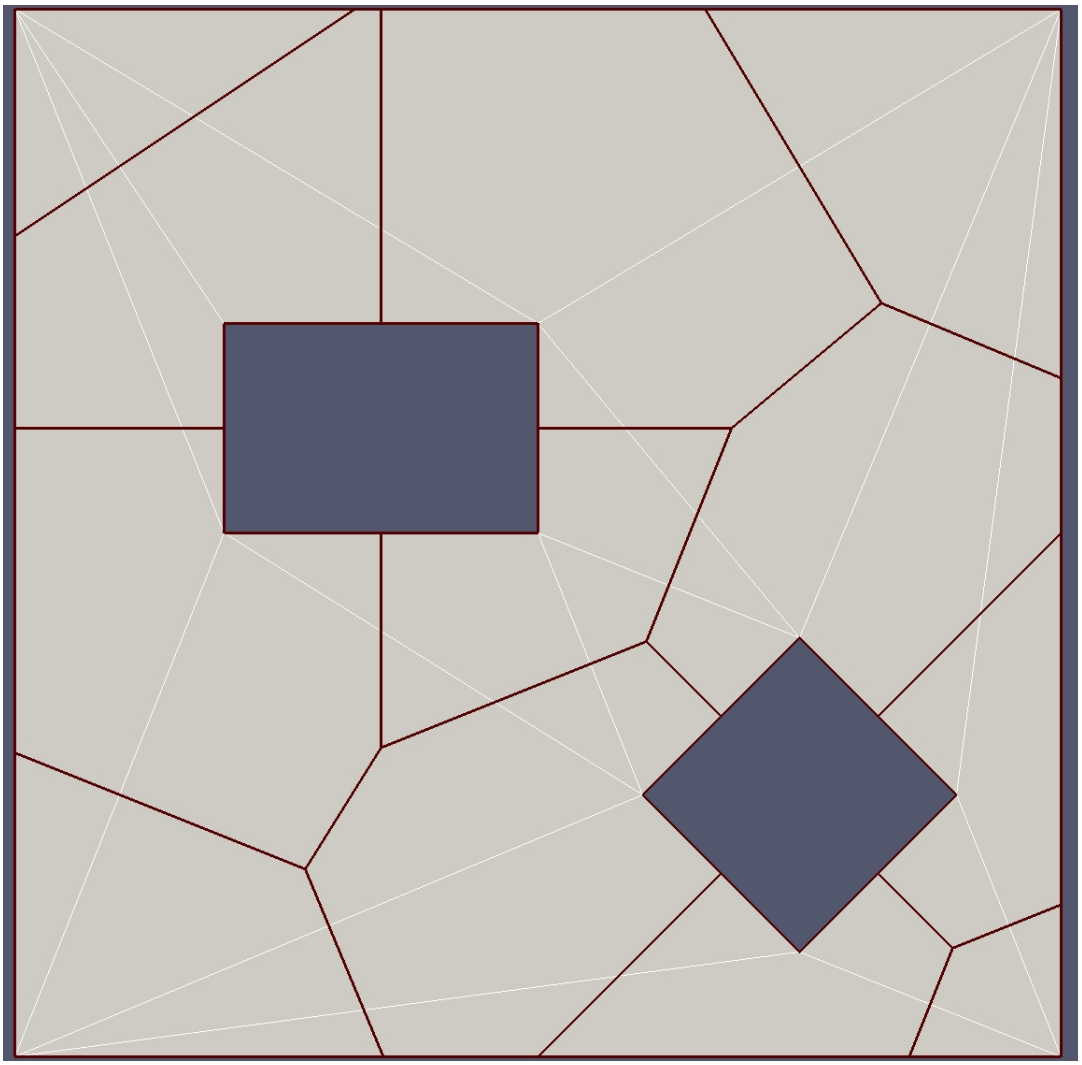

FIGURE 8 – Domaine non convexe (en gris) : triangulation de Delaunay (arêtes en blanc) et diagramme de Voronoï étendu correspondant (arêtes en marron) (image obtenue avec Paraview). Les cellules de Voronoï ne sont pas nécessairement convexes.

7 Références

Bibliographie

[1] Herbert Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2001.

[2] Rao Garimella, Jibum Kim, and Markus Berndt. Polyhedral Mesh Generation and Optimization for Non-manifold Domains. 10 2013.

[3] Jonathan Richard Shewchuk. Lecture Notes on Delaunay Mesh Generation. February 2012.

Logiciels

[4] CloudCompare. https://www.danielgm.net/cc/. « 3D point cloud and mesh processing software, Open Source Project ».

[5] Eigen. http://eigen.tuxfamily.org/index.php. « C++ template library for linear algebra : matrices, vectors, numerical solvers, and related algorithms ».

[6] Git. https://git-scm.com/. « Free and open source distributed version control system ».

[7] MeshLab. https://www.meshlab.net/.

[8] Paraview. https://www.paraview.org/.

[9] QtCreator. https://doc.qt.io/qtcreator/index.html. Environnement de développement intégré multi-plateforme.

Seconde bibliographie

[10] Guillaume Caumon, Bruno Lévy, Laurent Castanié, and Jean-Claude Paul. Visualization of grids conforming to geological structures : a topological approach. *Computers & Geosciences*, 31(6) :671 – 680, 2005.

[11] Xin Feng, Yuanzhen Wang, Yanlin Weng, and Yiying Tong. Compact combinatorial maps : A volume mesh data structure. *Graphical Models*, 75(3) :149 – 156, 2013. Computational Visual Media Conference 2012.

[12] Michael Kremer, David Bommes, and Leif Kobbelt. OpenVolumeMesh - A Versatile Index-Based Data Structure for 3D Polytopal Complexes. In *21st International Meshing Roundtable*, Proceedings of the 21st International Meshing Roundtable, pages 531–548, San Jose, United States, October 2012. Springer.

[13] François Lepage. *Génération de maillages tridimensionnels pour la simulation des phénomènes physiques en géosciences.* PhD thesis, INPL, Nancy, France, 2003.

[14] Romain Merland. *Génération de grilles de type volumes finis : adaptation à un modèle structural, pétrophysique et dynamique.* Theses, Université de Lorraine, April 2013.

[15] Thomas Viard, Claude Cavelius, Bradley Mallison, and Charles H. Sword. Data structure improvements for 3D polyhedral grids with application to unstructured discrete fracture models. . In *Proc. 32nd Gocad Meeting, Nancy* , 2012.