République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab Blida

N° D'ordre :.....

Faculté des sciences

Département d'informatique

Mémoire Présenté par :

Sebti Khaled Moulay Abdelhamid

En vue d'obtenir le diplôme de master

Domaine : Mathématique et informatique

Filière : Informatique
Spécialité : Informatique
Option : Ingénierie de logiciel

Sujet :

Modélisation Volumique des Pièces de Formes Complexes par Triple Dexels à Partir de Leurs Modèles STL

Soutenu le :

Mr. Ferfera	Président
Mr. Sidammou	Examinateur
Mme. Mnacer	Examinateur
Mme Abed.H.	Promotrice.
Mme Bouhadja , Mr Bey.M,	Encadreur.

Promotion
2014 / 2015

Remerciements

Nous tenons à remercier très vivement notre Promotrice de l'Université de Saad Dahleb de Blida, Madame Abed, qui a accepté d'encadrer notre travail durant ces mois de stage, s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de ce projet, et sans qui ce mémoire n'aurait jamais vu le jour.

Nous tenons à remercier très vivement nos encadreurs Mr El Bey.M et Mme K. Bouhadja pour ses entière disponibilités, ses amabilités, simplicités et ses conseils. D'avoir tout mis en œuvre pour que nous puissions donner le meilleur de nous même et pour tous les moyens qu'il a mis à notre disposition durant notre période de stage, nous le remercions sincèrement. Qu'il trouve ici l'expression de notre profonde gratitude d'avoir accepté de diriger et de corriger ce modeste travail.

Que Messieurs les membres du jury trouvent ici le témoignage de notre reconnaissance pour avoir bien voulu juger notre travail.

Qu'il nous soit permis de remercier toute personne ayant contribué à la réussite de ce travail.

Merci à tous et à toutes.

Dédicace

Par la grâce de Dieu je viens de terminer mon projet de fin d'étude.

Que je dédie :

A mes parents, qui grâce à leurs encouragements et leurs grands sacrifices, ils ont pu créer le climat affectueux et propice à la poursuite de mes études.

Aucune dédicace ne pourrait exprimer mon respect, ma considération et mes profonds sentiments envers eux.

A mes frères Farouk et sa femme, Khaled, Mohammed

A mes sœurs Fatima et son mari, et Meriem.

A tous mes neveux qui sont la joie de notre petite famille.

A mes amis Abdelhak Djamel, Yacine, Chouaib, Paco, Abdelhamid, les deux Youcef, Lila, Rahma, et qui m'ont aidé et encouragé.

A mon binôme Sebti khaled qui m'a encouragé et aidé pour faire ce travail.

Abdelhamid

Dédicaces

Je dédie ce modeste travail ;

A la mémoire à ma grand-Père ;

A vous très cher père et très chère mère, que dieu les protège;

A mes frères et A mes très chères sœurs ;

A mes nièces : Lina , ritedj et Lymare.

A mes neveux : abd errahim.

A tous mes amis.

A tous ceux qui mont soutenu, qu'ils trouvent ici l'expression de mon amour et ma profonde gratitude ;

Khaled

ملخـــص:

يتركز عملنا في مجال المحاكاة الهندسية على نطاق صغير باستعمال النمذجة الحجمية (الدكسل الثلاثي) لقطع ذات شكل معقد، وذلك لمحاكاة عملية نزع المادة عن طريق الحفر ذات الخمس محاور. ابتداء من نموذجه الأصلي (STL).

يتمحور هذا العمل على اقتراح وتطبيق طريقة تقسيم حجم ما بإدخال معنى كمية المادة، وحساب التقاطع مع نموذج الأصلي لتمثيل القطع بأحسن تمثيل ممكن مع تدقيق جيد مع الأخذ بعين الاعتبار كمية الذاكرة المستعملة بالإضافة إلى سرعة الحساب.

RESUME :

Notre travail se pointe sur la simulation géométrique à l'échelle macroscopique en utilisant la modélisation volumique des pièces de formes complexes Triple Dexel à partir de leurs modèles STL pour la simulation d'enlèvement de matière en usinage 05 axes.

Le travail consiste à proposer et à implémenter une méthodologie de discrétisation d'un volume en introduisant la quantité de matière. Et de calculer les intersections avec le modèle STL permettant de représenter ces pièces le plus fidèlement possibles avec une bonne précision afin de comptabiliser les défauts d'usinage sur l'état de surface, en répondant au compromis entre précision et temps de calcul.

ABSTACT

Our work points to the geometric simulation on a macroscopic scale using solid modeling of parts with complex shapes Triple Dexel from their STL models for machining material removal simulation 05 axes. The job is to propose and implement a methodology discretization of a volume by introducing the amount of material. And calculate the intersections with the STL model to represent these as closely as possible with good precision parts to account for the machining defects on the surface condition, responding to compromise between accuracy and computation time.

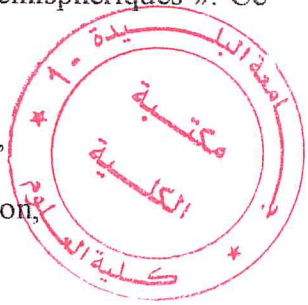
Table des matières

Table des figures

INTRODUCTION GENERALE

Présentation du sujet

Le présent travail s'inscrit dans le cadre du programme de recherche de l'équipe CFAO 2014- 2016 «Production des Surfaces de Formes Libres sur des Fraiseuses Numériques à 05 axes » au niveau du CDTA (Centre de développement des technologies avancés). Il est proposé afin de répondre à la tâche « Proposition et implémentation de méthodes de simulation de l'opération de finition en 05 axes en utilisant des outils hémisphériques ». Ce projet est une continuité de plusieurs travaux :



- Modélisation et conception des courbes et des surfaces gauches,
- Finition des surfaces gauches avec différentes stratégies de finition,
- Simulation virtuelle de l'usinage en ébauche et en finition,
- Finition des surfaces gauches à partir d'un nuage de points quelconque.

Le développement des techniques de simulation de l'usinage sur les machines multiaxes (03 axes et 05 axes) est en fait la clé de l'évolution de la productivité et de la qualité de fabrication des pièces mécaniques de formes complexes (moules, formes aérodynamiques, formes de style, …etc.). Ces pièces doivent répondre aux exigences fonctionnelles et / ou de style, qui nécessitent une attention particulière dans leur phase de production. Dans le cadre de notre travail, il nous a été demandé de focaliser sur la représentation volumique de la pièce à usinée afin de simuler l'état de surface finie.

1. Problématique

L'usinage des pièces mécanique passe par 2 étapes importantes qui sont : la conception et la fabrication. Dans la deuxième étape, La forme finale de la pièce est obtenue en trois opérations: ébauche, demi-finition et finition. Ces opérations exigent la sélection de plusieurs paramètres (stratégie d'usinage, outil, vitesse d'avance, …etc.). Avant de passer à l'usinage réel sur machine,il est indispensable de passer à la simulation virtuelle de l'usinage afin de vérifier le trajet d'outil, de détecter les collisions, de prédire les efforts de coupe et la rugosité de la surface finie, …etc. Cependant, cette technique est encore fortement pénalisée par les lacunes dans la connaissance de la coupe. Ce domaine est très vaste et touche différents niveaux

Dans la littérature, les différentes méthodes de simulation sont identifiées et présentées d'une manière permettant l'exploration dans ce domaine, en répondant aux questions clés: pourquoi (objectifs: simulation géométrique ou physique), à quelle échelle (humaine, macroscopique, microscopique), et comment (modèles géométrique, modèles dynamiques). Trois modèles de représentation géométrique sont à considérer à savoir: modèles de la pièce, modèle de l'enveloppe de l'outil et le modèle du volume balayé par l'outil. Les modèles géométriques utilisés dans ce domaine peuvent aller du plus simple, une série de points, au plus complexe, description facettisée de la surface ou volumique, représentation à base espace image de type Z-buffer, Dexel, Voxel et plus avancé le modèle Triple Dexel. Notre travail est focaliser sur la méthode Triple Dexel qui est une nouvelle méthode qu'on va utiliser dans ce projet.

2. Objectif

Notre travail se pointe sur la simulation géométrique à l'échelle macroscopique en utilisant la modélisation volumique des pièces de formes complexes Triple Dexel à partir de leurs modèles STL pour la simulation d'enlèvement de matière en usinage 05 axes. Le travail consiste à proposer et à implémenter une méthodologie de discrétisation d'un volume en introduisant la quantité de matière. Et de calculer les intersections avec le modèle STL permettant de représenter ces pièces le plus fidèlement possibles avec une bonne précision afin de comptabiliser les défauts d'usinage sur l'état de surface, en répondant au compromis entre précision et temps de calcul.

Le but de ce travail est le développement d'un module logiciel graphique et interactif sous Windows permettant la représentation volumique des pièces quelque soit la complexité des formes géométriques d'une manière précise et rapide.

3. Structuration du mémoire

Le présent mémoire est composé des chapitres suivants :

- Le premier chapitre est consacré à la présentation du processus de CFAO et le format d'échange de données « STL ». Ainsi qu'à la présentation des différentes méthodes de simulation.
- L'étude conceptuelle de notre application logicielle est menée dans le deuxième chapitre.

- L'application développée, les fonctions et les algorithmes utilisés pendant le développement informatiques sont présentés dans le troisième chapitre.
- Le dernier chapitre présente les tests et la validation des résultats.

Chapitre 1

Problématique

Et état de l'art

1. Introduction :

Dans ce chapitre nous introduisons le problème abordé dans ce mémoire en le divisons en deux partie. Dans la première, nous présentons d'une manière générale le système de Conception et de Fabrication Assisté par Ordinateur (CFAO), sa structure son processus qui est composé de deux modules principaux : le premier est dédié à la conception assistée par ordinateur (CAO), et le second est dédié à la fabrication assistée par ordinateur (FAO). Le passage d'un module à l'autre se fait systématiquement dans le cas où la CAO est intégré à la FAO, et dans le cas découplé ou interfacé le passage se fait par l'utilisation de formats d'échanges, dans notre travail nous considérons le format STL.

Dans la deuxième partie, nous recensons les principales méthodes pour solutionner les problèmes posés par la simulation d'usinage tout en mettant l'accent sur les techniques de représentation volumique : Voxel, Dexel et Triple Dexel.

2. Généralités sur la CFAO :

La CFAO est la combinaison de la CAO avec la FAO. Pour construire une forme, l'ingénieur possède des informations géométriques (points, courbes et surfaces) issues de calculs ou de mesures. Ces informations sont appelées données géométriques fonctionnelles. Elles servent de canevas pour la reconstruction de la surface dans l'environnement de CAO. En plus de ces données, l'ingénieur doit prendre en compte des contraintes fonctionnelles de construction (masse, volume, aires, …etc.) qui ne peuvent pas être exprimées sous formes d'éléments géométriques (points, droites, ...etc.) d'une base de données CAO. A partir de cette représentation numérique de la pièce, les logiciels de FAO génèrent les programmes d'usinage contenant les parcours d'outils et les fonctions annexes (mise en route de la broche, changement d'outil, ...etc.). Ces logiciels ne sont pas complètement automatiques, le choix du type de trajectoire à utiliser en fonction de l'entité choisie (profil, poche, trou, ...etc.) doit être fait par l'utilisateur.

Les systèmes de CFAO évoluent pour répondre aux points suivants :

- Rentabilité maximale ;
- Automatisation totale de la génération des trajectoires ;
- Communication et échange de données ;

- Qualité de finition des surfaces ;
- Contrôle de collisions.

Ces systèmes doivent vérifier les critères de performances suivantes :

- Productivité globale ;
- Applicables à des formes diverses et complexes;
- Applicables à divers modèles de surfaces : traitement des modèles de pièces réalisés avec différents modèles mathématiques ;
- Respect d'une qualité imposée et la validité de la pièce ;
- Contrôle de l'efficacité de l'usinage.

2.1 Processus de CFAO :

Le processus de CFAO est formé d'une suite d'activités importantes, de la conception à la production effective (Figure1). La réalisation d'une forme gauche passe par deux étapes :

- Construction du modèle géométrique,
- Génération de trajectoires d'outils.

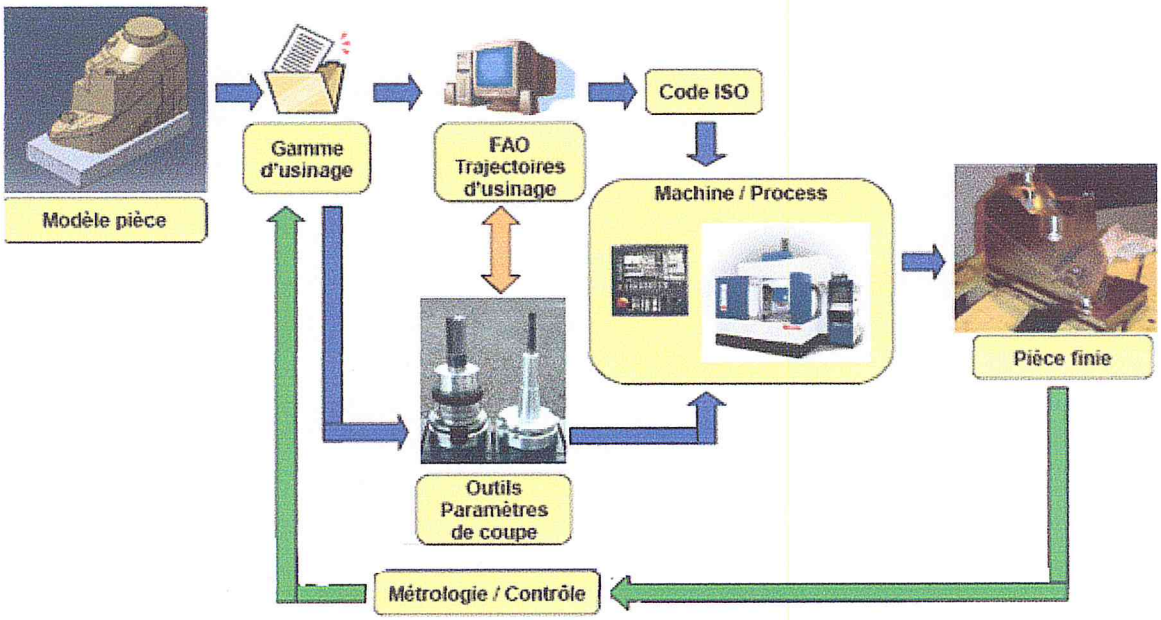


Figure 1: *Processus de fabrication d'une pièce mécanique. [1]*

- **Construction du modèle géométrique :** cette étape a pour but d'exprimer l'ensemble des contraintes fonctionnelles écrites dans le cahier des charges sous forme de surfaces ou d'éléments géométriques décrits à l'aide d'un système de CAO.

- **Génération de trajectoires d'outils :** cette étape est dite de fabrication et a pour but de transformer ces éléments en une forme réelle qui permet à l'objet de répondre à sa fonction.

- **Trajectoires en format neutre :** sont des trajectoires d'outils exprimées dans un repère lié à la pièce à usiner. Ces trajectoires ne tiennent compte ni de la cinématique de la machine ni de la position de la pièce sur le plateau de la machine. Elles ne sont pas modifiables dans la plupart des cas et seul un programmeur expérimenté peut modifier à ce niveau de la chaîne. Par ailleurs, il est à noter que chaque ligne du programme comporte des informations sur les trajectoires et sur l'attitude de la machine en tout point de la trajectoire. Pour exprimer ces trajectoires dans un repère lié à la machine et dans un langage compréhensible par la commande numérique, un post-processeur doit être utilisé. Le format neutre le plus courant est le CL-FILE *(Cutter Location -File)*.

- **Trajectoires en format "machine" :** sont les trajectoires exprimées dans un langage compréhensible par la commande numérique. Le langage de programmation ISO (souvent nommé G-Code ou M-Code) est le plus couramment utilisé. Il n'est malheureusement pas neutre et dépend fortement de la machine sur laquelle est réalisé l'usinage. Seules les fonctions de base du langage ISO sont communes à toutes les machines l'utilisant. Ces fonctions sont les lignes droites, les arcs de cercles et quelques fonctions annexes. Ces trajectoires sont modifiables dans la plupart des cas car le langage de programmation est généralement simple et compact. Le langage ISO comporte deux types d'ordre : les ordres "G" et les ordres "M". Les ordres "G" (comme GO) ont un rapport avec la trajectoire à suivre (avancer, tourner, percer de telle ou telle manière, décaler la trajectoire à droite ou à gauche en fonction du diamètre de la fraise, …etc.).Les ordres "M" s'occupent de tout ce qui n'a pas de rapport direct avec les mouvements de la machine (état de la broche, changement d'outils, fonctions pré-câblées, arrêt du programme, ...etc.).

- **Validation et simulation :** c'est l'étape où est validé le choix de la méthode de calcul des trajectoires d'outils avec la stratégie d'usinage optimale suivi d'une simulation d'usinage afin de détecter des anomalies (problèmes de collisions et d'interférences) et d'autres problèmes concernant le choix des paramètres d'usinage et des conditions opératoires.

- **Post-Processeur :** c'est un programme qui transforme les trajectoires en format neutre (CL-FILE) en des trajectoires comprises par la machine en prenant en compte la cinématique de la machine, ses courses et ses capacités. Il signale les erreurs et exprime les trajectoires dans le langage spécifique à la machine.

2.2 Structure générale d'un système de CFAO :

En général, il y a deux types de systèmes de CFAO :

- Le système de CFAO découplé ou interfacé voir (Figure2.a)
- le système de CFAO Intégré voir (Figure2.b)

Le premier système montre une organisation linéaire des différentes étapes à franchir depuis la phase conceptuelle jusqu'à la mise en place de l'usinage. Le passage d'une étape à l'autre s'effectue à travers une ou plusieurs bases de données. Les fichiers de passage représentent les interfaces entre les différentes étapes. Chaque fichier contient les résultats de l'activité précédente avec un enrichissement de la base de données (géométrique pour la CAO, gamme, parcours d'outils pour la FAO ...).

Le deuxième montre une organisation intégrée des modules CAO - FAO dans une même plateforme.

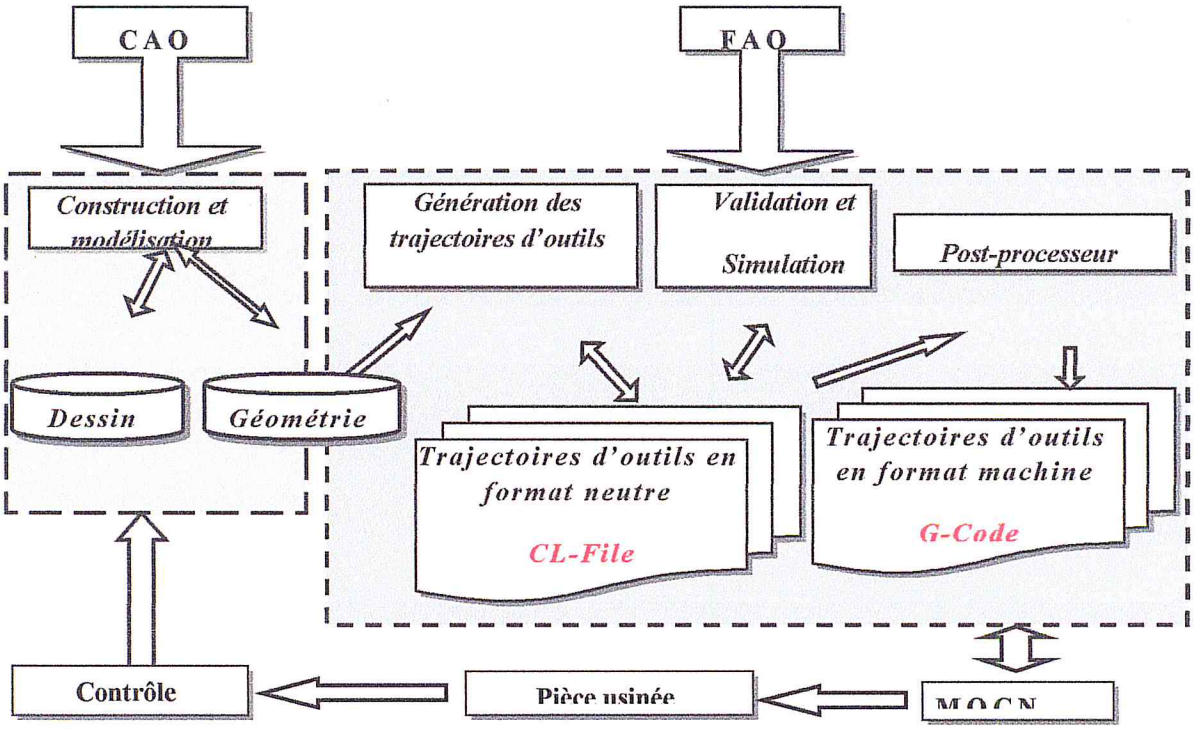


Figure 2a : Flux d'information sur le système de CAO-FAO découplé ou Interfacé

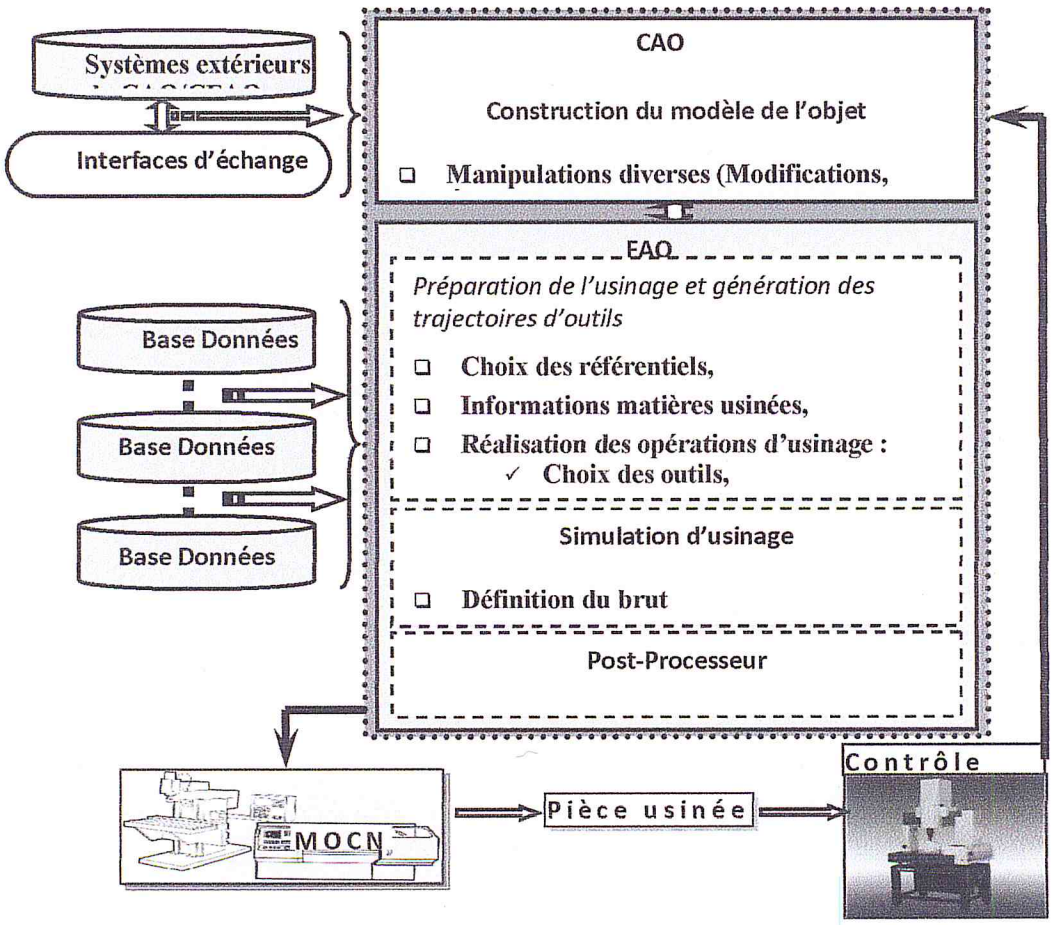


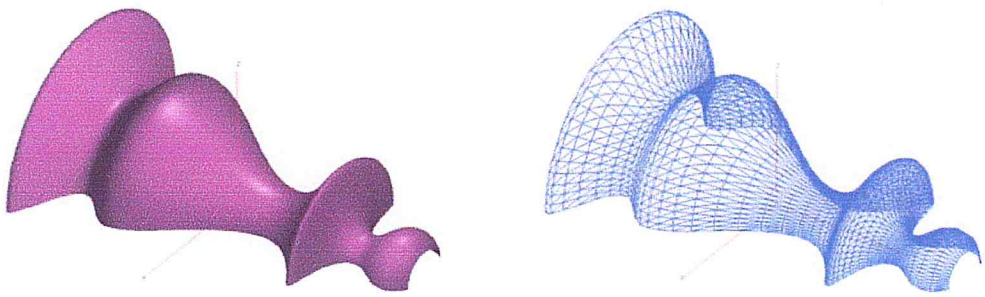
Figure 2b : Flux d'information sur le système de CFAO

3. Format d'échange STL :

Il existe deux types de système CFAO, le premier est découplé ou interfacé et le deuxième est Intégré. Pour le premier système, le problème qui se pose est l'échange des informations de conception entre les logiciels de CAO et de FAO. D'où la nécessité de développer des outils permettant de réaliser ces échanges dans un format neutre indépendant des logiciels. Pour cela, plusieurs formats d'échange de données sont créés tels que IGES, STL, VRML, DWG, DXF, STEP, ...etc. dans notre travail, nous avons choisi le format STL pour la représentation de pièces de forme complexe.

3.1 Description des surfaces facettisée :

Un fichier STL est une représentation triangulaire des domaines volumiques par leur frontière (B-Rep), où les surfaces qui représentent la peau du volume sont divisées en une série de triangles dont la taille dépend de la précision demandée (figure 3).



a. Surface de base. **b. Modèle STL.**

Figure 3 : Format STL d'une surface théorique.

La description des surfaces est basée sur une représentation topologique. Les entités élémentaires, dans cette description sont les sommets, les arêtes et les facettes (figure 4).

- les sommets sont des points,
- les arêtes sont des segments droits,
- les facettes sont des triangles plans.

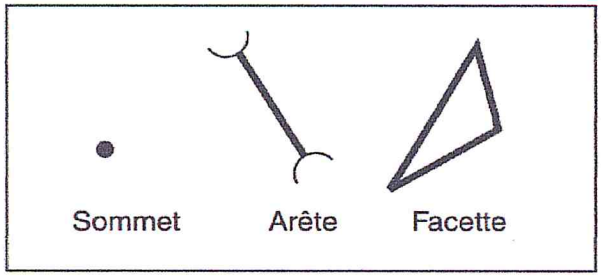
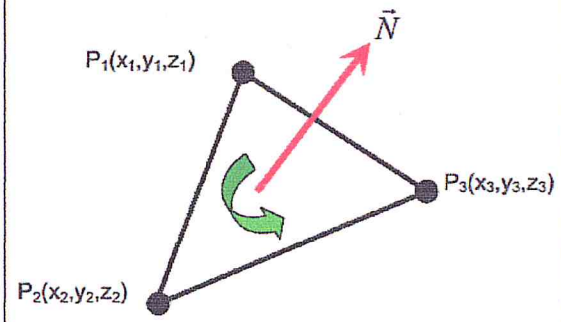


Figure 4 : les différentes entités [2]

Chaque triangle est défini de manière unique par sa normale unitaire et les coordonnées X, Y et Z de ses trois sommets orientés comme est indiqué sur la Figure5.

```
facet normal 0 -1 0
  outer loop
    vertex 41.437 0 60
    vertex 153.46 0. 0
    vertex 153.46 0. 60
  endloop
endfacet
facet normal 1 0 0
  outer loop
    vertex 153.464 150 60
    vertex 153.464 0 60
    vertex 153.464 150 0
  endloop
endfacet
```



a. Structure du fichier « STL ». *b. Paramètres d'un triangle.*

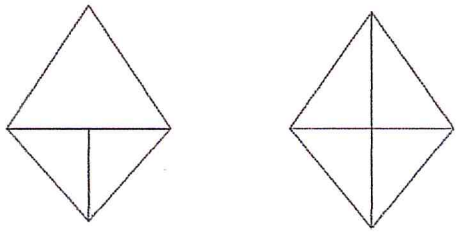
Figure 5 : Fichier STL et paramètres d'un triangle [3].

3.2 Caractéristiques du format STL :

Les principales caractéristiques du fichier STL sont les suivantes :

- Le format le plus simple qui existe pour décrire une géométrie 3D.
- Format de très faible niveau, il ne contient aucune notion d'attribut que l'on peut trouver dans les autres formats CAO (couleur, texture...etc.).
- Approximation de la frontière par des facettes triangulaires.
- Le fichier « STL » est volumineux. Le modèle CAO est remplacé par un nombre important de triangles dont le nombre dépend de la précisons désirée, de la forme de l'objet et de ses dimensions.

- Le fichier « STL » ne contient pas d'informations d'échelle, les coordonnées sont dans des unités arbitraires.
- Règle d'autorité, chaque triangle doit partager deux sommets avec chacun de ses triangles adjacents. Donc, un sommet d'un triangle ne peut pas être localisé sur le côté d'un autre triangle (Figure 6).



a. Non vérifiée. *b. Vérifiée.*

Figure 6 : Règle d'autorité [3].

3.3 Avantages et inconvénients du modèle STL :

Le format STL est simple et très répondu dans l'industrie, il est mathématiquement facile à trancher et il garantit la confidentialité par ce qu'il est difficilement réutilisable en CAO. Il garantit également la compatibilité et le respect des données [3]. Le format STL a beaucoup d'inconvénients comme la perte de précision et d'information et la redondance de cette dernière, mais il reste comme même intéressant dans notre cas l'application à la simulation d'usinage, d'où nous avons choisi l'utilisation du format STL dans notre étude.

4. Les méthodes de simulation d'usinage des formes gauches sur les machines multiaxes (03 axes et 05 axes) :

Comme nous avons vu dans la section 2.1, la simulation d'usinage dont le principe de base est celui de la représentation du phénomène de coupe le plus réaliste possible est une étape indispensable dans le processus de CFAO. Cependant, cette technologie est encore fortement pénalisée par les lacunes dans la connaissance de la coupe. Ce domaine est très vaste et touche différents niveaux. Nous allons présenter un travail de synthèse élaboré dans [4] afin de pouvoir sélectionner la technique qui correspond à notre problème. Ce travail identifie et présente les différentes méthodes de simulation dans une structure simplifiée (figure 7) permettant l'exploration rapide dans ce domaine, en répondant aux questions clés: pourquoi

(objectifs: simulation géométrique ou physique), à quelle échelle (humaine, macroscopique, microscopique), et comment (modèles géométrique, modèles dynamiques).

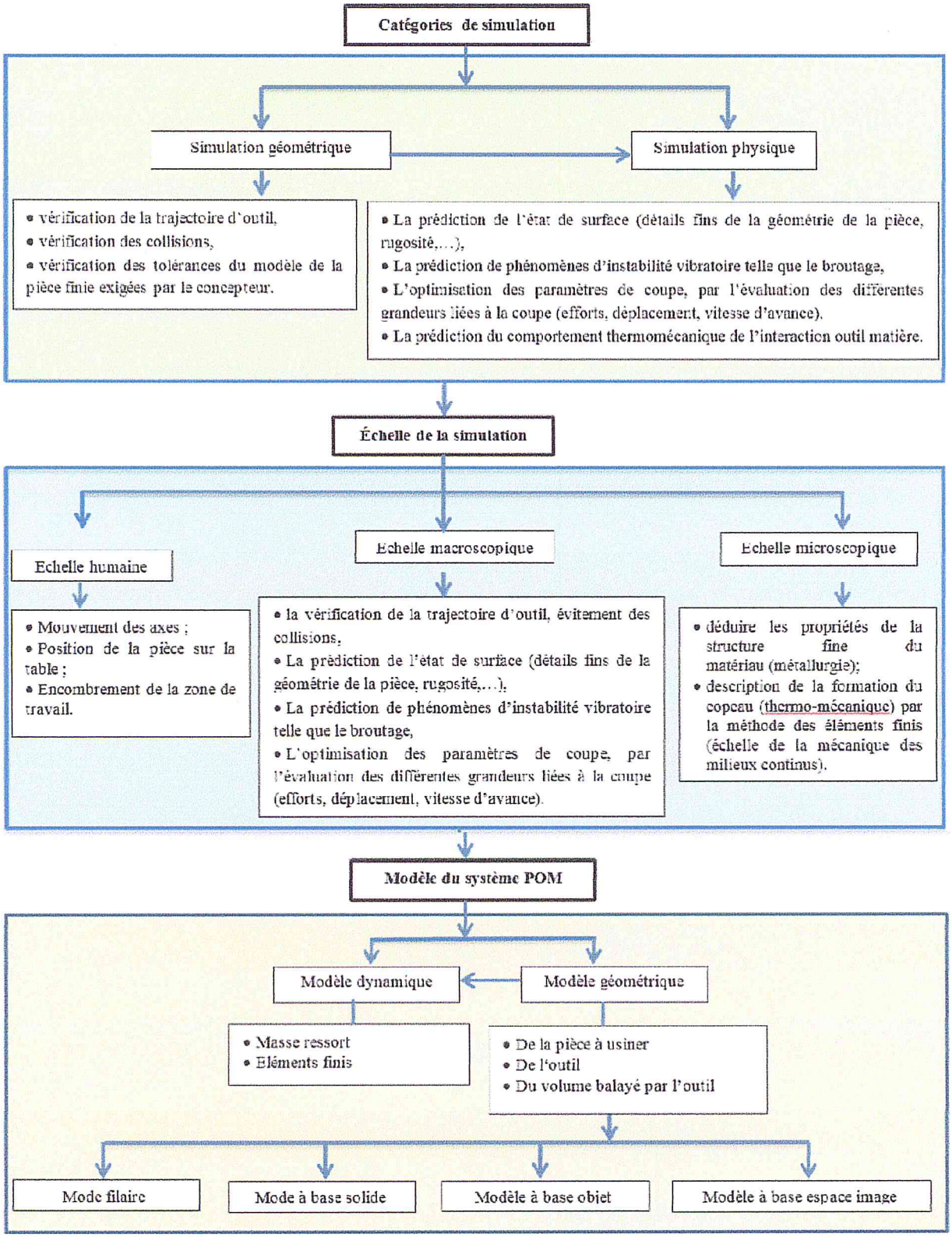


Figure 7:Classification des méthodes de simulation [4]

4.1 Catégories de simulation :

La simulation de l'usinage est distinguée en deux catégories: simulation géométrique et simulation physique (figure 8).

4.1.1 Simulation géométrique :

La simulation géométrique est utilisée pour vérifier graphiquement les interférences et les collisions, le respect des tolérances exigées dans le cahier des charges par le constructeur. En outre, elle peut fournir des informations géométriques telles que l'angle d'attaque de l'outil à la simulation physique.

4.1.2 Simulation physique :

Comme son nom l'indique, la simulation physique d'un processus d'usinage à commande numérique vise à révéler les aspects physiques d'un processus d'usinage, comme l'effort de coupe, les vibrations, rugosité de surface, température d'usinage et de l'usure de l'outil. Elle est basée sur la simulation géométrique et le choix du matériau de l'outil de coupe.

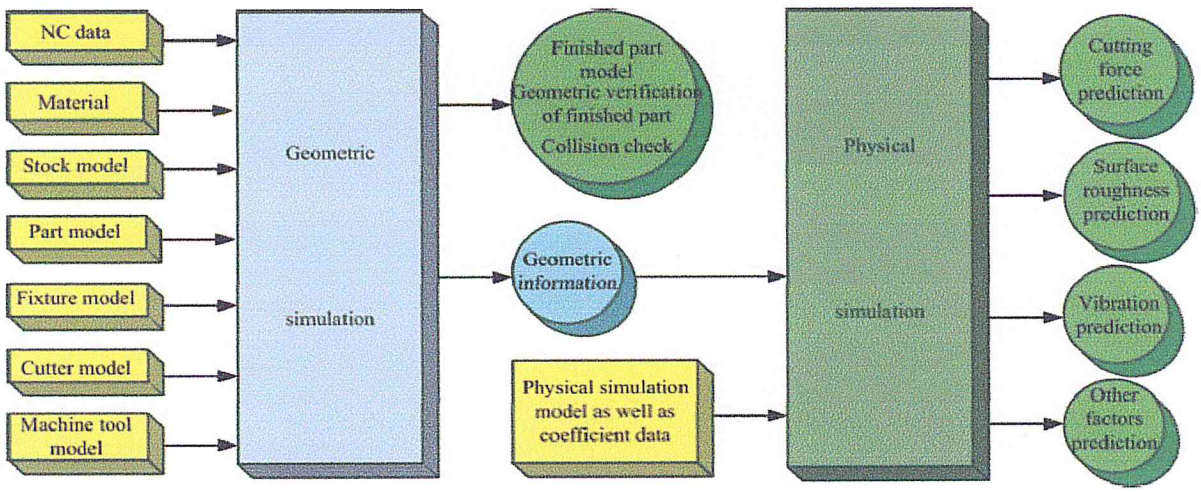


Figure 8 : Architecture générale de la simulation de l'usinage [5].

4.2 Echelles de simulation :

L'étude de l'usinage est souvent abordée à l'aide d'une approche multi-échelles, ceci permet de séparer les difficultés en limitant le nombre de phénomènes à prendre en compte et

la taille du modèle à une échelle donnée. Trois échelles d'analyse peuvent être distinguées : échelle humaine, échelle macroscopique et l'échelle microscopique.

4.2.1 Echelle humaine :

Est une simulation globale de l'environnement de l'usinage (figure2 page 9). L'objectif est de s'imprégner du comportement du moyen de production, afin de préparer l'usinage :

- Mouvement des axes ;
- Position de la pièce sur la table ;
- Encombrement de la zone de travail.

Cette étape est indispensable dans le cas où le moyen de production est très complexe et induit des mouvements relatifs de la pièce par rapport à l'outil difficiles à anticiper (machine multiaxes, robot d'usinage...). Elle permet effectivement de détecter d'éventuelles collisions pendant le processus. La plus part du temps, ce type de simulation est intégré dans les logiciels de FAO. Il existe cependant des logiciels de simulation indépendants spécialisés dans l'interprétation des codes ISO issus des logiciels de FAO.

4.2.2 Echelle macroscopique :

Toujours dans une approche industrielle, il est intéressant d'observer la pièce de plus près afin de visualiser l'enlèvement de matière proprement dit. Le but de cette simulation est de déterminer globalement le volume de matière enlevé pour chacune des phases d'usinage de la pièce.

A Cette échelle les techniques de simulation permet, entre autre, de visualiser et d'anticiper les défauts de surface liés purement à la stratégie programmée ou à la cinématique de la machine. Dans la littérature on distingue différents sortes de travaux, ceux qui s'appuient sur la représentation de la pièce à usiné. Et d'autres travaux se focalisent sur la génération du volume de matière balayée par l'outil de coupe. La difficulté à ce niveau-là est prononcée par la cinématique de la machine 05 axes, où l'outil subit un mouvement de rotation et de translation simultané. Et pour une haute précision d'autres travaux proposent la théorie de la cinématique des systèmes multi-corps.

4.2.3 Echelle microscopique :

La simulation à cette échelle est celle de l'étude des matériaux. Il s'agit de déduire certaines propriétés à partir de la structure fine du matériau. Ces propriétés sont, par exemple, les lois de comportement du matériau utilisées et à une échelle un plus supérieure. On trouve l'échelle Microscopique, à cette échelle en étudiant la description de la formation du copeau. En se basant sur une description thermomécanique faisant intervenir les différents phénomènes physiques et métallurgiques, mais à une échelle qui est celle de la mécanique des milieux continus, à cette échelle la simulation est souvent abordée par la méthode des éléments finis.

4.3 Modèles du Système Pièce Outil Machine (POM) :

Afin de simuler les phénomènes présents dans la dynamique du système POM à l'échelle macroscopique. Nous introduisons principalement, deux modèles à savoir : modèle dynamique et modèle géométrique.

4.3.1 Modèle dynamique :

Le modèle dynamique du système POM utilisé peut être un modèle simple masse ressort ou un modèle complexe élément finies.

La modélisation par éléments finies permet une discrétisation spatiale beaucoup plus fine et plus souple, elle permet aussi d'obtenir des modes de vibration beaucoup plus réaliste et d'aborder le cas où la pièce et/ou l'outil sont déformable dans la zone de travail.

4.3.2 Modèle géométrique :

Trois modèles de représentation géométrique sont à considérer à savoir: modèles de la pièce, modèle de l'enveloppe de l'outil et le modèle du volume balayé par l'outil. Les modèle géométrique utilisés dans ce domaine peuvent aller du plus simple, une série de points, au plus complexe, description facettisée ou volumique.

On distingue quatre grandes familles de modèles: filaire, à base solide, à base Objet, à base espace image. Dans ce dernier se retrouve les techniques de représentation volumique telle que le Dexel, le Voxel et le Triple Dexel.

Notre travail se pointe sur la simulation géométrique à l'échelle macroscopique en utilisant la modélisation volumique des pièces de formes complexes Triple Dexel à partir de leurs modèles STL pour la simulation d'enlèvement de matière en usinage 05 axes.

5. Techniques de représentation volumique :

5.1 Voxel :

Le Voxel (contraction de « volumetric pixel ») est un pixel en 3D (Figure9). Il consiste à stocker une information colorimétrique avec ses coordonnées spatiales, Ils s'inscrivent plus généralement dans des espaces matriciels bien que les espaces vectoriels leur soient favorables. Leur point faible est de nécessiter énormément de ressources, tant pour leur stockage que pour leur rendu. Aujourd'hui, la disponibilité de GPU a rendu possible d'obtenir un rendu en temps réel et l'interactivité avec les simulations voxels constitués de plusieurs milliards de voxels.

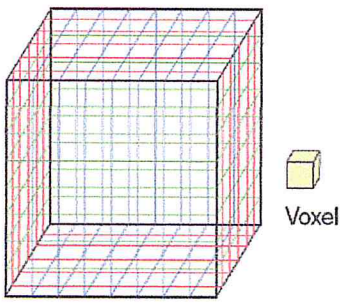
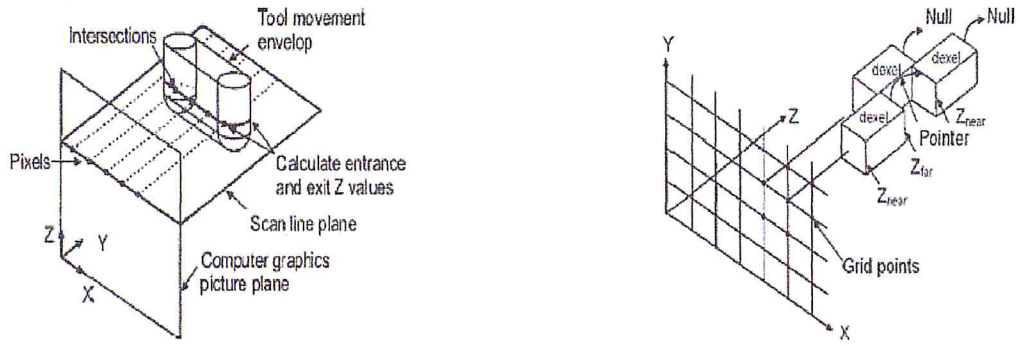


Figure 9 : Model Voxels [2]

5.2 Simple Dexel :

Le modèle dexel est l'extension du Z-buffer qui est utilisé pour l'élimination des surfaces cachées dans l'ordinateur, il découle des limites de Z-map, ce modèle est particulièrement performant, il est directement issue des méthodes de représentation de l'imagerie, et de rendu réaliste. La base de la méthode est de projeter une grille (soit un écran) selon une direction donnée sur la surface, suivant le point de vue choisi (figure10).



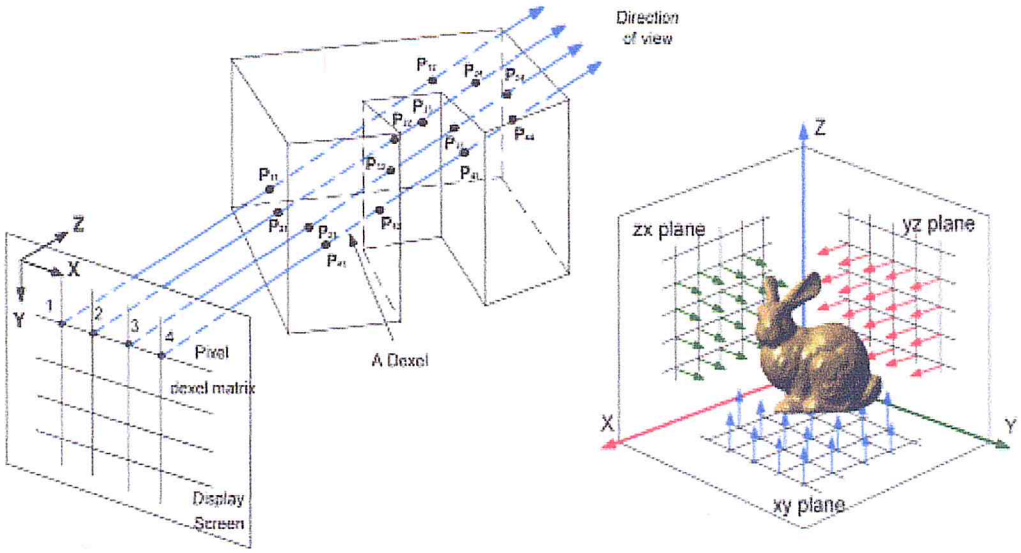
a. Intersection de Z-map avec le mouvement de l'outil *b. Extension de Z-map à des Dexel*

Figure 10 : Simulation à base espace image [4].

Elle s'adapte très bien à l'usinage à trois axes avec direction de projection, l'axe de l'outil, soit l'axe Z. Ainsi la construction de la surface est obtenue par intersection entre un ensemble de droites parallèles à Z et la surface. Les surfaces les plus adaptés ont une fonction qui réalise une bijection entre le plan de base XY et les points de la surface. Dans le cas d'un amas de surfaces, il n'est pas toujours possible d'usiner l'ensemble de l'amas. Pour chaque droite, on cherche toutes les intersections avec l'ensemble des surfaces, et on retient l'intersection la plus haute appartenant à la peau de la pièce, on n'usine ainsi que l'enveloppe externe de la pièce [5].

5.3 Triple Dexel

Le modèle simple Dexel est limité dans les régions où les surfaces normales sont presque perpendiculaires à la direction du rayon (Figure 11.a) [6]. Pour surmonter ce problème, une modélisation Triple- Dexel est présentée, qui est une extension de la modélisation simple Dexel, ce modèle est construit par les rayons orientés par les trois directions orthogonales (X, Y et Z) pour discrétiser le modèle (Figure 11.b). Les auteurs de [7] montrent clairement que la surface générée à partir des données Triple-Dexel (Figure 12.b et la Figure 12.d) est plus précise que la surface reconstruite à partir des données simples Dexel présentés dans la Figure 12.a et la Figure 12.c, en utilisant la même résolution.



a. Représentation Dexel *b. modèle Triple-Dexel.*

Figure 11 : Représentation en Dexel et en Triple-Dexel [6]

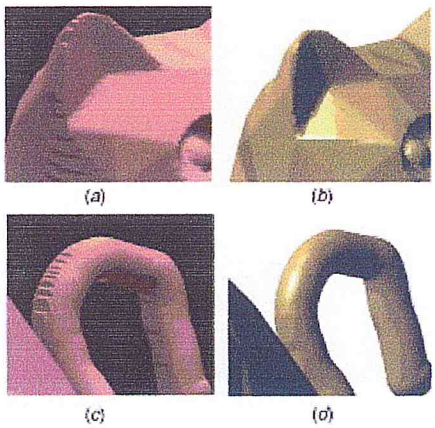


Figure 12 : Comparaison entre les données en simple-Dexel et en Triple-Dexel [7]

La modélisation Triple-Dexel est parfaitement adaptée pour les applications de simulation à base graphique en temps réel tels que la simulation d'usinage et la sculpture virtuelle. Dans [8], les auteurs présentent une méthode de reconstruction de surface à partir des données Triple-Dexel. La méthode développée est plus rapide est plus précise que la méthode de Voxel [8].

A partie de cette études restreinte sur les modèles de représentation 3D des pièces à usiné, nous avons opté pour le modèle Triple-Dexel qui représente une partie dans l'ensemble de traitement de la simulation d'usinage.

6. Conclusion

Nous avons considéré dans ce chapitre l'étude de l'état de l'art sur le processus de production des surfaces gauches en le divisant en deux parties. La première partie réservée aux généralités sur le système de CFAO ainsi que le format d'échange STL. La deuxième partie réservée à l'étude des différentes techniques de simulation de l'usinage des pièces complexes sur les fraiseuses 05 axes, Conclus par une comparaison restreinte sur les trois techniques de représentations volumiques à savoir Voxel, Dexel et Triple Dexel.

Dans ce qui suit. Notre travail se pointe sur la simulation géométrique à l'échelle macroscopique en utilisant la modélisation volumique des pièces de formes complexes Triple Dexel à partir de leurs modèles STL pour la simulation d'enlèvement de matière en usinage 05 axes.

Chapitre 2

Analyse et Conception

1. Introduction

Après une synthèse bibliographique dans le premier chapitre, nous allons nous intéresser dans ce chapitre à la conception de l'application en utilisant le modèle orienté objet et le langage de modélisation UML (*Unified Modeling Langage*). La conception a pour objectif de permettre de formaliser les étapes préliminaires du développement d'un système afin de le rendre plus fidèle aux besoins du client, mais avant nous allons analyser les différentes solutions existantes afin d'en tirer les avantages et d'éviter les problèmes déjà rencontrés par les autres développeurs on proposera notre conception en se basant sur les résultats de cette analyse.

2. Analyse :

Avant de commencer la partie conception de notre application, il faut d'abord faire une petite analyse pour justifier le but de développement de notre application, citer les insuffisances dans les applications précédentes et les nouveautés apportées dans notre application.

Dans le chapitre précédent nous avons vu les différentes techniques de représentation volumique comme le voxel, Dexel et le triple Dexel (qui est utilisé dans notre cas). Les travaux précédents concernant la simulation de l'usinage au niveau de CDTA, ils ont utilisé la technique de représentation Dexel, cette technique a des limites surtout là où il y'a des évidements et des courbures dans la pièce à usiner.

Le modèle de triple-dexel offre une meilleure résolution que le modèle unique dexel, en comparaison avec le modèle populaire de voxel, le modèle de triple-dexel nécessite moins de mémoire de l'ordinateur. Les données de triple-dexel ne consomment pas beaucoup de mémoire, et le travail se fait en un temps dans les normes, et le modèle de surface reconstruit est plus précis que la surface reconstruite à partir de la représentation de voxel ou de dexel. De plus il offre une bonne qualité de la vision de la surface au niveau de la simulation.

L'avantage le plus important du triple dexel par rapport au dexel est dans l'outil d'usinage, en dexel on utilise Fraiseuses à commande numérique 03 axes par contre en triple dexel on utilise Fraiseuses à commande numérique 05 axes. (La fraiseuse à commande numérique est

un ensemble comportant la machine-outil, le directeur de commande numérique, l'armoire électrique et le pupitre de commande).

En usinage à trois axes, pour une surface usinable, il existe une unique position de l'outil tangente à la surface en une position donnée. Par contre, en usinage à cinq axes, il existe une infinité de positions de l'outil tangente à la surface au point de contact, par conséquent le Tripe dexel c'est la solution qui répond au besoin d'usinage en cinq axes.

Dans la partie suivante nous allons commencer la conception de notre application.

3. Modélisation de l'application avec UML :

3.1 Méthode de modélisation :

Chaque application et chaque système avant d'être réalisé doit passé par une étape de conception avec une méthode ou bien langage de modélisation, cette étape permet de décrire les fonctionnalités du système, son comportement et tout le détail nécessaire pour la réalisation de ce système et son déroulement.

Les langages de modélisation orientés objet ont fait leur apparition entre le milieu des années soixante-dix et la fin des années quatre-vingt, quand les spécialistes méthode, confrontés à un nouveau genre de langages de programmation orientés objet et à des applications de plus en plus complexes.

UML (*Unified Modeling Language* - langage unifié pour la modélisation objet) est un moyen d'exprimer des modèles objet, c'est-à-dire que le modèle fourni par UML est valable pour n'importe quel langage de programmation. UML couvre toutes les phases du cycle de vie d'un logiciel, et à pour but de documenter les modèles objets.

UML est un langage standard conçu pour l'écriture de plans d'élaboration de logiciels, résultant de la fusion de 3 méthodes : OMT, OOSE et BOOCH qui possède les caractéristiques suivantes :

1. Est avant tout un support de communication performant, sa notation graphique permet d'exprimer visuellement une solution objet, ce qui facilite la comparaison et l'évaluation des solutions.

2. L'aspect formel de sa notation limite les ambiguïtés et les incompréhensions.

3. UML donc bien plus qu'un simple outil qui permet de dessiner des représentations mentales, il permet de parler un langage commun, normalisé mais accessible, car visuel.

Un modèle est une simplification de la réalité, donc la modélisation consiste à créer une représentation simplifiée d'un problème, elle comporte deux composants :

- **L'expression des besoins** : définir les besoins des Utilisateurs de système.
- **La conception** : la mise au point d'une solution.

UML fournit un moyen astucieux permettant de représenter diverses projections d'une même représentation grâce aux vues. Une vue est constituée d'un ou plusieurs diagrammes.

Dans notre projet, nous avons utilisé les diagrammes suivants :

- **L'expression des besoins :**

- Diagramme de cas d'utilisation.

- **Conception :**

- Diagramme de séquence,

- Diagramme de classes.

3.2 Diagramme de cas d'utilisation :

Les cas d'utilisation permettent de déterminer les besoins des utilisateurs et de capturer les exigences fonctionnelles d'un système. En d'autres termes, ils décrivent le comportement d'un système du point de vue de ses utilisateurs. Ils décrivent les interactions entre les utilisateurs du système et le système lui-même.

Un diagramme de cas d'utilisation permet de représenter graphiquement les cas d'utilisation.

Dans ce qui suit nous allons présenter les différentes fonctionnalités de notre système, Quels sont les offres qui proposent aux utilisateurs avec un diagramme de cas d'utilisation générale qui donne une vue générale à notre système.

1er niveau :

Notre système permet la simulation d'usinage des pièces mécaniques.

2eme niveau :

Ils existent deux grandes fonctionnalités qui sont :

- Acquisition du modèle de représentation d'une pièce.
- Remplissage du modèle.

3.2.1 Diagramme de cas d'utilisation générale :

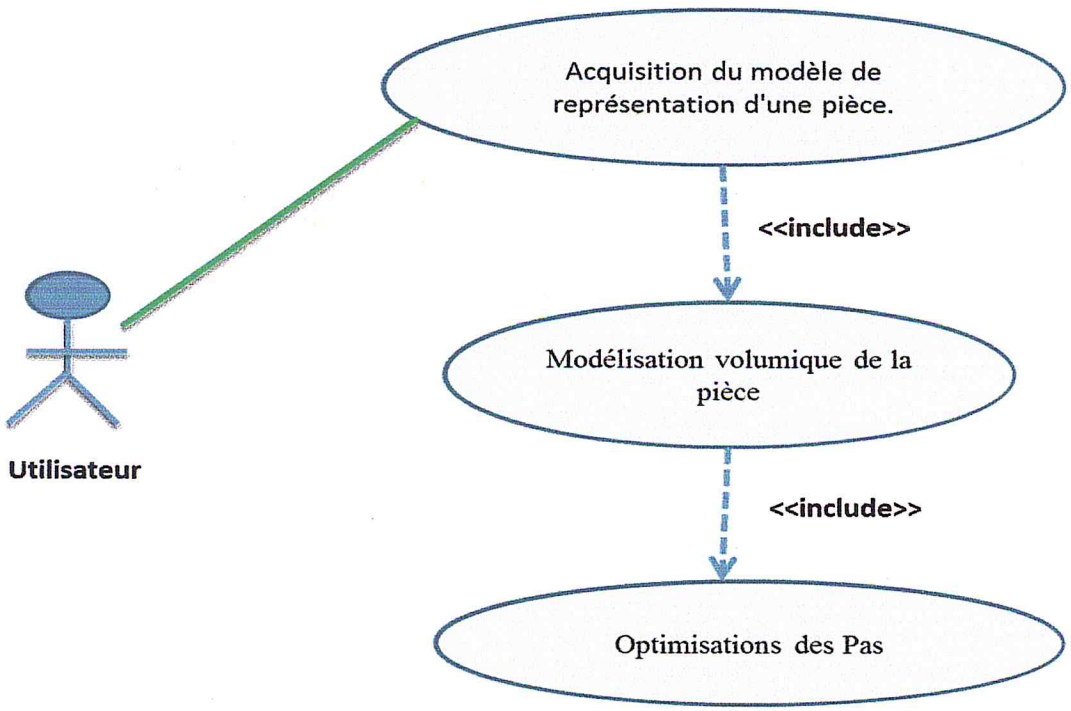


Figure 14 : ***Diagramme cas d'utilisation générale.***

Maintenant nous allons détailler ce diagramme général en détaillant les cas d'utilisations générales de notre système, commençant par le 1er cas d'utilisation qui est l'Acquisition du modèle de représentation d'une pièce:

3.2.1.1 Diagramme de cas d'utilisation "Acquisition du modèle de représentation d'une pièce":

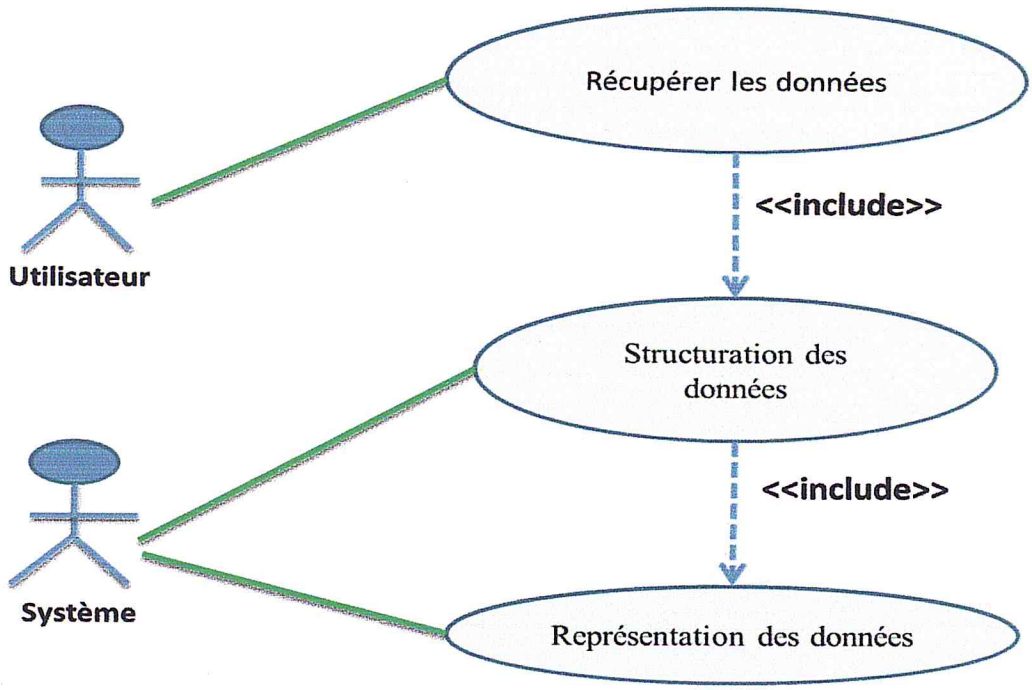


Diagramme Cas d'utilisation Acquisition du modèle

Figure 15: ***diagramme cas d'utilisation Acquisition des données.***

3.2.1.2 Le diagramme de cas d'utilisation suivant d'écrit le Remplissage du modèle :

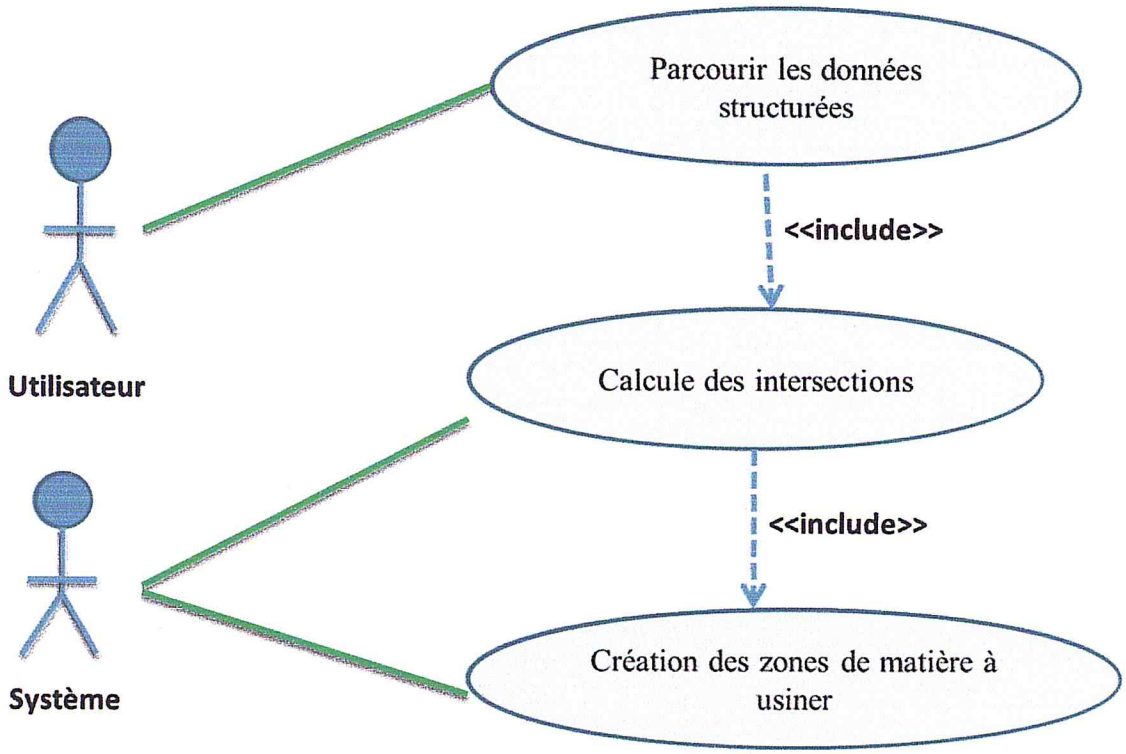


Diagramme Cas d'utilisation Modélisation volumique de la pièce

Figure 16 : ***Diagramme cas d'utilisation Remplissage de modèle.***

L'étude conceptuelle consiste à décrire le système futur. Ce nouveau système tiendra compte des nouveaux besoins des utilisateurs.

3.3 Diagrammes de séquence :

3.3.1 récupérer et structurer les points :

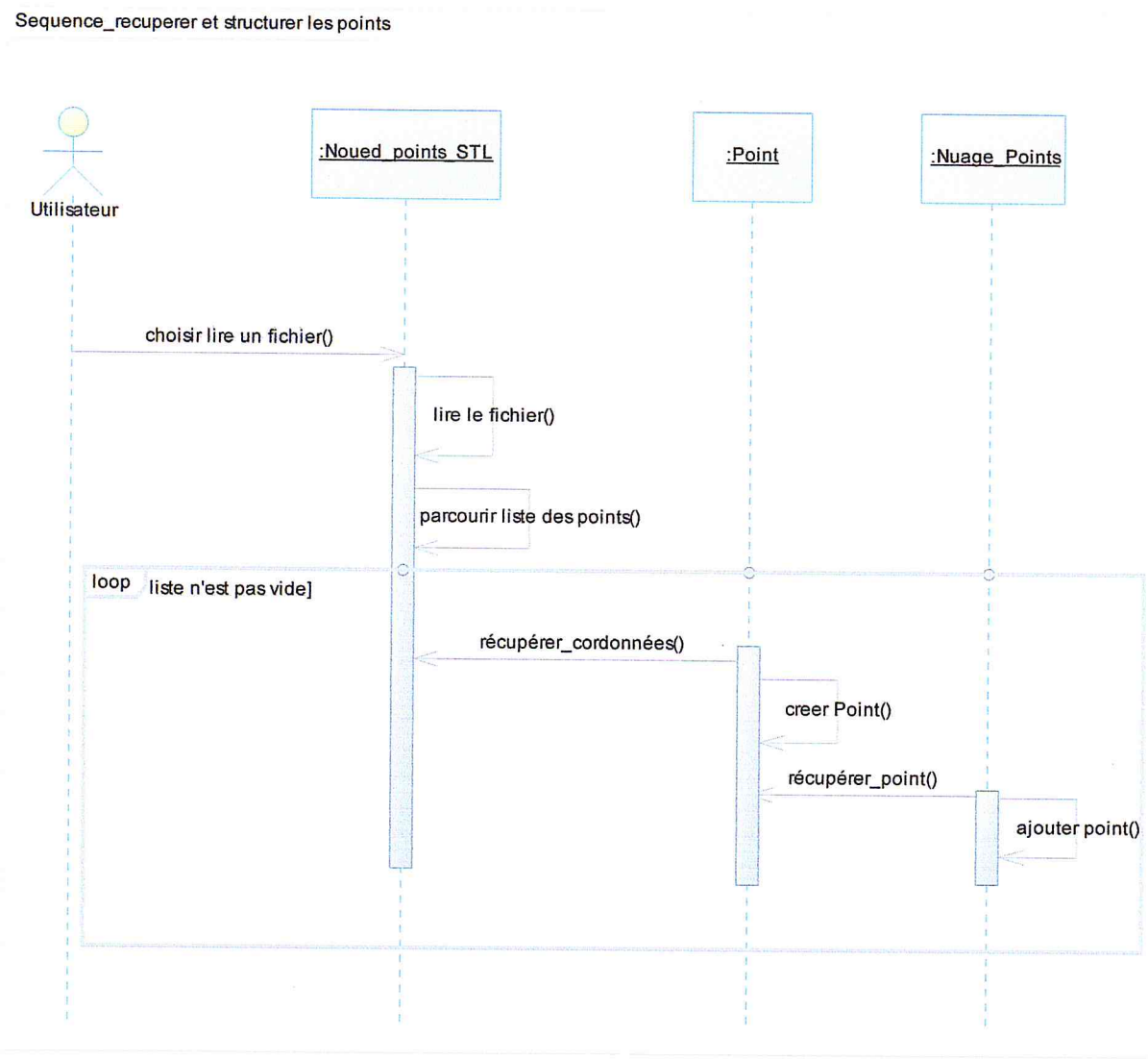


Figure 17 : ***diagramme de séquence récupérer et structurer point.***

3.3.2 récupérer et structurer les triangles :

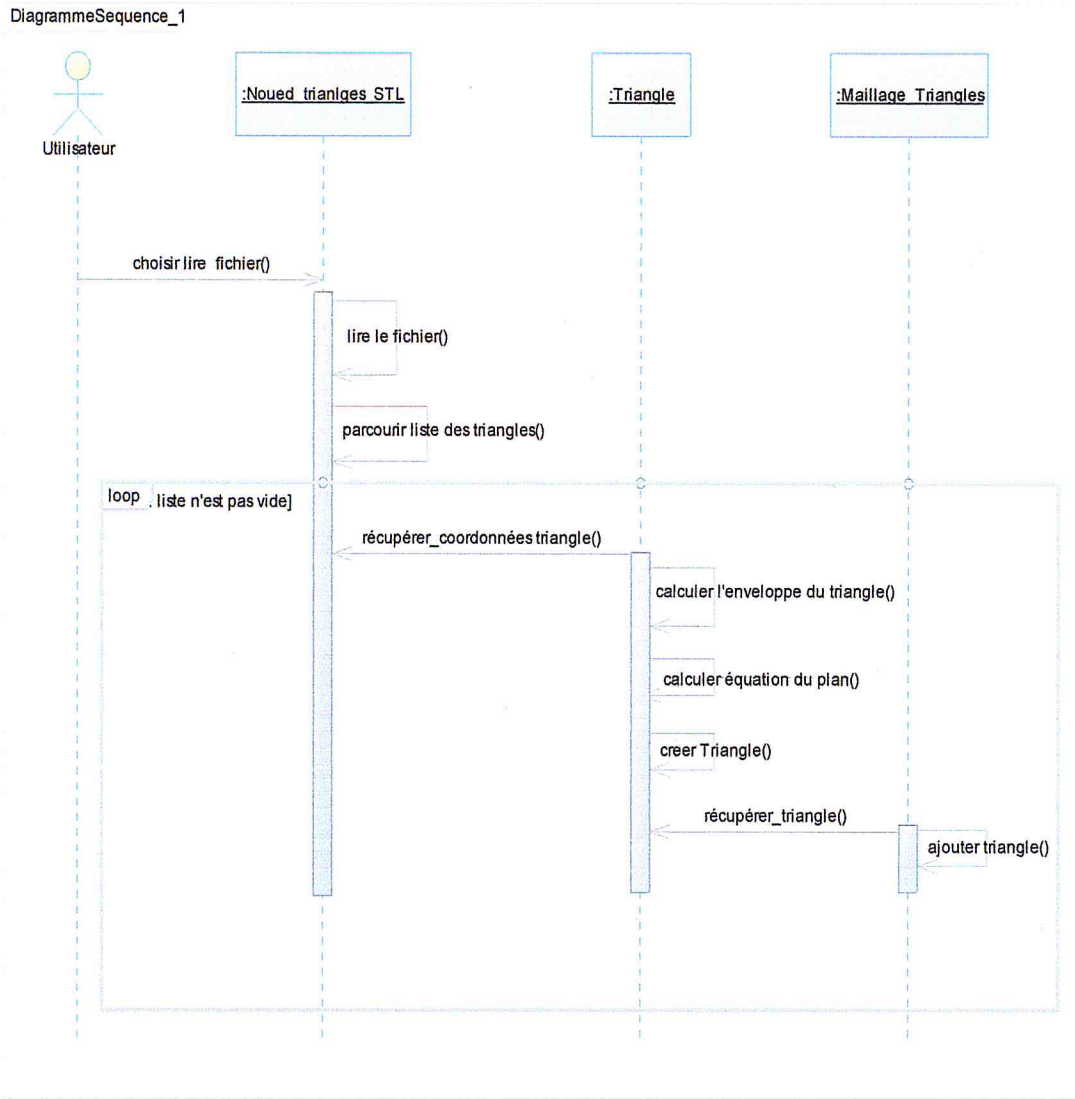


Figure 18 : ***diagramme de séquence récupérer et structurer triangles.***

3.3.3 Création de la grille :

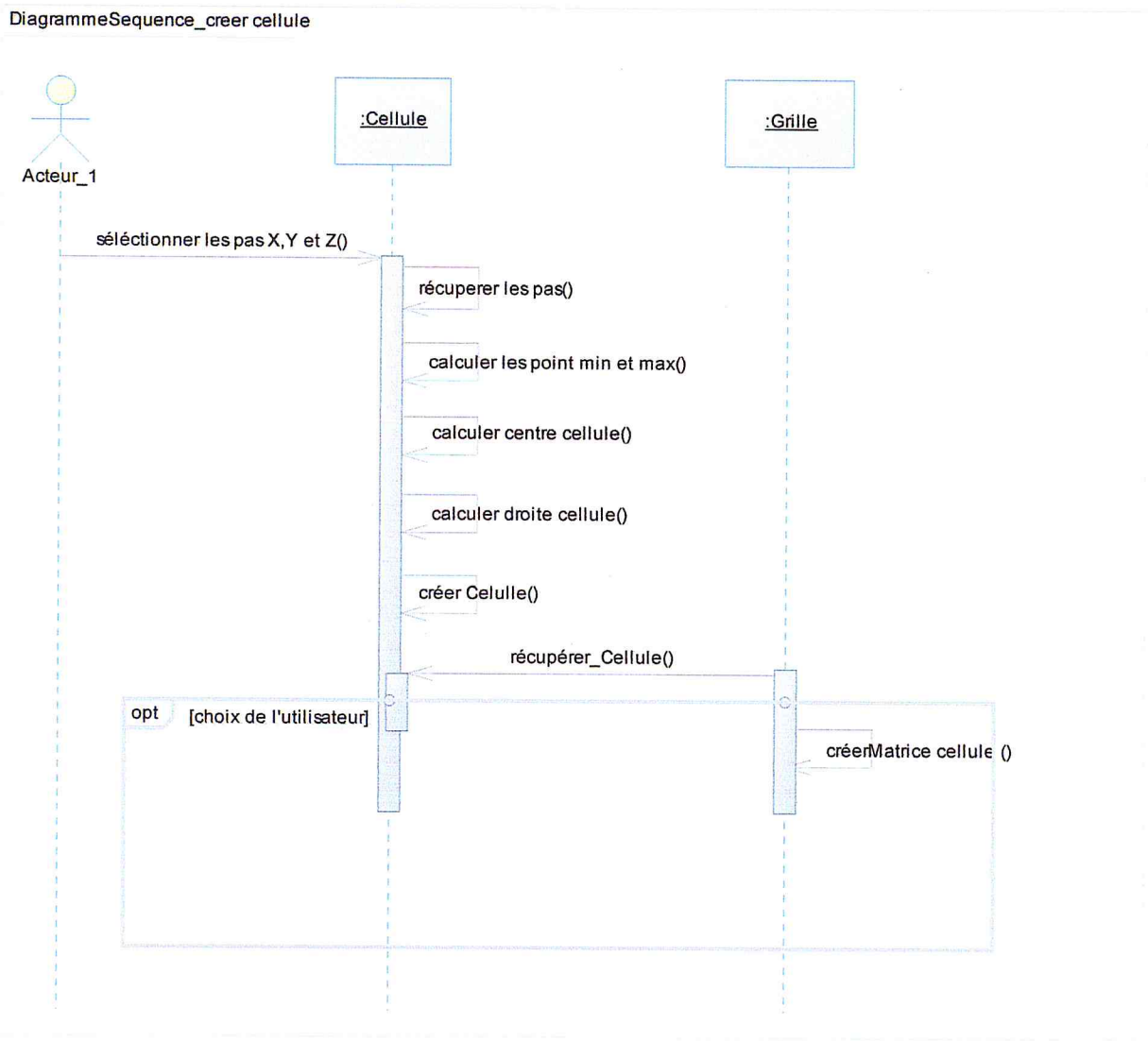


Figure 19 : ***diagramme de séquence créer Grille.***

3.3.4 Calculer les intersections :

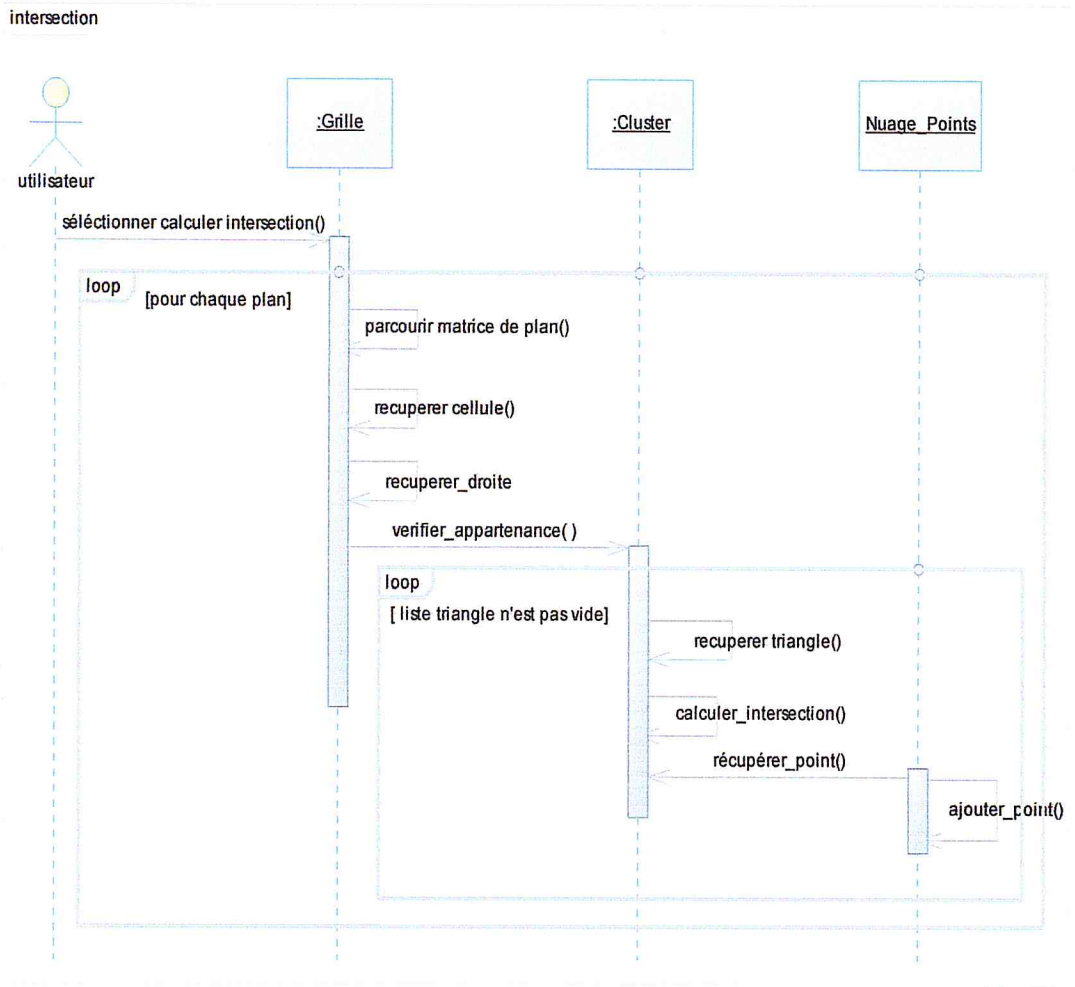


Figure 20: ***diagramme de séquence calculer intersection.***

3.3.5 Visualiser les points :

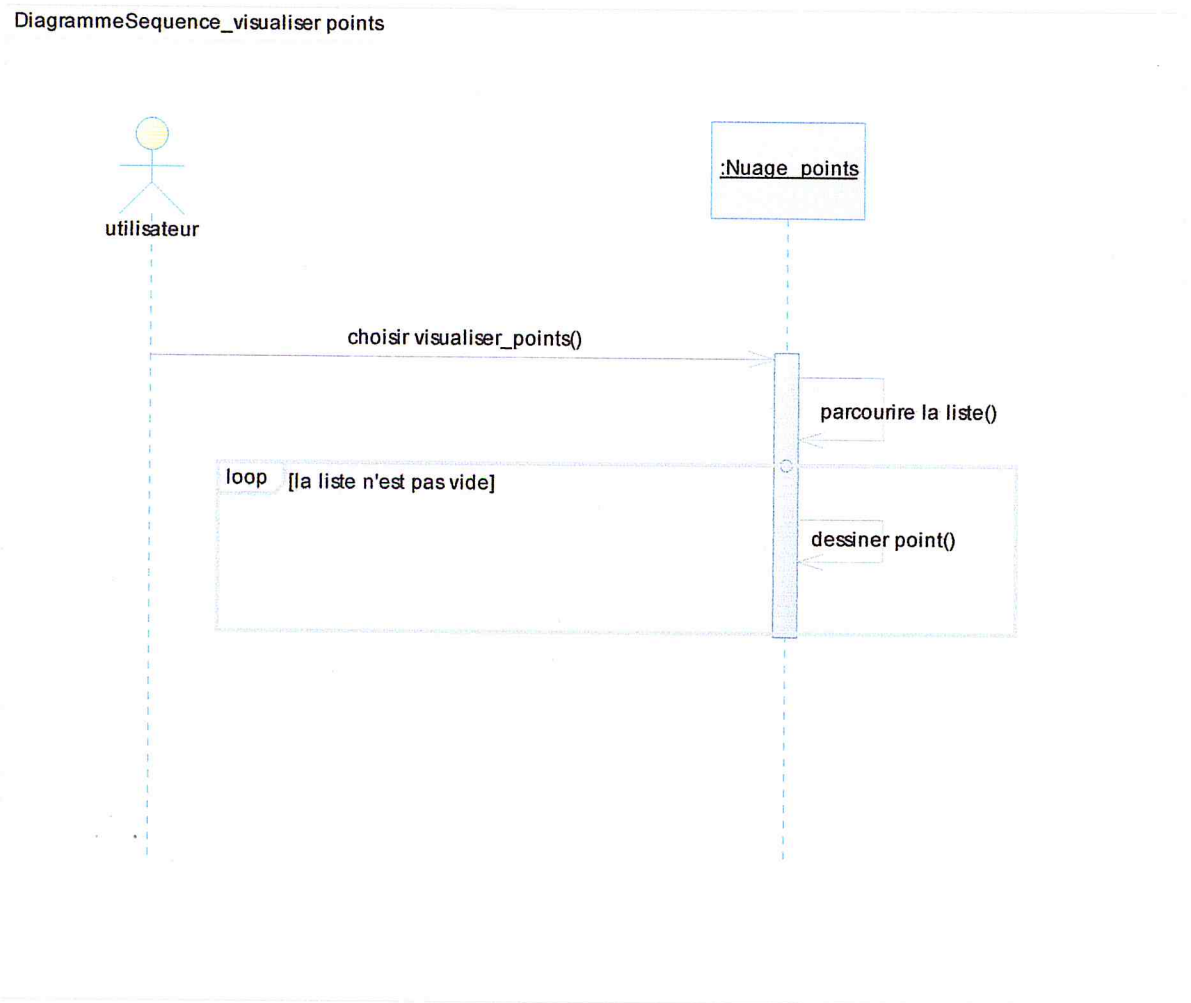


Figure 21 : ***diagramme de séquence visualiser points.***

3.3.6 Visualiser les triangles :

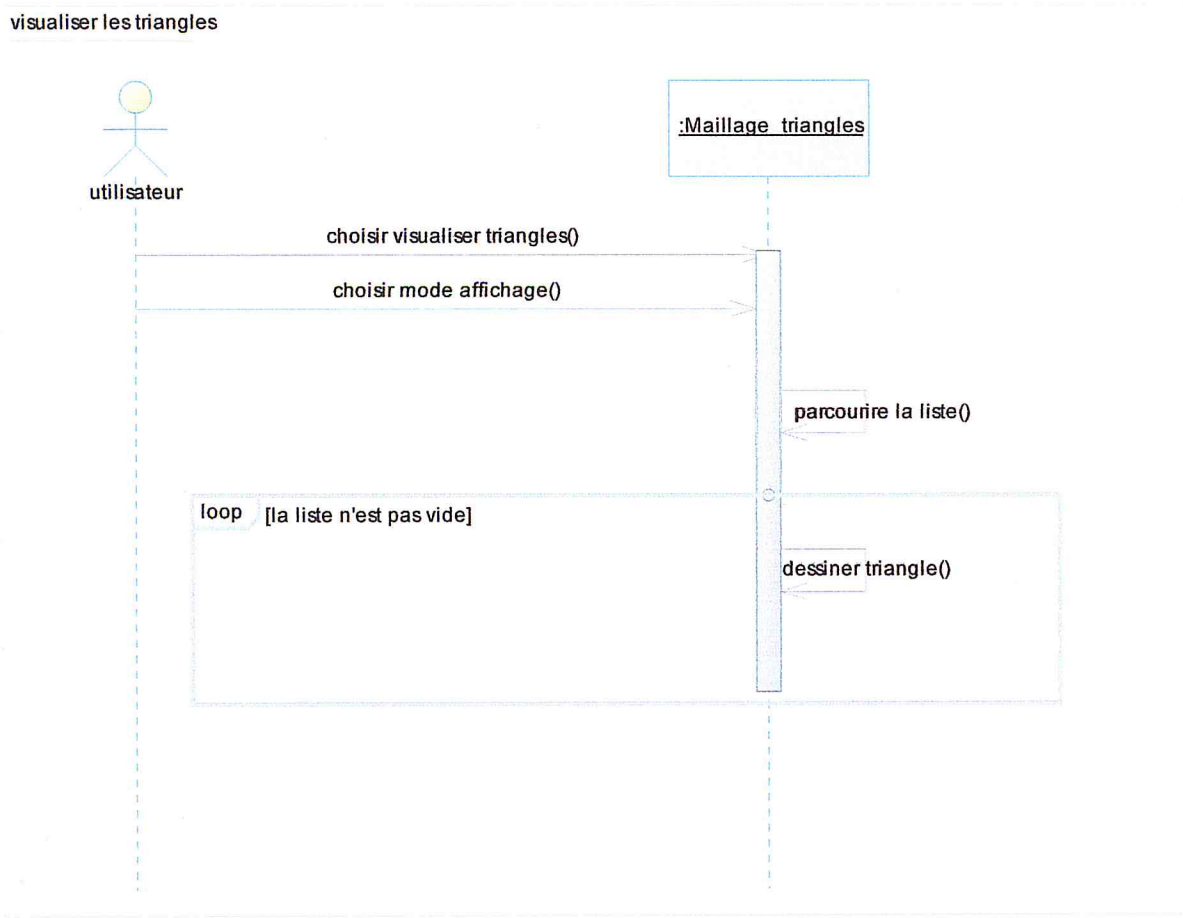


Figure 22 : ***diagramme de séquence visualiser triangles.***

3.3.7 Visualiser le Plan :

Figure 23 : ***diagramme de séquence visualisé plan.***

3.4 Diagramme de classes :

Les diagrammes de classes décrivent les types des objets qui composent un système et les différents types de relations statiques qui existent entre eux.

Les diagrammes de classes font abstraction du comportement du système.

Les classes qui composent notre système sont :

- Classe Point.
- Classe Triangle.
- Classe Nuage_Points.
- Classe Maillage.
- Classe Dexel.
- Classe Cellule.
- Classe Brute.
- Classe Droite.
- Classe Enveloppe.
- Classe Plan.
- Classe Cluster.

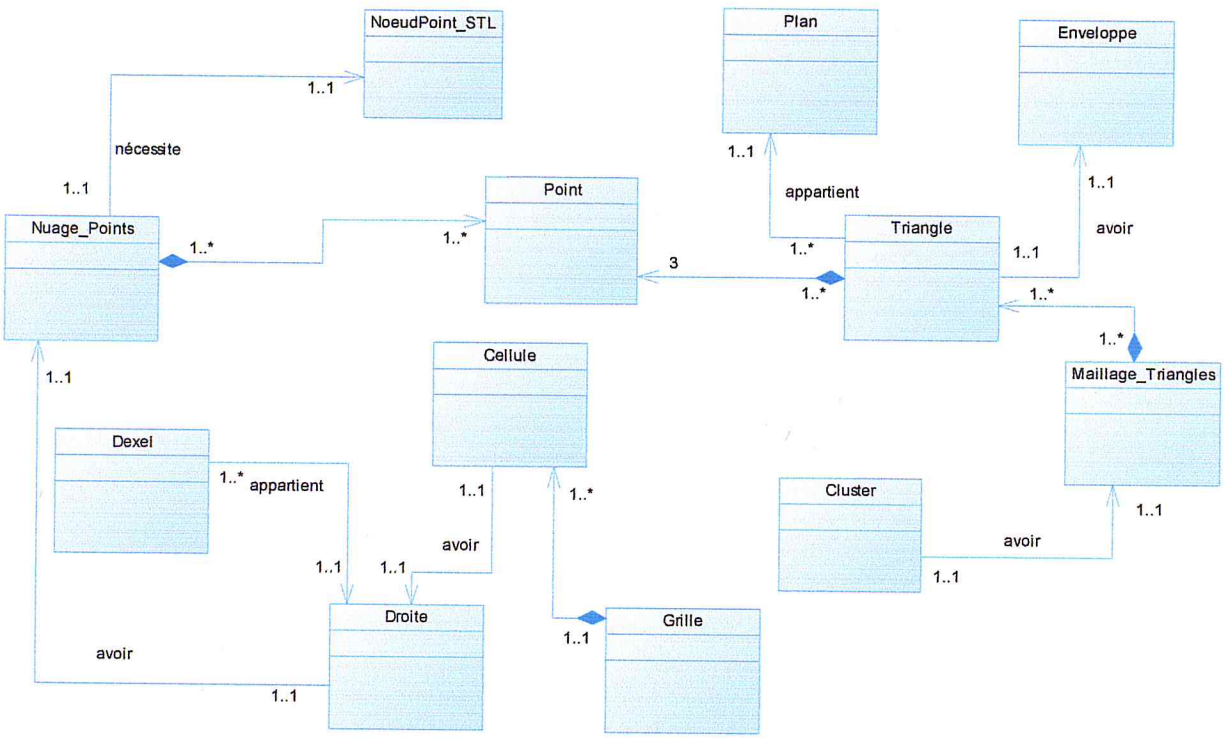


Figure 24: Diagramme de classes.

La représentation détaillée (les attributs et les fonctions) de chaque classe Implémentée est décrite dans les figures suivantes :

3.4.1 Classe Nuage_Points: c'est la classe qui regroupe les points du fichier STL. Nous pouvons facilement accéder à un point à partir d'une position donnée pour récupérer ses coordonnées. (Figure 25). Parmi les fonctions de la classe **Nuage_Points**, nous avons :

- getTaille () : permet de récupérer le nombre de point de fichier STL.
- dessiner_liste_point () : permet de dessiner tous les points du fichier STL.

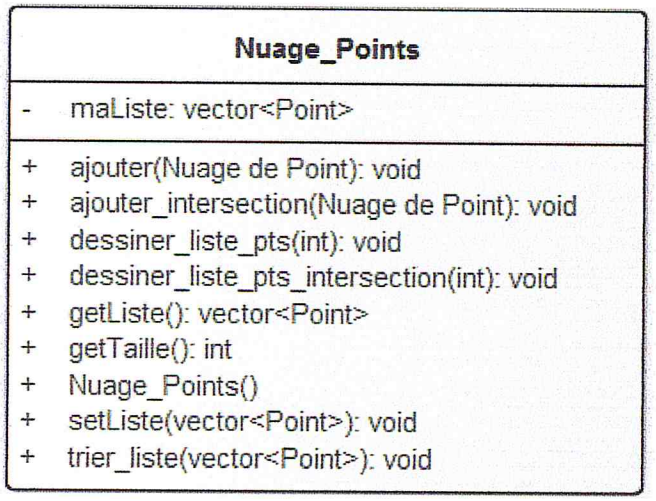


Figure 25: Classe Nuage_Points.

3.4.2 Classe Maillage : regroupe un ensemble de triangles dans une liste. Les attributs et fonctions de cette classe sont donnés par la Figure 26.

Parmi les fonctions de la classe **Maillage**, nous avons :

- getTaille () : permet de récupérer le nombre de triangle de fichier STL.
- dessiner_liste_triangle_filaire () : permet de dessiner tous les triangles du fichier STL.

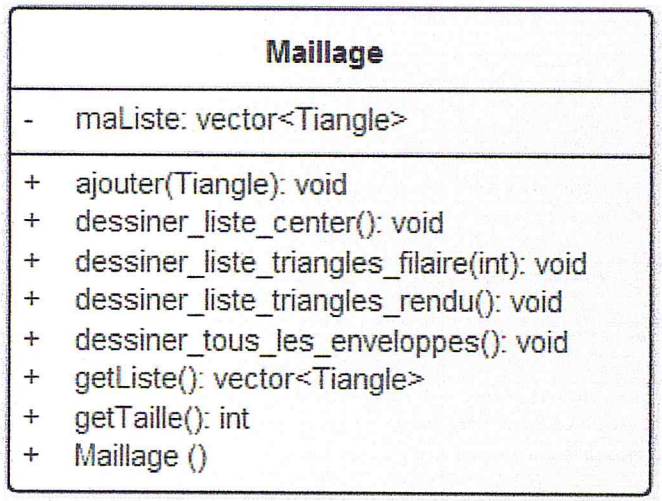


Figure 26: Classe Maillage.

3.4.3 Classe Cellule : c'est une composante de la grille, elle est comptée selon le pas de l'utilisateur et les limites de modèle de la pièce. Chaque cellule est caractériser pas ces limites, centre et droite.

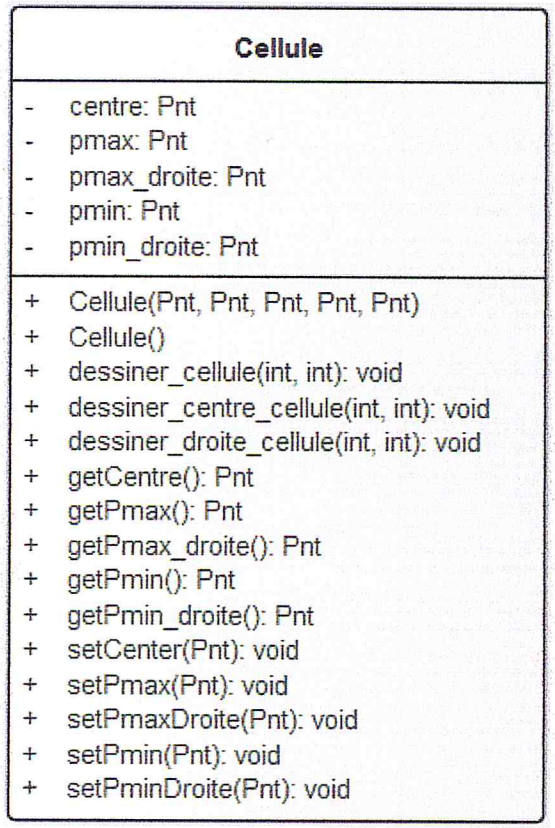


Figure 27: Classe Cellule.

3.4.4 Classe Grille : c'est l'ensemble des cellules, ses attributs et ses fonctions sont données par la figure 28.

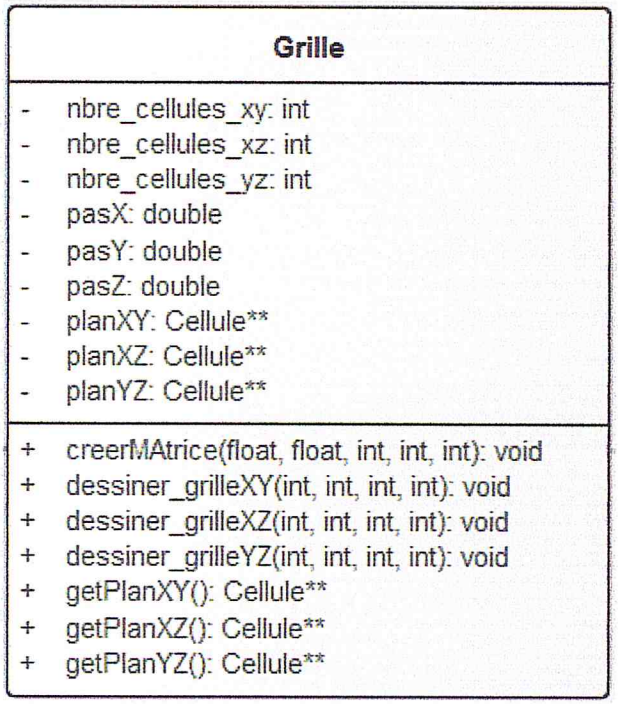


Figure 28: Classe Grille.

La classe grille comporte plusieurs fonctions, les plus importantes sont les suivantes :

- creerMatrice () :cette fonction sert à créer l'ensemble des cellules.
- dessiner_grille_XY (), dessiner_grille_XZ () et dessiner_grille_YZ () : permettent de dessiner les grilles dans les trois plan.

3.4.5 Classe Droite : est définie par deux points, pmin et pmax, elle est utilisé pour calculer les points d'intersection avec les triangles du modèle STL.

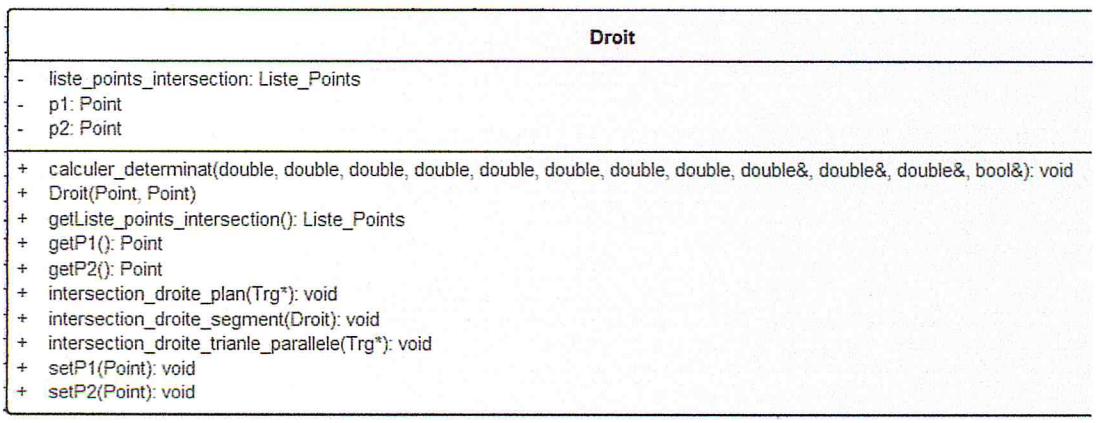


Figure 29: Classe Droite.

les fonctions inter_droite_plan(), inters_droite_segment(), inters_droite_triangle(), sont utilisées pour vérifier tous les cas d'intersections possible pour un droite et un triangle.

3.4.6 Classe Dexel : C'est une classe qui englobe les différentes caractéristiques des dexels. Ses attributs et fonctions sont donnés par la figure 30.

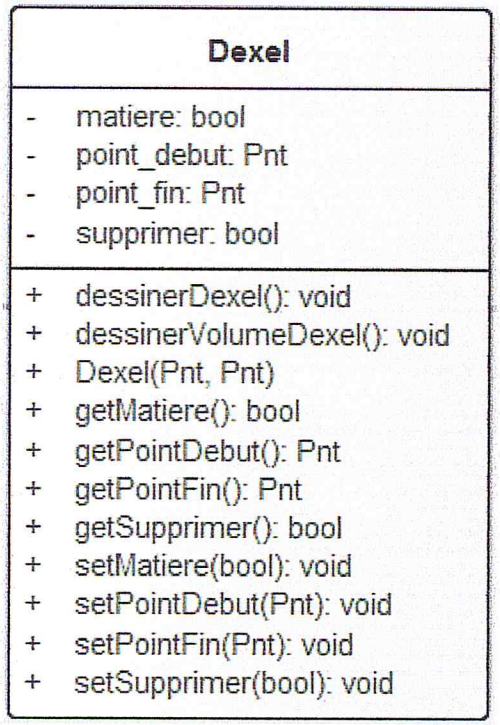


Figure 30: Classe Dexel.

- dessinerDexel () : permet de dessiner le dexel suivant un axe donnée qui est limité par deux points d'intersections.
- getMatiere () : permet de savoir le positionnement de dexel, s'il est dans la pièce ou non.

3.4.7 Classe Cluster : C'est une classe qui englobe les différentes caractéristiques des Clusters. Ses attributs et fonctions sont donnés par la figure 31.

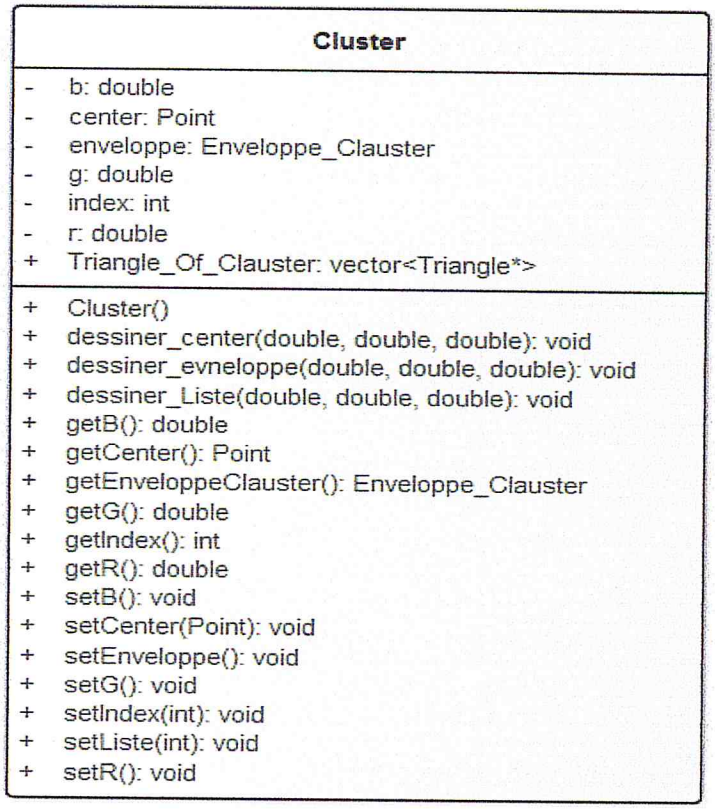


Figure 31: Classe Cluster.

- getListeTriangle () : elle permet de récupérer tous les triangles qui appartient à ce cluster.
- dessiner_center (), dessiner_enveloppe () et dessiner_liste () sont des fonctions qui permettent de dessiner les triangles de cluster, le centroïd et l'enveloppe de cluster.

4. Conclusion

Dans ce chapitre, nous avons définis la conception de notre application en commençant par la présentation de la problématique, ainsi que les différents objectifs visés à atteindre. Ensuite en passant à la présentation des solutions que nous avons proposées avec les différents diagrammes de conception en utilisant le modèle orienté objet et le langage de modélisation UML.

1. Introduction

Après une synthèse bibliographique dans le premier chapitre, nous allons nous intéresser dans ce chapitre à la conception de l'application en utilisant le modèle orienté objet et le langage de modélisation UML (*Unified Modeling Langage*). La conception a pour objectif de permettre de formaliser les étapes préliminaires du développement d'un système afin de le rendre plus fidèle aux besoins du client, mais avant nous allons analyser les différentes solutions existantes afin d'en tirer les avantages et d'éviter les problèmes déjà rencontrés par les autres développeurs on proposera notre conception en se basant sur les résultats de cette analyse.

2. Analyse :

Avant de commencer la partie conception de notre application, il faut d'abord faire une petite analyse pour justifier le but de développement de notre application, citer les insuffisances dans les applications précédentes et les nouveautés apportées dans notre application.

Dans le chapitre précédent nous avons vu les différentes techniques de représentation volumique comme le voxel, Dexel et le triple Dexel (qui est utilisé dans notre cas). Les travaux précédents concernant la simulation de l'usinage au niveau de CDTA, ils ont utilisé la technique de représentation Dexel, cette technique a des limites surtout là où il y'a des évidements et des courbures dans la pièce à usiner.

Le modèle de triple-dexel offre une meilleure résolution que le modèle unique dexel, en comparaison avec le modèle populaire de voxel, le modèle de triple-dexel nécessite moins de mémoire de l'ordinateur. Les données de triple-dexel ne consomment pas beaucoup de mémoire, et le travail se fait en un temps dans les normes, et le modèle de surface reconstruit est plus précis que la surface reconstruite à partir de la représentation de voxel ou de dexel. De plus il offre une bonne qualité de la vision de la surface au niveau de la simulation.

L'avantage le plus important du triple dexel par rapport au dexel est dans l'outil d'usinage, en dexel on utilise Fraiseuses à commande numérique 03 axes par contre en triple dexel on utilise Fraiseuses à commande numérique 05 axes. (La fraiseuse à commande numérique est

un ensemble comportant la machine-outil, le directeur de commande numérique, l'armoire électrique et le pupitre de commande).

En usinage à trois axes, pour une surface usinable, il existe une unique position de l'outil tangente à la surface en une position donnée. Par contre, en usinage à cinq axes, il existe une infinité de positions de l'outil tangente à la surface au point de contact, par conséquent le Tripe dexel c'est la solution qui répond au besoin d'usinage en cinq axes.

Dans la partie suivante nous allons commencer la conception de notre application.

3. Modélisation de l'application avec UML :

3.1 Méthode de modélisation :

Chaque application et chaque système avant d'être réalisé doit passé par une étape de conception avec une méthode ou bien langage de modélisation, cette étape permet de décrire les fonctionnalités du système, son comportement et tout le détail nécessaire pour la réalisation de ce système et son déroulement.

Les langages de modélisation orientés objet ont fait leur apparition entre le milieu des années soixante-dix et la fin des années quatre-vingt, quand les spécialistes méthode, confrontés à un nouveau genre de langages de programmation orientés objet et à des applications de plus en plus complexes.

UML (*Unified Modeling Language* - langage unifié pour la modélisation objet) est un moyen d'exprimer des modèles objet, c'est-à-dire que le modèle fourni par UML est valable pour n'importe quel langage de programmation. UML couvre toutes les phases du cycle de vie d'un logiciel, et à pour but de documenter les modèles objets.

UML est un langage standard conçu pour l'écriture de plans d'élaboration de logiciels, résultant de la fusion de 3 méthodes : OMT, OOSE et BOOCH qui possède les caractéristiques suivantes :

1. Est avant tout un support de communication performant, sa notation graphique permet d'exprimer visuellement une solution objet, ce qui facilite la comparaison et l'évaluation des solutions.

2. L'aspect formel de sa notation limite les ambiguïtés et les incompréhensions.

3. UML donc bien plus qu'un simple outil qui permet de dessiner des représentations mentales, il permet de parler un langage commun, normalisé mais accessible, car visuel.

Un modèle est une simplification de la réalité, donc la modélisation consiste à créer une représentation simplifiée d'un problème, elle comporte deux composants :

- **L'expression des besoins** : définir les besoins des Utilisateurs de système.
- **La conception** : la mise au point d'une solution.

UML fournit un moyen astucieux permettant de représenter diverses projections d'une même représentation grâce aux vues. Une vue est constituée d'un ou plusieurs diagrammes.

Dans notre projet, nous avons utilisé les diagrammes suivants :

- **L'expression des besoins :**

- Diagramme de cas d'utilisation.

- **Conception :**

- Diagramme de séquence,

- Diagramme de classes.

3.2 Diagramme de cas d'utilisation :

Les cas d'utilisation permettent de déterminer les besoins des utilisateurs et de capturer les exigences fonctionnelles d'un système. En d'autres termes, ils décrivent le comportement d'un système du point de vue de ses utilisateurs. Ils décrivent les interactions entre les utilisateurs du système et le système lui-même.

Un diagramme de cas d'utilisation permet de représenter graphiquement les cas d'utilisation.

Dans ce qui suit nous allons présenter les différentes fonctionnalités de notre système, Quels sont les offres qui proposent aux utilisateurs avec un diagramme de cas d'utilisation générale qui donne une vue générale à notre système.

1er niveau :

Notre système permet la simulation d'usinage des pièces mécaniques.

2eme niveau :

Ils existent deux grandes fonctionnalités qui sont :

- Acquisition du modèle de représentation d'une pièce.
- Remplissage du modèle.

3.2.1 Diagramme de cas d'utilisation générale :

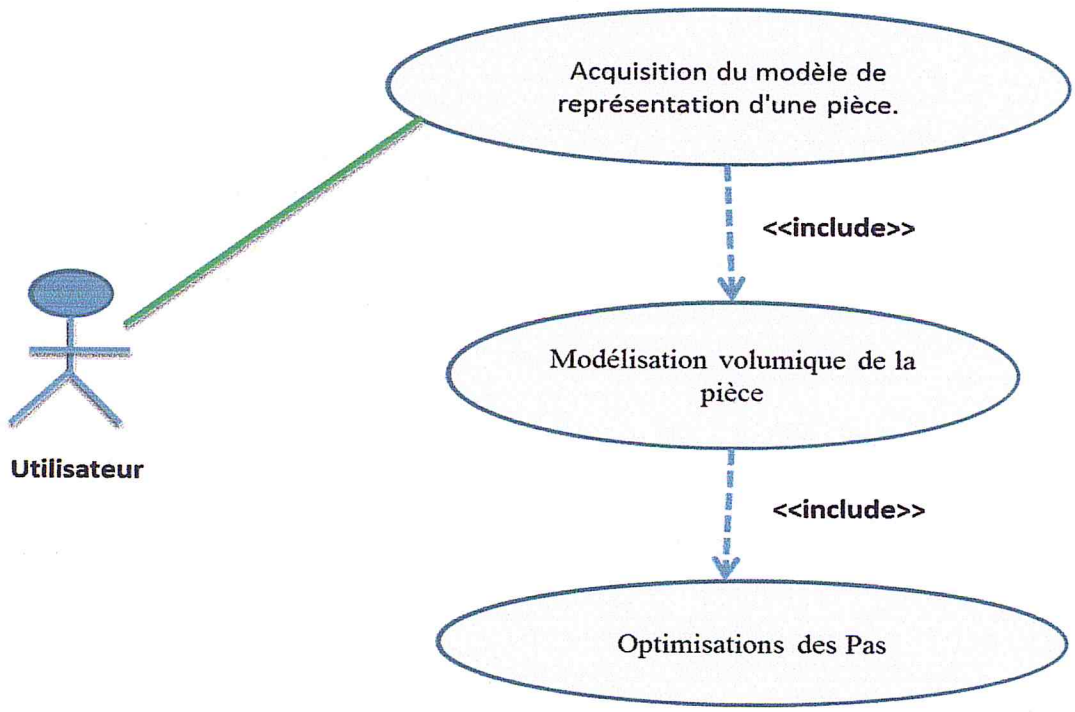


Figure 14 : ***Diagramme cas d'utilisation générale.***

Maintenant nous allons détailler ce diagramme général en détaillant les cas d'utilisations générales de notre système, commençant par le 1er cas d'utilisation qui est l'Acquisition du modèle de représentation d'une pièce:

3.2.1.1 Diagramme de cas d'utilisation "Acquisition du modèle de représentation d'une pièce":

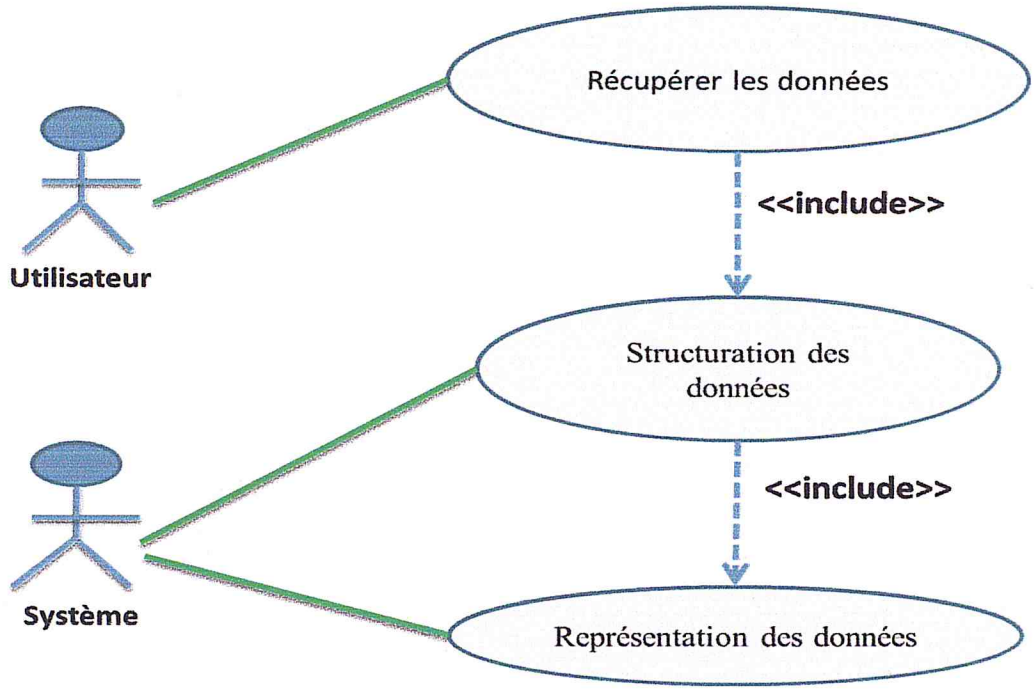


Diagramme Cas d'utilisation Acquisition du modèle

Figure 15: ***diagramme cas d'utilisation Acquisition des données.***

3.2.1.2 Le diagramme de cas d'utilisation suivant d'écrit le Remplissage du modèle :

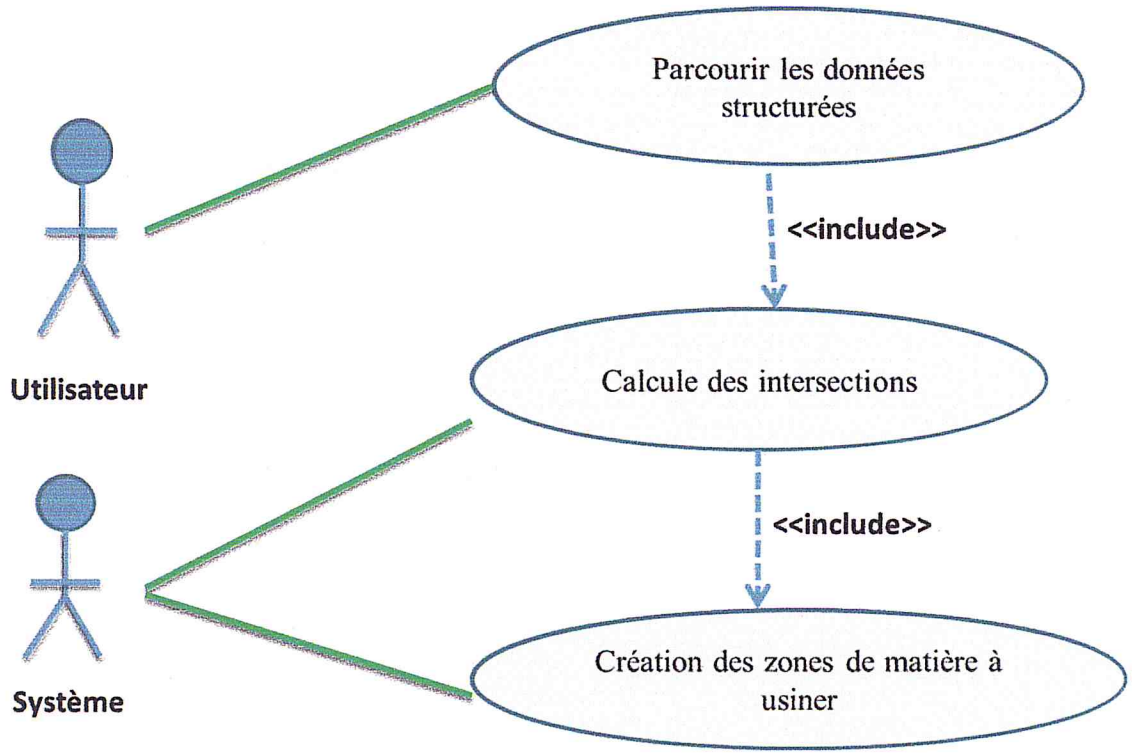


Diagramme Cas d'utilisation Modélisation volumique de la pièce

Figure 16 : ***Diagramme cas d'utilisation Remplissage de modèle.***

L'étude conceptuelle consiste à décrire le système futur. Ce nouveau système tiendra compte des nouveaux besoins des utilisateurs.

3.3 Diagrammes de séquence :

3.3.1 récupérer et structurer les points :

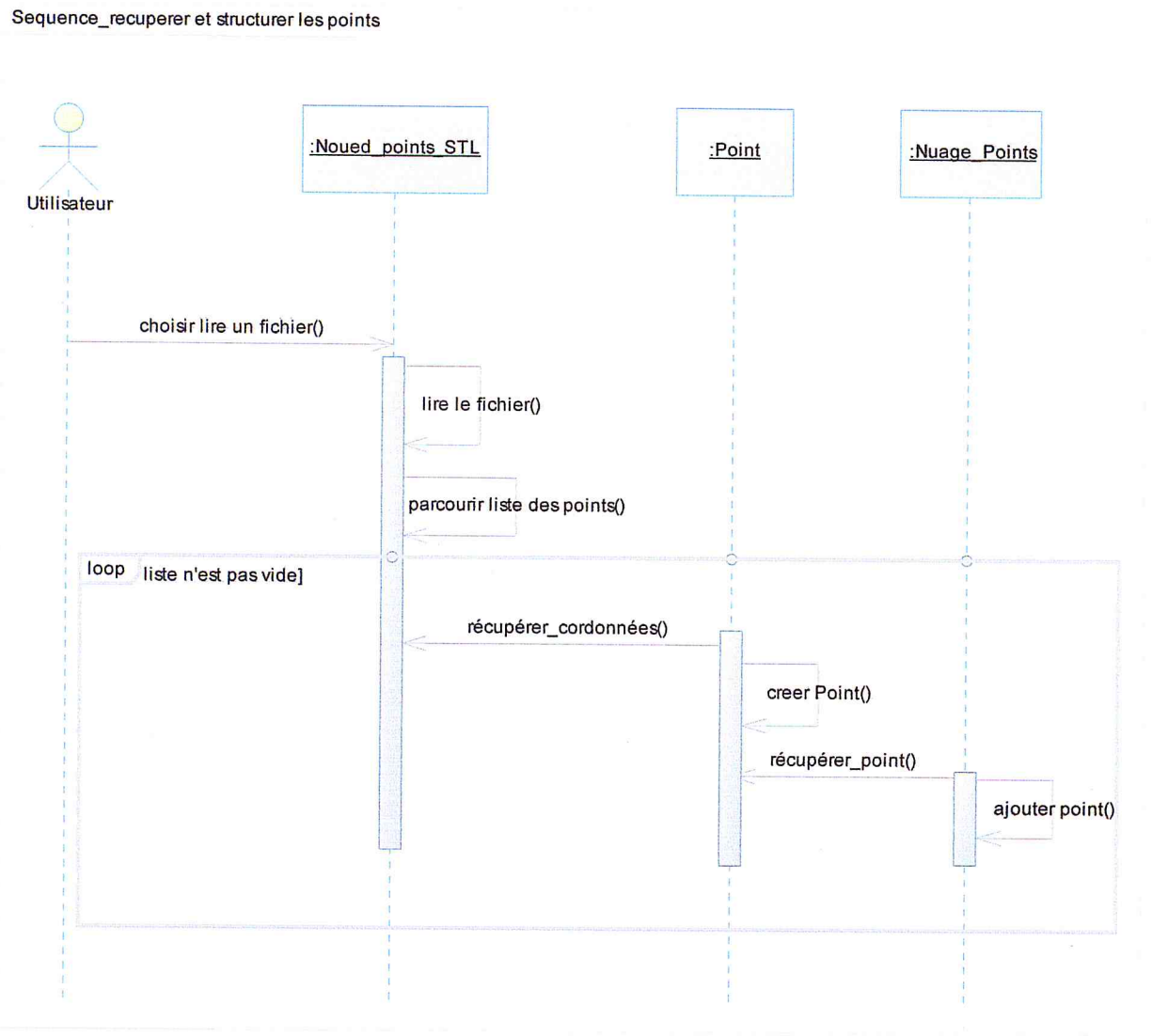


Figure 17 : ***diagramme de séquence récupérer et structurer point.***

3.3.2 récupérer et structurer les triangles :

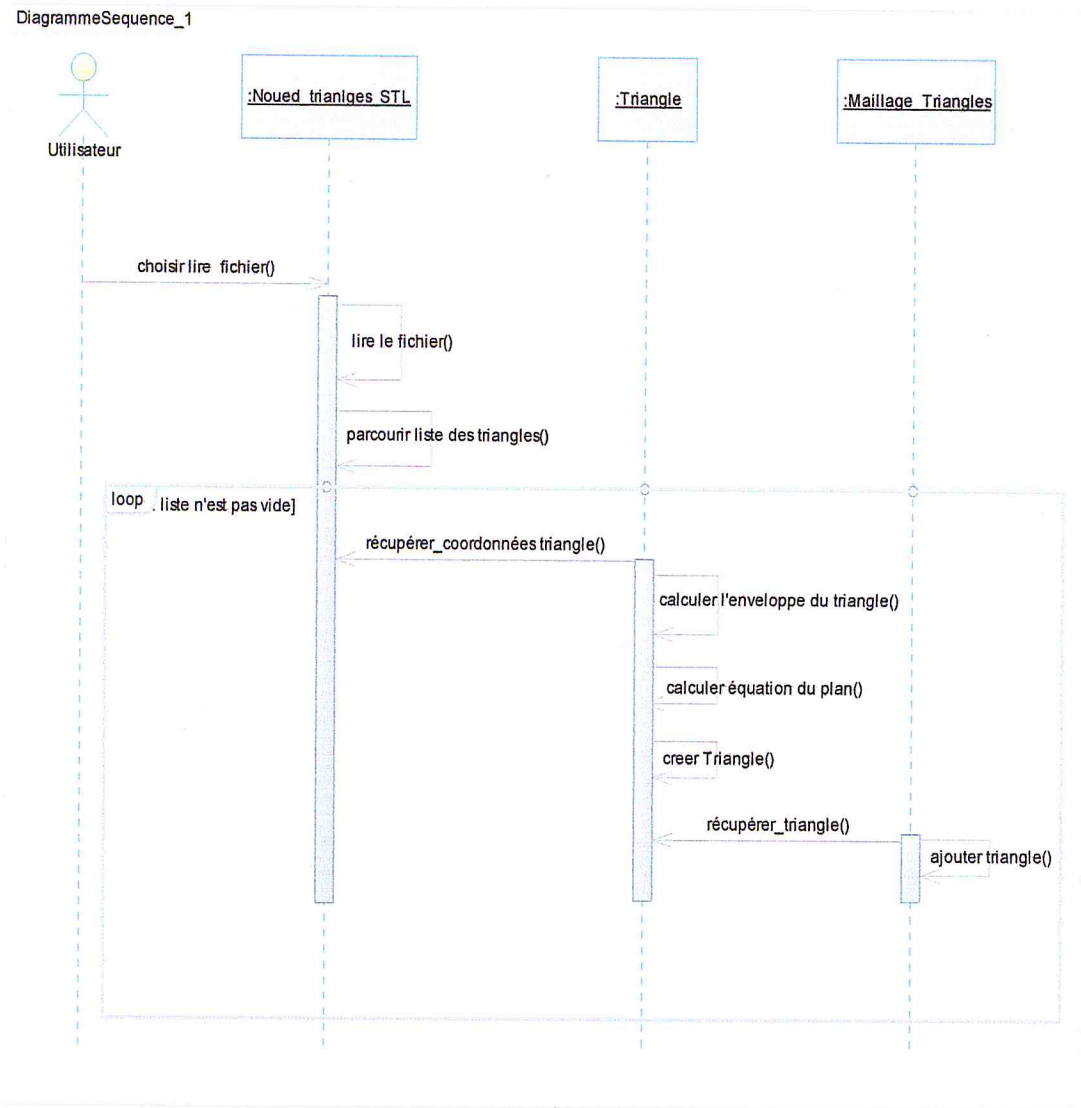


Figure 18 : ***diagramme de séquence récupérer et structurer triangles.***

3.3.3 Création de la grille :

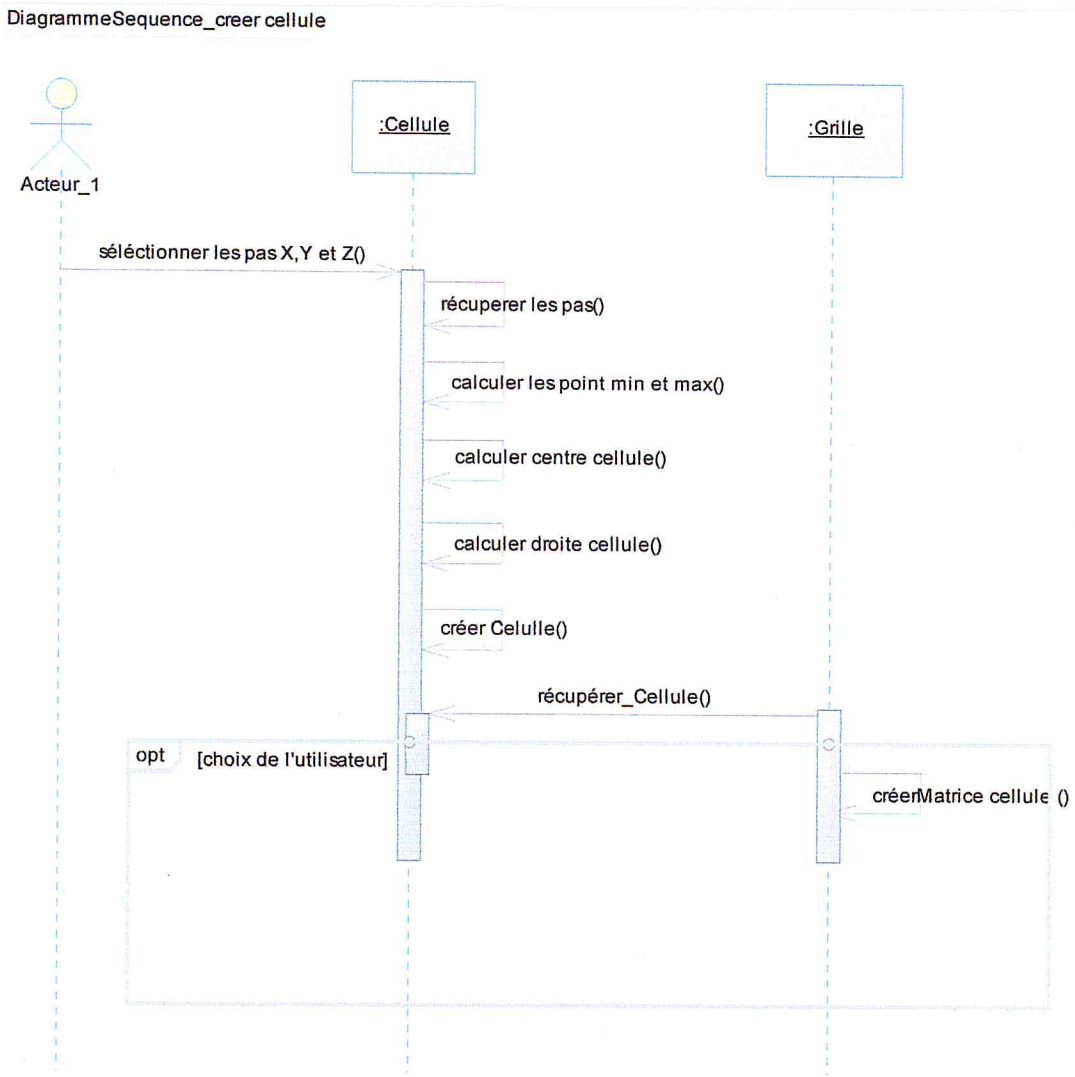


Figure 19 : ***diagramme de séquence créer Grille.***

3.3.4 Calculer les intersections :

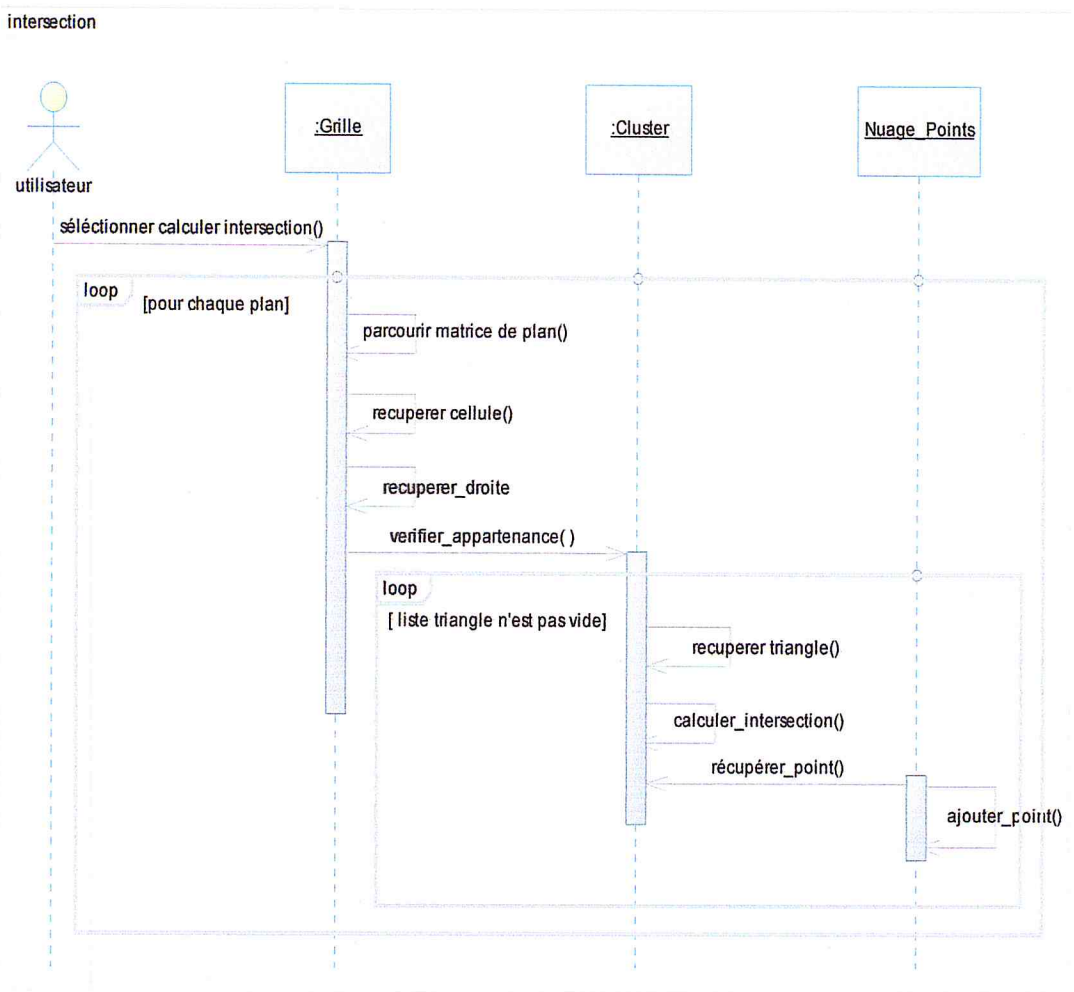


Figure 20: ***diagramme de séquence calculer intersection.***

3.3.5 Visualiser les points :

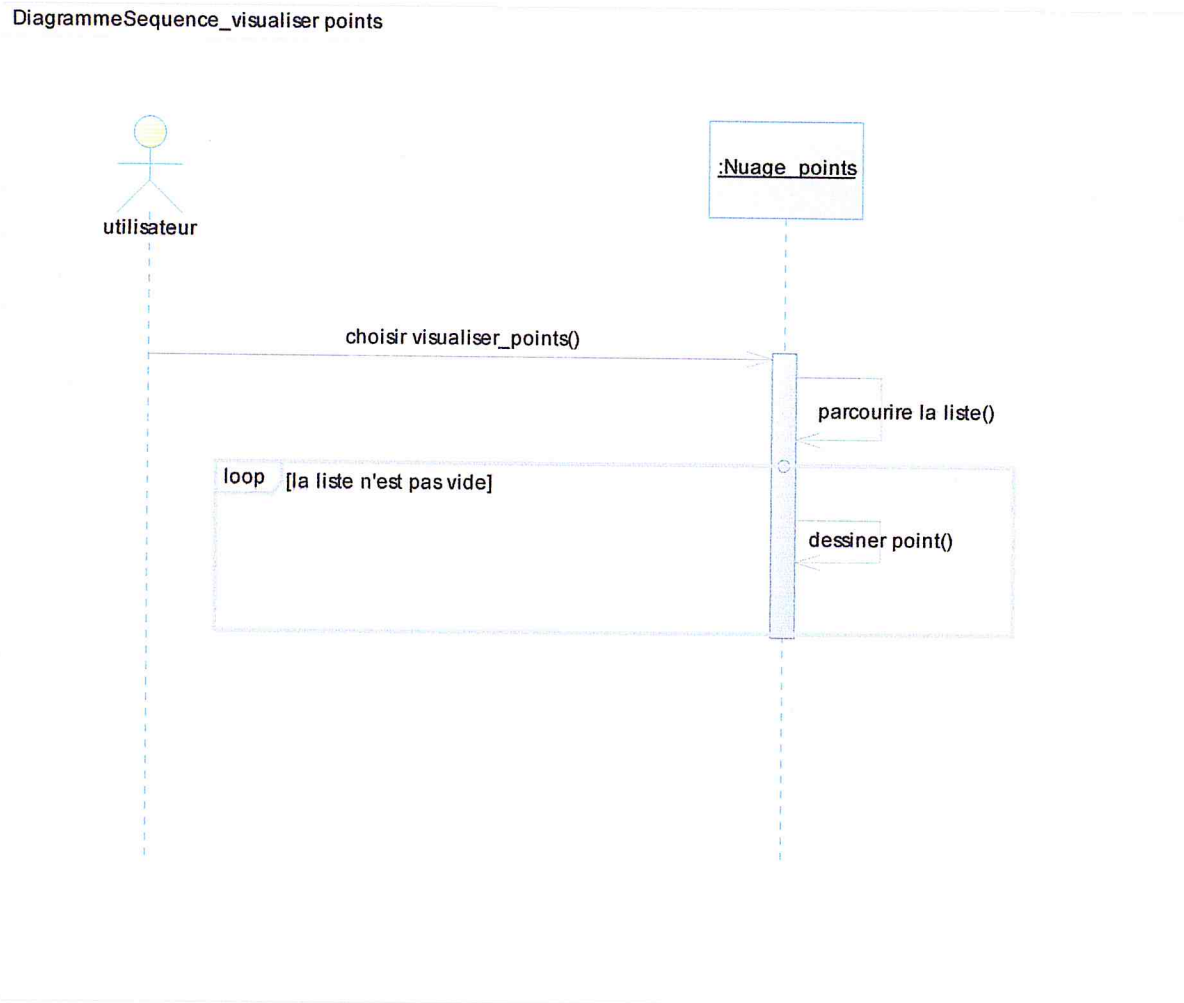


Figure 21 : ***diagramme de séquence visualiser points.***

3.3.6 Visualiser les triangles :

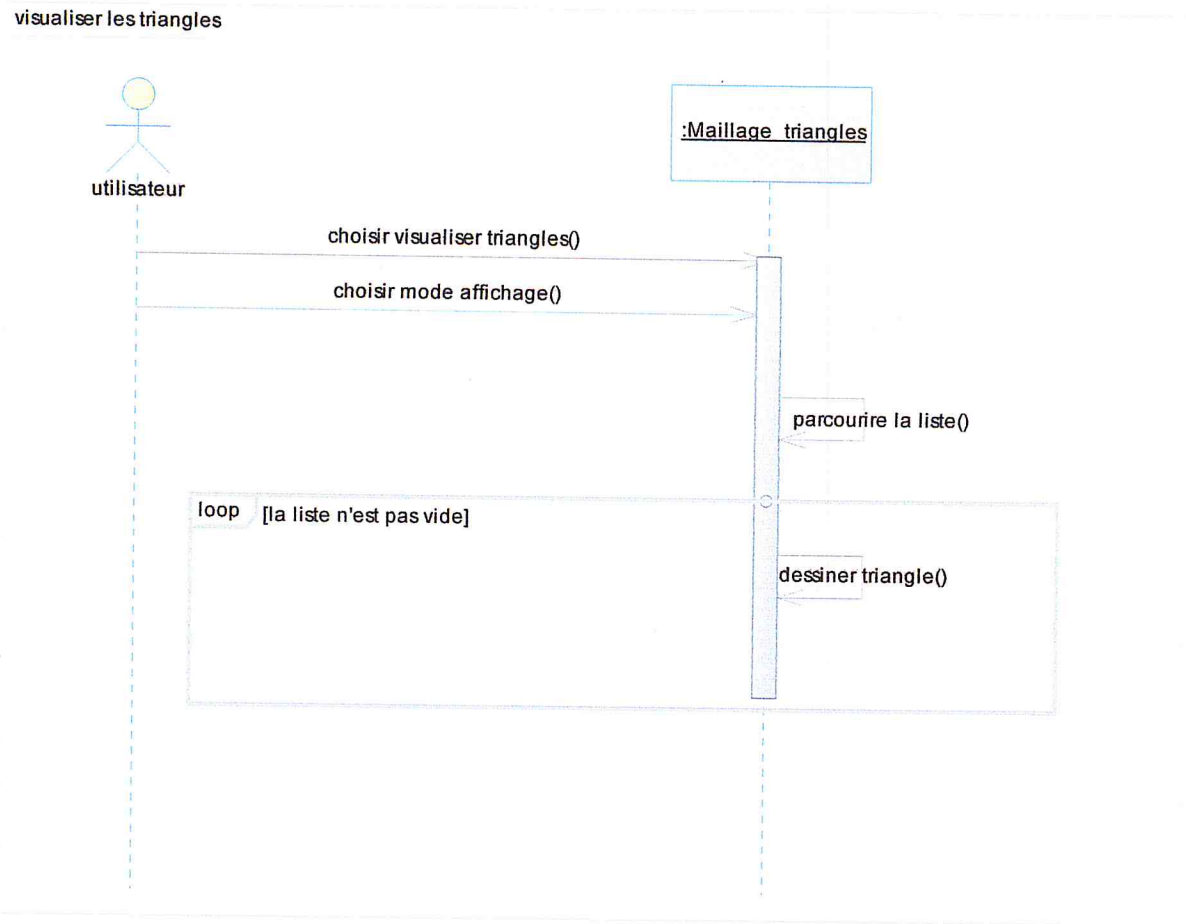


Figure 22 : ***diagramme de séquence visualiser triangles.***

3.3.7 Visualiser le Plan :

Figure 23 : ***diagramme de séquence visualisé plan.***

3.4 Diagramme de classes :

Les diagrammes de classes décrivent les types des objets qui composent un système et les différents types de relations statiques qui existent entre eux.

Les diagrammes de classes font abstraction du comportement du système.

Les classes qui composent notre système sont :

- Classe Point.
- Classe Triangle.
- Classe Nuage_Points.
- Classe Maillage.
- Classe Dexel.
- Classe Cellule.
- Classe Brute.
- Classe Droite.
- Classe Enveloppe.
- Classe Plan.
- Classe Cluster.

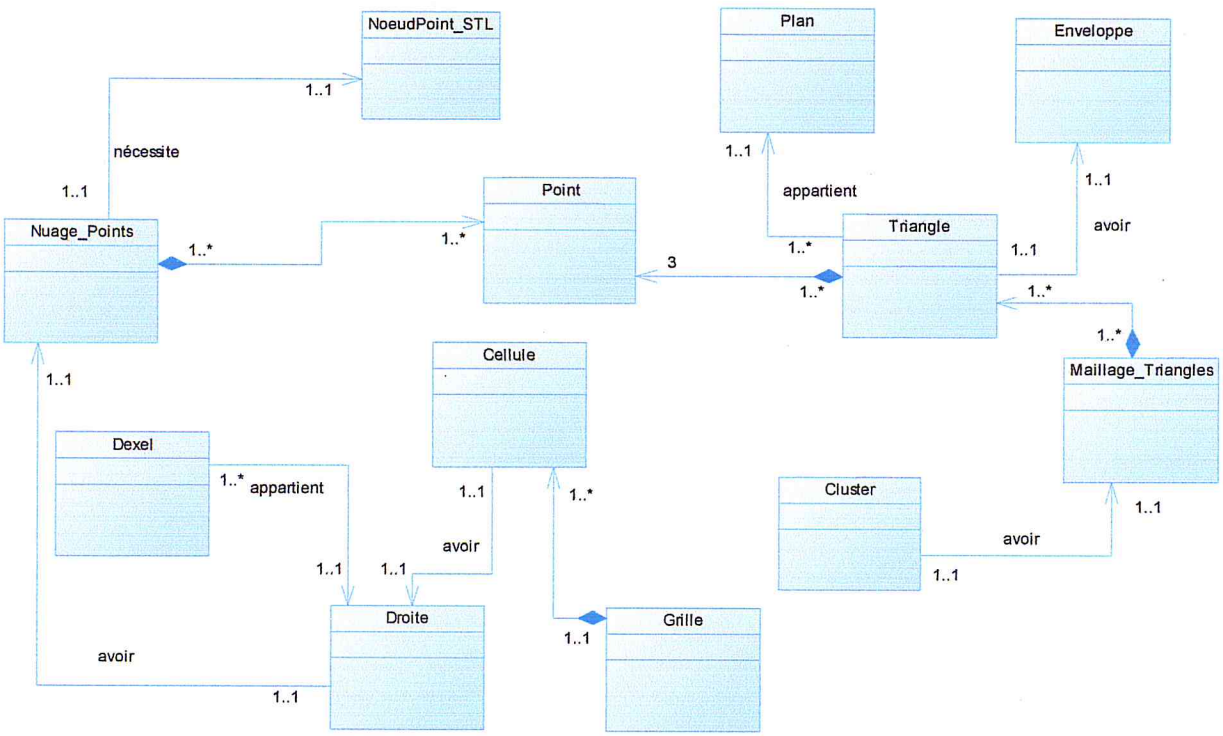


Figure 24: Diagramme de classes.

La représentation détaillée (les attributs et les fonctions) de chaque classe Implémentée est décrite dans les figures suivantes :

3.4.1 Classe Nuage_Points: c'est la classe qui regroupe les points du fichier STL. Nous pouvons facilement accéder à un point à partir d'une position donnée pour récupérer ses coordonnées. (Figure 25). Parmi les fonctions de la classe **Nuage_Points**, nous avons :

- getTaille () : permet de récupérer le nombre de point de fichier STL.
- dessiner_liste_point () : permet de dessiner tous les points du fichier STL.

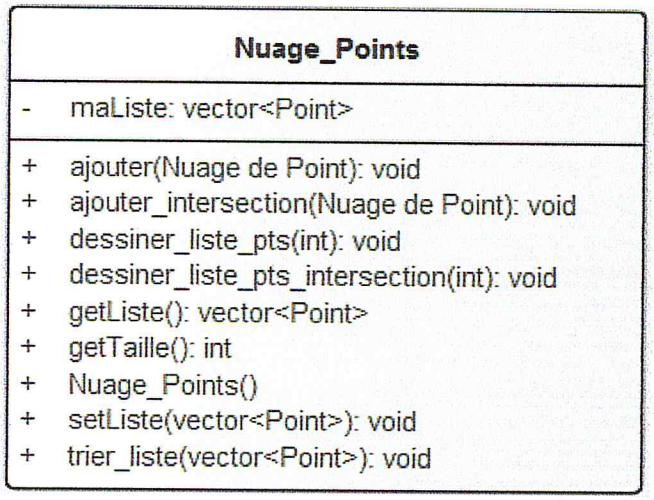


Figure 25: Classe Nuage_Points.

3.4.2 Classe Maillage : regroupe un ensemble de triangles dans une liste. Les attributs et fonctions de cette classe sont donnés par la Figure 26.

Parmi les fonctions de la classe **Maillage**, nous avons :

- getTaille () : permet de récupérer le nombre de triangle de fichier STL.
- dessiner_liste_triangle_filaire () : permet de dessiner tous les triangles du fichier STL.

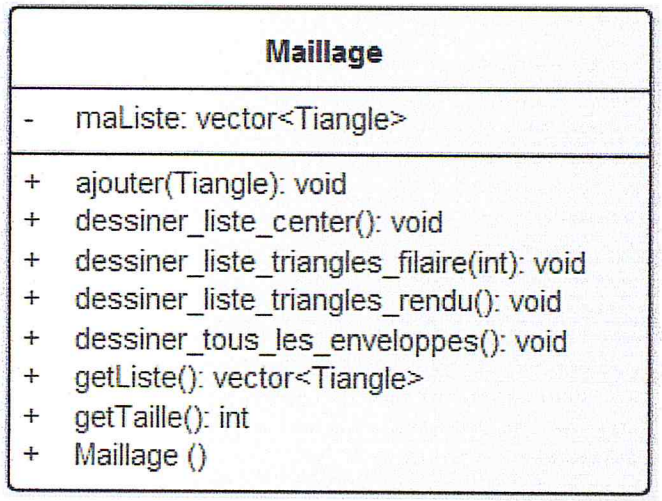


Figure 26: Classe Maillage.

3.4.3 Classe Cellule : c'est une composante de la grille, elle est comptée selon le pas de l'utilisateur et les limites de modèle de la pièce. Chaque cellule est caractériser pas ces limites, centre et droite.

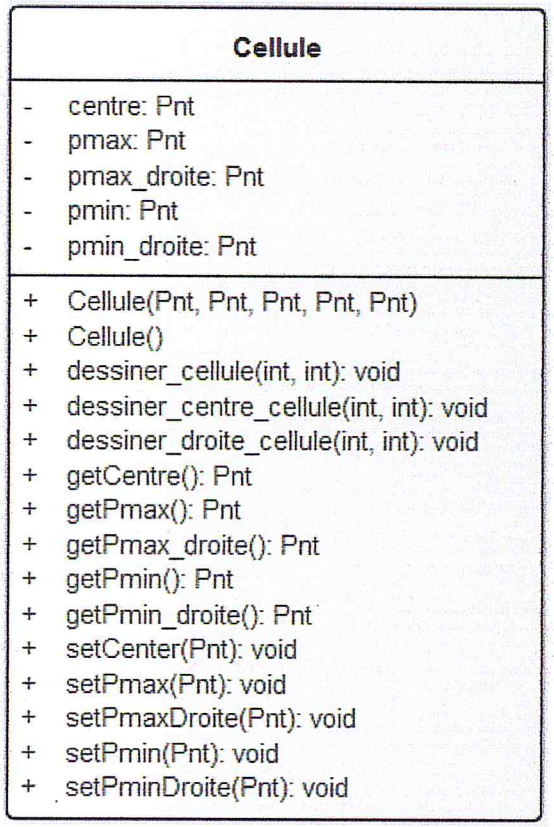


Figure 27: Classe Cellule.

3.4.4 Classe Grille : c'est l'ensemble des cellules, ses attributs et ses fonctions sont données par la figure 28.

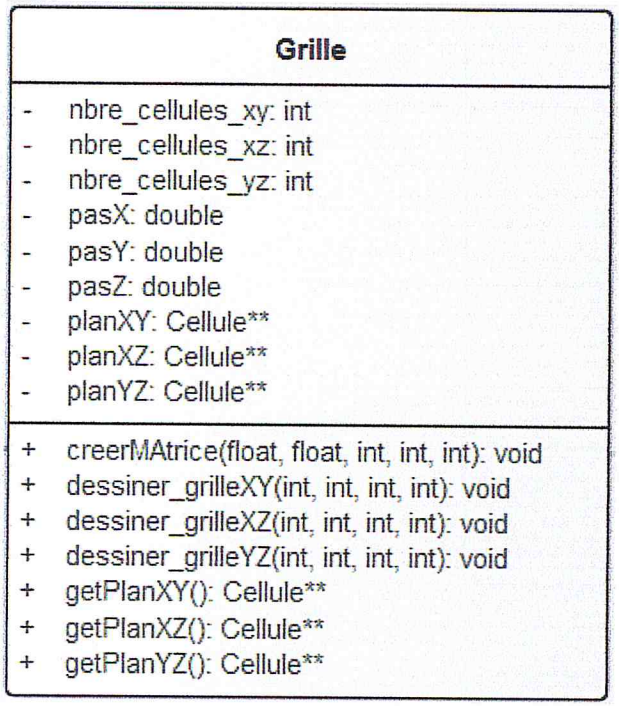


Figure 28: Classe Grille.

La classe grille comporte plusieurs fonctions, les plus importantes sont les suivantes :

- creerMatrice () :cette fonction sert à créer l'ensemble des cellules.
- dessiner_grille_XY (), dessiner_grille_XZ () et dessiner_grille_YZ () : permettent de dessiner les grilles dans les trois plan.

3.4.5 Classe Droite :

est définie par deux points, pmin et pmax, elle est utilisé pour calculer les points d'intersection avec les triangles du modèle STL.

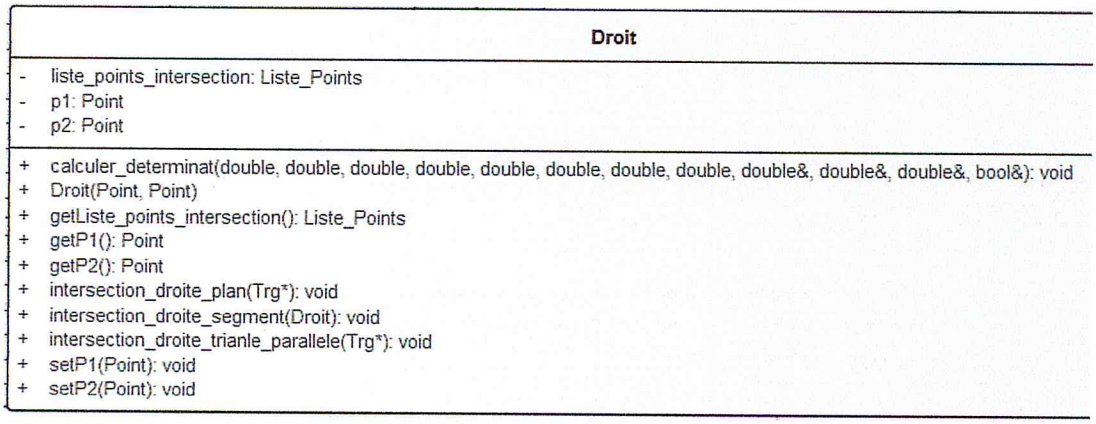


Figure 29: Classe Droite.

les fonctions inter_droite_plan(), inters_droite_segment(), inters_droite_triangle(), sont utilisées pour vérifier tous les cas d'intersections possible pour un droite et un triangle.

3.4.6 Classe Dexel : C'est une classe qui englobe les différentes caractéristiques des dexels. Ses attributs et fonctions sont donnés par la figure 30.

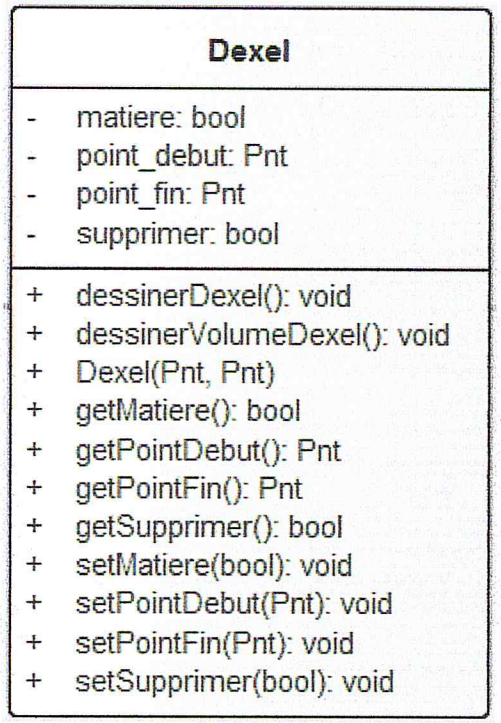


Figure 30: Classe Dexel.

- dessinerDexel () : permet de dessiner le dexel suivant un axe donnée qui est limité par deux points d'intersections.
- getMatiere () : permet de savoir le positionnement de dexel, s'il est dans la pièce ou non.

3.4.7 Classe Cluster : C'est une classe qui englobe les différentes caractéristiques des Clusters. Ses attributs et fonctions sont donnés par la figure 31.

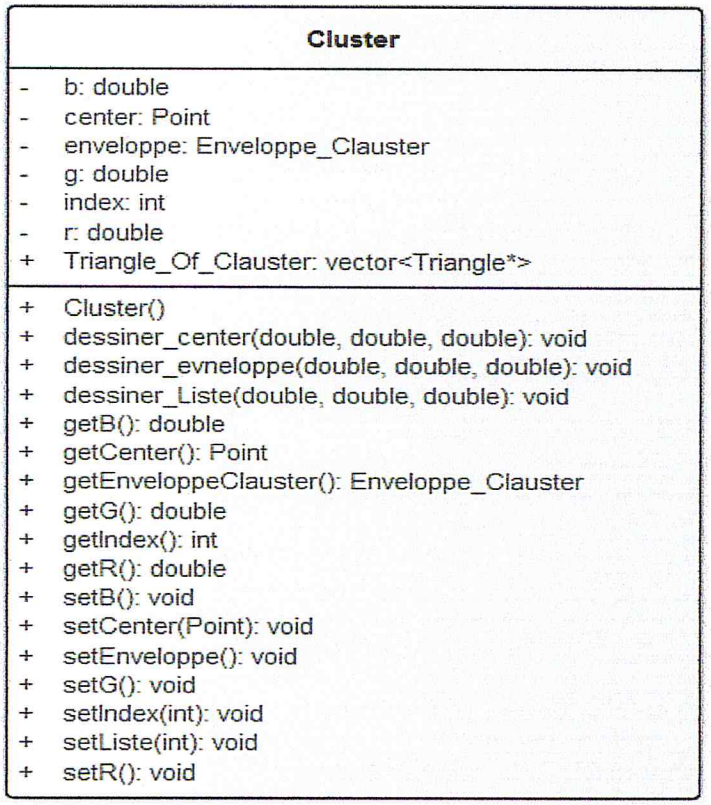


Figure 31: Classe Cluster.

- getListeTriangle () : elle permet de récupérer tous les triangles qui appartient à ce cluster.
- dessiner_center (), dessiner_enveloppe () et dessiner_liste () sont des fonctions qui permettent de dessiner les triangles de cluster, le centroïd et l'enveloppe de cluster.

4. Conclusion

Dans ce chapitre, nous avons définis la conception de notre application en commençant par la présentation de la problématique, ainsi que les différents objectifs visés à atteindre. Ensuite en passant à la présentation des solutions que nous avons proposées avec les différents diagrammes de conception en utilisant le modèle orienté objet et le langage de modélisation UML.

Chapitre 3

Implémentation

1. Introduction :

Après avoir exprimé les objectifs de notre travail ainsi que les solutions proposées en utilisant le modèle orienté objet et le langage de modélisation UML (Unified Modeling Langage), nous allons passer maintenant à la présentation de notre application logicielle développée en utilisant le langage de programmation C++ en utilisant l'EDI (Environnement de Développement Intégré) Builder C++ et la bibliothèque graphique OpenGL (Open Graphics Library) en mettant en évidence son interface, ses fenêtres et les principaux algorithmes que nous avons développés.

2. Présentation de l'application :

Notre Application est un module logiciel graphique et interactif sous Windows permettant la représentation volumique des pièces de n'importe quelles formes géométriques d'une manière précise et rapide.

3. Implémentation :

3.1 Fenêtre principale :

La fenêtre principale est composée de cinq pages (parties). La première partie permet de visualiser le modèle de la pièce en trois modes : points, triangles filaire et triangles rendue. La deuxième partie permet de regrouper les triangles en clusters suite à l'application de l'algorithme de k-means. La troisième partie permet de créer les grilles et les cellules dans les trois plans XY, XZ et YZ. La quatrième partie permet de générer les Dexels dans les trois directions X, Y et Z. La cinquième et dernière partie est une partie d'optimisation pour l choix des pas de discrétisation afin d'aider l'utilisateur à choisir les valeurs optimales des pas. Dans ce qui suit, nous allons présenter les fenêtres de ces parties.

3.2 Structuration et visualisation :

3.2-1. Lecture du fichier STL :

Dans cette partie, nous lisons le fichier de format STL qui contient la forme géométrique de la pièce à usiner sous forme de triangles avec les coordonnées de ses trois sommets.

Une fois le fichier est ouvert, une procédure est lancée qui permet de récupérer et de structurer les points et les triangles dans des listes. Par la suite, sont calculées les coordonnées des points

extrêmes du modèle (coordonnées minimales et maximales dans les directions X, Y et Z (Figure 32).

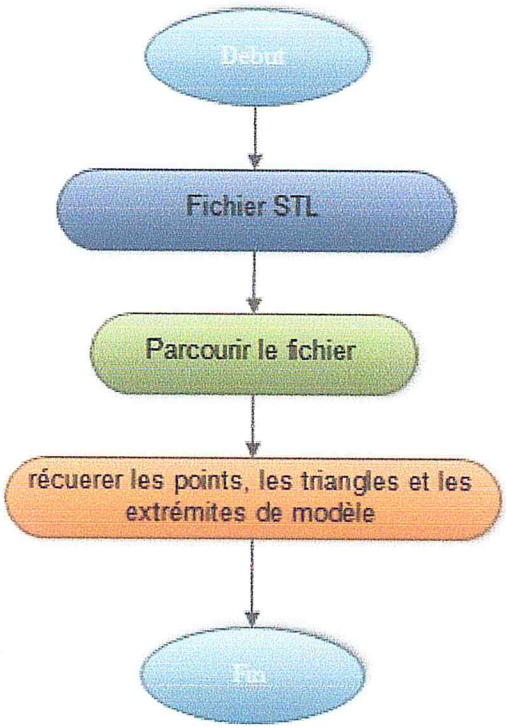


Figure 32: Lecture du fichier STL.

3.2-2. Visualisation du modèle STL :

Cette fenêtre permet de visualiser la pièce en trois modes : points, triangles filaire et triangles rendue. Après la structuration des points et des triangles, l'utilisateur peut visualiser le modèle de la pièce en trois modes selon son choix,

3.3 K-means :

Après lecture du fichier STL, nous devons créer les Dexels correspondant au modèle STL. Afin d'accélérer les calculs de détection des intersections entre les triangles de la surface et les droites des cellules et par conséquent minimiser les temps de calcul, nous avons procédé à l'adoption de la méthode de K-means (Figure 33). La création des clusters de triangles par la méthode K-means passe par :

- Initialisation de K clusters : une fois le nombre de groupes saisi, l'algorithme choisit arbitrairement k points comme centres «initiaux » des k groupes.

- Calculer des distances et groupement : consistent à calculer la distance entre chaque triangle et les k centres. La plus petite distance est retenue pour inclure ce triangle dans le groupe ayant le centre le plus proche.
- Calculer les nouveaux Centroids : pour chaque groupe, l'algorithme calcule le nouveau centre de gravité.

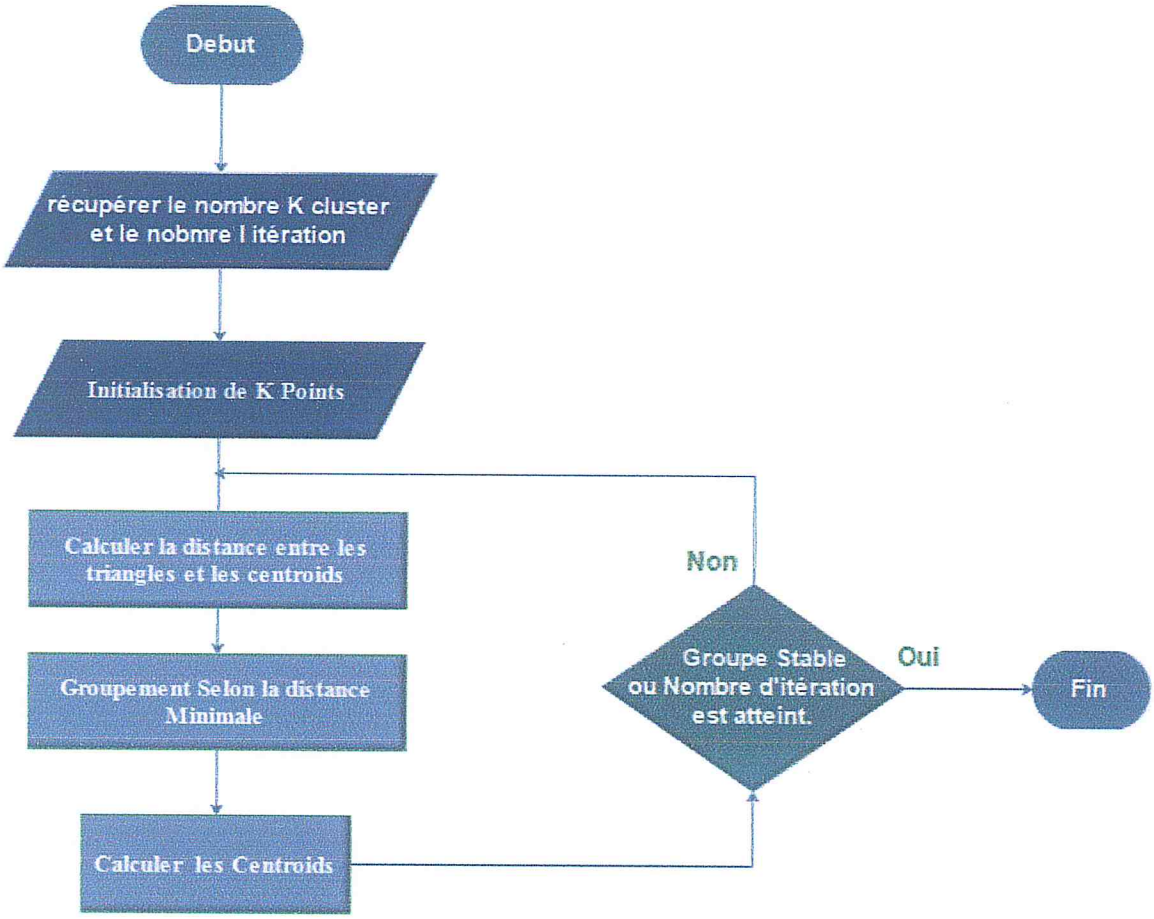


Figure 33 : Groupement des triangles en clusters par la méthode K-means.

3.4 Création des grilles :

Pour modéliser une pièce volumique donnée en Triple Dexels, deux étapes sont nécessaires. La première étape consiste à générer trois grilles : la première dans le plan XY, la deuxième dans le plan XZ et la troisième dans le plan YZ. Tandis que la deuxième étape consiste à créer les Dexels après le calcul des intersections entres les centres des cellules et le modèle STL suivant les trois axes X, Y et Z. Pour créer les Triples Dexels, il est nécessaire de spécifier des pas suivant les trois axes afin de discrétiser le modèle STL.

Une fois l'utilisateur introduit les pas selon les trois axes X, Y et Z, une procédure est lancée pour créer la grille. Par la suite, pour chaque grille nous procédons à la création de l'ensemble des cellules de cette grille. Les cellules de la grille sont comptées selon les pas de l'utilisateur et les limites du modèle de la pièce (Figure 34). Chaque cellule est caractérisée pas ces limites, son centre et sa droite.

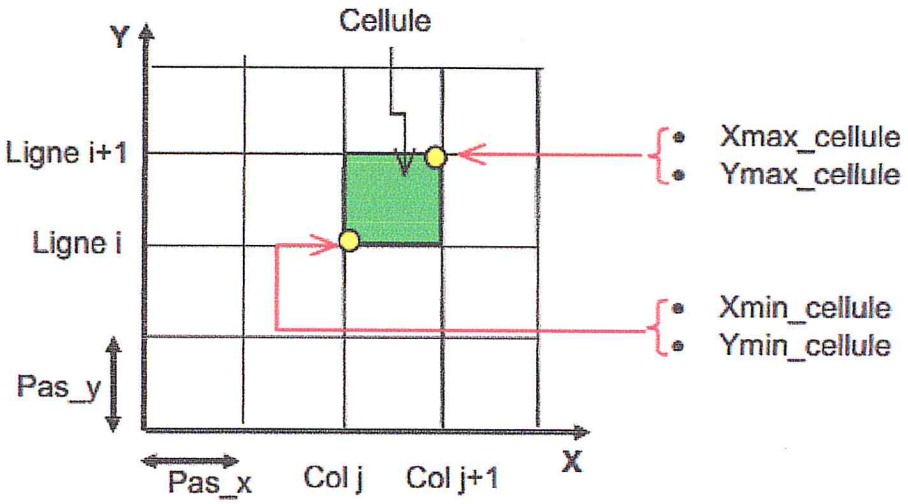


Figure 34 : Création de la grille dans le plan XY.

Procédure crée grille : (exemple grille xy) :

Pour créer la grille, nous avons besoins des pas selon les trois directions X, Y et Z ainsi que les limites maximales et minimales pour chaque direction de la pièce :

- Calculer les limites de la pièce longueur, largeur et hauteur :
  - Langueur = maxX-minX
  - Largeur = maxY-minY
  - Hauteur = maxZ-minZ
- Calculer le nombre de cellules pour chaque direction :
  - Nombre_cellule_X=longueur/pasX
  - Nombre_cellule_Y=largeur/pasY
  - Nombre_cellule_Y=hauteur/pasZ

Après ce calcul, nous pouvons trouver des valeurs non entières or les nombres doivent être entières. Dans ce cas, nous normalisons les valeurs pour obtenir des valeurs entières par la procédure suivante :

- Calcul des écarts écartX, écartY, écartZ par :
  - écartX= Nombre_cellule_X- valeur_entièr(Nombre_cellule_X).
  - écartY= Nombre_cellule_Y- valeur_entièr(Nombre_cellule_Y).
  - écartX= Nombre_cellule_Z- valeur_entièr(Nombre_cellule_Z).

- Si écartX≠0, alors Nombre_cellule_X=valeur_entière(Nombre_cellule_X) +1.
- Si écartY≠0, alors Nombre_cellule_Y=valeur_entière(Nombre_cellule_Y) +1.
- Si écartZ≠0, alors Nombre_cellule_Z=valeur_entière(Nombre_cellule_Z) +1.

Une fois les valeurs sont calculées, nous commençons la création des grilles. Le processus de création de la grille est le suivant (exemple pour le plan XY) :

- ✓ I et J deux variables initialisé à 1.
- ✓ Pour I =1 jusqu'à Nombre_cellule_X.
- ✓ Pour J =1 jusqu'à Nombre_cellule_Y.
- ✓ Nous créons la cellule : pour chaque cellule nous définissons le point minimum, le point maximum, le centre de la cellule et la droite de la cellule :
  - Point_min= (minX+pasX*I, minY+pasY*J, minZ).
  - Point_max=((minX+pasX)+(pasX*I), ((minY+pasY)+(pasY*J), minZ).
  - Centre=(Point_min.getX()+(pasX/2), Point_min.getY()+(pasY/2),minZ).
  - Droite : une droite est définit par deux points :
  - Point1= (centre.getX(), centre.getY(),minZ).
  - Point2= (centre.getX(), centre.getY(),maxZ).

Par la suite, la cellule est ajoutée à la grille. La création des grilles et des cellules dans les plan XY et YZ suit les mêmes étapes où il suffit de faire des rotations des coordonnées.

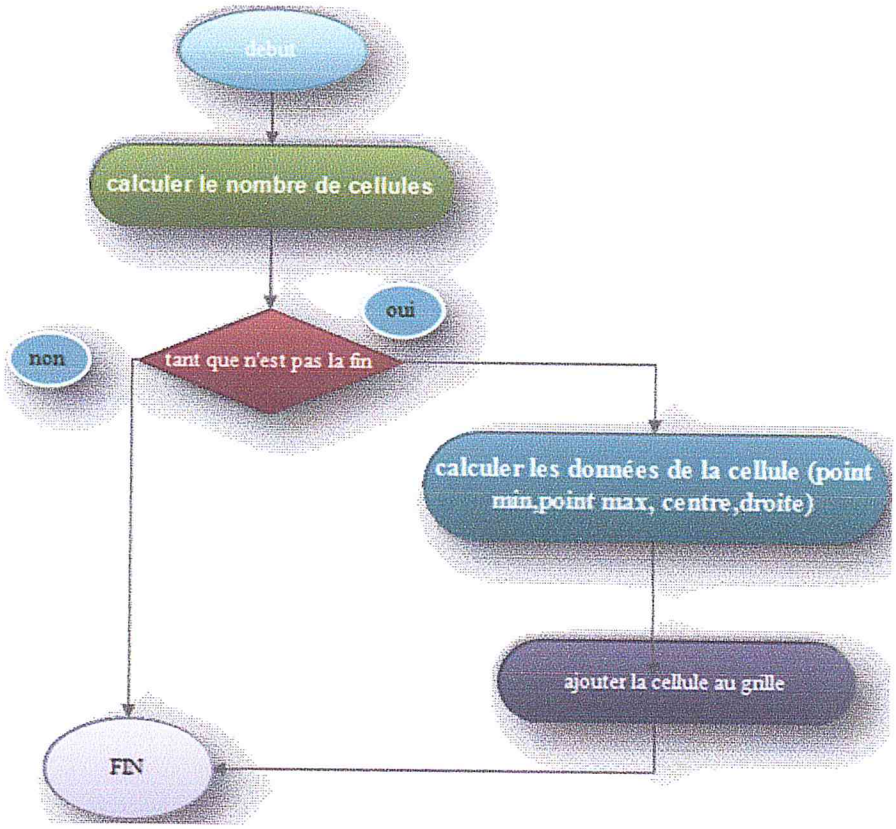


Figure 35 : Organigramme de création de la grille.

3.5. Génération des Dexels dans les trois directions :

3.5.1. Création des Dexels :

Une fois les grilles sont créées, nous passons à la création des Dexels (Figure 36). L'étape de création des Dexels est compliquée puisqu'elle passe par plusieurs procédures afin de calculer les Dexels. Lorsque l'utilisateur clique sur le bouton créer Dexel, la création passe par plusieurs étapes :

- parcourir la grille.
- Pour chaque cellule de la grille :
  - Calculer l'intersection entre le centre de la cellule et le modèle STL.
  - Récupérer les points d'intersection et générer les Dexels (Figure 37).
- A partir de ces points, la liste des Dexels est créée (Figure 38).

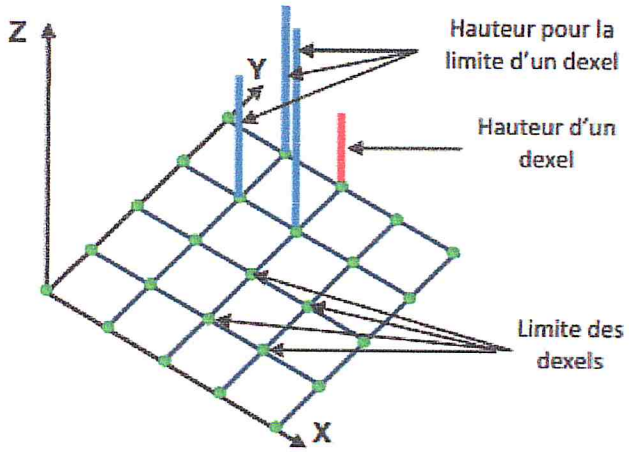


Figure 36: Représentation d'un Dexel.

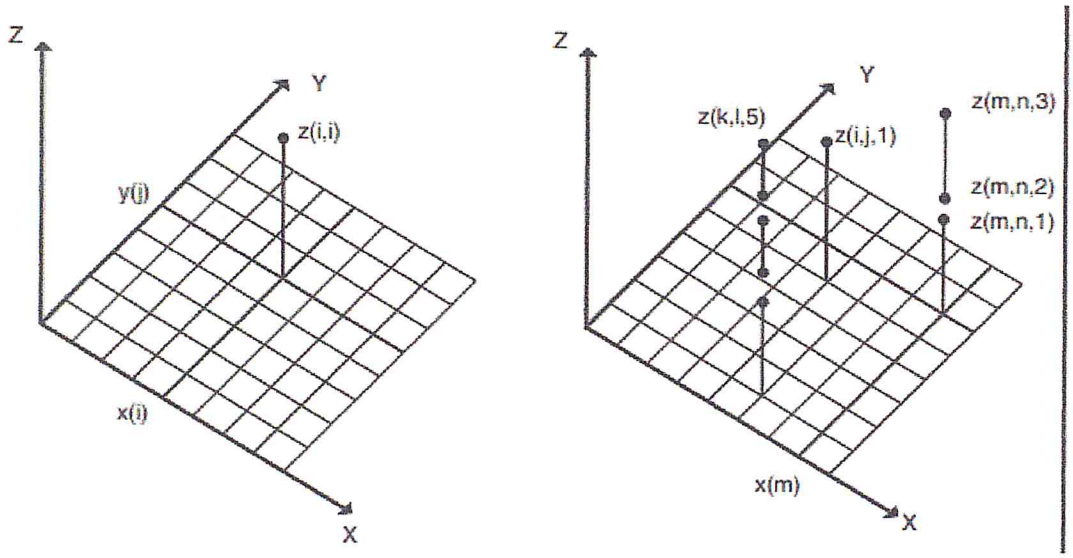


Figure 37: Liste des Dexels de la même droite.

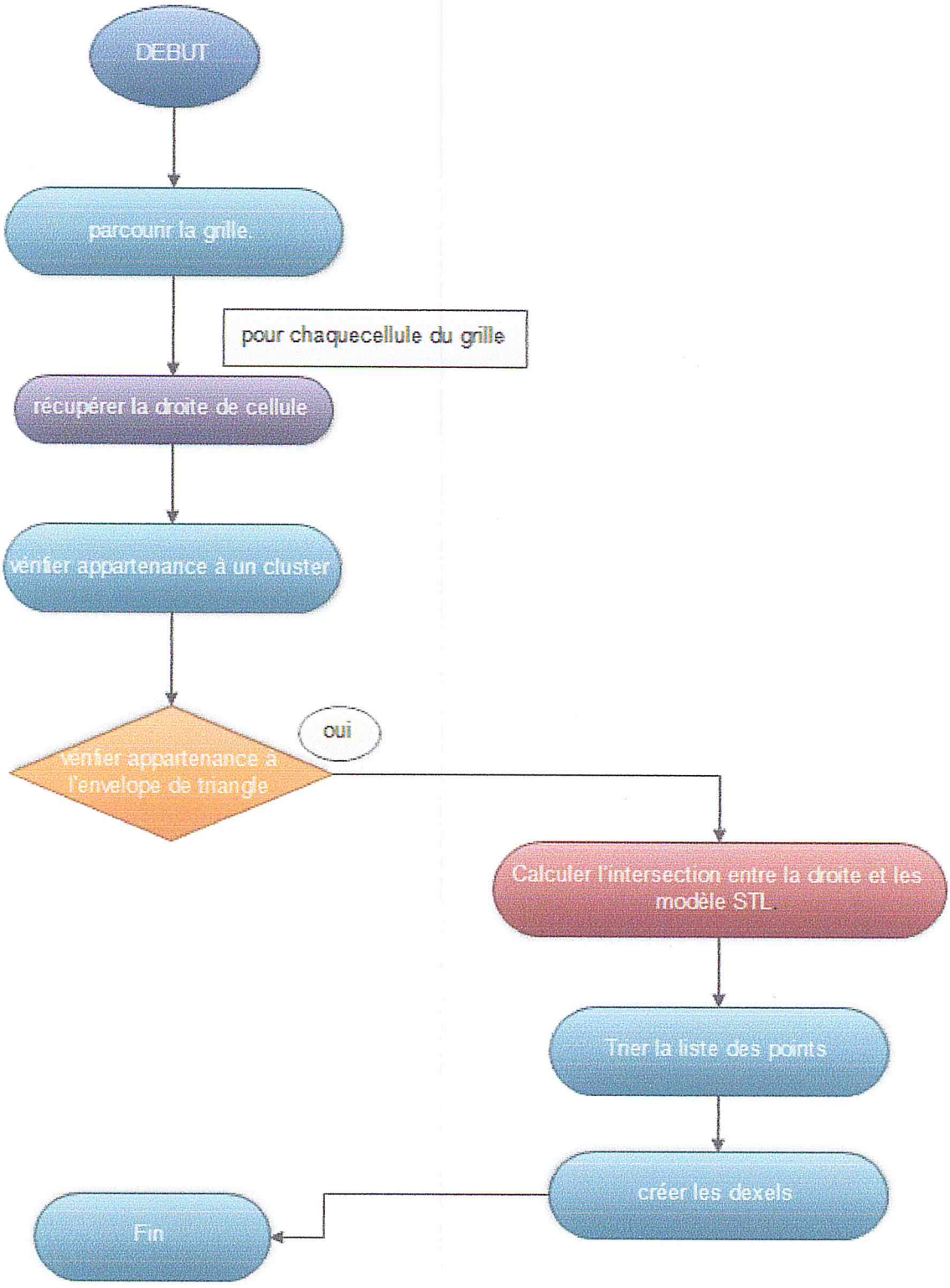


Figure 38: Organigramme de l'algorithme créerDexel.

Lors de la création des Dexels, nous avons besoins de calculer des intersections entre la droite de la cellule et l'ensemble des triangles qui appartiennent à un cluster. Dans notre cas, nous devons calculer plusieurs types d'intersections : intersection droite plan, intersection droite

triangles parallèle et intersection droite segment. Dans ce qui suit nous allons détailler comment calculer ces intersections.

3.5.2. Intersection droite-triangle :

Dans cette partie, nous allons aborder comment déterminer les points d'intersection entre une droite et un triangle. Pour cela, il faut calculer l'intersection entre la droite et le plan du triangle :

- La droite est définie par l'équation :

  X=x1+α (x2- x1).

  Y=y1+α (y2- y1).

  Z=z1+α (z2- y1).

- Le Plan est définit par l'équation : **AX+BY+CZ+D=0.**

- Calculer le vecteur unitaire U de la droite :

  Ux= X2-X1.
  Uy= Y2-Y1.
  Uz= Z2-Z1.

- Calculer la somme des alphas et le reste :

  Somme_alpha= A*Ux+B*Uy+C*Uz.
  Somme_reste= A*X1+B*Y1+C*Z1+D.

- Deux cas sont à distingués :

1er cas : la valeur absolue de Somme_alpha n'est pas nulle : c'est-à-dire que la droite n'est pas parallèle au plan et donc un seul point d'intersection donné par :

alpha=-somme_reste/somme_alpha.
X=X1+alpha*Ux.
Y=Y1+alpha*Uy.

Z=Z1+alpha*Uz.

Le point obtenu est le résultat de l'intersection entre la droite et le plan. Maintenant, il faut vérifier si le point appartient au triangle ou non (Figure 39). Pour vérifier si un point M appartient à un triangle, nous devons vérifier le système d'équations suivant :

$$\begin{cases} (\overrightarrow{p1p2} \wedge \overrightarrow{p1M}) * (\overrightarrow{p1M} \wedge \overrightarrow{p1p3}) \geq 0 \\ (\overrightarrow{p2p1} \wedge \overrightarrow{p2M}) * (\overrightarrow{p2M} \wedge \overrightarrow{p2p3}) \geq 0 \\ (\overrightarrow{p3p1} \wedge \overrightarrow{p3M}) * (\overrightarrow{p3M} \wedge \overrightarrow{p3p2}) \geq 0 \end{cases}$$

Le point M appartient au triangle si les trois conditions du système sont vérifiées.

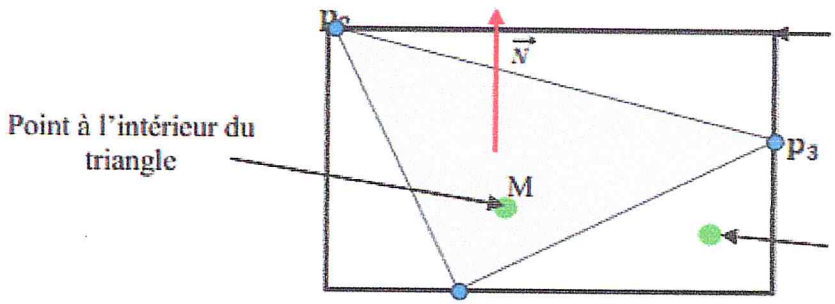


Figure 39: Appartenance point à un triangle.

2ème cas : la valeur absolue de Somme_alpha est égale à 0 : c'est-à-dire que la droite est parallèle au plan. Dans ce cas, soit la droite est dans le plan du triangle, soit la droite et le plan sont disjoints. Pour vérifier si la droite appartient au plan du triangle, il suffit de calculer la distance entre la droite et le plan du triangle :

- Si la distance =0, alors la droite est dans le plan.
- Si la distance≠0, alors la droite et le plan sont disjoints.

La distance entre un point de coordonnées X, Y et Z de la droite et le plan est donnée par :

$$\textbf{Distance} = (A*X+B*Y+C*Z+D) / \sqrt{A^2 + B^2 + C^2}$$

Dans le cas où la distance est nulle, alors plusieurs cas sont à considérer (Figure 40) :

- La droite passe par un des sommets du triangle.
- La droite passe par un des segments du triangle.

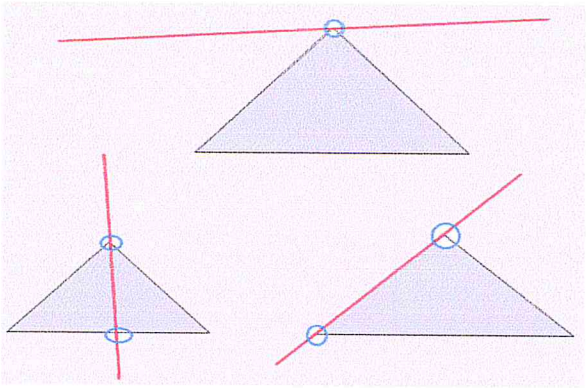


Figure 40: Différents cas d'intersections.

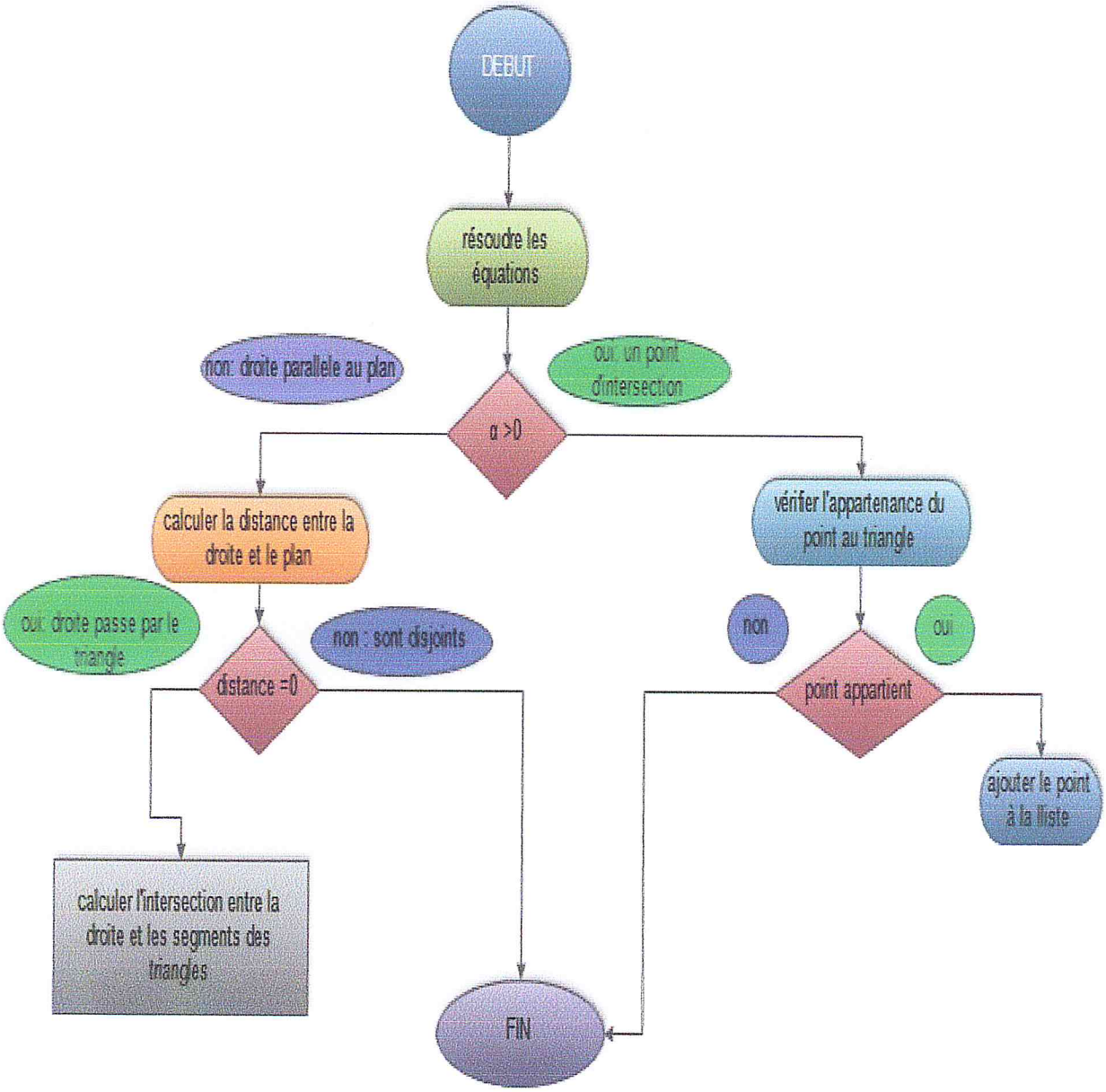


Figure 41 : Intersection droite plan.

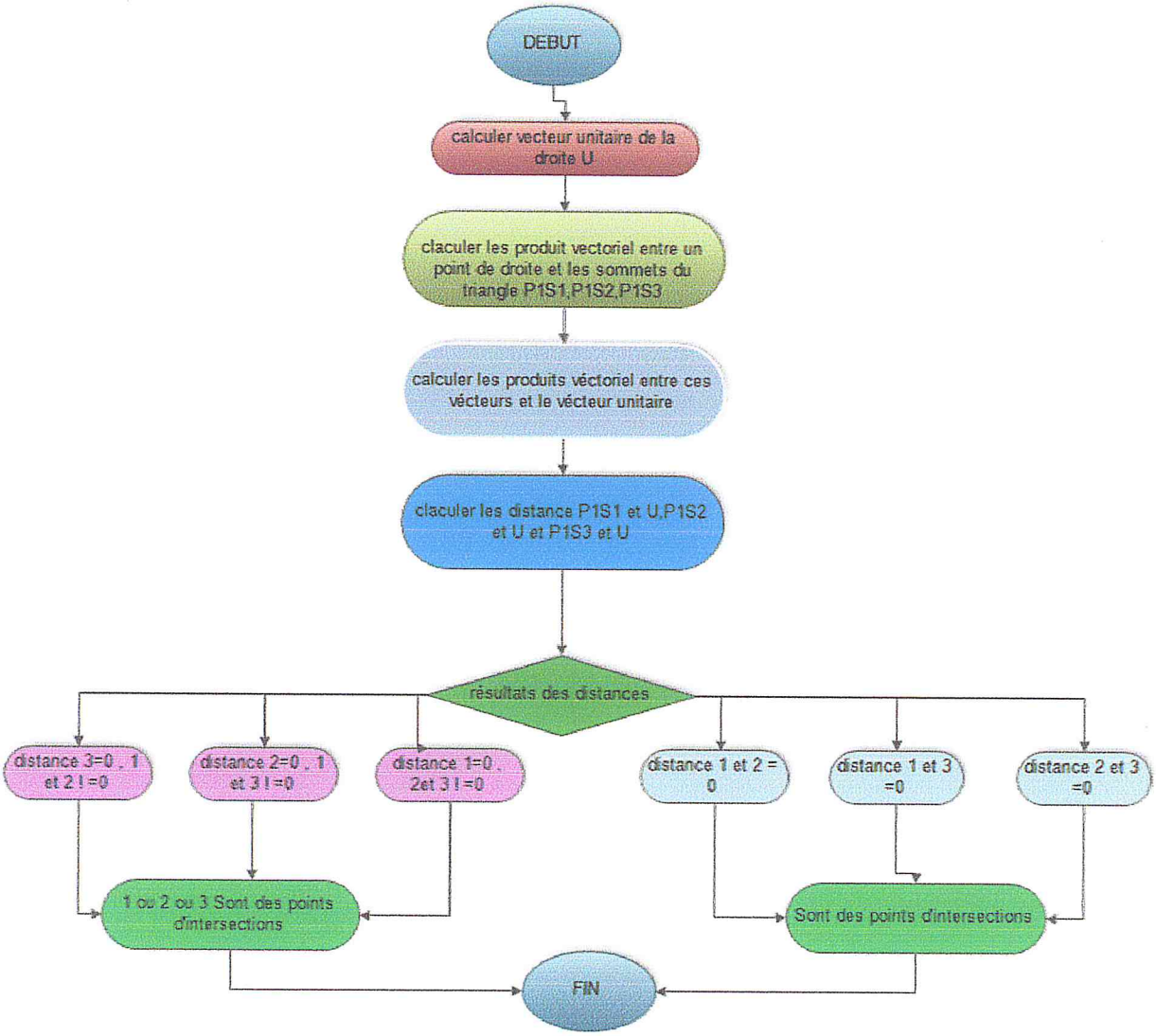


Figure 42 : Intersection droite plan parallèle.

3.5.3. Intersection droite segment :

Le dernier cas est quand la droite passe par un des sommets du triangle. Dans cette partie nous calculons l'intersection entre une droite et un segment du triangle.

La droite est définit par deux points et une équation qui est :

D1 : $X=X1+\alpha\ (X2-X1)$.

Y=Y1+α (Y2-Y1).

Z=Z1+α (Z2-Z1).

Le segment forme une droite :

D2 : X=Xs1+β (Xs2-Xs1).

Y=Ys1+ β (Ys2-Ys1).

Z=Zs1+ β (Zs2-Zs1).

En mettant l'égalité suivante :

α (X2-X1) - β (Xs2-Xs1) = (Xs2- X1).

α (Y2-Y1) - β (Ys2-Ys1) = (Ys2- Y1).

Ce système d'équations est résolu par la méthode de Cramer.

Déterminant maintenant la matrice A :

$$A=\begin{pmatrix} X2-X1 & -Xs2+Xs1 \\ Y2-Y1 & -Ys2+Ys1 \end{pmatrix}.$$

- Calculant maintenant le déterminant det(A).
- Si det(A)>0 :
  - Calcule de α et β par :

    α= (1/det_A)*det_alpha

    β = (1/det_A)*det_beta;

  - Remplacer α dans l'équation de la droite pour déterminer les coordonnées du point d'intersection.
- Si det(A)=0 rien à faire.

3.5.4. Initialiser Liste Dexels :

Une fois le calcul des intersections est terminé, nous obtenons une liste qui contient des points d'intersection. Cette liste est triée et par la suite un ensemble de procédures sont lancées pour créer les Dexels.

La première procédure appliquée après le tri est l'initialisation (Figure 43). Dans cette procédure :

- Nous parcourons la liste des points d'intersection.
- Nous prenons les points deux à deux.
- Initialiser chaque segment créé « ne contient pas de la matière ».

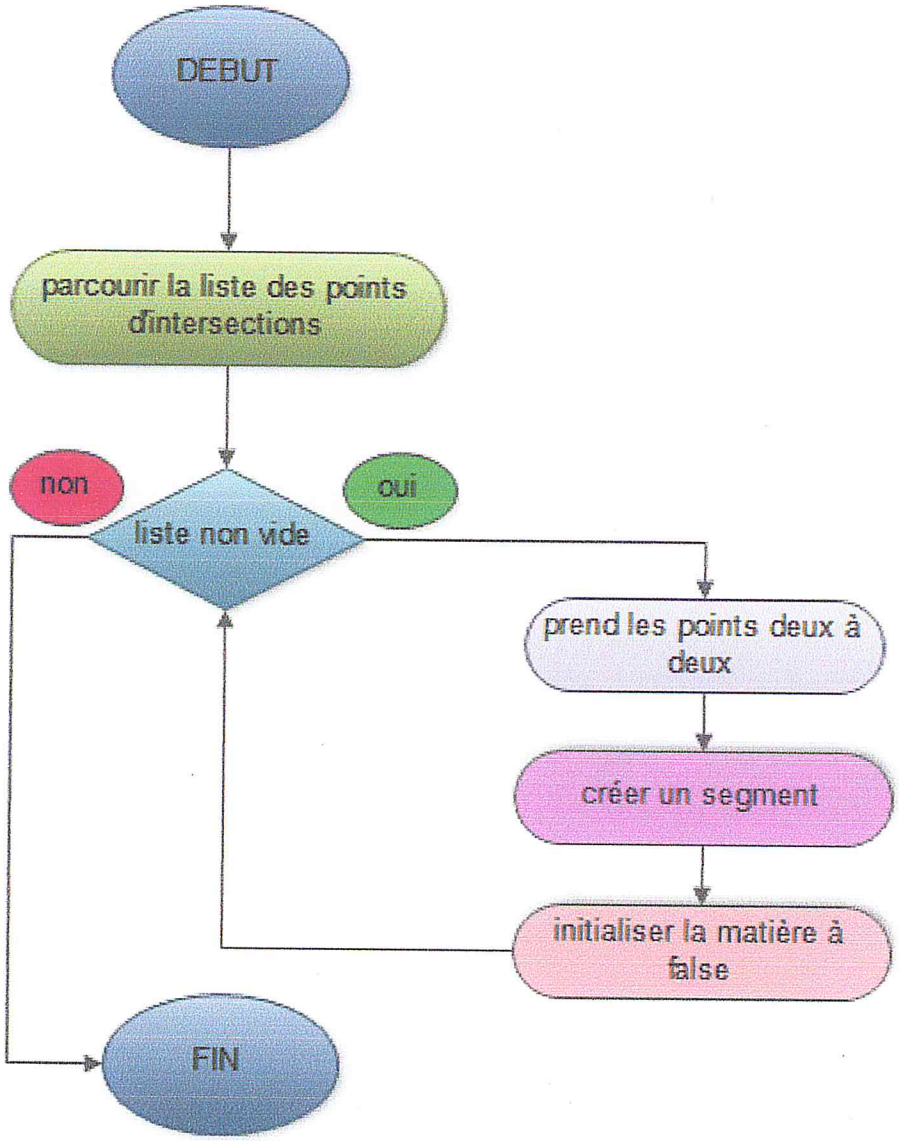


Figure 43 : Initialiser liste Dexels.

3.5.5. Vérification :

Après la fin de l'étape d'initialisation, nous passons à l'étape de vérification qui permet de vérifier si les segments passent par un des triangles ou non (Figure 44).

3.5.6. Créer Liste Dexels :

Cette procédure permet de créer la liste finale des Dexels (Figure 45). Pour cela, la procédure suivante est utilisée :

- Parcourir la liste des Dexels.
  - Si le Dexel ne contient pas de la matière :
    - Calculer le produit scalaire entre la normale du triangle auquel les points du segment appartiennent et le vecteur unitaire de la droite :
      - Si le produit scalaire1<=0 et produit sclaire2>=0, alors ce Dexel contient de la matière.

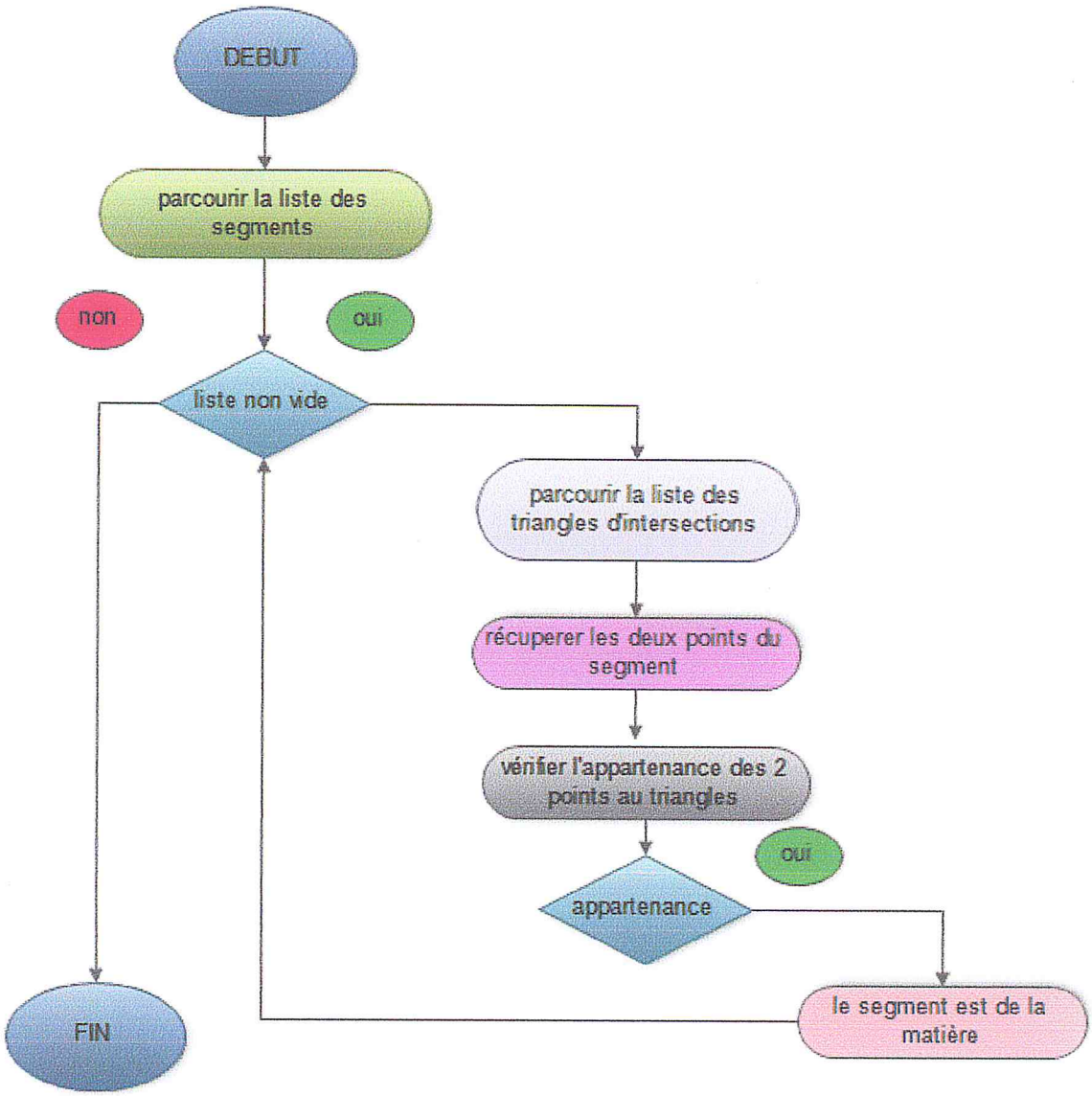


Figure 44: Procédure de vérification.

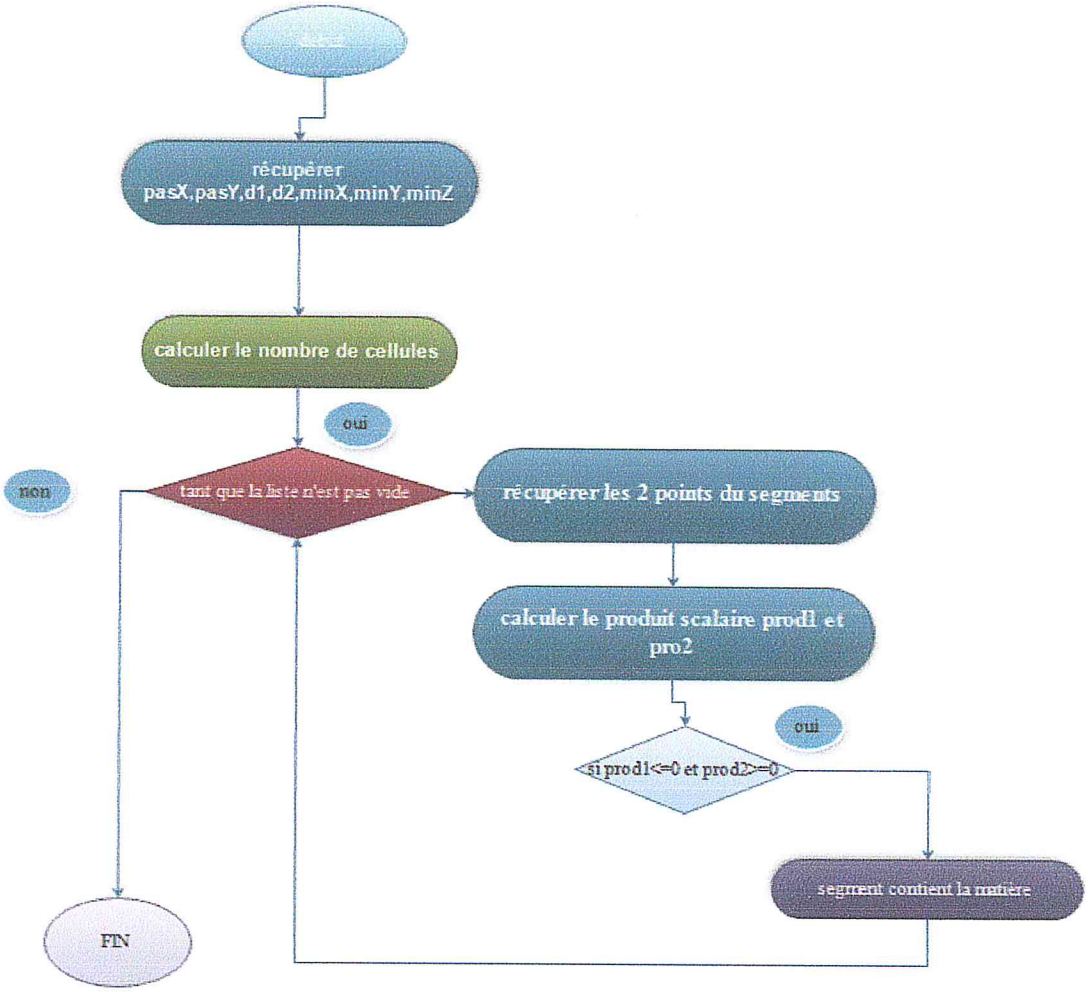


Figure 45 : Créer Dexels finaux.

Une fois le processus de création des Dexels, nous obtenons notre modèle en triple Dexels (Figure 46). Triple Dexel est une extension de la modélisation simple Dexel.

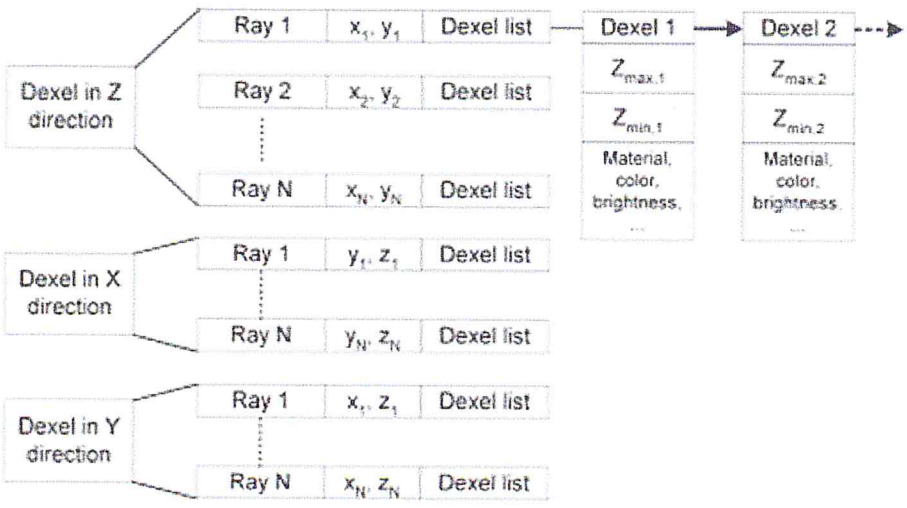


Figure 46 : Structure de données de Triple Dexel.

3.6 Optimisation :

Dans cette partie, l'utilisateur introduit des pas initiaux suivant les directions X, Y et Z ainsi qu'une valeur d'erreur (% pourcentage). Une fois l'utilisateur saisit les valeurs et clique sur « pas optimums », une procédure est lancée qui calcule les pas optimums pour faire les calculs nécessaires pour la simulation. Le déroulement de cette procédure est représenté par la Figure 47.

- L'utilisateur introduit les pas initiaux et le pourcentage de l'erreur.
- Calculer la grille pour le plan.
- Calculer les Dexels.
- Quantifier le volume de la matière.
- Comparer les pas : toujours nous divisons le plus grand pas par 2.
- Calculer le volume avec les nouveaux pas.
- Appliquer une formule pour déterminer l'écart entre l'ancien et le nouveau volume.
- Répéter la procédure tant que la condition n'est pas vérifiée.

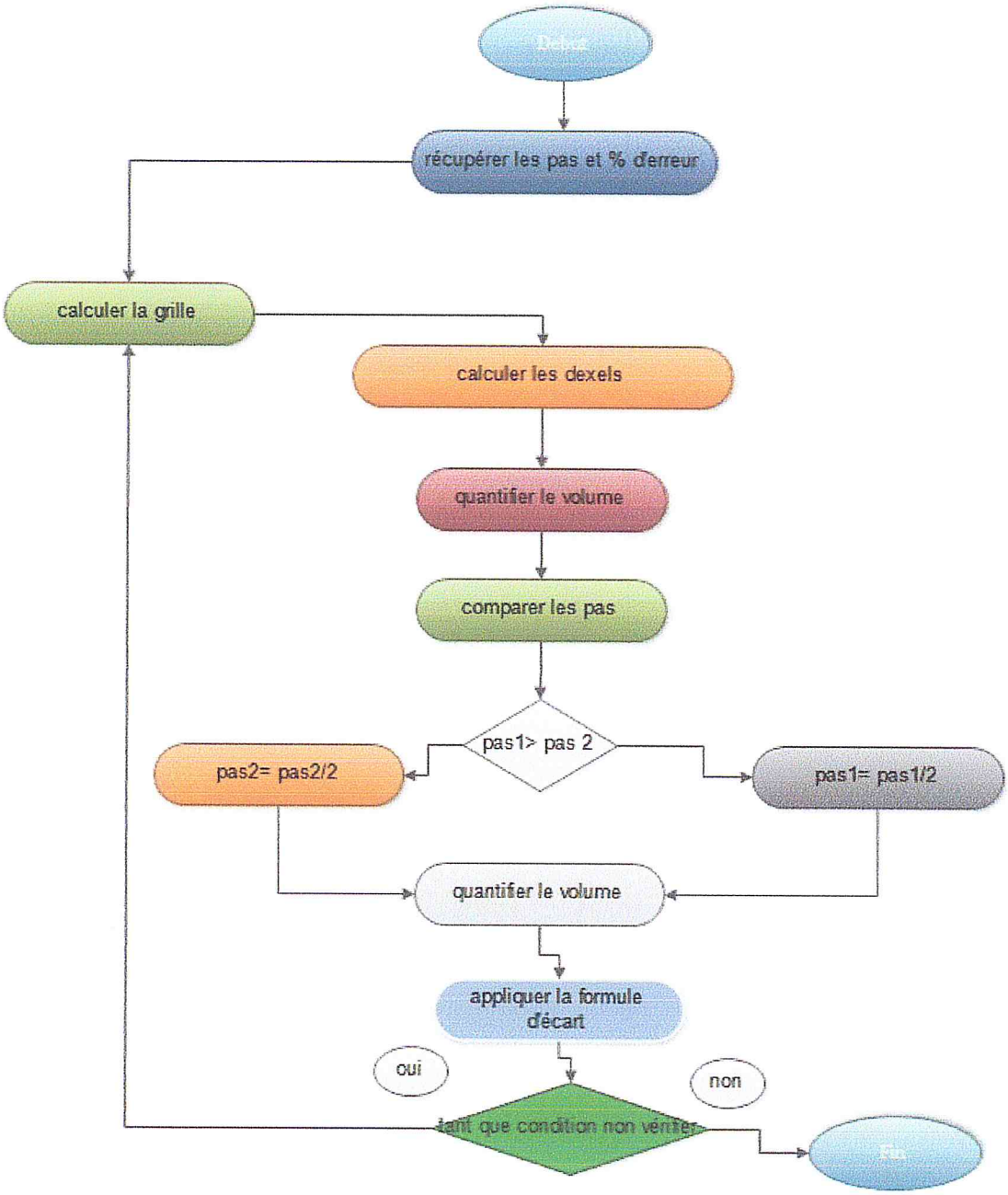


Figure 47 : Procédure d'optimisation.

4. Conclusion:

Dans ce chapitre, nous avons donné les détails de la partie implémentation avec les solutions proposées, les algorithmes et les procédures développés.

Le test et la validation de notre application sur des surfaces réelles seront considérés dans le chapitre suivant.

Chapitre 4

Tests Et Validations

1. Introduction :

Après les étapes de conception, de proposition des solutions et d'implémentation informatique et d'intégration des différents modules que nous avons développés, nous passons maintenant à l'étape des tests et des validations de notre travail. Afin de valider notre application un ensemble de tests ont été préparés pour chaque fonctionnalité de l'application. Nous commençons par le choix de la forme sur laquelle nous allons faire les tests, ensuite pour chaque tâche nous allons visualiser le résultat obtenu après l'exécution de la commande sélectionnée par l'utilisateur. L'affichage est géré par la bibliothèque OpenGL. L'utilisateur peut aussi modifier les couleurs des objets affichés pour une bonne visualisation.

Une grande partie de cette application est consacrée à la visualisation. Donc, l'utilisateur a la possibilité de vérifier même les petits détails avec le zoom et les différentes possibilités de rotations de la scène affichée. Comme il peut aussi faire des projections sur des combinaisons d'axes choisis afin de choisir l'angle idéal de visualisation.

2. Présentation de l'interface graphique de l'application :

Les tests et les validations sont effectués sur trois pièces représentées par les Figures 48, 49 et 50. Ces pièces conçues dans un logiciel spécialisé de CAO où elles sont composées de surfaces de formes très complexes. La première pièce est une roue à ailettes utilisées dans les compresseurs tandis que les deux autres pièces contiennent plusieurs évidements de géométries très complexes. Dans ce même logiciel, les pièces sont approximées par un ensemble de triangles. Ces triangles sont stockés dans des fichiers STL (un pour chaque pièce) qui seront utilisés comme des entrées pour le logiciel.

Pour les tests de validation, nous allons dérouler l'algorithme étape par étape en visualisant les résultats de chaque étape.

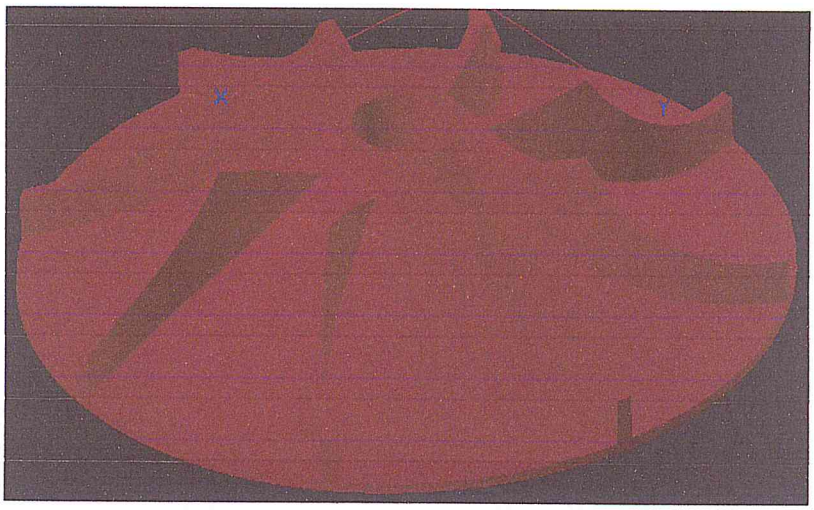


Figure 48 : Modèle de la pièce 1 (roue à ailettes).

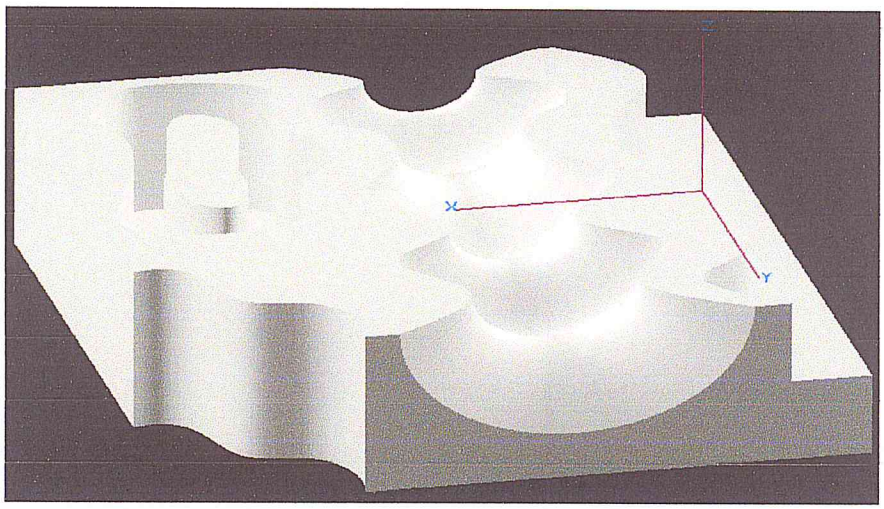


Figure 49 : Modèle de la pièce 2.

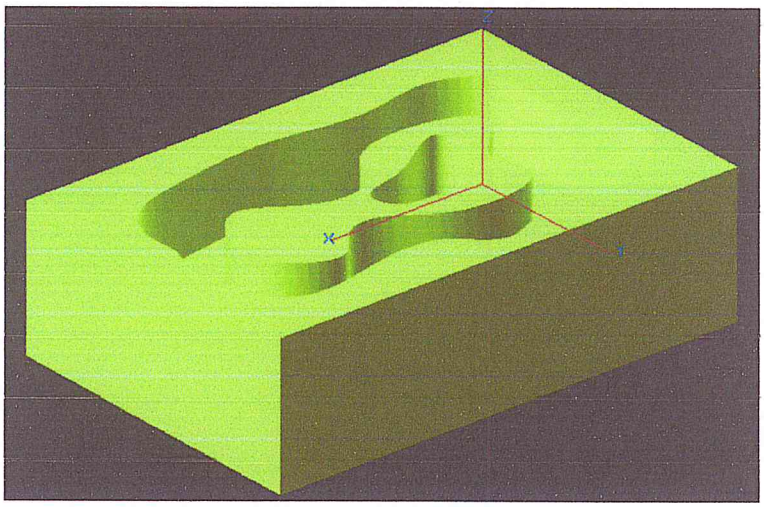


Figure 50 : Modèle de la pièce 3.

2.1. Lecture du fichier STL :

Dans l'onglet lecture fichier STL, l'utilisateur doit cliquer sur le bouton lire fichier STL pour pouvoir sélectionner un fichier STL à ouvrir (Figure 51).

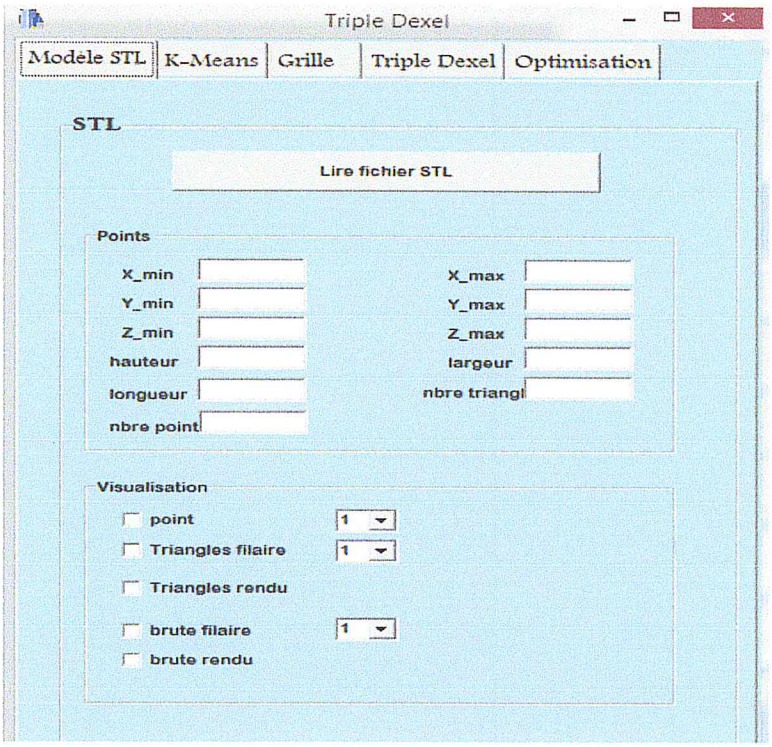


Figure 51 : Onglet lecture fichier STL de l'interface.

Après l'ouverture du fichier, nous pouvons :

- Récupérer les paramètres de la forme brute.
- Visualiser les sommets du modèle (Figure 52).
- Visualiser les triangles du modèle en deux modes filaire et rendu (Figure 53).

Pour le deuxième modèle de test (Figure 49), le nombre total des sommets et le nombre total des triangles sont égaux respectivement à 52137 et 104266.

- Les paramètres de la pièce brute sont :

Xmin = 0.393 Ymin = 0.393 Zmin = 3.6747

Xmax =130.139 Ymax = 130.139 Zmax =33.6747.

Longueur = 30mm Largeur =129.8mm Hauteur =129.8mm

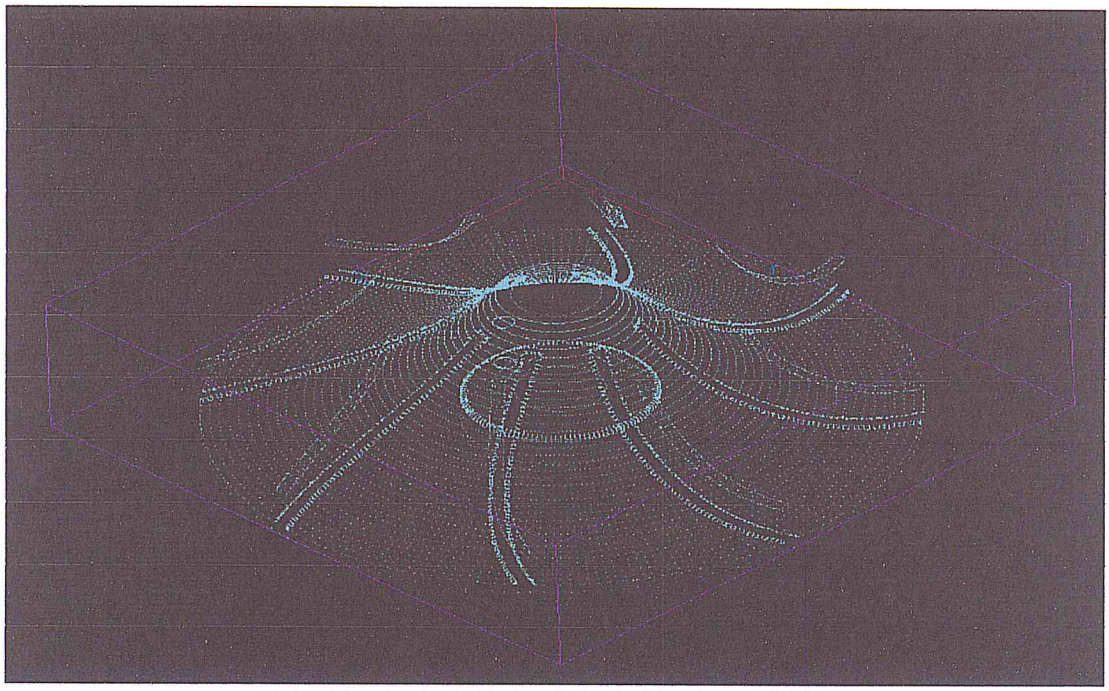
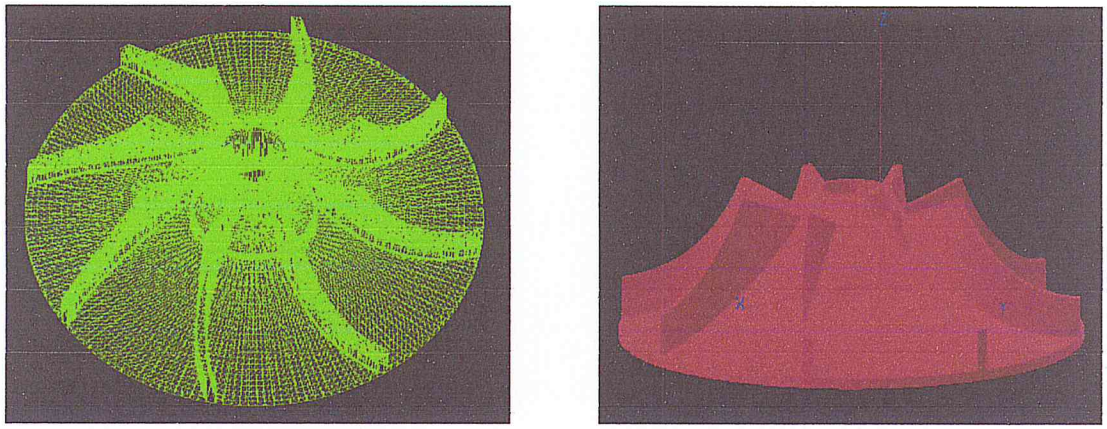


Figure 52: Sommets du modèle STL de la pièce 1.



a. Triangles filaires *b. Triangles rendus.*

Figure 53 : Triangles du modèle STL de la pièce 1.

2.2. Regroupement des triangles en clusters :

Afin d'optimiser et de réduire le temps de calcul, nous avons utilisé l'algorithme de k-means qui sert à regrouper les triangles dans des clusters. Pour cela, l'utilisateur doit choisir le nombre de clusters et le nombre d'itérations (Figure 54).

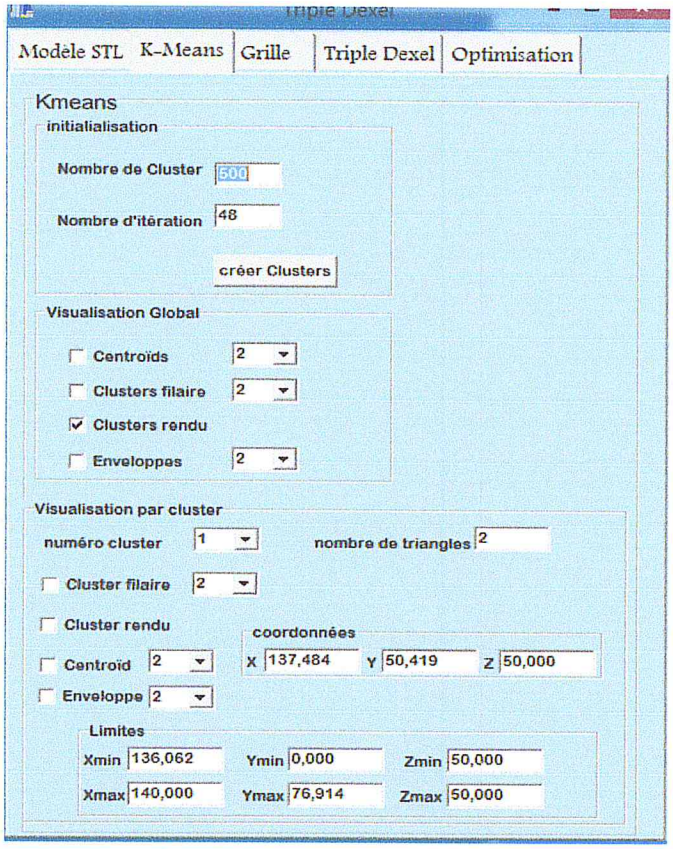


Figure 54 : Onglet de k-means.

Avant tous, nous avons utilisé le modèle de la deuxième pièce pour appliquer l'algorithme de calcul des intersections (chapitre 3, Page 53) pour créer les Dexels sans utiliser le K-means comme nous avons détaillé dans le chapitre précédent où nous avons calculé le temps d'exécution. Pour le test de k-means, nous avons opté pour 500 clusters et 80 itérations. L'algorithme peut s'arrêter avant que le nombre d'itérations est atteint si deux itérations successives donnent les mêmes partitions. Dans notre cas, le processus de création des clusters s'est arrêté à l'itération 44. Nous avons également calculé le temps d'exécution. Les temps de calcul des intersections pour les deux cas sont :

- Sans K-means : 135,727 s
- Avec K-means : 1,394 s

Nous remarquons qu'il y a un très grand écart entre les deux temps (un rapport de plus de 97). Plus le nombre de triangles augmente ou le pas de discrétisation diminue, plus cet écart augmente. D'où l'utilité de regrouper les triangles dans des clusters.
Nous pouvons ensuite choisir le mode de visualisation :

- Visualisation globale des clusters, des centroïds et des enveloppes (Figure 55).

- Visualisation partielle où nous pouvons visualiser chaque cluster, son centroïd et son enveloppe (Figure 56).

On peut également afficher les paramètres de chaque cluster : nombre de triangles, l'enveloppe du cluster dans l'espace ainsi que les coordonnées de son centroïd.

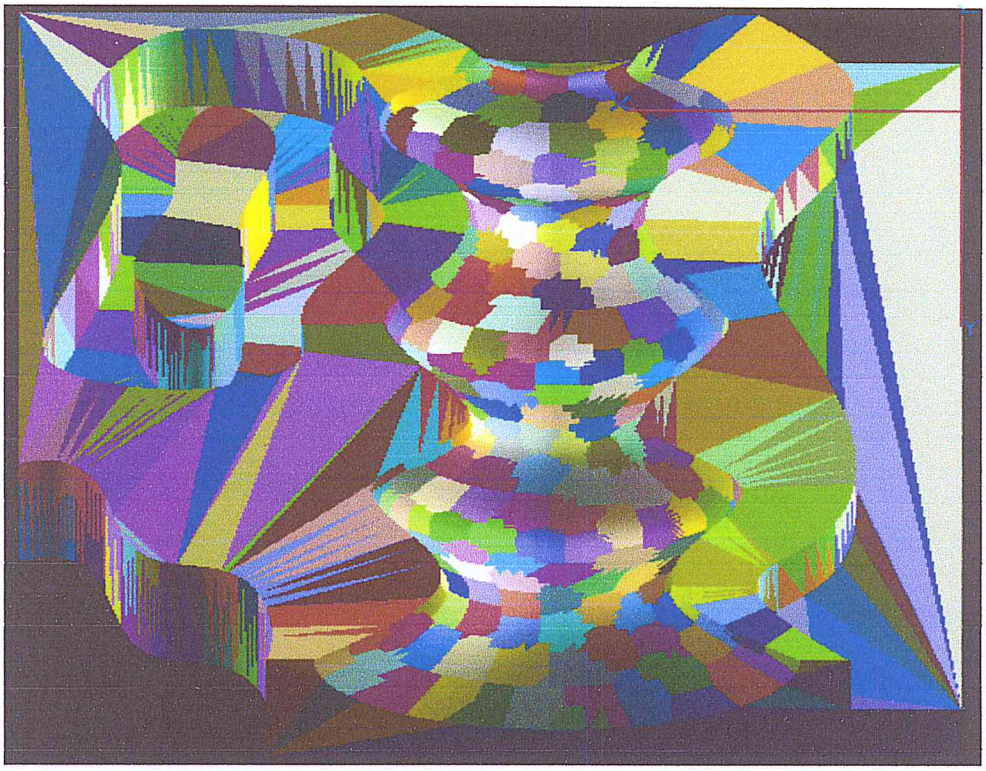


Figure 55 : Visualisation globale des clusters.

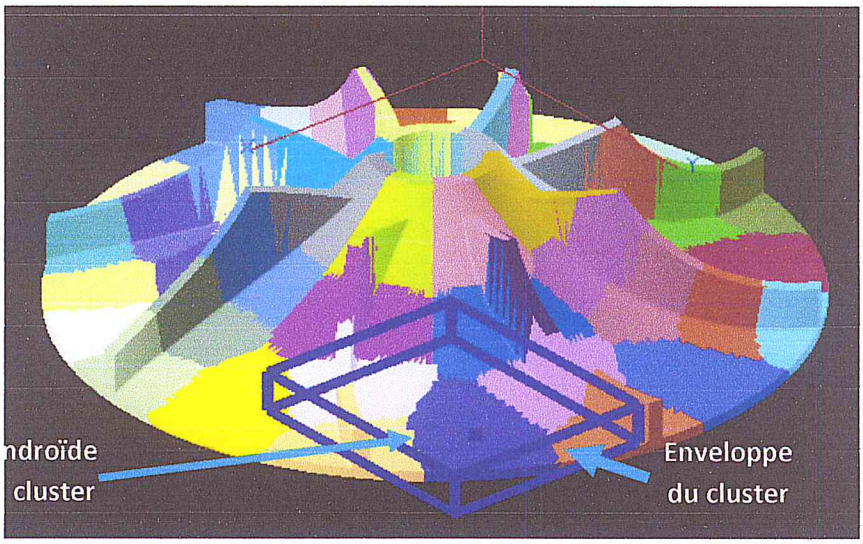


Figure 56 : Visualisation partielle du cluster, de l'enveloppe et du cendroïd.

2.3. Création des grilles :

Pour la création des grilles, nous avons choisi juste pour le test un pas de 5 suivant les trois axes X, Y et Z (Figure 57).

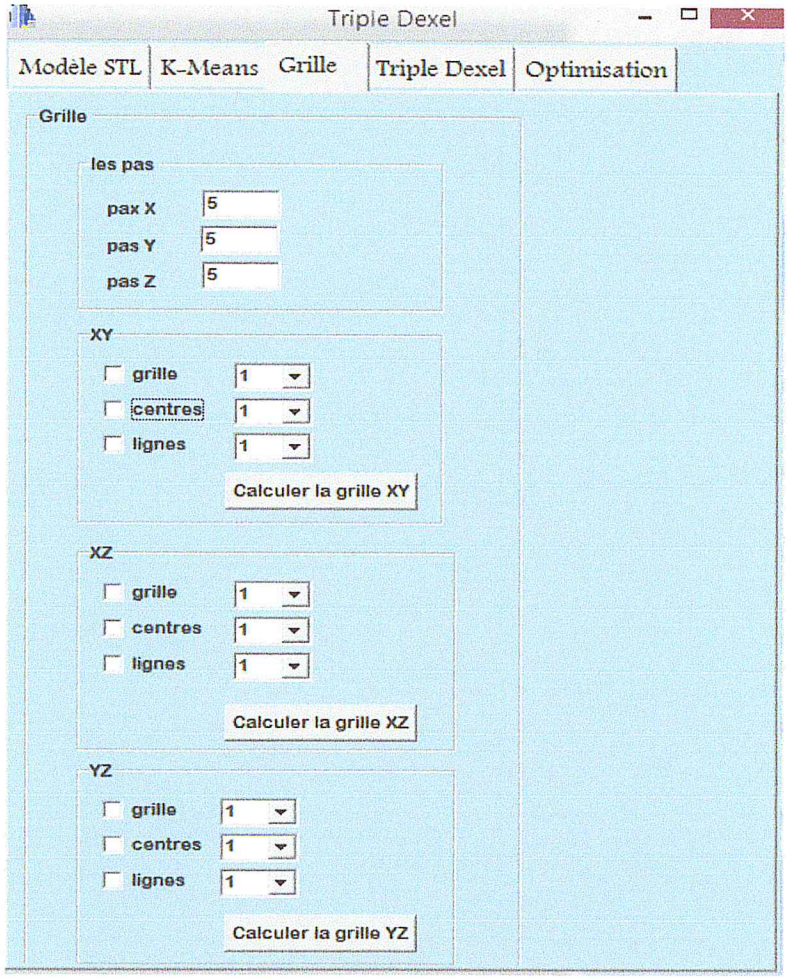


Figure 57 : Onglet de création des grilles.

Nous avons choisi le modèle de la pièce 1 pour visualiser le résultat. L'exécution de cette fonction nous permet de :

- Visualiser les grilles dans les trois plans (Figure 58).
- Visualiser les centres des cellules (Figure 59).
- Visualiser les droites des cellules (Figure 60).

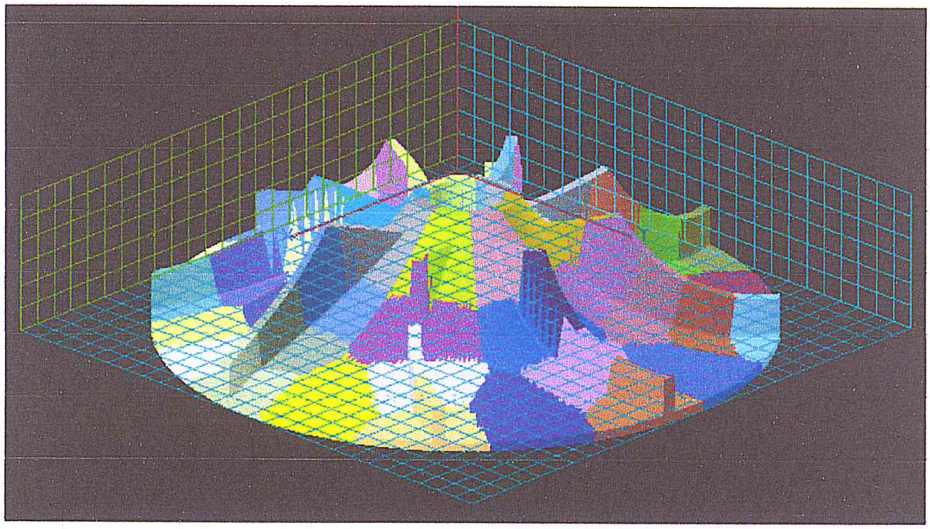


Figure 58 : Visualisation des grilles dans les trois plans.

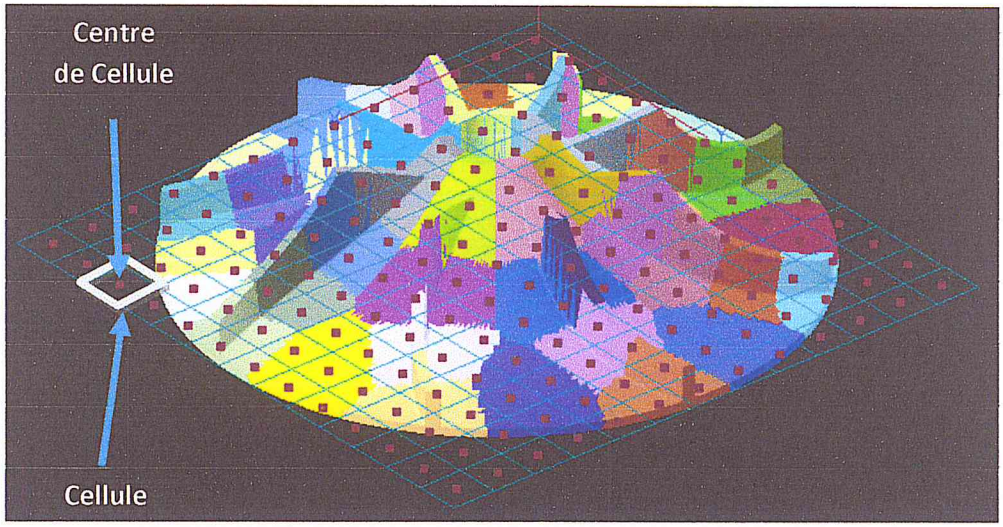
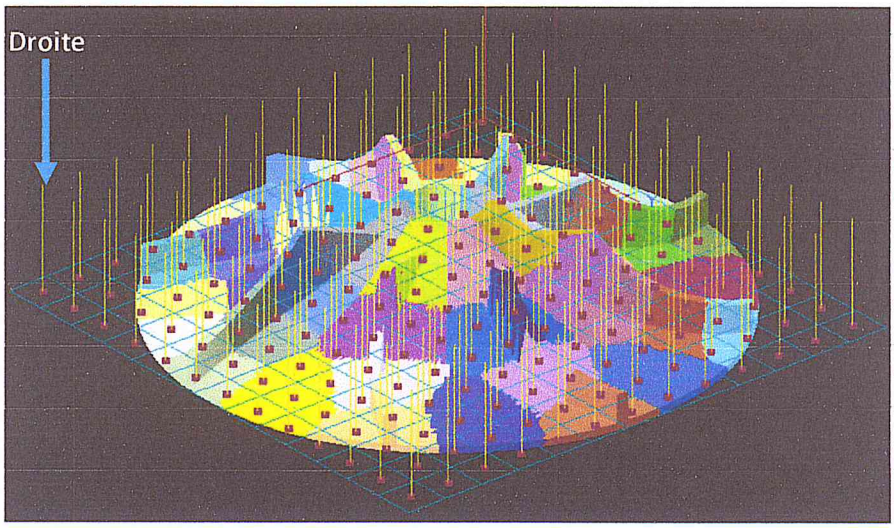
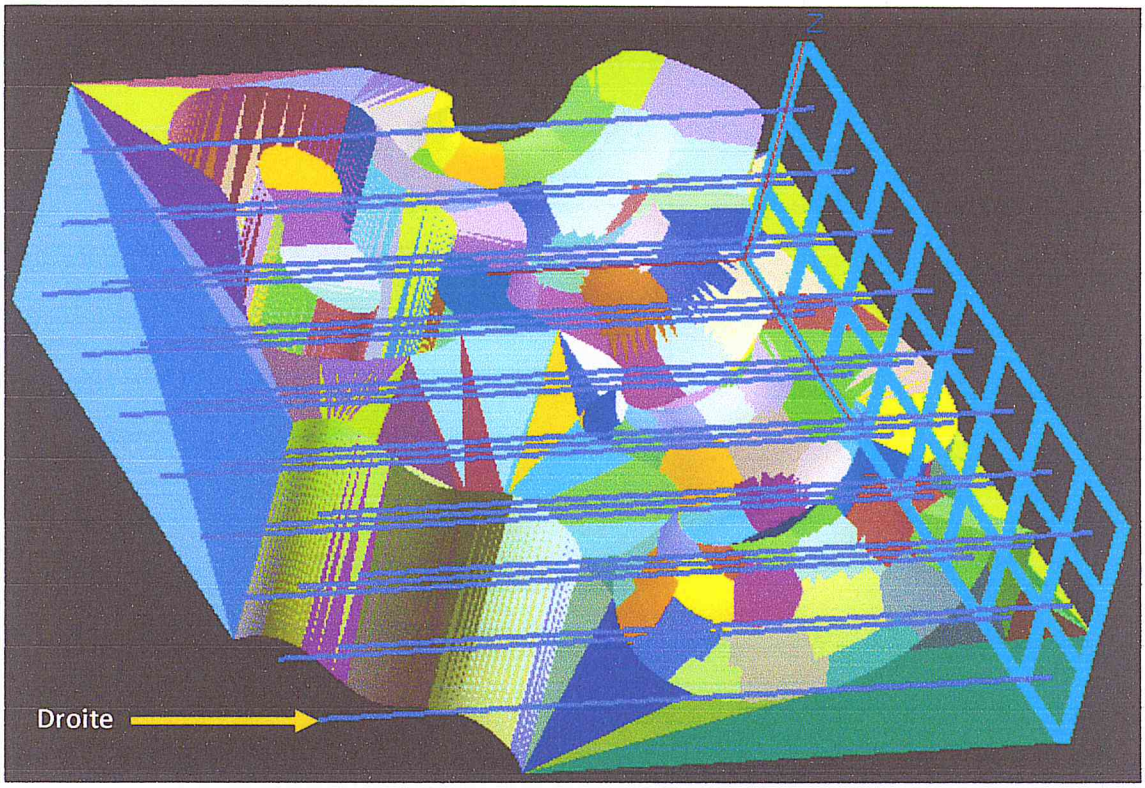


Figure 59 : Visualisation des centres des cellules sur le plan XY.



a. Droites suivant l'axe Z pour le modèle de la pièce 1.



b. Droites suivant l'axe X pour le modèle de la pièce 2.

Figure 60 : Visualisation des droites des cellules dans différentes directions.

2.4. Création des Triple-Dexels :

Après la création des droites des grilles sur les trois plans, nous allons calculer les points d'intersection entre ces droites avec les triangles des clusters.

Pour le test, nous avons fixé les pas suivant les X et Y égaux à 1 pour le modèle de la pièce 1. Pour le modèle de la pièce 3, les pas suivant les X et Z sont égaux à 1. L'exécution de la fonction créer Dexels sur un plan choisi (Figure 61), permet de créer les Dexel à partir des points d'intersection. Ensuite, nous pouvons :

- Visualiser les points d'intersection (Figure 62).
- Visualiser les Dexels (Figure 63).
- Visualiser le volume du modèle STL selon les trois plans (Figure 64, 65 et 66).
- Visualiser les Triple-Dexels (Figure 67).

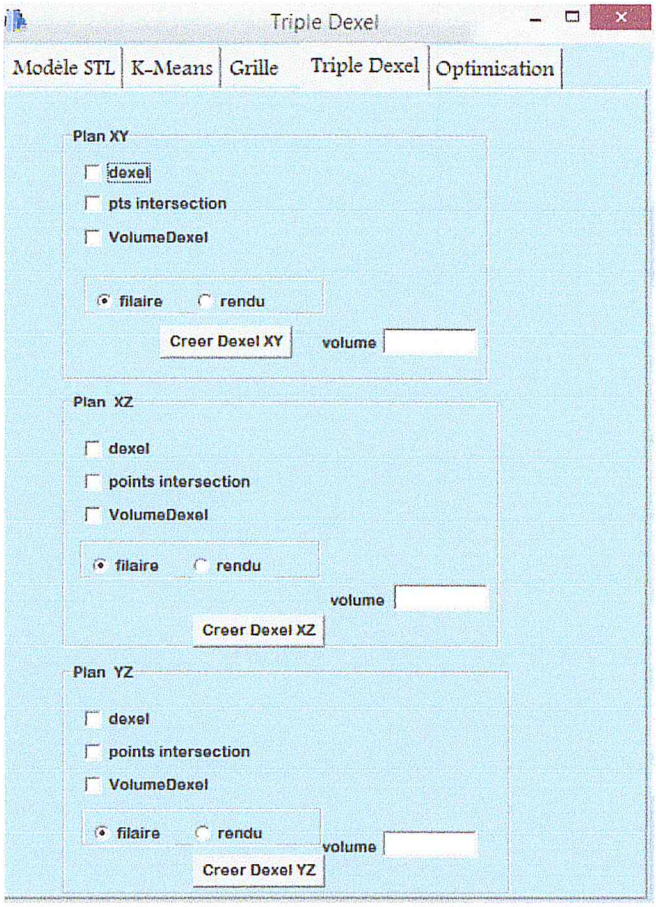


Figure 61: Onglet Création du Triple Dexels.

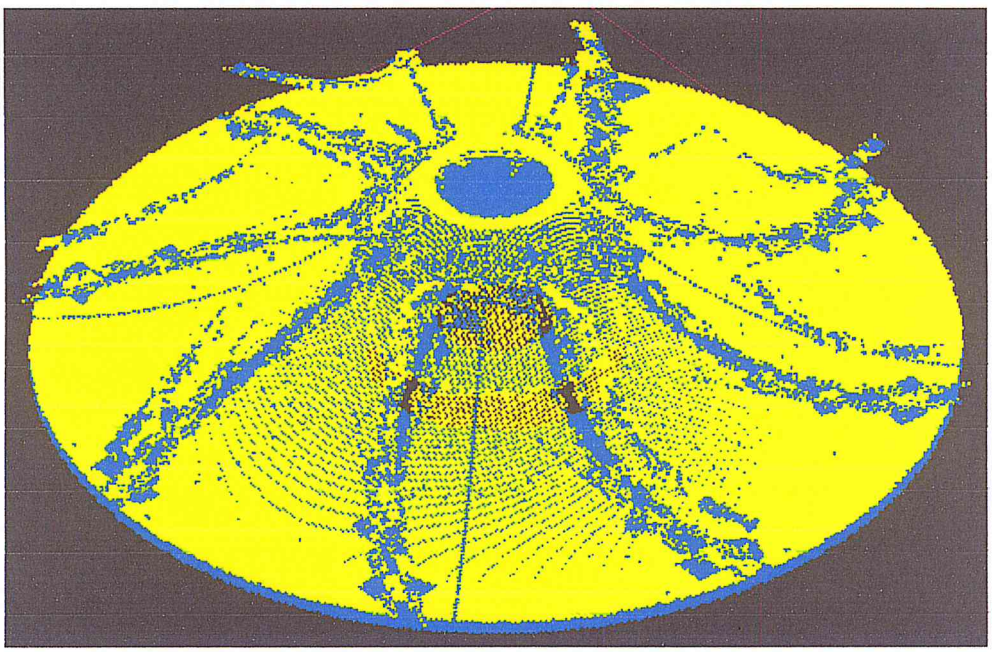
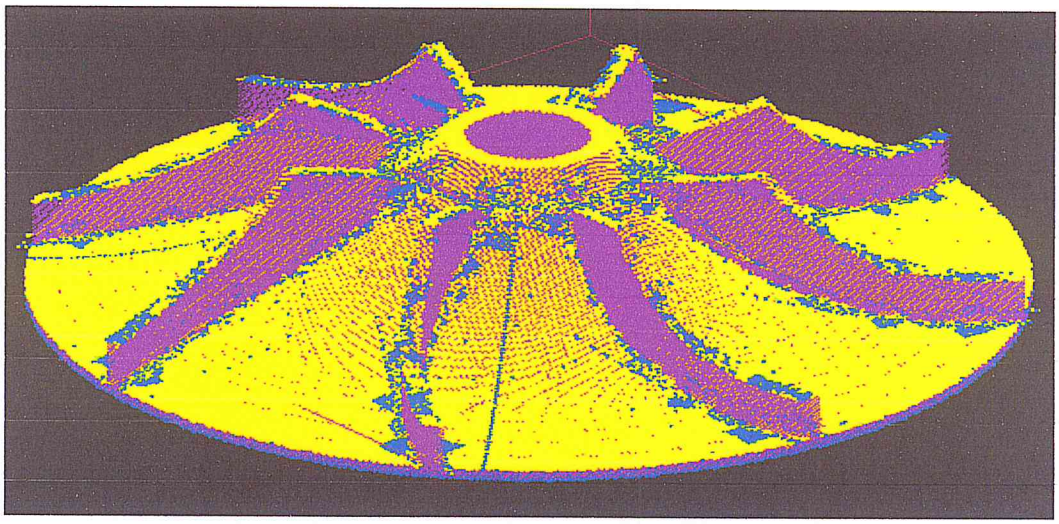
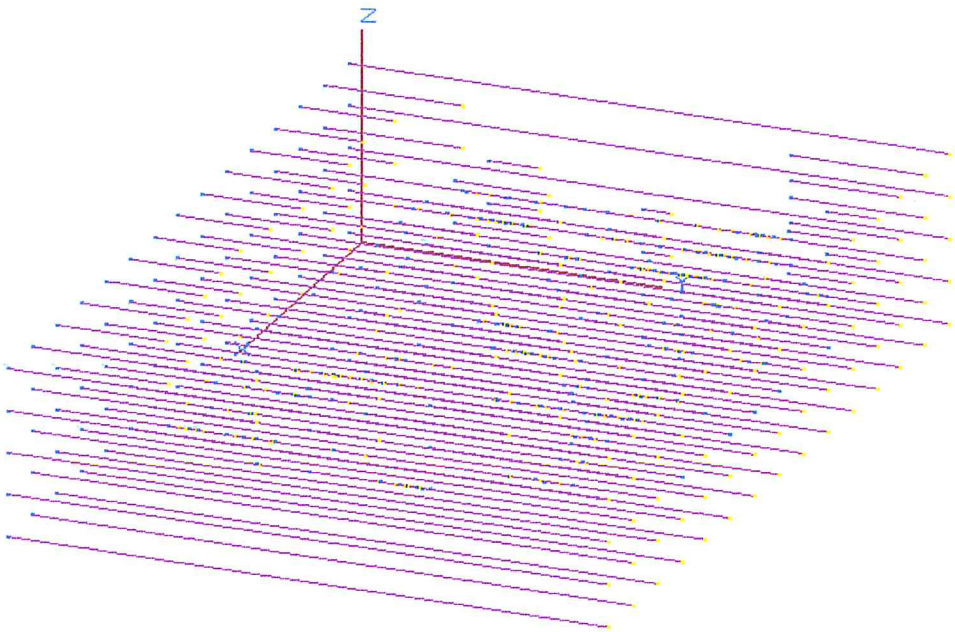


Figure 62 : Visualisation des points d'intersection.



a. Résultat pour le modèle de la pièce 1 sur le plan XY.



b. Résultat pour le modèle de la pièce 3 sur le plan YZ.

Figure 63 : Visualisation des Dexels et des points d'intersection.

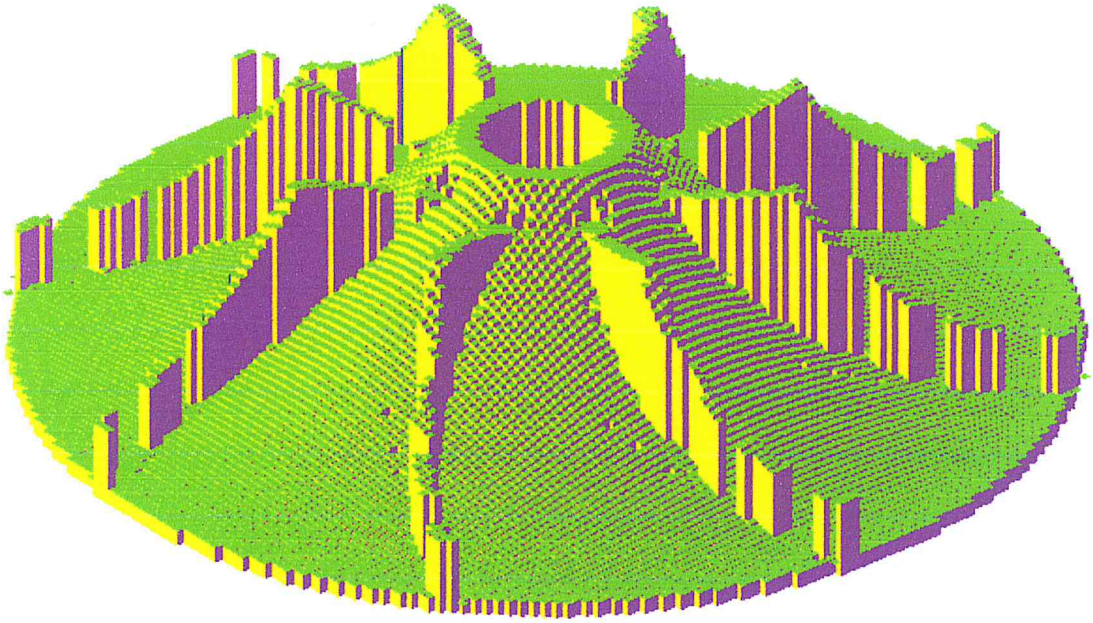


Figure 64 : Visualisation volumique des Dexels suivant l'axe Z pour la pièce 1.

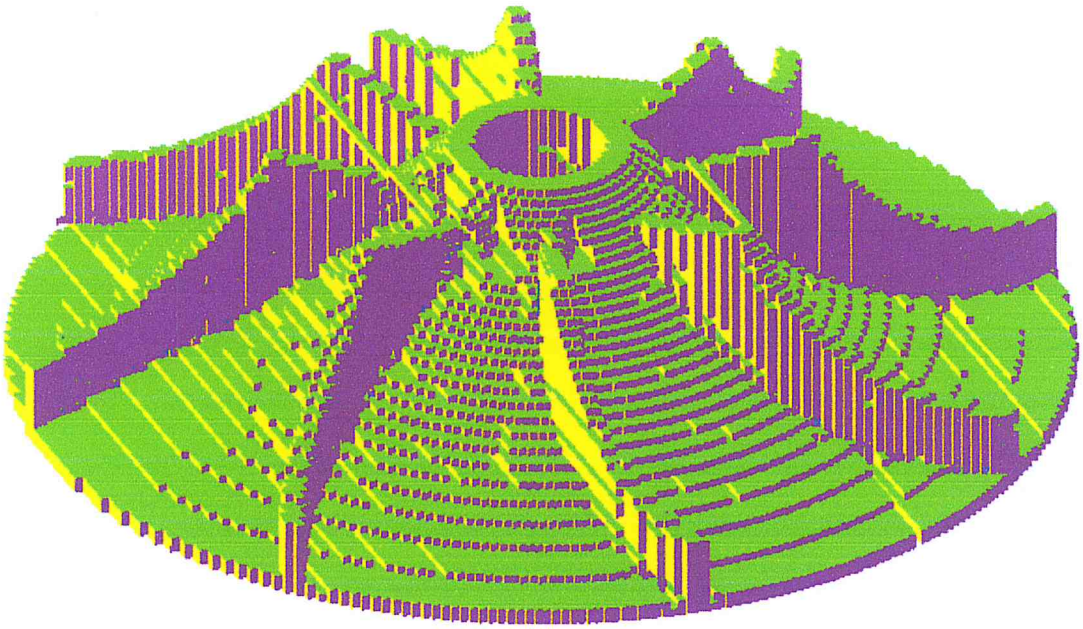


Figure 65 : Visualisation volumique des Dexels suivant l'axe Y pour la pièce 1.

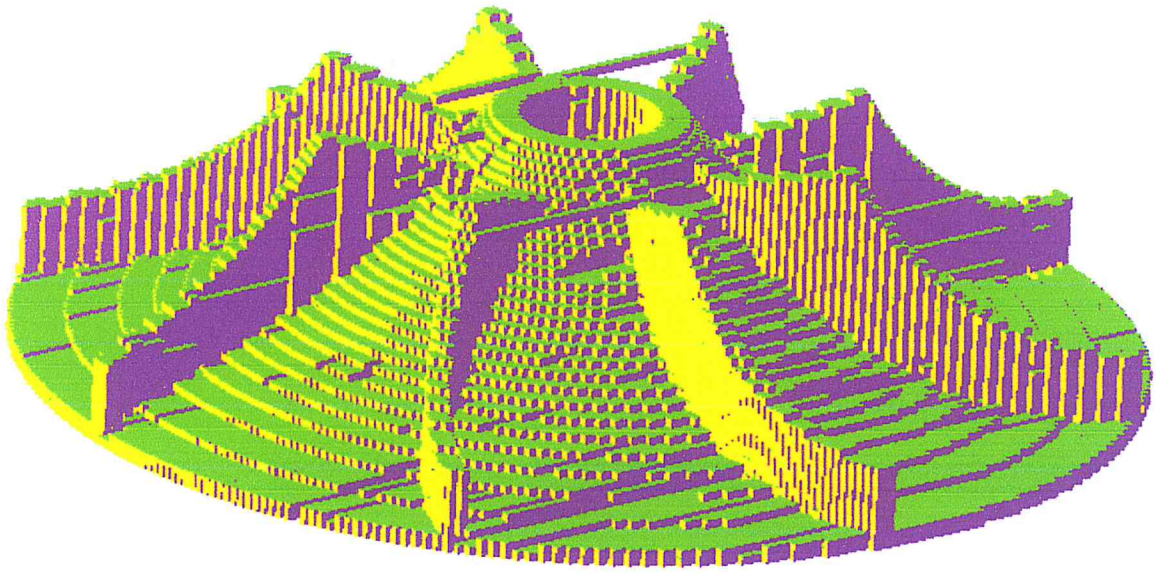


Figure 66 : Visualisation volumique des Dexels suivant l'axe X pour la pièce 1.

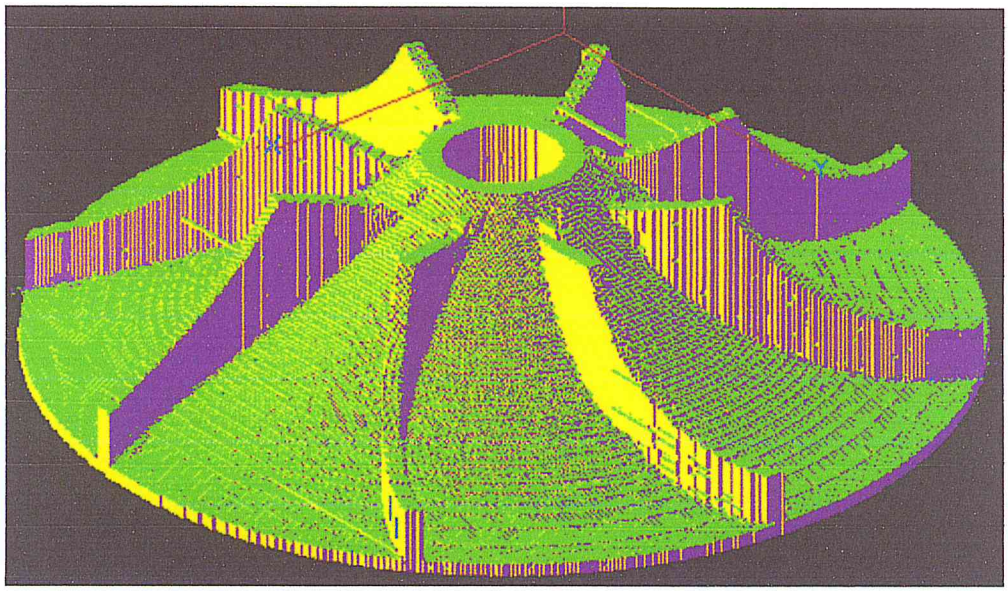


Figure 67 : Visualisation volumique des Dexels suivant l'axe Z pour la pièce 1.

2.5. Optimisation des pas de la grille :

L'utilisateur peut saisir les pas de la grille sur les trois plans, mais il ne pourra jamais prédire le meilleur pas pour avoir la meilleure présentation volumique possible de la pièce. Cette optimisation est faite par rapport aux pas initiaux et le pourcentage d'erreur par rapport au volume de la pièce saisis par l'utilisateur (Figure 66).

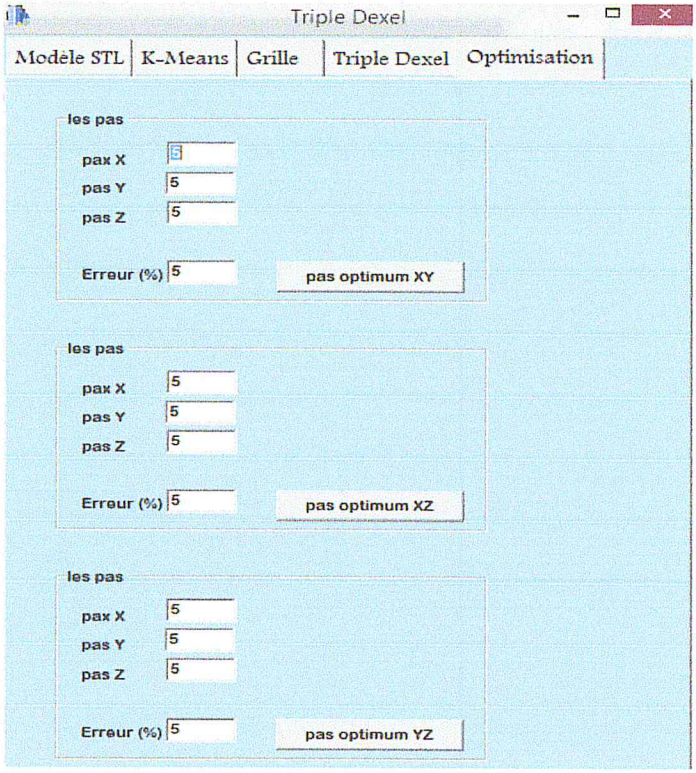


Figure 66 : Onglet d'optimisation.

Pour le test, nous avons fixé les pas sur le plan XY égaux à 10 et le pourcentage de l'erreur est fixé égal à 3%. La Figure 67 montre les Dexels avant l'optimisation et la Figure 68 montre les Dexels après l'optimisation.

Avant l'optimisation :

- Le pas suivant X: 10mm
- Le pas suivant Y: 10mm
- Le volume de la pièce : 700234,555mm^3

Après l'optimisation :

- Le pas suivant X: 2,5mm
- Le pas suivant Y: 1,25mm
- Le volume de la pièce : 708517,376mm^3

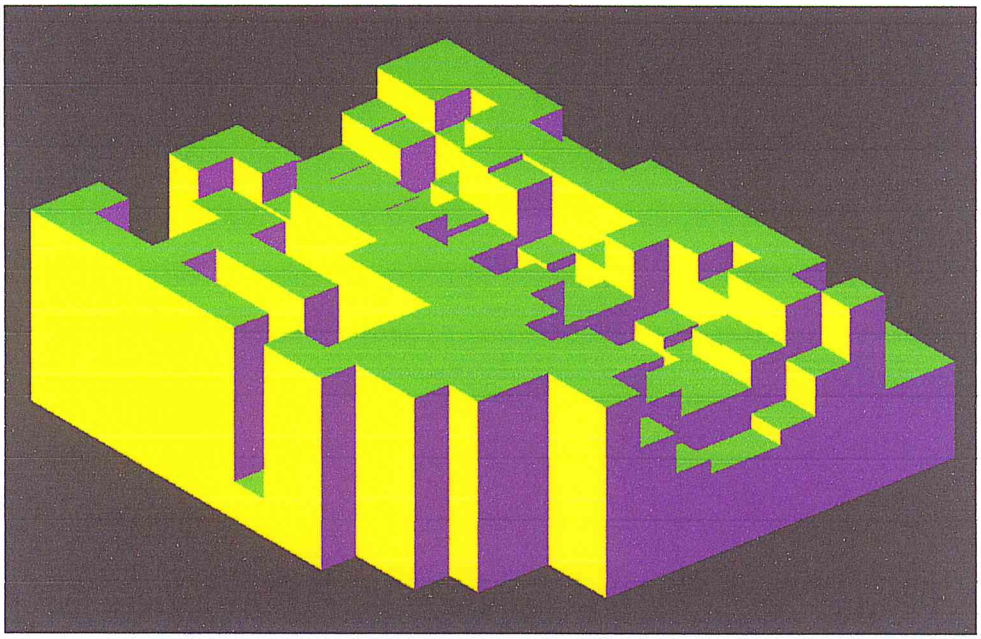


Figure 67 : Visualisation volumique de la pièce avant l'optimisation.

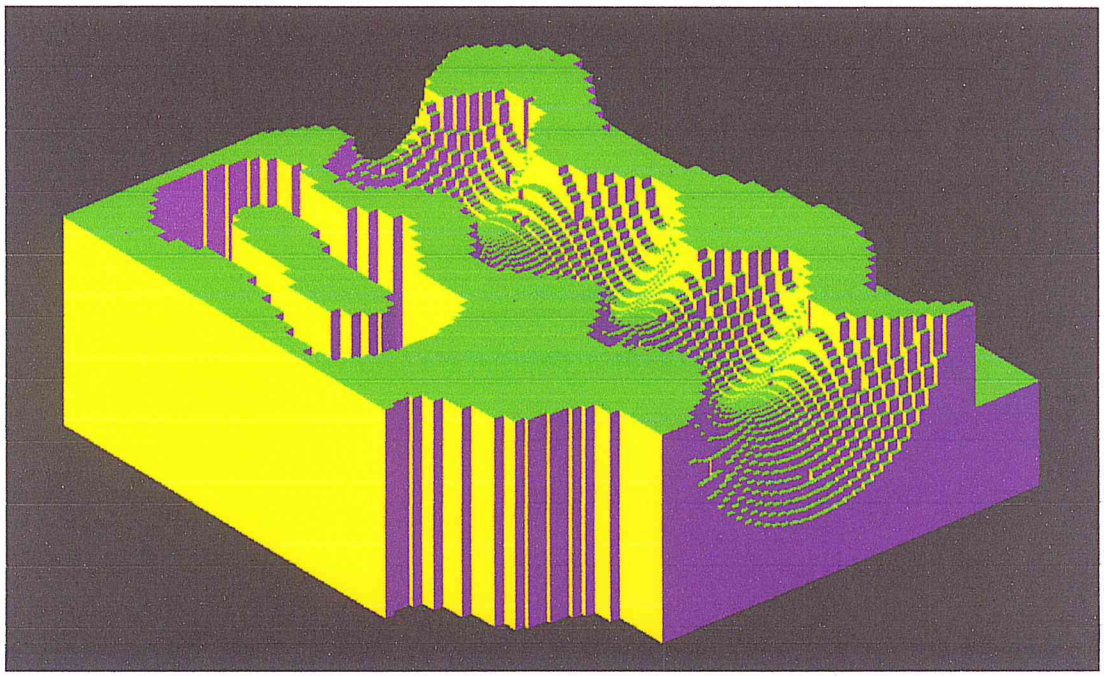


Figure 68 : Visualisation volumique de la pièce après l'optimisation.

3. Conclusion :

Dans ce chapitre, nous avons présenté toutes les formes que nous avons créées et intégrées dans une application de CFAO, permettant la représentation volumique des pièces quel que soit la complexité des formes géométriques d'une manière précise et rapide.

CONCLUSION GENERALE

Conclusion Générale

Le projet que nous avons présenté dans ce mémoire consiste en la conception et le développement d'une application logicielle graphique et interactive permettant la représentation volumique des pièces de formes complexes par la technique Triple Dexel à partir de leurs modèles STL pour la simulation d'enlèvement de matière en usinage 05 axes. Cette technique permet la discrétisation d'un volume en introduisant la quantité de matière. Et de calculer les intersections avec le modèle STL permettant de représenter ces pièces le plus fidèlement possibles avec une bonne précision afin de comptabiliser les défauts d'usinage sur l'état de surface, en répondant au compromis entre précision et temps de calcul.

Lors de la réalisation de ce projet, nous avons commencé par une étude bibliographique sur le processus de CFAO et le format d'échange de données « STL». Aprés, nous avons présenté les différentes méthodes de simulation d'usinage. Par la suite, l'accent est mis sur la technique Triple Dexel. A la fin de notre mémoire, nous avons présenté la conception, les algorithmes développés et l'implémentation informatique de notre application logicielle ainsi que des tests de validation.

Les principaux résultats de notre application logicielle sont les suivants :

- Minimisation du temps de calcul en utilisant k-means.
- Création des modèles en Dexels suivant les trois axes.
- Création des modèles en « Triple Dexels » d'objets de n'importe quelle forme,
- Détermination des pas optimums suivant les trois axes.

En perspective de notre travail, nous recommandons de traiter les thématiques suivantes :

- Faire une combinaison entre les techniques de simulation géométrique, pour la représentation de l'outil de coupe et le volume balayé par l'outil,
- Intégration de cette technique dans le calcul d'effort de coupe afin d'optimiser les paramètres de coupes,

- Application des méthodes d'optimisation pour réduire le temps de calcul avec le gain en qualité,
- Voir la possibilité de faire une mesure virtuelle sur une pièce modélisée par le Triple Dexel.

References Bibliographies

Références Bibliographique:

[1] K. Bouhadja," Fabrication des surfaces de forme gauche". Rapport de recherche CFAO/CDTA . 2013

[2] S.C. Assouline. « Simulation numérique de l'usinage à l'échelle macroscopique : prise en compte d'une pièce déformable ». Thèse Doctorat, Ecole Nationale Supérieure d'Arts et Métiers - CER de Paris, 2005.

[3] M.E.H.Bendifallah. « Approche pour la Finition des Surfaces Gauches par la Stratégie 'Gradient Conjugué' à Partir de Modèle STL ». Mémoire de Master,Faculté Des Sciences et Sciences de L'ingénieur- Universite Amar Telidji Laghouat, 2011.

[4] K. Bouhadja, M. Bey, "Classification of Simulation Methods in Machining on Multi-axis Machines". Proceedings of the World Congress on Engineering 2014 Vol II, WCE 2014, July 2 - 4, 2014, London, U.K.

[4] Seok Won Lee, Andreas Nestler. « Virtual workpiece: workpiece representation for material removal process ». ManufTechnol 58:443-463, 2012.

[5] Yu Zhang, Xun Xu, Yongxian Liu. « Numerical control machining simulation: a comprehensive survey ». International Journal of Computer Integrated Manufacturing, 24:7, 593-609, 2011.

[6] X.Peng and W. Zhang, « A Virtual Sculpting System Based on Triple Dexel Models with Haptics ». Computer-Aided Design and Applications 01/2009; 6(5). DOI:10.3722/cadaps. 2009.645-659.

[7] W. Zhang, «Virtual Prototyping with Surface Reconstruction and Freeform Geometric Modeling Using Level-set Method ». Missouri University of Science and Technology, 98-99 pages; 2008.

[8] W. Zhang, and M.C. Leu, « Surface reconstruction using dexel data from three sets of orthogonal rays » (2009). Faculty Research & Creative Works. Paper 3678. http://mst.bepress.com/faculty_work/367.

Annex

Annexe

1. INTRODUCTION :

K-means (MacQueen, 1967) est l'un des plus simples algorithmes d'apprentissage non supervisé et un problème d'optimisation combinatoire qui permettent de partitionner de données. La procédure suit un moyen simple et facile de classer un ensemble à travers un certain nombre de grappes données fournies (supposer k grappes) fixé a priori.

2. OBJECTIF DE LA METHODE DE « K-MEANS » :

La méthode des "K-means" reste actuellement la méthode la plus utilisée surtout pour les grands fichiers de données qui contiennent plus de 40 000 individus. En effet, cette méthode a été utilisée pour classer 40 000 personnes. Ceux-ci ont répondu à une enquête sur les ventes par correspondance d'une entreprise afin d'obtenir des profils types de clientèle [Celeux, G., et al., (1989)].

Cette méthode à l'instar de la méthode hiérarchique, a l'avantage d'être efficace et très rapide. La classification hiérarchique a l'inconvénient d'user toutes les ressources de l'ordinateur. Elle procède par le calcul pour chaque point sa distance à tous les autres. Elle effectue ensuite un tri et enfin elle agrège les individus les plus proches. La méthode hiérarchique est itérative et elle est inefficace pour les grands fichiers de données. Le principe de la méthode des "k-means" c'est que la classification se fait sur la base du critère des plus proches voisins. Celui-ci signifie que chaque individu est affecté à une classe s'il est très proche de son centre de gravité.

La particularité de la méthode des "k-means" c'est que le nombre de classes doit être spécifié préalablement. La méthode la plus utilisée pour estimer ce nombre c'est de mener une classification hiérarchique sur un échantillon représentatif de l'ensemble I des individus. Une autre manière de procéder est de se baser sur le nombre de classes obtenues par des classifications ayant les mêmes objectifs que la présente étude.

3. L'algorithmes de la méthode des "k-means" :[2]

Dans la méthode des "k-means", le choix des centres initiaux s'effectue sur la base d'un tirage aléatoire sans remise de **k** individus à partir de la population à classifier. La partition des classes est modifiée avec chaque affectation d'un individu i de I.

Les individus sont géométriquement représentés dans l'espace vectoriel ^{P} muni d'une distance notée **d**. L'algorithme de la méthode des "k-means" se déroule comme suit :

Etape (0)

1- On choisit par un tirage aléatoire sans remise k individus parmi n individus composant l'ensemble I. Ces k centres notés $\{C_1^0, C_2^0, ..., C_k^0\}$ sont provisoires.

2- Chaque individu i de I est affecté à une classe et une seule. Chacune de ces classes est localisée par son centre. La procédure d'affectation est la suivante : i est affecté à la classe notée P_l^0 de centre C_l^0 si et seulement si $d[i,C_l^0] = \inf_{j=1,...,k}\{d[i,C_j^0]\}$.

Après avoir affecté tous les individus on obtient k classes notées $\{P_1^0, P_2^0, ..., P_k^0\}$ de centres respectifs $\{C_1^0, C_2^0, ..., C_k^0\}$.

Etape (1) En considérant les k classes obtenues à l'étape -0-, on calcule ses centres de gravité. On obtient donc k nouveaux centres notés $\{C_1^1, C_2^1, ..., C_k^1\}$. On utilise la même règle d'affectation qu'à l'étape -0-, on obtient k nouveaux classes $\{P_1^1, P_2^1, ..., P_k^1\}$ de centres respectifs $\{C_1^1, C_2^1, ..., C_k^1\}$.

Etape -h- On détermine k nouveaux classes en calculant les centres de gravité des classes obtenues à l'étape (h-1). La règle d'affectation reste la même qu'à l'étape précédente et on obtient par la suite une nouvelle typologie de l'ensemble I : $\{P_1^h, P_2^h, ..., P_k^h\}$ de centres respectifs $\{C_1^h, C_2^h, ..., C_k^h\}$

L'arrêt de l'algorithme

L'arrêt de l'algorithme de la méthode des "k-means" se fait

- Lorsque deux itérations successives conduisent à une même partition.
- Lorsqu'on fixe un critère d'arrêt tel que le nombre maximal.
-

4. Comment appliquer k-means dans notre projet :

Dans notre cas, les individus sont les centres des triangles, alors on suit les mêmes étapes pour ces centres, et le nombre k est choisi par l'expert. Ensuit :

1- On crée une enveloppe pour chaque ensemble des points (centre des triangles).
2- On vérifie l'intersection de chaque droite de cellule avec chaque enveloppe.
3- Si oui on vérifie l'intersection de cette droite avec tous les triangles de cette clauster (ensemble), sinon on ne fait rien.

5. Avantages et inconvénients pour la méthode K-means :

5.1 **Avantages** : l'algorithme de k-means est simple et compréhensible et correspondent à une complexité de calculs en O(n). il est ainsi applicable à des données de grande taille, sans problème de mémoire et de temps calcul. La complexité est plus précisément en O(k.n) pour chaque itération, où **n** et **k** sont respectivement le nombre d'objets à classer et le nombre de classes. Le nombre n d'objets étant très important par rapport au nombre de k de

classes, la complexité des calculs de calculs de méthodes de k-means et en O(n).

5.2 **Inconvénient** : Dans ce type de méthode le nombre de classe doit être fixé au début. Et ce méthode ne permet pas de détecter des données bruitées ou la présence d'outliers et fournit des classes convexes. Le résultat dépend fortement au tirage initial des points représentant les centres des classes d'où l'obtention d'une solution locale. Il est donc conseillé d'effectuer plusieurs tirages et comparer les déférentes partitions correspondantes obtenues.