REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLEB DE BLIDA
FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

CDTA

MEMOIRE DE FIN D'ETUDES

Pour l'obtention

D'un Diplôme de Master en Informatique

Option : Ingénierie des Logiciels

THÈME :

Tréflage des Surfaces Gauches Discrètes sur des Fraiseuses 05-axes

Réalisé par:

Mr. DRIOUECH Abdelkader

Mr.LARBI CHERIF Dhia Eddine

Soutenu devant :

Mr. BEY Mohamed	CDTA,	Encadreur
Mr. BENDIFALLAH Hassène	CDTA,	Encadreur
Mme. TCHANTCHANE Zahida	CDTA,	Encadreuse
Mr. KAMECHE Abdellah Hichem	USDB,	Promoteur
Mr. BALA Mahfoud	USDB,	Président
Mme OUAHRANI Leila	USDB,	Examinatrice

Année Universitaire 2017/2018

REMERCIEMENTS

Nous remercions avant tout le bon Dieu qui nous a donné la force de réaliser ce modeste travail.

Nous tenons à remercier du plus profonds nos encadreurs Mr « *BEY Mohamed* » et Mr « *BENDIFALLAH Hassène* » et Mme « *TCHANTCHANE Zahida* » pour leurs aides, leurs patiences, leurs disponibilités et surtout pour leurs soutient.

Nous remercions également l'ensemble des personnes du Centre de Développement des Technologies Avancées (CDTA) en particulier l'équipe « CFAO » de la Division Productique et Robotique.

Nous remercions aussi l'ensemble du département d'Informatique et tous les enseignants.

Un remerciement spécial à notre promoteur Mr « *KAMECHE Abdellah Hichem* » pour son aide et son soutient.

Nous remercions tous ceux qui ont participé à la réalisation de ce travail de prêt ou de loin.

Dédicace

Je dédie ce modeste travail:
A mes très chers parents pour leurs sacrifices et
la confiance qu'ils m'accordent
Que Dieu me les garde
A mes frères et ma sœur
A ma fiancée
A tous les membres de ma famille
A tous mes amis qui me soutiennent
La liste est longue !

Dhia Eddine

Dédicace

Je dédie ce modeste travail:

A mes très chers parents pour leurs sacrifices et

la confiance qu'ils m'accordent

Que Dieu me les garde

A mon petit frère et ma sœur

A tous les membres de ma famille

A tous mes amis qui me soutiennent

La liste est longue !

Abdelkader

Résumé :

Ce travail s'insère dans le cadre de développement d'un environnement de fabrication de surfaces gauches initié par l'équipe « CFAO » du Centre de Développement des Technologies Avancées « CDTA ».

Dans ce travail, il s'agit de développer un module logiciel graphique et interactif sous Windows, à intégrer à la plate-forme développée par l'équipe CFAO du CDTA, pour le Tréflage des surfaces gauches sur des fraiseuses numériques à 05-axes. Il s'agit de déterminer les positions de plongées, les orientations de l'outil valides et les profondeurs de plongées.

Mots Clés : Surfaces Gauches, Tréflage, Modèle STL, Fraiseuse 05-axes, positions de plongées, Profondeur de Plongée, Orientation de l'Outil.

Abstract:

This work is a part of the software developed by "CAD/CAM" team of the Center for the Development of Advanced Technologies "CDTA" dedicated to the machining of free form surfaces on CNC milling machines.

In this work, it is a question of developing a graphic and interactive software module under Windows, to integrate to the platform developed by the "CAD/CAM" team of CDTA, for the "Plunge Milling" of sculptured surfaces on 05-axes CNC milling machines. This involves determining plunging positions, valid tool orientations and plunging depths.

Key Words: Sculptured Surface, Plunge Milling, STL Model, 05-axis CNC Milling Machines, Plunging Positions, Plunging Depths, Tool Orientations.

ملخص:

هذا العمل هو جزء من تطوير وحدات البرامج لإنتاج أسطح الأشكال المعقدة التي بدأها فريق التصميم والتصنيع بمساعدة الحاسوب (CFAO) في قسم الإنتاجية و الروبوتات التابع لمركز تطوير التكنولوجيا المتقدمة (CDTA).

الهدف من هذا العمل هي مسألة تحسين عملية التخشين عن طريق تطبيق عملية الغوص على الأشكال المعقدة التي تتطلب استخدام آلات الطحن العددي 05 محاور على وجه التحديد يتضمن هذا تحديد مواقع الغوص وعمق الغوص. الغرض من هذا العمل هو تطوير وحدة برمجية رسومية و تفاعلية تحت نظام ويندوز ,ودمجها في بيئة المعالجة التي طورها فريق التابع لCDTA لأتمتة عملية معالجة 05 محاور.

الكلمات المفتاحية: الأشكال المعقدة ,الغوص في 5 محاور ،نموذج STL, ماكينات 5 محاور, مناطق الغوص, عمق الغاطس , توجيه أداة الحفر.

Table des matières

Liste des figures

Introduction Générale

INTRODUCTION GENERALE

Présentation du sujet :

Avec le développement qu'a connu l'outil informatique en termes de fiabilité et de rapidité de calcul, celui-ci est devenu ces dernières années un outil indispensable dans tous les domaines de la technologie et de l'automatisation puisqu'il permet aux industriels d'augmenter la productivité tout en assurant une meilleure qualité des produits. Comme toute autre spécialité, l'industrie mécanique a aussi été développée par l'utilisation de cet outil. Cette industrie ne cherche pas seulement la quantité et la qualité mais avec la concurrence industrielle, elle cherche à réduire le cycle de développement d'un produit (temps de développement) depuis la conception, l'analyse jusqu'à la fabrication. Pour réduire les temps de production, les grandes et même les moyennes et les petites entreprises utilisent des logiciels de CFAO (Conception et Fabrication Assistées par Ordinateur). Ces logiciels permettent d'assister les ingénieurs dans les phases de conception, d'analyse et de planification de l'usinage sur des machines-outils à commande numérique.

Dans l'industrie, les surfaces de formes libres (surfaces gauches) sont utilisées dans la conception et la fabrication des moules, des matrices, des formes esthétiques, des formes aérodynamiques… etc. En raison de leurs géométries très complexes, elles sont usinées sur des machines-outils à commande numérique en particulier les fraiseuses à 03, à 04 ou à 05-axes. L'usinage de ces surfaces passe généralement par trois étapes : ébauche pour enlever le maximum de matière, demi-finition pour s'approcher de la forme finale et finition pour obtenir les formes voulues. Toutes ces étapes nécessitent un choix judicieux des formes et des dimensions des outils, des stratégies d'usinage et des conditions de coupe pour respecter les exigences du cahier des charges.

L'opération d'ébauchage consiste à enlever l'excédent de matière, généralement, par le choix entre deux stratégies « Plans Parallèles » et « Contours Décalés ». Afin de minimiser les

temps d'ébauchage de certaines formes notamment les cavités profondes, une nouvelle stratégie a été développée c'est le « Tréflage ». Pour réduire le nombre de mise en position de la pièce à ébaucher et par conséquent les temps d'usinage, il est nécessaire de réaliser l'ébauchage sur des fraiseuses à 05-axes. Le résultat du Tréflage est une géométrie en escaliers d'épaisseurs non uniformes.

Problématique :

L'opération d'ébauchage consiste à enlever l'excédent de matière. Afin d'optimiser les temps d'ébauchage de certaines formes notamment les cavités profondes, une nouvelle stratégie a été développée c'est le « Tréflage » où plusieurs modes de balayage d'outils sont possibles (One-Way, Zig-Zag, Spirale, ...etc.). Pour réduire le nombre de mise en position de la pièce à ébaucher et par conséquent les temps d'usinage, il est nécessaire de réaliser l'ébauchage sur des fraiseuses à 05-axes. Cette stratégie devient plus complexe si elle est appliquée sur des surfaces gauches en contre dépouille ce qui nécessite de développer des procédures spécifiques pour automatiser et optimiser le Tréflage. Son application en 05-axes nécessite le choix judicieux du nombre et des dimensions des outils, des modes de balayage, des positions et des orientations de plongée évitant les interférences et les collisions entre l'outil et l'environnement d'usinage, des profondeurs de plongée et des conditions de coupe. Le résultat du Tréflage est une géométrie en escaliers d'épaisseurs non uniformes.

Objectifdu travail :

Ce travail s'insère dans le cadre de développement de modules logiciels pour la production des surfaces de formes complexes initié par l'équipe Conception et Fabrication Assistées par Ordinateur (CFAO) de la Division Productique et Robotique (DPR) du Centre de Développement des Technologies Avancées (CDTA).

Dans ce projet, à partir des modèles STL des surfaces, il s'agit d'optimiser l'opération d'ébauchage par l'application du procédé de Tréflage sur des formes gauches nécessitant le recours à des fraiseuses numériques à 05-axes. Il s'agit plus précisément de déterminer les positions de plongées, les orientations de l'outil valides et les profondeurs de plongée.

Structuration du mémoire :

Le présent mémoire est composé des parties suivantes :

- Le premier chapitre est consacré à l'étude bibliographique du processus de production des surfaces gauches et en particulier le Tréflage.
- Le deuxième chapitre est réservé à l'étude conceptuelle de notre application ainsi que la présentation des fonctions et des algorithmes utilisés pendant le développement informatique.
- Le dernier chapitre détaille l'implémentation informatique et les tests réalisés pour valider l'approche proposée.
- Ce mémoire se termine par une conclusion générale et des perspectives.

Chapitre 1

Généralités sur le Processus d'Usinage des Surfaces Gauches

Introduction :

Dans l'industrie, les surfaces de formes libres (surfaces gauches) sont utilisées dans la conception et la fabrication des moules, des formes aérodynamiques…etc. Pour raccourcir leur cycle de développement, la conception est réalisée dans des logiciels de Conception Assistée par Ordinateur « CAO » et la préparation de la fabrication est effectuée dans des logiciels de Fabrication Assistée par Ordinateur « FAO ». Pour augmenter la productivité, des logiciels intégrant simultanément la conception et la fabrication sont apparus vers les années 1970. Cette intégration a donné la naissance des logiciels de CFAO. Les formes des surfaces gauches sont très complexes et sont conçues à partir d'une image de la forme que l'ingénieur concepteur veut obtenir. Pour concrétiser son idée, il faut mettre à sa disposition des outils et des méthodes permettant l'édition et la manipulation de ces surfaces d'une manière interactive et en temps réel. Une fois que les modèles mathématiques sont obtenus, il est souvent nécessaire d'échanger ces modèles avec des logiciels métiers pour effectuer d'autres analyses (résistance, cinématique, dynamique, …etc.). D'où la nécessité de développer des moyens permettant de réaliser ces échanges. Ce sont les formats d'échange de données.

Dans ce chapitre, nous allons étudier les différentes méthodes utilisées dans la conception des surfaces gauches et en particulier les surfaces paramétriques. Ensuite, nous présentons le format d'échange de données « STL ».Par la suite, nous allons présenter l'opération d'ébauchage des surfaces gauches et les stratégies utilisées. A la fin, l'accent sera mis sur l'opération de « Tréflage » en 03-axes et en 05-axes.

1. Méthodes de représentation des surfaces [1]:

Les surfaces peuvent être représentées par deux méthodes :

1.1. Surfaces non paramétriques :

Ces surfaces peuvent être représentées sous deux différentes formes :

- **Forme explicite :** la surface est donnée par l'équation suivante :

$$Z = F(X, Y) \qquad \text{(I.1)}$$

Où pour chaque couple (x, y) lui correspond une seule valeur de Z.

- **Forme implicite** : la surface est donnée par l'équation suivante :

$$F(X, Y, Z) = 0 \qquad \text{(I.2)}$$

1.2. Surfaces paramétriques:

Cette représentation nécessite (Figure I. 1) :

- Deux paramètres ***u*** et ***v*** appartenant à l'intervalle [0.1].
- Un ensemble de trois fonctions ***f***, ***g*** et ***h*** pour déterminer les coordonnées X, Y et Z d'un point de coordonnées paramétriques ***u*** et ***v***.

Ce type de surfaces est donné par la formule suivante :

$$F(u,v) = (f(u,v),\ g(u,v),\ h(u,v)) \tag{I.3}$$

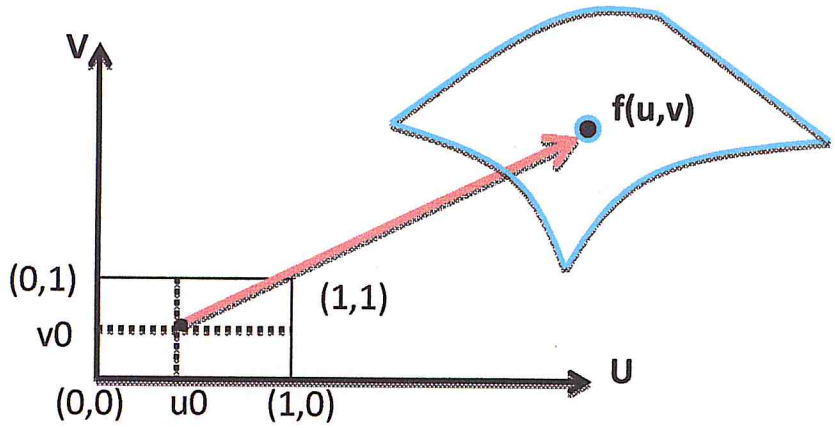


Figure I. 2. Localisation d'un point sur une surface paramétrique.

1.3. Propriétés géométriques des surfaces paramétriques:

Pour l'usinage des surfaces, il est nécessaire de calculer les propriétés géométriques intrinsèques en des points de ces surfaces. Les principales propriétés sont les suivantes.

1.3.1. Vecteurs tangents et vecteur normal à la surface [2]:

Le vecteur tangent dans la direction ***u*** T_u et le vecteur tangent dans la direction ***v*** T_v à la surface paramétrique au point (***u***,***v***) sont données respectivement par :

$$T_u = \frac{\partial F}{\partial u} = \left(\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u}\right) \tag{I.4}$$

$$T_v = \frac{\partial F}{\partial v} = \left(\frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v}\right) \tag{I.5}$$

Ces deux vecteurs définissent le plan tangent à la surface en un point. Le vecteur normal unitaire $\vec{n}$ à la surface au point (***u***, ***v***) est donné par (Figure I. 2) :

$$\vec{n}=\frac{\frac{\partial F}{\partial u}\times\frac{\partial F}{\partial v}}{\frac{\partial F}{\partial u}\times\frac{\partial F}{\partial v}} \qquad (I.6)$$

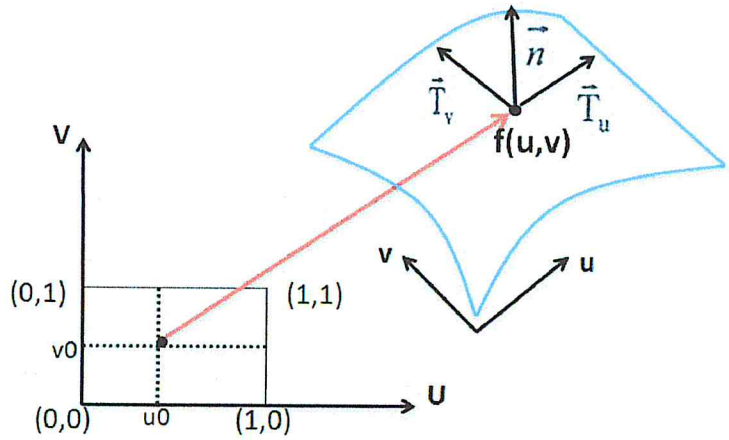


Figure I. 2. Vecteurs tangents et vecteur normal d'une surface paramétrique.

1.3.2. Courbure d'une surface:

➢ **Notion de courbure [3] :** en un point d'une courbe, le cercle osculateur est le cercle de rayon R (rayon de courbure) qui approxime le mieux la courbe (Figure I. 3). La courbure k en ce point est l'inverse du rayon.

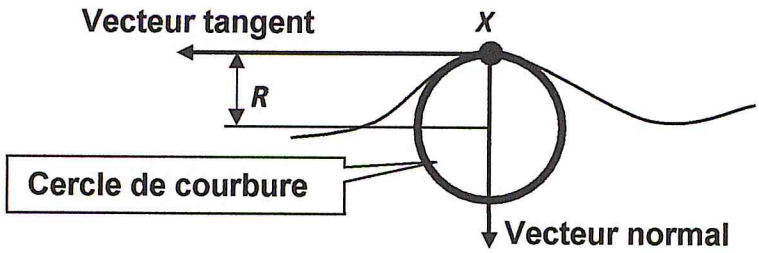


Figure I. 3. Courbure d'une courbe en un point

En chaque point de la surface, un nombre infini de courbes passent par ce point et appartiennent à cette surface. Donc, à chaque courbe est associée une courbure. La plus petite valeur est appelée courbure minimale $\boldsymbol{K_1}$ et la plus grande valeur est appelée courbure maximale $\boldsymbol{K_2}$.Ces courbures sont appelées courbures principales de la surface. A partir de ces courbures, la courbure moyenne $\boldsymbol{H}$ et la courbure gaussienne $\boldsymbol{K}$ sont données par :

$$H=\frac{K_1+K_2}{2} \qquad (I.7)$$

$$K=K_1+K_2 \qquad (I.8)$$

En un point x de la surface et en fonction des valeurs des deux courbures (H, K), la forme locale de la surface au voisinage de ce point peut être concave, concave développable, convexe, convexe développable, selle de cheval et plane.

2. Méthodes de conception des surfaces [4,5,6,7] :

Deux classes de méthodes sont utilisées dans la conception des surfaces.

2.1. Méthode basée sur les courbes :

Pour ces méthodes, la conception de la surface passe par la conception de quelques courbes clés. Cette méthode est utilisée dans la conception des surfaces balayées, lissées, Gordon, Coons, réglées et extrudées.

2.2. Méthode basées sur les points:

Pour ces méthodes, l'information élémentaire est le point. Cette méthode est utilisée dans la conception des surfaces suivantes :

- Interpolation d'un nuage de points (globale ou locale).
- Approximation d'un nuage de points (globale ou locale).
- Surfaces de Bézier et de Bézier Rationnelle.
- Surfaces B-Spline.
- Surfaces NURBS (Non-Uniform Rational B-Spline).

3. Conception et Fabrication Assistées par Ordinateur « CFAO » [8] :

Le développement de produits a toujours impliqué deux processus : la conception et la fabrication. Pour faciliter la conception d'un produit et accroître la productivité, la conception et la fabrication assistées par ordinateur « CFAO » sont apparues. L'idée générale de ce procédé est d'utiliser les capacités de l'ordinateur pour concevoir la pièce en trois (03) dimensions et la fabriquer grâce à une machine automatique programmable. Les objectifs de la CFAO sont l'atteinte d'une extrême précision, le gain de temps et la minimisation le plus possible de l'intervention humaine source d'erreurs.

3.1. Conception Assistée par Ordinateur « CAO » [9]:

La conception assistée par ordinateur « CAO »rassemble des outils informatiques (logiciels et matériels) permettant de réaliser une modélisation géométrique d'un objet afin de pouvoir simuler des tests en vue d'une fabrication. Ses avantages sont :

- Augmenter la productivité et améliorer la qualité de la conception.
- Valider par simulation des solutions adoptées avant la réalisation effective.

3.2. Fabrication Assistée par Ordinateur« FAO » :

La fabrication assistée par ordinateur « FAO » est l'ensemble des outils informatiques permettant de faciliter et d'automatiser les processus de fabrication. Le module « FAO » utilise le modèle « CAO » pour commander la machine-outil. Ses avantages sont :

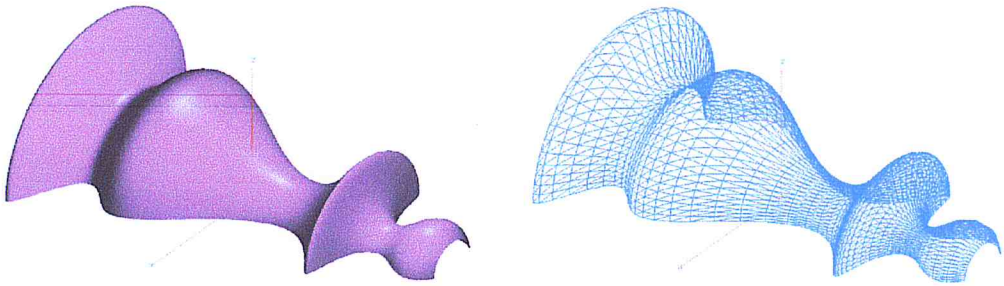
- Obtenir des pièces identiques.
- Assure un niveau de précision très élevé dans la production à grande échelle.

4. Formats d'échange de données [10]:

Pour entamer la fabrication, la « FAO » nécessite certaines informations contenues dans le modèle conçu. Dans un système homogène (« CAO » et« FAO » complètement intégrés dans le même environnement), les informations circulent facilement et directement entre les deux modules. Dans le cas où ces deux modules ne sont pas intégrés, la circulation des données nécessite l'utilisation des translateurs (Format d'échange de données).C'est-à-dire la convention qui permet d'échanger des données entre divers logiciels par l'intermédiaire de fichiers neutres. Plusieurs formats d'échange sont utilisés tels que IGES, STEP, STL, ...etc. Le choix d'un format dépend du besoin et des contraintes de l'application.

4.1. Format STL [11]:

Le modèle STL signifie **S**téréolithographie **T**esselation **L**angage. Par définition, un modèle STL et la représentation des formes tridimensionnelles avec des facettes triangulaires dont la taille est déterminée par une tolérance appliquée aux surfaces d'un modèle solide (Figure I. 4). Dans le format STL, chaque facette est définie par les coordonnées X, Y et Z de ses trois sommets et par un vecteur normal unitaire dirigé vers l'extérieur de l'objet (Figure I. 5). Le format inclus également des règles pour assurer que les triangles sont tous correctement connectée aux nœuds.



a. Surface de base. b. Modèle STL.

Figure I. 4. Format STL d'une surface théorique.

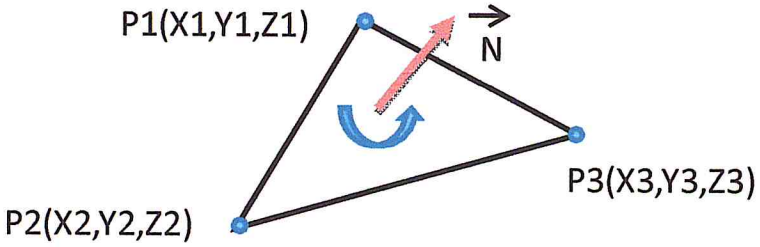


Figure I. 5. Paramètres d'un triangle.

4.1.1. Caractéristique du fichier STL :

- Format de très faible niveau (aucune notion de couleur, texture ou autres attributs).
- Approximation de la frontière par des facettes triangulaires.
- Fichier volumineux puisqu'il est formé d'un nombre important de triangles qui dépend de la précision désirée, de la forme de l'objet et de ses dimensions.
- Utilisé pour le prototypage rapide.

4.1.2. Avantages :

- Fichier plus simple que les autres formats d'échange de données.
- Sa simplicité. Le plus fondamental des entités planaires, le triangle est utilisé pour décrire tout.
- Libère le processus de traduction du fichier et la génération de trajectoire de l'outil de beaucoup de fautes.

4.1.3. Inconvénients :

- Le modèle en facettes triangulaires est une approximation du modèle CAO. La fidélité du modèle d'approximation dépend de la précision imposée.
- Le fichier généré est souvent très volumineux.
- Les rayons de courbures très faibles exigent un grand nombre de triangles.

- Redondance de l'information parce que les sommets de chaque triangle sont partagés par plusieurs triangles ce qui augmente considérablement la taille du fichier.

5. Stratégie d'usinage

Une stratégie d'usinage est une méthodologie utilisée pour générer une série d'opérations dans le but de réaliser une forme donnée. Elle permet d'associer un processus d'usinage à une entité d'usinage, c'est-à-dire un ensemble d'opérations comprenant la définition des outils, des conditions de coupe et des trajets d'usinage.

5.1. Processus d'usinage d'une surface gauche [12,13,14]:

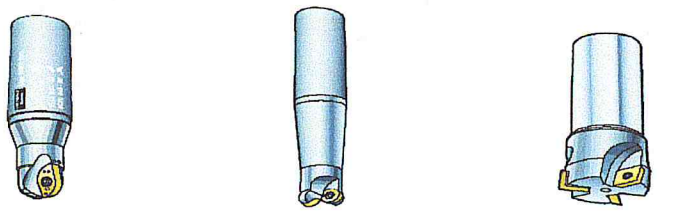
La complexité géométrique des surfaces gauches rend leur usinage très difficile. Pour cela, la fabrication des pièces passe par plusieurs étapes par la prise en compte des formes et des dimensions des outils, des contraintes géométriques des pièces et du taux d'enlèvement de matière. La fabrication d'une pièce passe, généralement, par trois opérations d'usinage :

- **Ebauche :** consiste à enlever le maximum de matière en un temps minimum.
- **Semi-Finition :** consiste à laisser une faible surépaisseur d'usinage pour s'approcher de la forme désirée.
- **Finition :** consiste à obtenir la forme désirée tout en respectant les spécifications dimensionnelles et de forme inscrites dans le cahier des charges.

5.2. Outils d'usinage [15,16]:

Comme pour toute pièce mécanique, l'usinage des surfaces gauches nécessite le choix des formes et des dimensions des outils d'usinage. Pour l'usinage des surfaces gauches, trois formes d'outils peuvent être utilisées (Figure I. 6) :

- **Fraise cylindrique :** généralement utilisée pour l'opération d'ébauchage.
- **Fraise sphérique :** généralement utilisée pour l'opération de finition.
- **Fraise torique :** généralement utilisée pour l'opération d'ébauchage.



a. Sphérique. b. Torique. c. Cylindrique.

Figure I. 6. Types de fraises.

6. Ebauchage des surfaces gauches [17]:

6.1. Définition :

L'opération d'ébauchage est la première étape du processus de fabrication. Elle consiste à enlever le maximum de matière en un temps réduit sans tenir compte de la qualité de l'usinage tout en générant une forme plus ou moins proche de la forme finale avec une surépaisseur d'usinage(Figure I. 7). Le trajet d'usinage le plus économique est celui qui nécessite un minimum de temps d'usinage. Ce temps est évalué par le taux d'enlèvement de matière. Donc, il est important de choisir le plus grand outil possible.

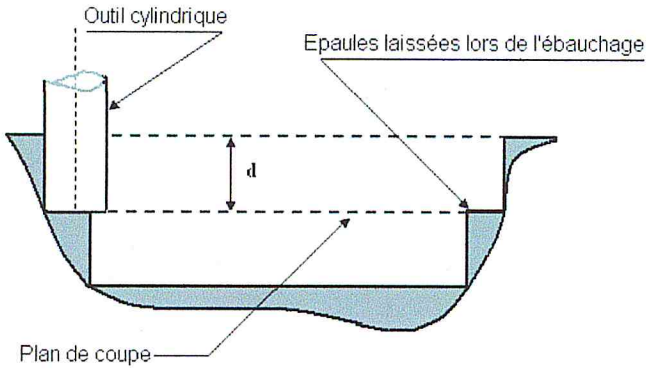


Figure I. 7. Ebauchage avec un outil cylindrique.

6.2. Outils d'ébauchage :

L'ébauchage peut être réalisé par un outil cylindrique, hémisphérique ou torique en fonction de la géométrie de la pièce, de la profondeur et de l'étendue de la forme à usiner.

6.3. Stratégies d'ébauchage :

Pour ébaucher des surfaces gauches, plusieurs stratégies peuvent être utilisées.

6.3.1. Stratégie des plans parallèles [13,18] :

Cette stratégie est composée de deux étapes. La première étape consiste à calculer les contours d'intersection entre des plans horizontaux et la surface à usiner. La deuxième étape consiste à calculer les intersections entre les contours et des plans verticaux parallèles ayant une certaine orientation par rapport au plan XZ (Figure I.8). A la fin, l'usinage en ébauche se fait suivant des segments parallèles entre eux.

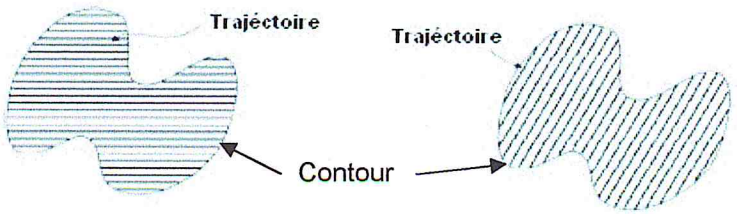
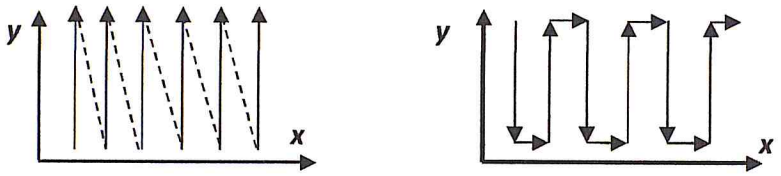


Figure I.8. Directions d'usinage.

Pour cette stratégie, deux modes de balayage de l'outil sont possibles.

- **One-Way (Aller simple) :** l'outil suit une droite jusqu'à sa fin. Ensuite, l'outil quitte la surface et reprend l'usinage au début de la droite suivante. Ce processus est répété jusqu'à l'usinage de toute la surface (Figure I. 9.a).
- **Zig-Zag (Aller et retour) :**l'outil suit une droite jusqu'à sa fin. Ensuite, l'outil se déplace à la fin de la droite suivante sans quitter la surface et reprend l'usinage en sens inverse. Ce processus est répété jusqu'à l'usinage de toute la surface (Figure I. 9.b).



a. One-Way. b. Zig-Zag

Figure I. 9. Modes de balayage de l'outil pour la stratégie des plans parallèles.

6.3.2. Stratégie des contours décalés :

Cette stratégie est composée de deux étapes. La première étape consiste à calculer les contours d'intersection entre des plans horizontaux et la surface à usiner. La deuxième étape consiste à calculer les contours décalés d'une distance « ω », inférieure au rayon de l'outil, fixée par l'utilisateur (Figure I. 30). Les contours décalés constituent le trajet d'usinage. L'inconvénient de cette stratégie est la possibilité d'existence de zones non usinées si la distance de décalage est plus grande que le rayon de l'outil ou si le contour est très courbé.

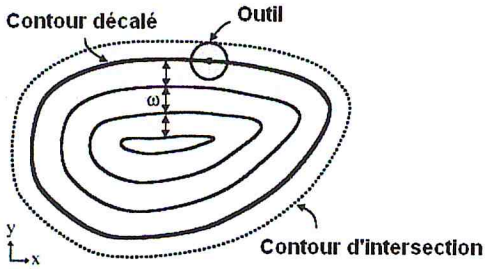


Figure I. 40. Stratégie des contours décalés.

6.4. Problèmes d'usinage [19]:

6.4.1. Problème d'interférence :

L'interférence est la pénétration de la partie active de l'outil dans la matière au-delà de la surface nominale de la pièce (Figure I. 51).

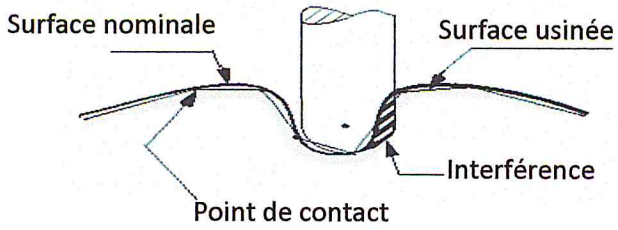


Figure I. 61. Outil en intérference.

6.4.2. Problème de collision :

La collision est observée dans les parties où l'angle formé entre la normale à la surface au point de contact et l'axe de l'outil est supérieur à 90° (Figure I. 72).

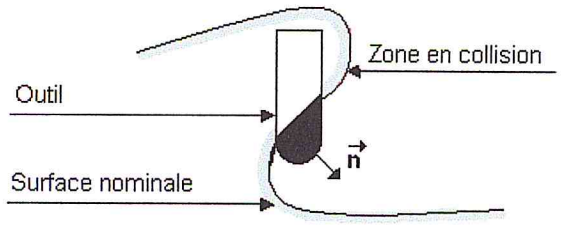


Figure I. 82. Outil en collision.

7. Opération de Tréflage [20] :

7.1. Définition :

Le Tréflage est une nouvelle opération d'usinage utilisée en ébauchage. Son principe est fondé sur l'enlèvement de matière en ayant des avances selon la direction de l'axe de l'outil. La fraise travaille, comme en perçage, suivant l'axe Z de la machine. Cette opération est plus efficace lorsque la hauteur de l'usinage est importante. Le Tréflage est préconisé pour réaliser des cavités profondes des moules et des matrices car il permet de réduire les vibrations en sollicitant l'axe le plus rigide de la machine, à savoir l'axe Z (Figure I. 93).

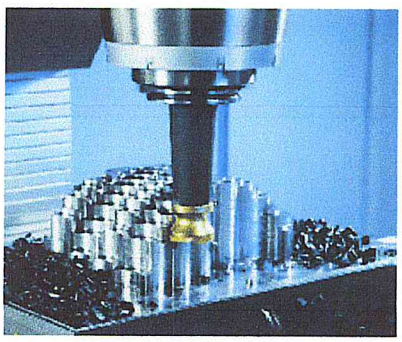


Figure I. 103. Opération de Tréflage.

7.2. Outils de Tréflage :

Initialement, les outils de Tréflage ont été détournés de leurs utilisations initiales. Puis, de nouvelles configurations d'outils et géométries de plaquettes sont développées. Les outils de Tréflage peuvent être classés de plusieurs manières (monoblocs, brasés, indexés ou à plaquettes rapportées, nombre de tailles, montage, ...etc.). La première spécificité de ces fraises porte sur la coupe dite centrale pour effectuer des plongées en pleine matière. L'outil de Tréflage est composé d'une porte plaquette montée sur un système d'attachement (cône de base, adaptateur et rallonge). Ce système permet de positionner l'outil dans la broche de la machine (Figure I. 114). Les caractéristiques telles que le diamètre, la longueur, le pas, le nombre de dents, la géométrie des plaquettes permettent de décrire l'outil de Tréflage.

- Le choix du diamètre d'une fraise est guidé par les dimensions de la pièce.
- Le Tréflage est limité en profondeur par le corps et les rallonges de la fraise utilisée.
- Le pas d'une fraise est la distance entre deux points correspondants sur deux arêtes successives. Une fraise est caractérisée par le nombre de dents (Zn).
- La géométrie de la partie active doit assurer un enlèvement de matière correct.

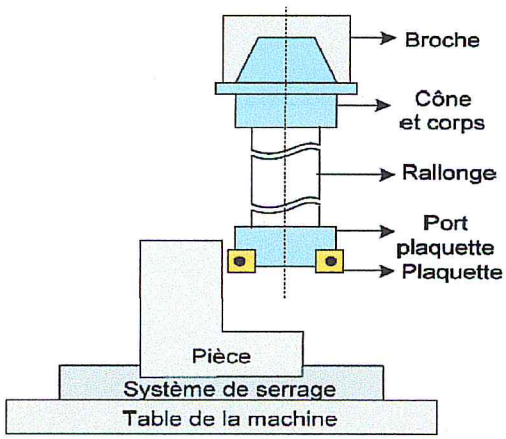


Figure I. 124. Montage de l'outil de Tréflage.

7.3. Paramètres limites de coupe [21]:

Les paramètres limites de coupe en Tréflage sont les suivants (Figure I. 135) :

- **Engagement radial (a_e) (mm) :** est la distance d'engagement de l'outil dans la pièce. Il est limité par la dimension de la plaquette utilisée. Il est déterminé en fonction de la surépaisseur d'usinage, de la géométrie de la pièce, de la puissance et de la rigidité de la machine et de l'outil.
- **Epaisseur de coupe (h) (mm) :** est l'épaisseur axiale de copeau. Elle est constante, à l'exception lors de l'entrée de la fraise dans la pièce, elle passe de zéro à la valeur constante et inversement en sortie de la pièce.

- **Pas radial (P) (mm) :** est la distance entre deux plongées de la fraise dans la matière. Plus le pas radial augmente, plus la longueur d'arête en contact avec la matière augmente. Le pas radial est un paramètre influençant la topographie de la surface.
- **Avance par dent (f_z) (mm/dent) :** est la distance linéaire parcourue par une dent de l'outil. Elle représente la distance couverte entre la pénétration de deux dents successives dans la pièce. La vitesse d'avance (V_f) est exprimée en fonction du nombre de dents Z_n, de l'avance par dent f_z et de la vitesse de rotation de la fraise N par :

$$V_f = Z_n f_z N$$

- **Vitesse de coupe (V_c) (m/min) :** est la vitesse périphérique de l'arête de coupe de l'outil en usinage. Elle dépend des matériaux de la pièce et de l'outil et de l'opération.

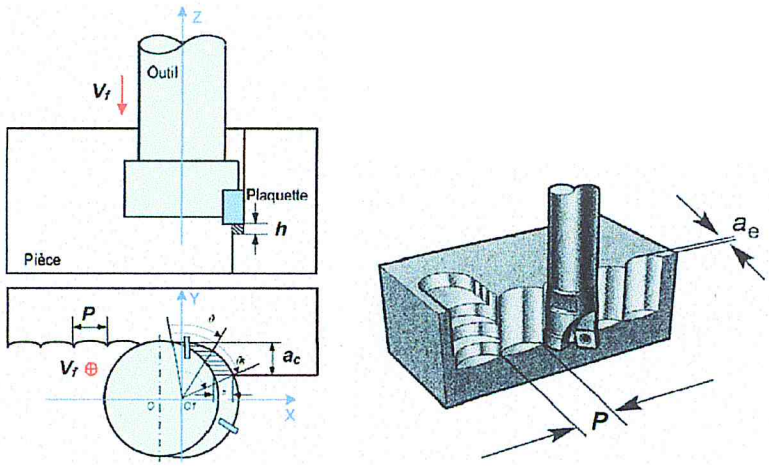
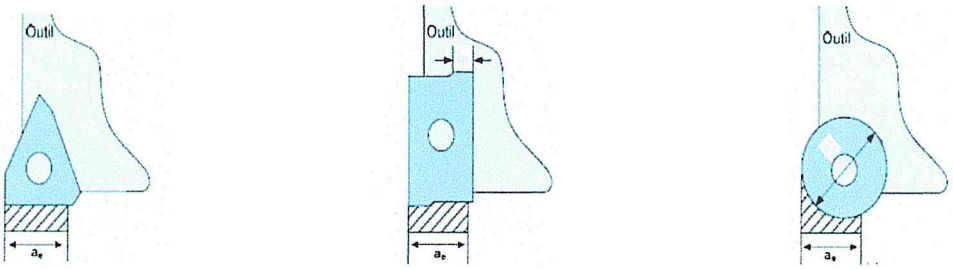


Figure I. 145. Paramètres limites de coupe.

7.4. Types de géométrie de plaquettes :

Les plaquettes employés en Tréflage peuvent être rondes, rectangulaires, carrées et triangulaires (Figure I. 156). Les plaquettes rectangulaires et carrées sont les plus utilisées.



a. Triangulaire. b. Rectangulaire. c. Rondes

Figure I. 166. Types de plaquettes.

7.5. Paramètres pilotant l'opération de Tréflage [21] :

Les principaux paramètres pilotant l'opération de Tréflage sont (Figure I. 177) :

- **Paramètres limites :** engagement radial, épaisseur de coupe, pas radial, avance par dent et vitesse de coupe.
- **Paramètres de liaison :** débit de copeaux, section du copeau, …etc.
- **Paramètres auxiliaires :** énergie spécifique de coupe, puissance de coupe, efforts de coupe, …etc.

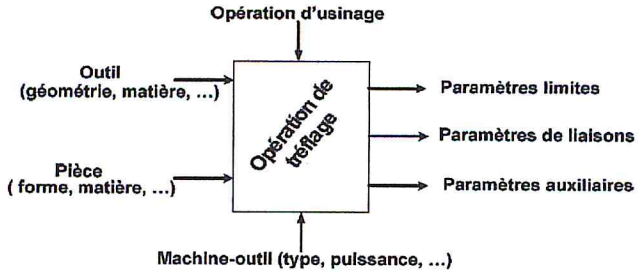


Figure I. 187. Paramètres pilotant l'opération de Tréflage.

7.6. Mouvements de l'outil de Tréflage [21]:

Lors de l'usinage, la course de l'outil peut être descendante ou ascendante dans la pièce, bien que l'usinage en descendant soit le plus courant. Le nom d'usinage en plongée est souvent adopté pour désigner l'usinage par Tréflage. La cinématique de l'opération de Tréflage est caractérisée par deux mouvements spécifiques de l'outil (Figure I. 18) :

- **Mouvement de coupe :** développé par la broche et qui correspond à la rotation de l'outil autour de son axe.
- **Mouvement d'avance :** qui correspond au déplacement de l'outil parallèlement à son axe (suivant l'axe Z).

La spécificité du Tréflage réside dans le sens de la direction d'avance qui est colinéaire à l'axe portant le mouvement de coupe.

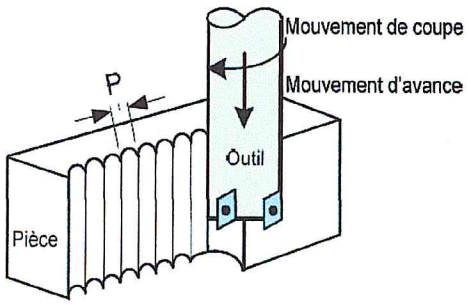


Figure I. 18. Mouvements de l'outil en Tréflage.

7.6.1. Mouvement de coupe :

Le mouvement de coupe est le mouvement principal transmis à l'outil par la broche de la machine-outil. Par ce mouvement, la face de coupe de l'outil attaque le matériau. Il est caractérisé par la direction et la vitesse de coupe :

- **Direction de coupe :** direction instantanée du mouvement de coupe du point considéré de l'arête par rapport à la pièce.
- **Vitesse de coupe (*Vc*) (m/min) :** est la vitesse instantanée du mouvement de coupe du point considéré de l'arête par rapport à la pièce.

7.6.2. Mouvement d'avance :

C'est le mouvement relatif entre l'outil et la pièce nécessaire à la génération de la surface de la pièce. Il peut être composé en plusieurs mouvements. Le mouvement d'avance est caractérisé par la direction et la vitesse d'avance *Vf*).

- **Direction d'avance :** direction instantanée du mouvement d'avance du point considéré de l'arête par rapport à la pièce.
- **Vitesse d'avance *Vf* (mm/min) :** est la vitesse instantanée du mouvement d'avance du point considéré de l'arête par rapport à la pièce.

7.6.3. Mouvement résultant de coupe :

Le mouvement de coupe et le mouvement d'avance combinés constituent le mouvement résultant de coupe. La vitesse résultante de coupe V_e est donnée par :

$$\vec{V}_e = \vec{V}_c + \vec{V}_f$$

7.7. Trajectoires d'usinage en Tréflage [21]:

Un trajet d'usinage est un ensemble de mouvements de type Approcher-Usiner-Dégager. Approcher et Dégager sont des trajectoires hors matière. L'action Usiner comporte une garde d'engagement, la coupe proprement dite et une garde de dégagement. L'approche est un mouvement de l'outil en vitesse rapide dans l'espace. Ses contraintes essentielles sont l'évitement des collisions. Le dégagement est un mouvement avec les mêmes principes que l'approche. En Tréflage, trois modes de dégagement de l'outil sont possibles : 1) dégagement selon l'axe Z (Figure I. 19.a), 2) dégagement incliné (Figure I. 19.b) et 3) dégagement en deux demi-cercles (Figure I. 19.c). Le premier mode requiert des performances spécifiques à la machine-outil (arrêt en fin de plongée et changement de direction) et à l'outil (résistance au choc). Les deux autres modes assurent une remontée d'outil sans contact avec la surface.

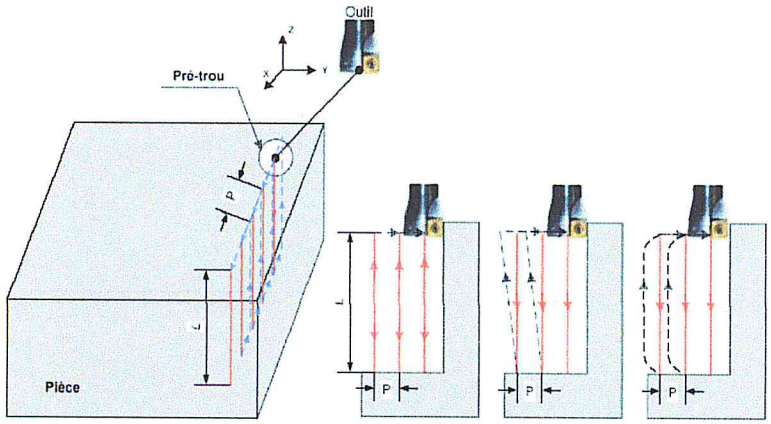


Figure I.19. Modes de dégagement de l'outil en Tréflage.

7.8. Stratégies d'usinage en Tréflage [21]:

Les principaux objectifs d'une stratégie d'usinage sont le respect des conditions d'usinage, de la qualité de surface et de la productivité. En Tréflage, le choix d'une stratégie d'usinage tient compte du type d'outil utilisé, de la variation d'engagement, du pas radial et des contraintes technologiques. Plusieurs stratégies peuvent être utilisées en Tréflage telles que « Zig-Zag », « Contour Parallèle », « Spirale », « Spirale-Contour parallèle», « One-Way », « Zig-Zag-Triangle », ...etc. (Figure I.20).

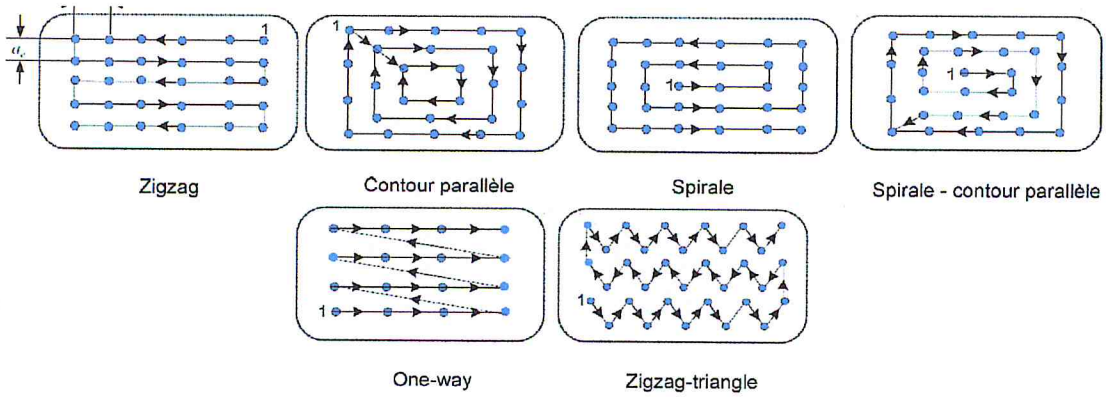


Figure I.20. Stratégies d'usinage en Tréflage.

7.9. Avantages du Tréflage [21]:

- Débit de copeau élevé par rapport aux autres méthodes d'usinage conventionnelles.
- Réduction de la charge appliquée à la broche de la machine-outil.
- Limitation des efforts et moments fléchissant sur l'outil notamment l'effort radial.
- Fraisage profond par l'emploi de fraises avec des attachements de longueur supérieure au fraisage classique.
- Sens de travail principalement en poussant ou en tirant suivant l'axe Z.
- Possibilité d'usiner des formes inclinées.

- Diminution de la déflection de l'outil d'où amélioration de la qualité des surfaces.

7.10. Inconvénients du Tréflage [21] :

- Besoin d'une FAO en vue de définir les stratégies d'usinage.
- Définition des réelles conditions opératoires pour des géométries d'outil données.
- Connaissance des types de dégradation des outils.
- Réalisation en semi finition et finition de surfaces en Tréflage.

7.11. Tréflage en 05-axes :

7.11.1. Définition [22]:

La méthode de coupe utilisant la machine-outil à 05-axes et de la méthode de génération de trajectoire d'outil pour usiner efficacement des formes compliquées nécessite l'inclinaison de l'outil de coupe. En général, les méthodes conventionnelles de coupe utilisent le fraisage de contour avec un contrôle à 03-axes. Cependant, ces méthodes n'ont pas toujours été efficaces car plusieurs changements de réglages sont nécessaires pour usiner des formes complexes. L'étude propose la nouvelle méthode de coupe pour l'usinage 05-axes qui utilise le Tréflage. Par conséquent, la coupe avec le rendement élevé du Tréflage et la flexibilité de la commande à 05-axes peuvent être réalisée. Pour réaliser le procédé, il est nécessaire que la trajectoire de l'outil considère les problèmes d'interférences et de collisions (Figure I.21).

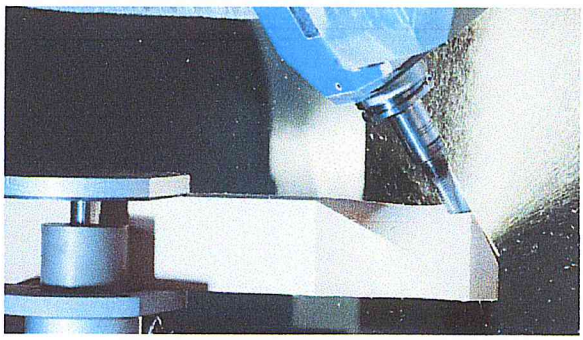


Figure I.21. Opération de Tréflage en 05-axes.

7.11.2. Stratégies de Tréflage en 05-axes :

Pour l'opération de Tréflage en 05-axes plusieurs stratégies peuvent être utilisées. Les contraintes principales sont l'enlèvement du maximum de la matière tout en évitant les problèmes d'interférences et de collisions. Des stratégies spécifiques peuvent être mises en œuvre pour ébaucher des pièces avec des géométries particulières. Le problème posé c'est le développement d'une stratégie indépendante des géométries des pièces à usiner.

7.11.3. Machine 05-axes :

Les surfaces gauches peuvent être réalisées avec un centre d'usinage 03-axes où l'outil est fixé à une broche et peut se déplacer en translation suivant trois axes X, Y et Z. Mais elle pose un problème lorsque le point générateur de l'outil se trouve sur l'axe de rotation de la fraise. Pour résoudre ce problème on utilise la machine en 05-axes qui est l'association d'un centre d'usinage 03-axes et d'une tête rotative à 02-axes (Figure I.22). De même, ces machines sont utilisées dans le cas où la pièce est composée de régions non-accessibles ou des régions en contre dépouille.

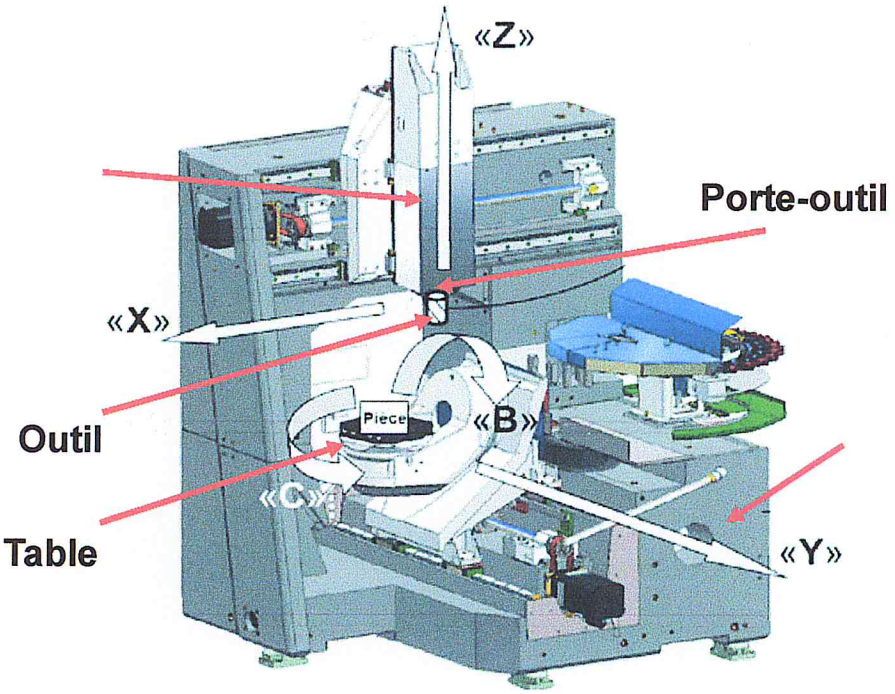


Figure I.22. Fraiseuse 05-axes.

7.11.4. Avantages du Tréflage en 05-axes :

- Amélioration de l'état de surface (bonne condition)
- Diminution des retournements de pièce.
- Eviter les interférences.
- minimiser le temps d'usinage.
- Élargir la distance de la zone à usiner.

7.11.5. Inconvénients du Tréflage en 05-axes :

- Difficulté de programmation.
- Risques plus élevés de collisions.

- Coûts machine, opérateurs, périphérique.

CONCLUSION :

Dans ce chapitre, nous avons présenté les différentes méthodes de conception des surfaces gauches et en particulier les surfaces les plus utilisées dans les logiciels de CFAO ainsi que leurs importantes propriétés. Par la suite, nous avons étudié le format d'échange de données « STL ».

A la suite nous avons mis l'accent sur le processus d'ébauchage et de Tréflage en 03-axes et 05-axes des surfaces gauches.

Chapitre 2

Etude Conceptuelle

Introduction :

Après avoir présenté dans le chapitre précédent un état de l'art et des généralités indispensables pour la compréhension de notre travail et pour donner une vue générale de notre application. Le présent chapitre sera consacré à cerner les étapes entreprises pour réaliser ce travail. Ces étapes sont, en premier lieu, présentées dans l'architecture générale de l'application. Par la suite, elles seront détaillées chacune dans un diagramme à part.

1. Architecture générale de l'application :

L'architecture générale de l'application logicielle à développer est composée de huit parties essentielles :

- Lecture du fichier STL et création des limites du brut.
- Création des cellules.
- Affectation des points des surfaces aux cellules.
- Détermination des paramètres des triangles et des sommets.
- Détermination des zones visibles et des zones accessibles.
- Détermination des points de plongée finales et les profondeurs de plongée en 03-axes.
- Détermination des points de plongée finales et les profondeurs de plongée en 05-axes.
- Génération du trajet d'usinage.

1.1. Lecture du fichier STL et création du brut :

Avant de commencer le processus de Tréflage, le fichier STL décrivant la pièce à usiner doit être analysé. Cette dernière consiste à vérifier son extension et sa syntaxe et en fin calculer les dimensions minimales du brut (Figure II.1).

1.1.1. Vérification de l'extension du ficher :

La lecture d'un fichier STL a pour but de récupérer les informations sur le modèle STL de la pièce à usiner. La première étape consiste à vérifier l'extension et la syntaxe du ficher (Figure II.2).

1.1.2. Calcul des limites du brut :

Une fois le fichier est validé, le brut doit être calculé. La création du brut a pour but d'englober tous les points du modèle STL dans une enveloppe parallélépipédique de faces parallèles aux axes *X*, *Y* et *Z*. Il s'agit de déterminer les coordonnées X_{min}, X_{max}, Y_{min}, Y_{max}, Z_{min} et Z_{max} et par suite ses dimensions minimales longueur, largeur et hauteur (Figure II.3).

$$\text{Longueur} = X_{max} - X_{min}$$

$$\text{Largeur} = Y_{max} - Y_{min}$$

$$\text{Hauteur} = Z_{max} - Z_{min}$$

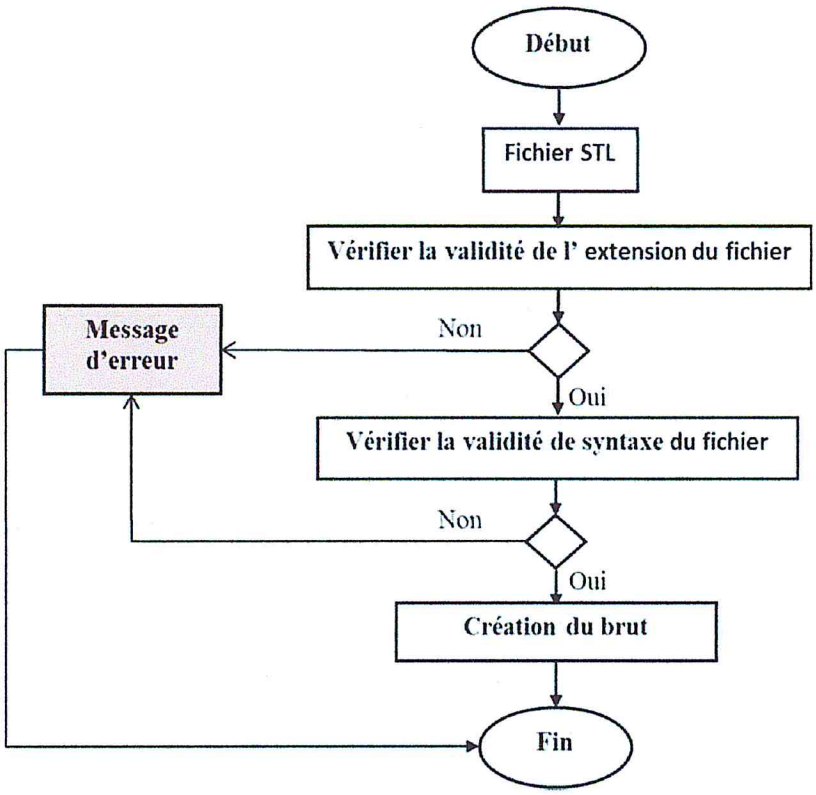


Figure II.1. Vérification du fichier STL et création du brut.

```
facet normal 0.418 0.436 0.796
  outerloop
    vertex 1.836 0 29.100
    vertex 3.673 1.836 27.127
    vertex 1.836 1.836 28.092
  endloop
endfacet
```

a.Syntaxe valide.

```
facet normal 0.418 0.436 0.796
  outerloop
    vert 1.836 0 29.100
    vertex 3.673 1.836 27.127
    vertex 1.836 1.836 28.092
  endloop
endfacet
```

b.Syntaxe invalide.

Figure II.2. Syntaxe du fichier STL.

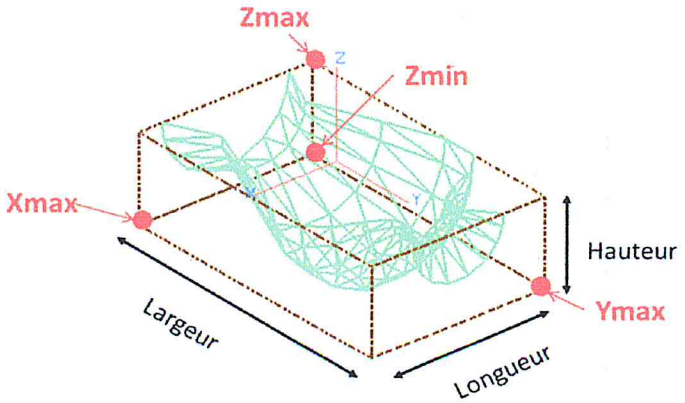


Figure II.3. Limites du brut.

1.2. Création des cellules :

La création des cellules consiste à subdiviser le brut en cellules de mêmes dimensions (Figure II.4). Pour cela, le nombre de subdivisions N_x, N_y et N_z suivant les trois axes X, Y et Z doivent être spécifiées par l'utilisateur. A partir de ces données, les pas suivant les axes X, Y et Z sont calculés par :

$$\text{Pas x} = \text{longueur du brut}/ N_x$$
$$\text{Pas y} = \text{largeur du brut}/ N_y$$
$$\text{Pas z} = \text{hauteur du brut}/ N_z$$

Les limites de chaque cellule sont calculées comme suit :

- **Extrémités minimales :**

Pour i de 0 à Nx $\quad X_{min} = Brut\_X_{min} + i * \text{Pas x}$

Pour j de 0 à Ny $\quad Y_{min} = Brut\_Y_{min} + j * \text{Pas y}$

Pour k de 0 à Nz $\quad Z_{min} = Brut\_Z_{min} + k * \text{Pas z}$

- **Extrémités maximales :**

Pour i de 0 à Nx $\quad X_{max} = X_{min} + \text{Pas x}$

Pour j de 0 à Ny $\quad Y_{max} = Y_{min} + \text{Pas y}$

Pour k de 0 à Nz $\quad Z_{max} = Z_{min} + \text{Pas z}$

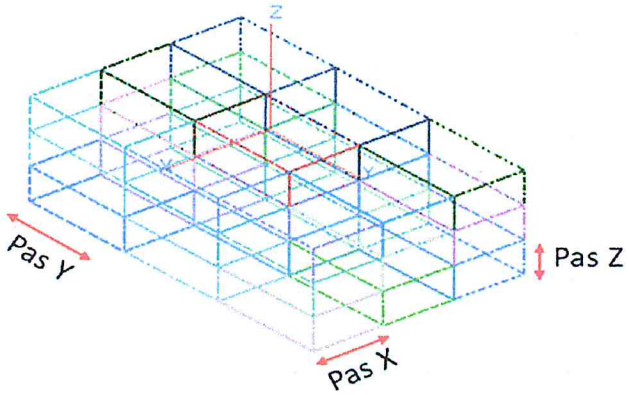


Figure II.4. Création des cellules.

1.3. Affectation des points aux cellules :

Après la création des cellules, chaque sommet des triangles du modèle STL doit être affecté à la cellule correspondante. L'affectation des sommets aux cellules consiste à déterminer les indices de la cellule correspondante (Figure II.5).Les indices de la cellule sont calculés en se basant sur les coordonnées (x, y, z) du point et des coordonnées minimales et maximale du brut comme suit :

$$I = \frac{x - X_{min\ brut}}{Pas\ x}$$

$$J = \frac{y - Y_{min\ brut}}{Pas\ y}$$

$$K = \frac{z - Z_{min\_brut}}{Pas\ z}$$

Pour éviter la duplication des sommets dans la même cellule, des vérifications sont prises en charge avant l'affectation des points dans la cellule. Après la réalisation de ce processus, les sommets et les triangles seront stockés dans deux vecteurs différents.

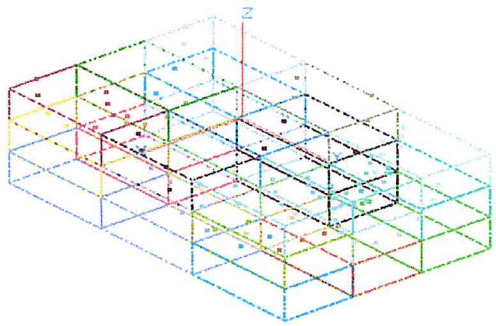


Figure II.5. Affectation des points aux cellules.

1.4. Détermination des paramètres des triangles et des sommets :

1.4.1. Détermination des paramètres de chaque triangle :

Afin de déterminer la normale pour chaque sommet, les différents paramètres des triangles sont calculés :

➢ **Aire d'un triangle :** pour un triangle défini par ses trois sommets $S_1(x1, y1, z1)$, $S_2(x2, y2, z2)$ et $S_3(x3, y3, z3)$, les longueurs A (longueur du segment S_1S_2), B (longueur du segment S_2S_3), C (longueur du segment S_3S_1) et le demi-périmètre P du triangle sont donnés par (Figure II.6) :

$$A = \sqrt{(x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2}$$

$$B = \sqrt{(x2 - x3)^2 + (y2 - y3)^2 + (z2 - z3)^2}$$

$$C = \sqrt{(x1 - x3)^2 + (y1 - y3)^2 + (z1 - z3)^2}$$

$$P = \frac{A + B + C}{2}$$

A partir de ces paramètres, l'aire du triangle est donnée par :

$$\text{aire} = \sqrt{P.(P-A).(P-B).(P-C)}$$

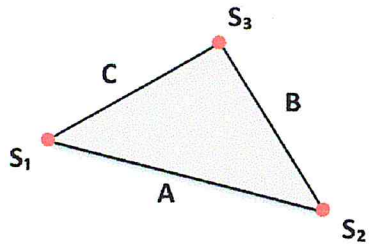


Figure II.6. Aire du triangle.

➢ **Angles aux sommets d'un triangle** : pour calculer les angles aux sommets du triangle (Figure. II.7), la loi des sinus est utilisée :

$$\frac{A}{\sin(\widehat{S_3})} = \frac{B}{\sin(\widehat{S_1})} = \frac{C}{\sin(\widehat{S_2})}$$

En même temps, application du théorème d'Al-Kashi (loi des cosinus) :

$$C^2 = A^2 + B^2 - 2.A.B\cos(\widehat{S_2})$$

A partir de ces deux lois, les angles aux sommets des triangles sont calculés par :

$$\textbf{Angle } \widehat{S_2} : \cos(\widehat{S_2}) = \frac{C^2 - A^2 - B^2}{(-2).A.B}$$

$$\textbf{Angle } \widehat{S_1} : \sin(\widehat{S_1}) = \frac{B.\sin(\widehat{S_2})}{C}$$

$$\textbf{Angle } \widehat{S_3} : \sin(\widehat{S_3}) = \frac{A.\sin(\widehat{S_2})}{C}$$

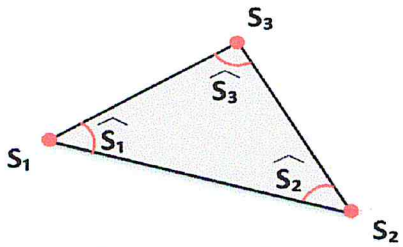


Figure II.7. Angles d'un triangle.

➢ **Voisins d'un triangle :** pour un triangle défini par les sommets S_1, S_2 et S_3, les triangles partagent au moins deux sommets représentent les voisins du triangle.

Pour déterminer les triangles voisins, la convention suivante est utilisée (Figure II.8) :

- Si les sommets S_1 et S_2 (côté A) du triangle T_1 sont communs avec deux autres sommets du triangle T_i, le triangle T_i est le premier voisin noté V_1.
- Si les sommets S_2 et S_3 (côté B) du triangle T_1 sont communs avec deux autres sommets du triangle T_i, le triangle T_i est le deuxième voisin noté V_2.

- Si les sommets S_1 et S_3 (côté C) du triangle T_1 sont communs avec deux autres sommets du triangle T_i, le triangle Ti est le troisième voisin noté V_3.

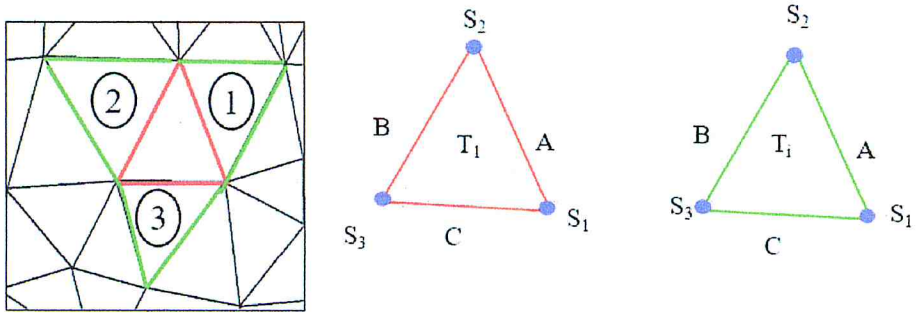


Figure II.8. Voisins d'un triangle.

➢ **Points supplémentaires d'un triangle :** afin d'enrichir le modèle STL, pour chaque triangle, des points sont insérés aléatoirement. Le nombre de points insérés est fonction de l'aire du triangle et de la densité de points introduite par l'utilisateur (Figure II.9).

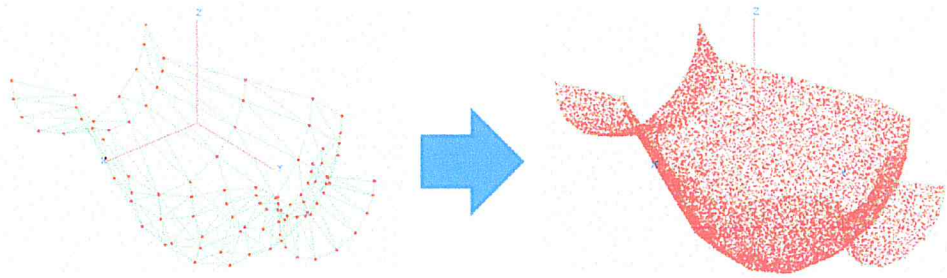


Figure II.9. Points supplémentaires.

1.4.2. Détermination des paramètres des sommets :

Les différents paramètres suivants doivent être calculés (Figure. II.10) :

➢ **Triangles communs :** représentent les triangles partageant un sommet donné.

➢ **Voisins d'un sommet :** représentent les sommets qui l'entourent.

➢ **Normale d'un sommet :** la normale au point P est le vecteur normé calculé en tant que moyenne pondérée des vecteurs normaux aux faces voisines à P. Elle est donnée par :

$$\vec{N} = \frac{\sum_{i=1}^{k} w_i \cdot \vec{n}_i}{\sum_{i=1}^{k} w_i}$$

Où n_i sont les vecteurs normaux des triangles partageant le même point et w_i sont les facteurs de pondération. Ces facteurs peuvent être de trois modes :

- Pondération par les aires des triangles. Dans ce cas, w_i représente l'aire du triangle (Figure II.11.a).
- Pondération par les angles des triangles. Dans ce cas, w_i représente l'angle au sommet du triangle (Figure II.11.b).

- w_i est pris égal à 1. Dans ce cas, c'est la moyenne des vecteurs normaux des triangles partageant ce sommet (Figure II.11.c).

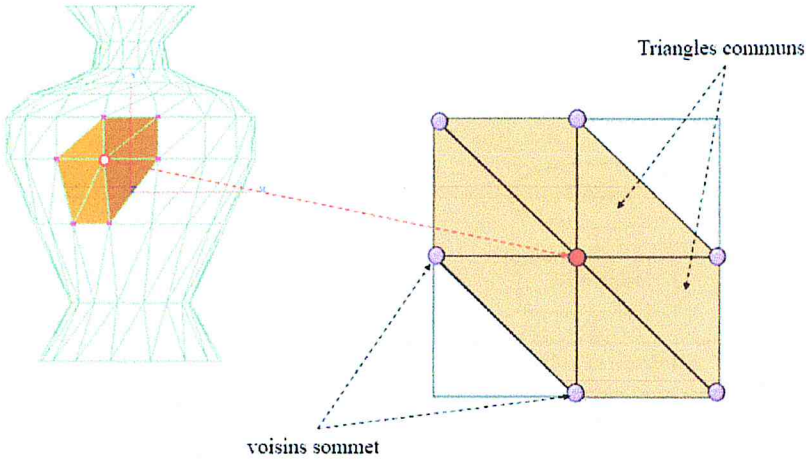
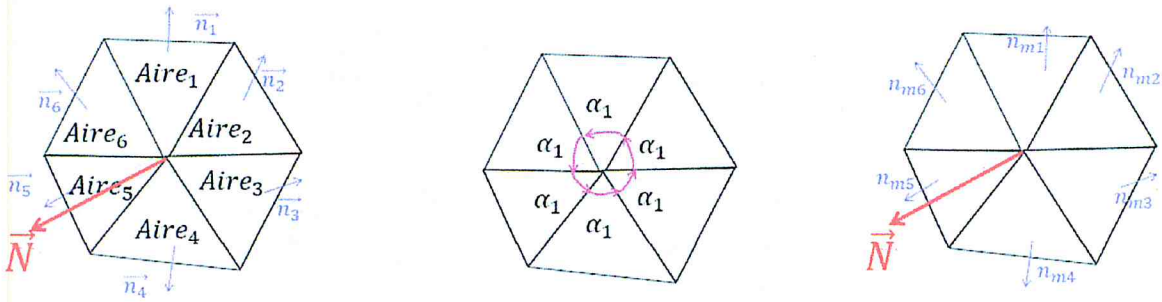


Figure II.10. Triangles communs et voisins du sommet.



a. Par les aires. b. Par les angles. c. Moyenne des normales.

Figure II.11. Modes de pondération de la normale en un point.

1.5. Détermination des zones visibles et des zones accessibles :

1.5.1. Détermination des sommets visibles :

Pour déterminer la visibilité d'un sommet, il faut que l'axe de visibilité et la normale du sommet soient dans la même direction. Pour cela, le produit scalaire entre eux est calculé :

- Si produit scalaire > 0, alors le sommet est visible.
- Sinon, le sommet est invisible.

1.5.2. Détermination des triangles visibles :

Pour déterminer la visibilité d'un triangle, il est nécessaire de vérifier que ces trois sommets sont visibles :

- Si les trois sommets S_1, S_2 et S_3 sont visibles, alors le triangle est visible.
- Sinon, le triangle est invisible.

- Si $\alpha > 0$, alors il y a une intersection entre la droite du sommet et le plan du triangle.
- Sinon, pas d'intersection entre la droite du sommet et le plan du triangle.

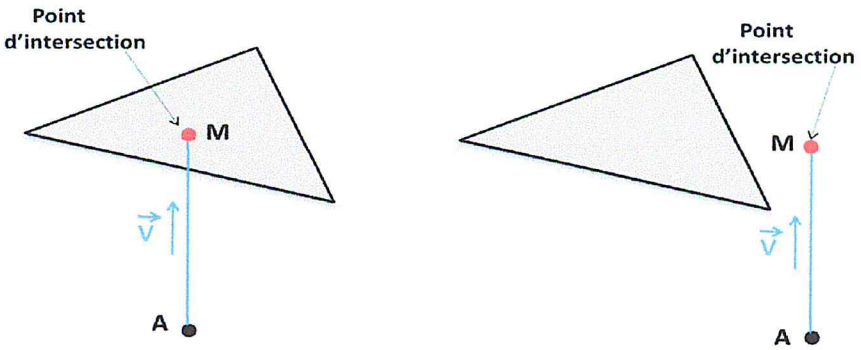
Le point d'intersection est calculé comme suit :

$$\overrightarrow{AM} = \alpha \vec{V}$$

La projection de cette équation vectorielle sur les trois axes donne :

$$\begin{cases} X_M = \alpha X_V + X_A \\ Y_M = \alpha Y_V + Y_A \\ Z_M = \alpha Z_V + Z_A \end{cases}$$

Le point calculé n'est valide que s'il appartient au triangle (Figure II.13).Donc, nous devons vérifier si le point d'intersection appartient dans le triangle.



a. Point appartient au triangle. b. Point n'appartient pas au triangle.

Figure II.13. Point d'intersection entre une droite et un plan.

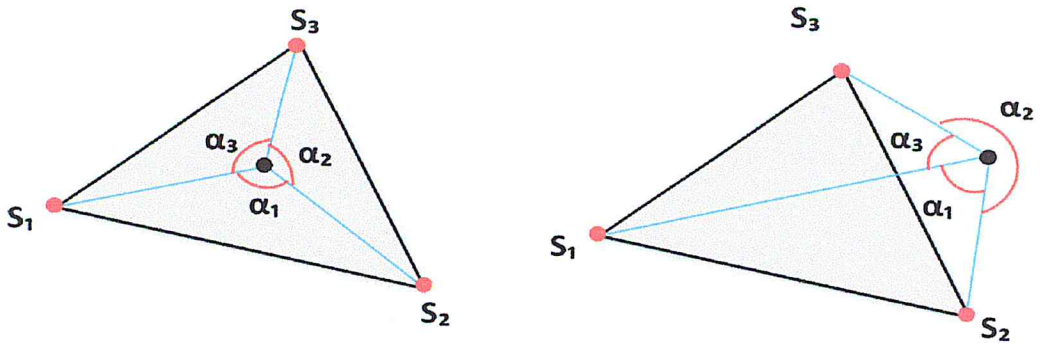
Pour vérifier l'appartenance d'un point **S** à un triangle T défini par ses trois sommets $\mathbf{S_1}$, $\mathbf{S_2}$ et $\mathbf{S_3}$, les paramètres suivants sont calculés (Figure II.14) :

$$\alpha_1 = \text{acos}\left((\mathbf{SS_1}.\mathbf{SS_2}) / |\mathbf{SS_1}|.|\mathbf{SS_2}|\right)$$

$$\alpha_2 = \text{acos}\left((\mathbf{SS_2}.\mathbf{SS_3}) / |\mathbf{SS_2}|.|\mathbf{SS_3}|\right)$$

$$\alpha_3 = \text{acos}\left((\mathbf{SS_3}.\mathbf{SS_1}) / |\mathbf{SS_3}|.|\mathbf{SS_1}|\right)$$

- Si $\alpha_1 + \alpha_2 + \alpha_3 = 360°$, alors le point appartient au triangle.
- Sinon, le point n'appartient pas au triangle.



a. Point appartient au triangle. b. Point n'appartient pas au triangle.

Figure II.14. Vérification de l'appartenance d'un point à un triangle.

1.5.4. Détermination des triangles accessibles :

Pour déterminer l'accessibilité d'un triangle, il est nécessaire de vérifier que ces trois sommets sont accessibles :

- Si les trois sommets S_1, S_2 et S_3 sont accessible, alors le triangle est accessible.
- Sinon, le triangle est inaccessible.

1.6. Détermination des points de plongée finales pour le Tréflage en 03-axes :

1.6.1. Génération des contours décalés à partir des limites du brut :

D'abord, nous devons générer les contours décalés à partir des limites du brut $X_{min}, Y_{min}, Z_{min}, X_{max}, Y_{max}, Z_{max}$ selon un engagement radial et une distance d'engagement spécifiés par l'utilisateur (Figure II.15). Les limites des contours sont calculées comme suit :

Les limites du premier contour sont :

P1 (X_{min}, Y_{min} , Z_{max}+ distance d'engagement)

P2 (X_{min}, Y_{max}, Z_{max}+ distance d'engagement)

P3 (X_{max}, Y_{max}, Z_{max}+ distance d'engagement)

P4 (X_{max}, Y_{min}, Z_{max}+ distance d'engagement)

Pour les autres contours, leurs limites sont calculées par décalage du contour précédent jusqu'à arriver au du contour initial. Leurs limites sont données par :

P1 (X_{min}+ engagement radial, Y_{min}+ engagement radial,Z_{max}+ distance d'engagement)

P2 (X_{min}+ engagement radial, Y_{max}- engagement radial,Z_{max}+ distance d'engagement)

P3 (X_{max}- engagement radial, Y_{max}- engagement radial,Z_{max}+ distance d'engagement)

P4 (X_{max}- engagement radial, Y_{min}+ engagement radial,Z_{max}+ distance d'engagement)

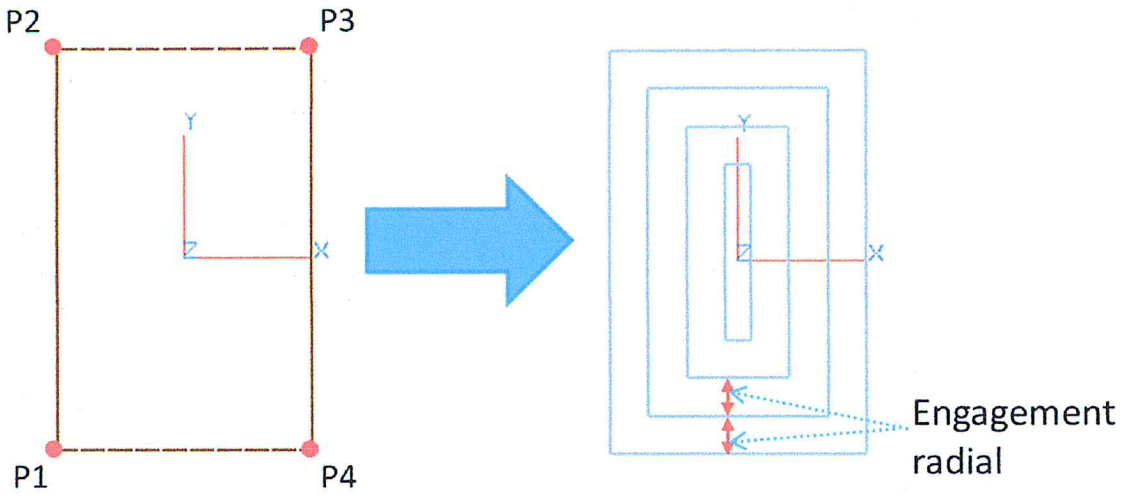


Figure II.15. Génération des contours à partir des limites du brut.

1.6.2. Génération des points de plongée initiaux :

Une fois les contours sont générés, nous devons générer les points de plongées initiaux dans les contours selon un pas radial spécifié par l'utilisateur (Figure II.16).

Pour chaque contour, les points de plongées initiaux sont calculés comme suit :

- **Entre P1 et P2 :**

$$X = X_{\text{point precedent}} + \text{pas radial}$$

$$Y = Y_{\text{point precedent}}$$

$$Z = Z_{\text{point precedent}}$$

- **Entre P2 et P3 :**

$$X = X_{\text{point precedent}}$$

$$Y = Y_{\text{point precedent}} + \text{pas radial}$$

$$Z = Z_{\text{point precedent}}$$

- **Entre P3 et P4 :**

$$X = X_{\text{point precedent}} - \text{pas radial}$$

$$Y = Y_{\text{point precedent}}$$

$$Z = Z_{\text{point precedent}}$$

- **Entre P4 et P1 :**

$$X = X_{\text{point precedent}}$$

$$Y = Y_{\text{point precedent}} - \text{pas radial}$$

$$Z = Z_{\text{point precedent}}$$

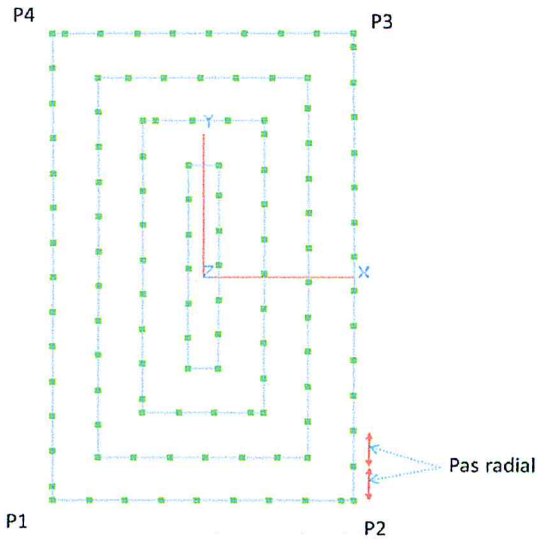


Figure II.16. Génération des points de plongée à partir des contours.

1.6.3. Détermination des positions de plongées :

Pour déterminer la position de plongée de chaque point de plongée initial, nous devons déterminer le point d'intersection entre la droite du point de plongée initial selon l'axes Z et le brut. Pour chaque point de plongée initial PI, la position de plongée P est calculée par :

$$X_P = X_{PI}$$

$$Y_P = Y_{PI}$$

$$Z_P = Z_{MAX\ BRUT}$$

1.6.4. Détermination des points de plongées finaux :

En raison de la géométrie complexe des surfaces gauches, la profondeur de plongée de l'outil change d'une position à une autre. Le calcul de la profondeur de plongée en chaque position de plongée sans interférence passe par les étapes suivantes (Figure II.17) :

- Générer les points supplémentaires dans chaque triangle.
- Pour chaque point de plongée P, vérifier l'intersection de l'outil avec chaque point supplémentaire S.
  - Si ($X_S > X_P$-rayon d'outil et$X_S < X_P$ +rayon d'outil et $Y_S > Y_P$-rayon d'outil et $Y_S < Y_P$ + rayon d'outil), alors il existe une intersection.
  - Sinon, pas d'intersection, donc le point de plongée final prend$Z_{MIN\ BRUT}$.
- Calculer la distance entre le point de plongée initial et chaque point supplémentaire positionné au-dessus d'outil, et prendre le point le plus proche :

Profondeur = distance minimale

- Déterminer la position de plongée finale PF de chaque point de plongée initial PI :

$$X_{PF}=X_{PI}$$

$$Y_{PF}=Y_{PI}$$

$$Z_{PF}=Z_{PI} - \text{Profondeur}$$

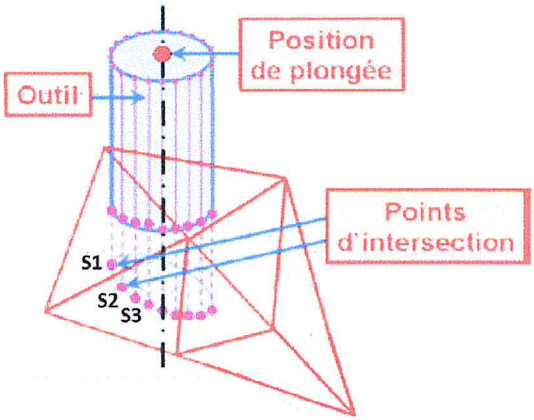


Figure II.17. Calcul de la profondeur de plongée.

1.6.5. Détermination des positions de plongée finales valides :

Pour déterminer la validité de chaque position de plongée finale, nous devons vérifier que le point de plongée final est au-dessus de la position de plongée.

- Si$Z_{PF}<Z_P$, alors la position de plongée finale est valide.
- Sinon, la position de plongée finale est invalide.

1.7. Détermination des points de plongée finales pour le Tréflage en 05-axes :

1.7.1. Génération de la sphère :

La stratégie de Tréflage adoptée consiste à positionner les points de plongée sur une sphère centrée au brut de la pièce. Pour générer la sphère (Figure II.18), nous devons calculer le centre du brut. Par la suite, le rayon de la sphère est calculé comme suit :

$$rayon = \sqrt{(X_{MAX} - X_{CENTRE})^2 + (Y_{MAX} - Y_{CENTRE})^2 + (Z_{MAX} - Z_{CENTRE})^2}$$

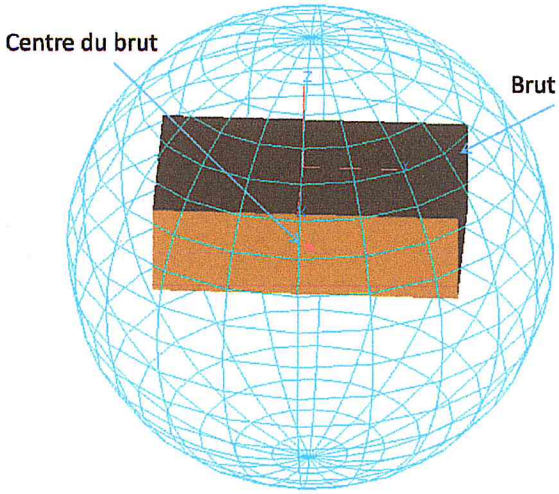


Figure II.18. Génération de la sphère.

1.7.2. Génération des points de plongée initiaux :

La génération des points de plongées initiaux requiert la spécification par l'utilisateur de deux pas (p et q) pour échantillonner la sphère (Figure II.19).

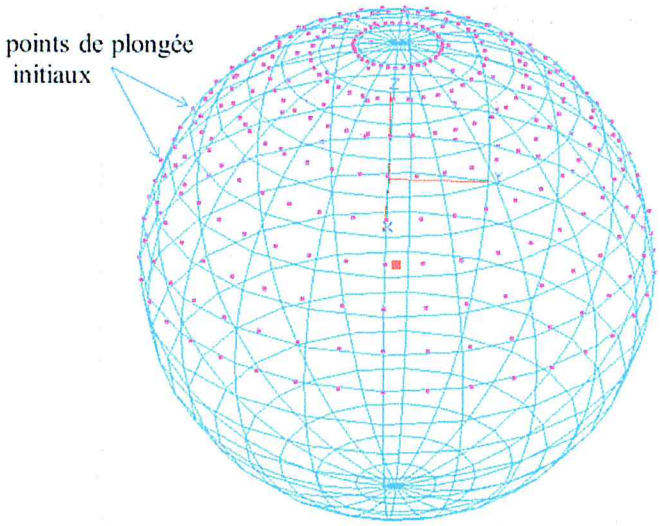


Figure II.19. Génération des points de plongée initiaux.

1.7.3. Détermination des vecteurs de plongée :

Le vecteur de plongée de chaque point de plongée initial PI est calculé comme suit :

$$X_U = X_{PI} - X_{Centre}$$

$$Y_U = Y_{PI} - Y_{Centre}$$

$$Z_U = Z_{PI} - Z_{Centre}$$

1.7.4. Détermination des positions de plongée :

Pour déterminer la position de plongée P de chaque point de plongée initial PI, nous devons déterminer le point d'intersection entre la droite du point plongée initial selon le vecteur de plongée V et le brut. Le calcul passe par les étapes suivantes :

- Affectation des normales de cinq facettes sauf la facette d'appui du brut :

Facette $_1$ (1,0,0)

Facette $_2$ (0,1,0)

Facette $_3$ (0,0,1)

Facette $_4$ (-1,0,0)

Facette $_5$ (0,-1,0)

- Calcul du point d'intersection de chaque point de plongée initial avec ces facettes.
- Prendre le point d'intersection le plus proche par rapport au point de plongée initial.

1.7.5. Détermination des positions de plongées finales :

La détermination de la position de plongée finale de chaque point de plongée initial passe par les étapes suivantes :

- Générer les points supplémentaires des triangles.
- Vérifier le chevauchement de l'outil pour chaque triangle :

❖ Changement du repère des sommets du triangle a_1, b_1 et c_1. Ce changement de repère consiste à déterminer les coordonnées du point dans un repère local lié à l'outil dont l'origine est le point de plongée initial de l'outil C_L. Son premier vecteur unitaire T_1 est perpendiculaire au vecteur directeur de l'axe de l'outil U. Son deuxième vecteur unitaire T_2 est perpendiculaire au plan formé par les vecteurs U et T_1. Son troisième vecteur unitaire est le vecteur directeur de l'axe d'outil U. Donc, les coordonnées des points seront exprimées dans le repère (T1, T2, U). Le changement de repère se fait donc du repère global (repère de conception) vers le repère local (repère de vérification lié à l'outil).

Soit un P de coordonnées (X1, Y1, Z1) exprimées dans un repère global R_1.Les coordonnées du point P de coordonnées (X2, Y2, Z2) exprimées dans le repère local R_2 sont données par :

$$P_{/R1} = \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} P_{/R2} = \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix}$$

Le passage du repère R_2 vers le repère R_1 est donné par :

$$P_{/R1} = M * P_{/R2}$$

Avec :

$$M = \begin{bmatrix} T_{1x} & T_{1y} & T_{1z} & x_0 \\ T_{2x} & T_{2y} & T_{2z} & y_0 \\ U_x & U_y & U_z & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Le passage du repère R_1 vers le repère R_2 est donné par :

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} * M^{-1}$$

Avec :

M^{-1} : matrice inverse de la matrice M.

X_0, Y_0, Z_0 : coordonnées de l'origine du repère R_2 exprimées dans le repère R_1.

Donc, les nouveaux trois sommets du triangle sont A_1,B_1 etC_1

- Vérification si le triangle est positionné au-dessus de l'outil :

$$x_{min} = \min(\min(X_{A_1}, X_{B_1}), X_{C_1})$$

$$x_{max} = \max(\max(X_{A_1}, X_{B_1}), X_{C_1})$$

$$y_{min} = \min(\min(Y_{A_1}, Y_{B_1}), Y_{C_1})$$

$$y_{max} = \max(\max(Y_{A_1}, Y_{B_1}), Y_{C_1})$$

Si (x_{max}< -rayon et y_{max}< -rayon et x_{min}> rayon et y_{min}> rayon), alors le triangle est positionné au-dessus de l'outil.

- Eliminer les points supplémentaires dont la distance entre eux et les points de plongée initiaux est supérieure au rayon de l'outil.
- Déterminer la distance minimale entre le point de plongée initial et les points supplémentaires de chaque triangle positionné au-dessus de l'outil.

Profondeur = distance minimale

- Déterminer la position de plongée finale PF de chaque point de plongée initial PI selon une surépaisseur spécifiée par l'utilisateur :

$$X_{PF} = X_{PI} - (\text{profondeur+surépaisseur}) * X_U$$

$$Y_{PF} = Y_{PI} - (\text{profondeur+surépaisseur}) * Y_U$$

$$Z_{PF} = Z_{PI} - (\text{profondeur+surépaisseur}) * Z_U$$

- Ajouter la distance de sécurité pour éviter les problèmes de collision (Figure II.20).

Distance de sécurité = rayon/ tangent (angle de rotation).

$$X_{PF}=X_{PF}- \text{Distance de sécurité} * X_U$$

$$Y_{PF}=Y_{PF} -\text{Distance de sécurité} * Y_U$$

$$Z_{PF}=Z_{PF} - \text{Distance de sécurité} * Z_U$$

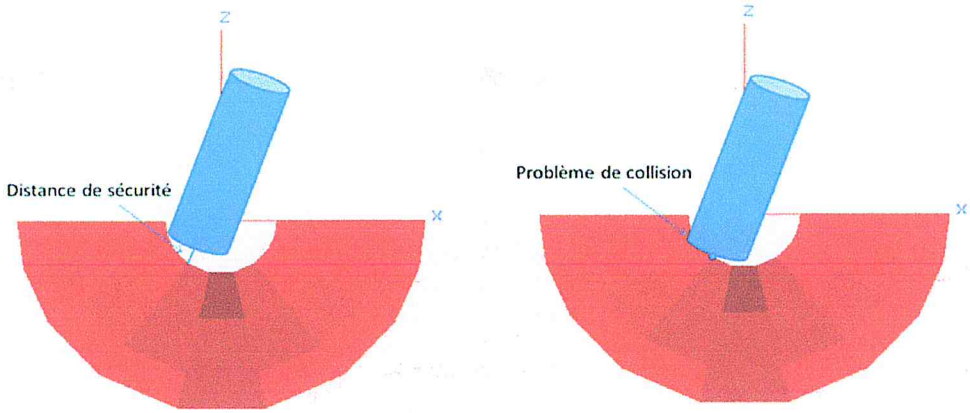


Figure II.20. Distance de sécurité.

1.8. Génération du trajet d'usinage :

Le trajet d'usinage est généré à partir des étapes suivantes :

- Générer la trajectoire d'outils en 03-axes et en 05-axes.
- Simuler les mouvements de l'outil par rapport à la surface en 03-axes et en 05-axes.
- Générer le programme d'usinage « G-Code ».
- Usiner la pièce sur machine.

2. Modélisation UML de l'application :

Pour présenter les différentes fonctionnalités de notre application, nous avons opté pour les diagrammes du langage UML. La modélisation UML permet de :

- Faciliter l'analyse, la compréhension et la réduction de la complexité du système.
- Simuler le système et reproduire ses comportements.
- Penser objet dès le départ.

Dans notre projet, nous avons utilisé les diagrammes suivants :

- Diagramme de cas d'utilisation.
- Diagramme de classe.

2.1. Diagramme de cas d'utilisation :

Les cas d'utilisations permettent de recueillir, d'analyser, d'organiser les besoins et de recenser les grandes fonctionnalités du système. Nous commençons par le digramme de cas d'utilisation générale qui donne une vue globale du fonctionnement du système.

2.1.1. Diagramme de cas d'utilisation générale :

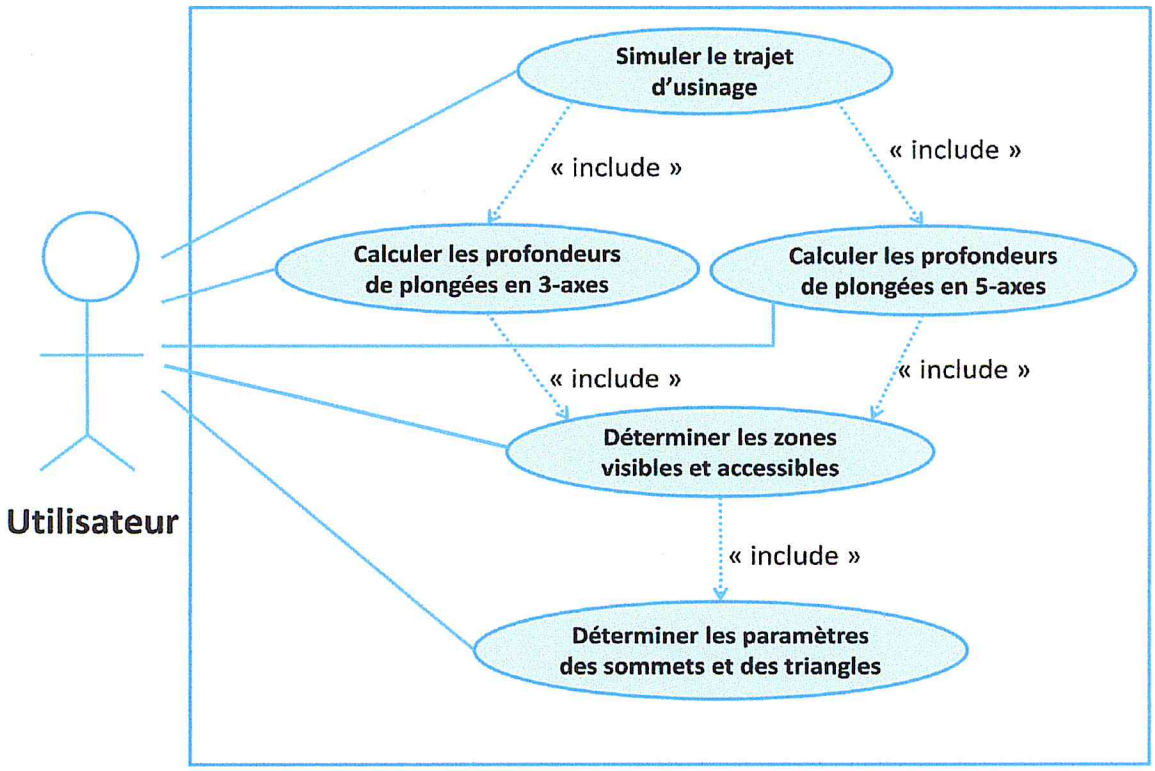


Figure II.21. Diagramme de cas d'utilisation globale.

2.1.2. Diagramme de cas d'utilisation « déterminer les paramètres des sommes et des triangles » :

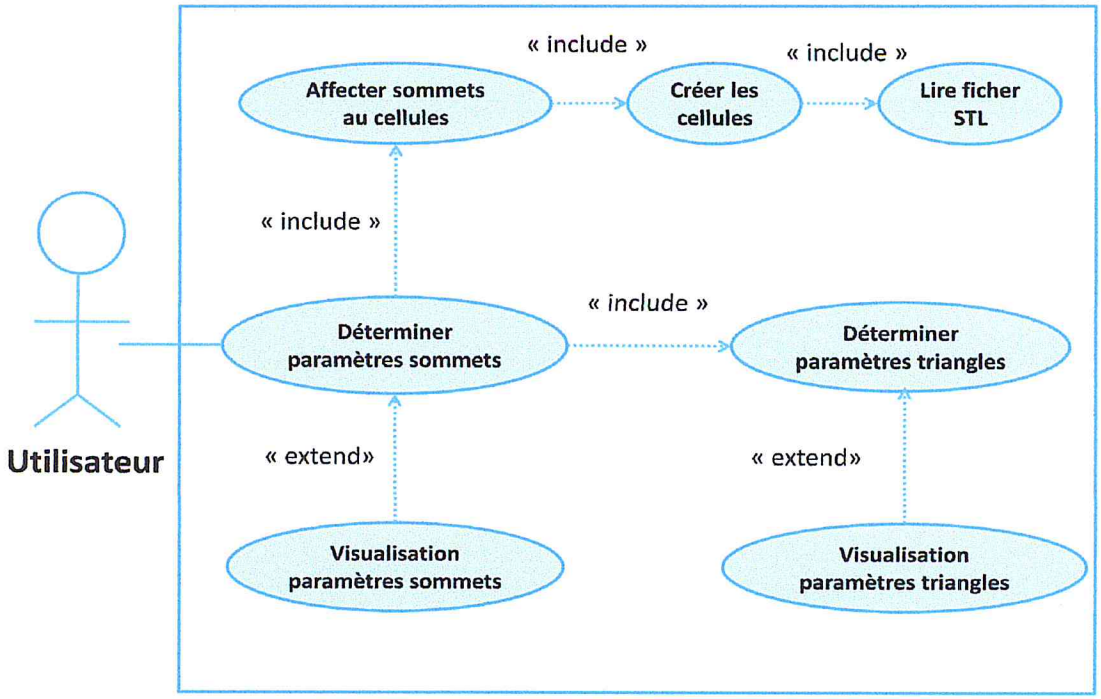


Figure II.22. Diagramme de cas d'utilisation « déterminer les paramètres des sommes et des triangles ».

2.1.3. Diagramme de cas d'utilisation « déterminer les zones visibles et les zones accessibles » :

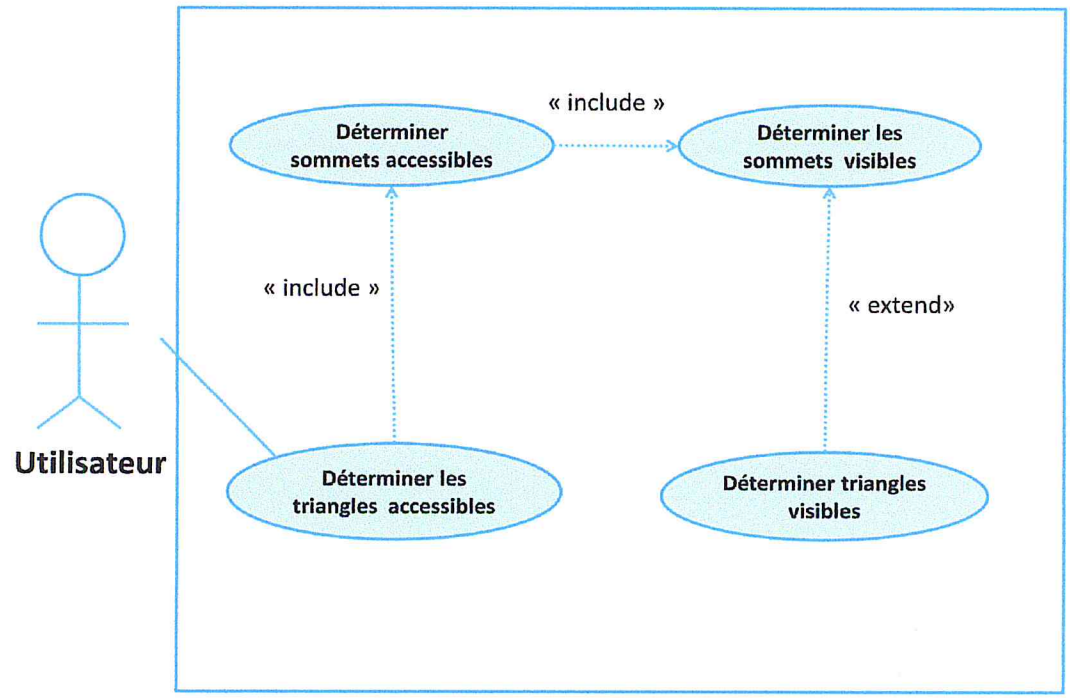


Figure II.23. Diagramme de cas utilisation « déterminer les zones visibles et les zones accessibles ».

2.1.4. Diagramme de cas d'utilisation« calculer les profondeurs de plongées en 03-axes » :

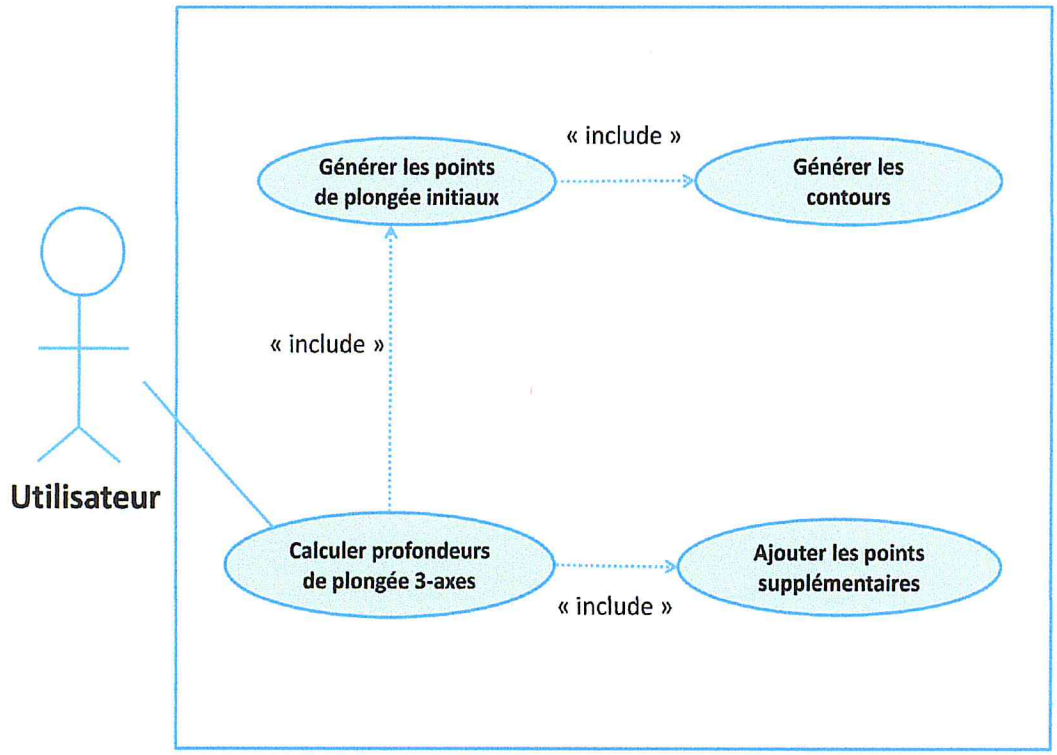


Figure II.24. Diagramme de cas utilisation« calculer les profondeurs de plongées en 03-axes ».

2.1.5. Diagramme de cas d'utilisation« calculer les profondeurs de plongées en 05-axes » :

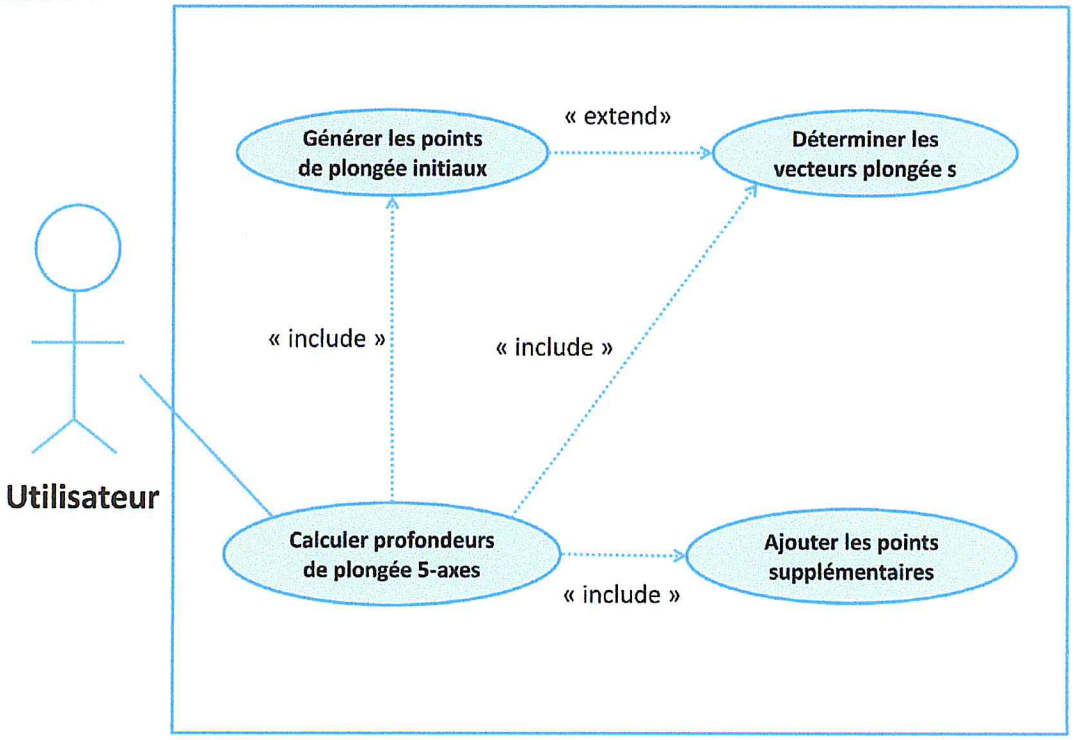


Figure II.25. Diagramme de cas utilisation« calculer les profondeurs de plongées en 05-axes ».

2.1.6. Diagramme de cas d'utilisation « Simuler le trajet d'usinage » :

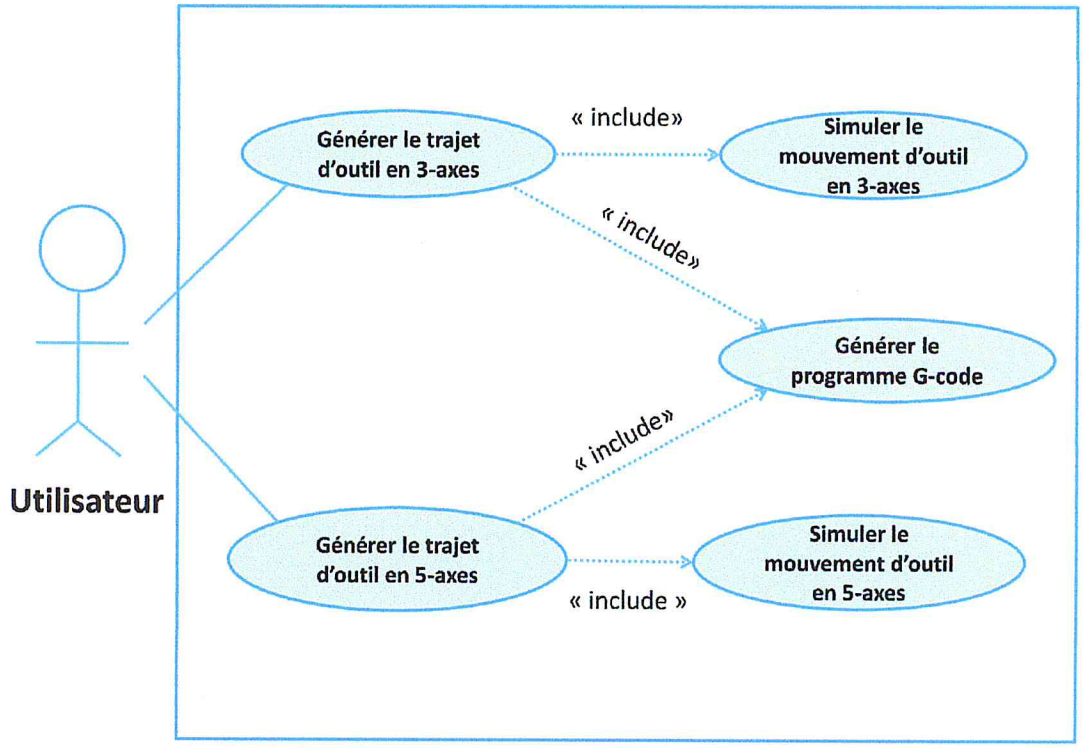


Figure II.26. Diagramme de cas utilisation« simuler le trajet d'usinage ».

2.2. Diagramme de classes :

C'est un diagramme qui permet de représenter toutes les classes d'un système ainsi que les différentes relations entre celles-ci par la définition de tous les objets et leurs composants. Dans notre conception, les différentes classes créées sont les suivantes :

1. Class COULEUR.
2. Class BRUT.
3. Class CELLULE.
4. Class POINT_COORD.
5. Class NORMALE.
6. Class SOMMET.
7. Class TRIANGLE_STL.
8. Position_plongée_stl.
9. Class contour_stl.
10. Class TRAJET_TREFLAGE.
11. Class simulation.
12. Class modèle_stl.

Les relations entre les différentes classes sont montrées dans le diagramme de classes suivant (figure 27).

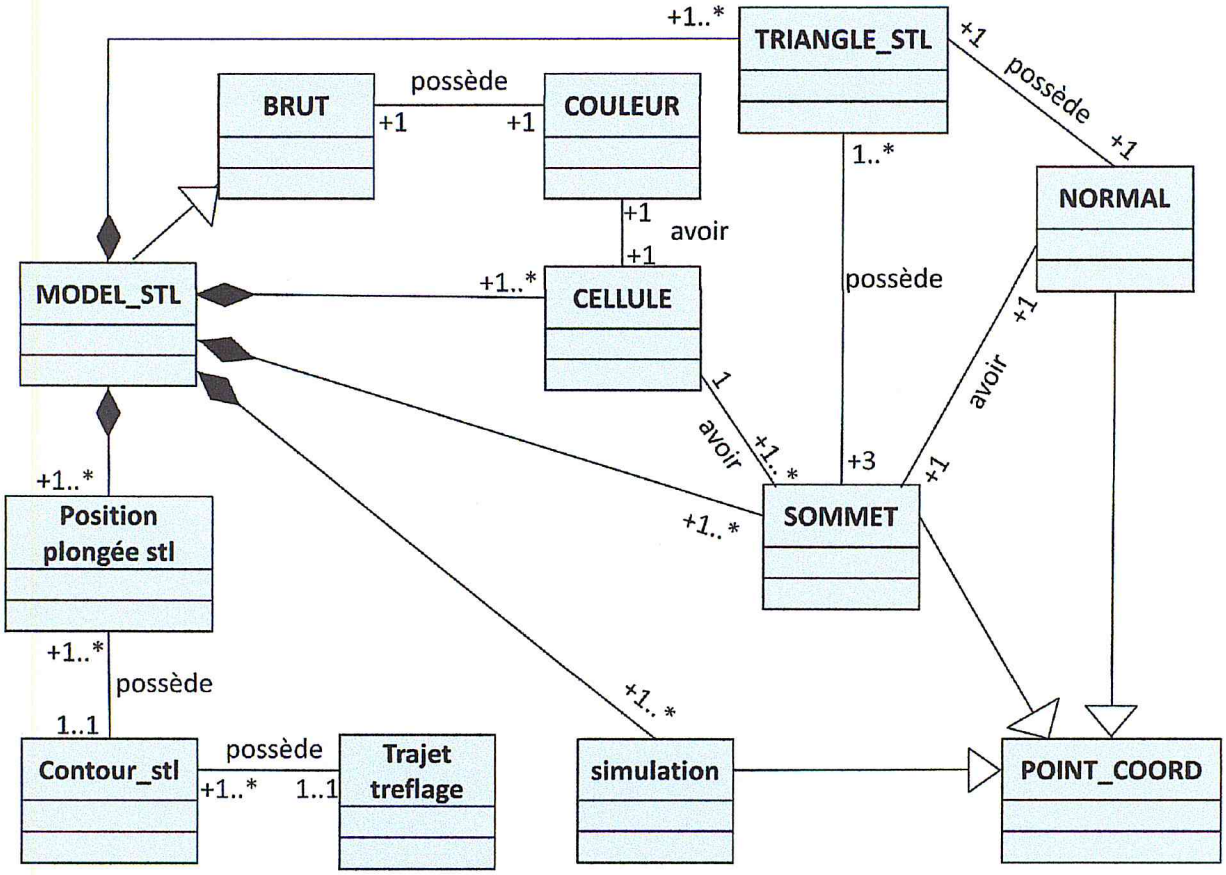


Figure II.27. Diagramme de classe.

Les détails des classes sont donnés dans les figures suivantes :

- **Classe « COULEUR » :** classe qui représente les couleurs des différents objets.

COULEUR
- R:double - B:double - G:double
+ setR(double):void + setB(double):void +setG(double):void +getR():double +getB():double +getG():double +affecter_couleur(): void

Figure II.28. Classe COULEUR.

- **Classe« BRUT » :** représente l'enveloppe des différents objets créés.

BRUT
- xmax:double - ymax:double - zmax:double

- xmin:double - ymin:double - zmin:double
+ set xmax (double):void + set ymax (double):void + set zmax (double):void + set xmin (double):void + set ymin (double):void + set zmin (double):void + getxmax():double + getymax ():double + getzmax ():double + getxmin():double + getymin ():double + getzmin ():double + getlargeur():double + getlongeur():double + gethauteur():double + dessiner_brut_filaire(): void + dessiner_brut_rendu(): void

Figure II.29. Classe BRUT.

- **Classe« CELLULE » :** classe qui définit une cellule par un vecteur d'indice de sommets (Figure II.28).

CELLULE
- indice_sommets_triangles :vector<int>
+ getindice_sommets_triangles() : vector<int> + getindice_sommets_triangles_element(int) : int + Ajouter_indice_sommets_triangles(int) : void + dessiner_points_une_cellule() :void

Figure II.30. Classe CELLULE.

- **Classe« POINT_COORD » :** représente les coordonnées d'un point.

POINT_COORD
-x:double -y:double -z:double
+setx(double):void +sety(double):void +setz(double):void +getx():double +gety():double +getz():double +dessiner_point():void

Figure II.31. Classe POINT_COORD.

- **Classe« NORMALE » :** c'est la classe qui définit la normale des sommets et des triangles.

NORMALE
+ dessiner_normale() : void

Figure II.32. Classe NORMALE.

- **Classe« SOMMET » :** c'est la classe qui définit les differentes carctéristiques d'un sommet (Figure II.29).

SOMMET
- indice_triangles : vector<int> - indice_sommets_voisins :vector<int> - vecteur_normal : NORMALE - vecteur_normal_aire : NORMALE - vecteur_normal_angle : NORMALE - vecteur_normal_moyenne : NORMALE - intindice_sommet; - num_sommet : int - visibilite : bool - visibilite_tech : bool - p: double
- getvisibilite_sommet() : bool - getvisibilite_sommet_tech() : bool - getp() : double - setp(double) : void - getindice_sommet() : int - setindice_sommet(int) : void - setnum_sommet(int) : void - getindice_triangles() : vector<int> - getelementindice_triangles(int) : int - Ajouter_indice_triangles(int) :void - Ajouter_indice_sommets_voisins(int) :void - getindice_sommets_voisins() : vector<int> - getelementindice_sommets_voisins(int) : int - getvecteur_normal() : NORMALE - setvecteur_normal(NORMALE) : void - getvecteur_normal_aire() : NORMALE - setvecteur_normal_aire(NORMALE) : void - getvecteur_normal_angle() : NORMALE - setvecteur_normal_angle(NORMALE) : void - getvecteur_normal_moyenne() : NORMALE - setvecteur_normal_moyenne(NORMALE) : void

- comparer (SOMMET) :bool - calculer_normale_ponderation_aire() : void - voidcalculer_normale_avec_moyenne() : void - calculer_normale_ponderation_angles() : void - creer_sommets_voisins() : void - chercher_indice_sommet(int) bool - D_plan_tangent() : double - visibilite_sommet() :void - visibilite_sommet_tech() :void - dessiner_sommet() : void - dessiner_sommet_voisins() : void - dessiner_normale() : void - dessiner_triangles_sommet() : void - dessiner_triangles_sommet_rendu() : void

Figure II.33. Classe SOMMET.

- **Classe« TRIANGLE_STL » :** c'est la classe qui définit les differentes carctéristiques d'un triangle.

TRIANGLE_STL
- indice_sommet_1 : int - indice_sommet_2 : int - indice_sommet_3 : int - num_triangle :int - vecteur_normal : NORMALE - aire_triangle : double - angle_sommet_1 : double - angle_sommet_2 : double - angle_sommet_3 : double - voisins_triangle :int - centre_triangle : POINT_COORD - visibilite_t : bool - visibilite_tech_t : bool - x_mint : double - y_mint : double - z_mint : double - x_maxt : double - y_maxt : double - z_maxt : double - d :double - tab_poin_sup : vector<SOMMET>
+ gettab_poin_sup() : vector <SOMMET> + getelementtab_poin_sup(int) : SOMMET + getvisibilite_t() :bool

+ getvisibilite_tech_triangle() :bool
+ setindice_sommet_1(int) :void
+ setindice_sommet_2(int) :void
+ setindice_sommet_3(int) :void
+ getindice_sommet_1() :int
+ getindice_sommet_2() :int
+ getindice_sommet_3() :int
+ getnum_triangle() : int
+ setnum_triangle(int) :void
+ getx_mint() : int
+ gety_mint() : int
+ getz_mint() : int
+ setx_mint(double) : void
+ sety_mint(double) : void
+ setz_mint(double) : void
+ getx_maxt() :int
+ gety_maxt() :int
+ getz_maxt() :int
+ setx_maxt(double) : void
+ sety_maxt(double) : void
+ setz_maxt(double) : void
+ setvecteur_normal(NORMALE) :void
+ getvecteur_normal() : NORMALE
+ getcentre_triangle() : POINT_COORD
+ setcentre_triangle(POINT_COORD) :void
+ getaire_triangle() :double
+ setaire_triangle (double) :void
+ getangle_sommet_1() :double
+ getangle_sommet_2() :double
+ getangle_sommet_3() :double
+ setangle_sommet_1 (double a) :void
+ setangle_sommet_2 (double a) :void
+ setangle_sommet_3 (double a) :void
+ setvoisins_triangle_v1(int a) :void
+ setvoisins_triangle_v2(int a) :void
+ setvoisins_triangle_v3(int a) :void
+ initialiser_voisins() :void
+ creer_triangles_voisins(int) : void
+ calculer_aire_triangle_angles_sommets() :void
+ calcul_centre_triangle() : void
+ récupérer_indice_sommet(vector<int>) :void
+ comparer_indice(int,int,int,int) :void
+ voidcalculer_d() :void
+ visibilite_triangle():void
+ voidcalculer_limite_triangle():void
+ voidvisibilite_techn_triangle():void
+ intersection_plan_droite(SOMMET, POINT_COORD,bool&,SOMMET,bool&) : void
+ apartenance_point_triangle(SOMMET) :bool

+ produit_vectoriel(double, double, double, double, double, double) : SOMMET + longeurr (double ,double, double) :double + produit_scalaire(X1, Y1, Z1, X2, Y2, Z2) :double + intersection_point_plongee_triangle(POINT_COORD ,bool&) : SOMMET + calculer_vecteur(SOMMET,SOMMET) : SOMMET + calculer_dalta_alpha (SOMMET ,SOMMET ,SOMMET) :double + calculer_dalta_beta (SOMMET ,SOMMET) :double + calculer_Dalta_globale (SOMMET ,SOMMET) :double + intersection_demi_droit_segment(SOMMET ,SOMMET,SOMMET) :bool + ajouter_point_sup() :void + voiddessiner_triangle_filaire():void + dessiner_triangle_rendu():void + dessiner_normale():void + dessiner_voisins_triangle():void + dessiner_voisins_triangle_rendu():void + dessiner_point_sup() :void

Figure II.34. Classe TRIANGLE_STL.

- **Classe« position_plongee_stl » :** c'est la classe qui définit les differentes carctéristiques d'uneposition de plongée.

position_plongee_stl
- point_plongee : POINT_COORD - point_extremite_droite : POINT_COORD - usinee : bool - pos_finale_plongee : SOMMET - direction : POINT_COORD - profondeur_plongee : double - u : POINT_COORD - rayon :double - D : double - axe_t1 : POINT_COORD - axe_t2 : POINT_COORD
+ setu(POINT_COORD): void +getu():POINT_COORD + getusinee():bool + getpos_finale_plongee():SOMMET + getpoint_plongee():POINT_COORD + setusinee(bool): void + setpoint_plongee(POINT_COORD):void + getaxe_t1():POINT_COORD + getaxe_t2():POINT_COORD + setaxe_t1(POINT_COORD):void + setaxe_t2(POINT_COORD):void + voidsetpos_finale_plongee(SOMMET) + calculer_prodondeur_plongee():void

+ determiner_usinage():void + calculer_t1_t2():void + calculer_D():void + determiner_point_plonge_finale_rot():void + dessiner_point_finale_rot():void + distance_points_axes(vector<SOMMET>): SOMMET + acces_point_plongee():void + dessiner_disque_point():void + Distance_point_axe(SOMMET): double + calculer_profondeur_5axes(SOMMET):double + dessiner_outil_cylindrique():void + dessiner_outil_cylindrique_plongee():void

Figure II.35. Classe position_plongee_stl.

- **Classe« contour_stl » :** c'est la classe qui définit les differentes carctéristiques d'un contour.

contour_stl
- num_con : int - p1 : POINT_COORD - p2 : POINT_COORD - p3 : POINT_COORD - p4 : POINT_COORD - tab_point_plonge : vector<position_plongee_stl>
+ getp1() : POINT_COORD + getp2() : POINT_COORD + getp3() : POINT_COORD + getp4() : POINT_COORD + setp1(POINT_COORD) : void + setp2(POINT_COORD) : void + setp3(POINT_COORD) : void + setp4(POINT_COORD) : void + getelement_tab_point_plonge(int) : position_plongee_stl + gettab_point_plonge() : vector <position_plongee_stl> + settab_point_plonge(vector<position_plongee_stl>) :void + setnum_contour(int xx) :void + dessiner_contour_decale() : void + calculer_point_plongee(): void + dessiner_point_plongee(): void + determiner_point_usinee(): void + dessiner_point_usinee(): void + determiner_profondeur_plongee(): void + dessiner_point_plonge_finale(): void + dessiner_disk_plongee_finale(): void + determiner_usinage_cont(): void

Figure II.36. Classe contour_stl.

➢ **Classe« TRAJET_TREFLAGE » :** elle représente les différentes informations de l'outil et le trajet de Tréflage.

TRAJET_TREFLAGE
- rayon_outil :double - longeur_outil:double - pas_radial:double - tab_contour_stl :vector<contour_stl> - engagement_radial : double - distance_engagement : double - centre_rayon : POINT_COORD
+ getrayon_outil() :double + getlongeur_outil():double + getpas_radial():double + setpas_radial(double) :void + double getengagement_radial() + setengagement_radial(double) :void + setrayon_outil(double) :void + setlongeur_outil(double) :void + getelement_tab_contour_stl(int) : contour_stl + gettab_contour_stl() : vector<contour_stl> + setelement_tab_contour_stl(int , contour_stl) :void + calculer_contour_stl() :void + dessiner_contours_decales() :void + point_contour(POINT_COORD,POINT_COORD,POINT_COORD):POINT_COORD + dessiner_points_plongees() :void + dessiner_points_usinees() : void + dessiner_points_plongee_finales() : void + voiddessiner_disk_plongees_finales() :void + voiddeterminer_usinage_traj():void + voiddessiner_point_usinee_traj();:void + dessiner_points_finale_plonges_rot():void

Figure II.37. Classe TRAJET_TREFLAGE.

➢ **Classe « simulation » :**

simulation
- t1 :POINT_COORD - t2 :POINT_COORD - u :POINT_COORD - mode_deplacement :bool - rayon : double - Longeur_partie_active; double - Longeur : double

+ gett1():POINT_COORD + gett2():POINT_COORD + getu() : POINT_COORD + sett1(POINT_COORD): void + sett2(POINT_COORD): void + setu(POINT_COORD): void + getmode_deplacement():bool + setmode_deplacement(bool): void +dessiner_outilLL_cylindrique_simulation():void

Figure II.38. Class simulation

- **Classe« modele_stl » :**c'est la classe principale qui regroupe tous les sommets et triangles du fichier STL et les cellules ainsi que les points de la simulation.

modele_stl
- tab_point_visible_tech : vector<SOMMET> - tab_triangles : vector<TRIANGLE_STL> - tab_triangles_visibles : vector<TRIANGLE_STL> - tab_points_sphere : vector<position_plongee_stl> - tab_sommets_triangles : vector< SOMMET> - tab_cellules_points :CELLULE *** - nre_cellule_x : unsignedint - nre_cellule_y : unsignedint - nre_cellule_z : unsignedint - pasx :double - pasy:double - pasz:double - aire_min:double - aire_max:double - centre : POINT_COORD - rayon_sphere:double - pas_teta:double - pas_delta:double
+ double getpas_teta() :double + getpas_delta():double + setpas_teta(double) :void + setpas_delta(double) :void + getcentre() : POINT_COORD + getrayon_sphere() :double + setcentre(POINT_COORD) : void + setrayon_sphere(double) : void + getelement_tab_sommets_triangles(int) : SOMMET + setelement_tab_sommets_triangles(SOMMET, int) :void + gettab_sommets_triangles() : vector< SOMMET> + gettab_point_visible_tech() : vector< SOMMET> + getelement_tab_points_sphere(int) : position_plongee_stl

+ setelement_tab_points_sphere(position_plongee_stl, int) : void
+ gettab_points_sphere() : vector <position_plongee_stl>
+ gettab_triangles() : vector <TRIANGLE_STL>
+ getelement_tab_triangles(int) : TRIANGLE_STL
+ getelement_tab_point_visible_tech(int) : SOMMET
+ gettab_triangles_visibles() : vector <TRIANGLE_STL>
+ getelement_tab_triangles_visibles(int) : TRIANGLE_STL
+ getindice_triangle (int) :int
+ getelement_tab_cellules_points(int, int, int) : CELLULE
+ getpasx() :double
+ getpasy() :double
+ getpasz() :double
+ getnre_cellule_x() :int
+ setnre_cellule_x(int) :void
+ getnre_cellule_y() : int
+ setnre_cellule_y(int) :void
+ intgetnre_cellule_z() :int
+ setnre_cellule_z(int) :void
+ verifier_extension_fichier_stl(AnsiString) :bool
+ calculer_limites_brut_verifier_syntaxe(AnsiString, bool&) : void
+ creer_cellule(): void
+ affecter_indices_sommets_aux_triangles(AnsiString) : void
+ verifier_sommet_double(SOMMET,int,int,int,int&) : int
+ recuperer_indice_sommet(SOMMET, int&) : int
+ creer_sommet_voisins() :void
+ determiner_voisins_sommets() :void
+ determiner_voisins_triangles():void
+ calculer_normales_sommets_triangles(): void
+ verifier_nbr_point():void
+ voiddeterminer_visibilite_sommets() :void
+ determiner_visibilite_triangles():void
+ determiner_visibilite_sommets_tech():void
+ determiner_limites_triangles():void
+ determiner_visibilite_tech_triangles():void
+ ajouter_point_sup_triangles():void
+ choisi_aire():void
+ determiner_profondeur_final():void
+ calculer_centre_demi_sphere():void
+ determiner_point_sphere():void
+ dessiner_points_sphere() :void
+ dessiner_vecteurs_plongee():void
+ dessiner_point_global() :void
+ dessiner_points_cellule():void
+ dessiner_triangles_rendus():void
+ dessiner_Triangles_filaires():void
+ dessiner_cellules_filaires():void
+ dessiner_cellules_rendues():void
+ dessiner_normales_triangles():void

+ dessiner_normales_sommets():void
+ dessiner_sommets():void
+ dessiner_sommet_visible():void
+ dessiner_triangle_visible():void
+ dessiner_triangle_visible_rendu():void
+ dessiner_sommet_invisible():void
+ dessiner_triangle_invisible():void
+ dessiner_triangle_invisible_rendu():void
+ dessiner_sommets_visibles_tech():void
+ dessiner_sommets_invisibles_tech():void
+ dessiner_triangle_visible_tech():void
+ dessiner_triangle_visible_tech_rendu():void
+ dessiner_triangle_invisible_tech():void
+ dessiner_triangle_invisible_tech_rendu():void
+ dessiner_points_sups():void
+ dessiner_point_plonge_finale_rot():void
+ determiner_point_plonge_finale_rot():void
+ dessiner_sphere():void
+ dessiner_point_sphere():void
+ determiner_vecteurs_plonges_finale_rot():void
+ dessiner_vecteur_plongee() : void
+ calculer_D_toutpoints(): void
+ calculer_t1_t2_toutpoints(): void
+ determiner_points_plonges_finale_rot(): void
+ modele_stl::dessiner_outlis_cylindriques(): void
+ modele_stl::dessiner_outlis_cylindriques_plongee(): void

Figure II.39. Classe modele_stl.

Conclusion :

Dans ce chapitre, nous avons présenté l'architecture générale de notre application d'un point de vue conceptuel, où nous avons recensé nos besoins en spécifiant toutes les étapes suivies et toutes les solutions proposées.

Dans le chapitre suivant, l'implémentation et la mise en œuvre de notre application seront présentées et la validation de l'application sera prouvée à travers deux exemples.

Chapitre 3

Implémentation et Validation

Introduction :

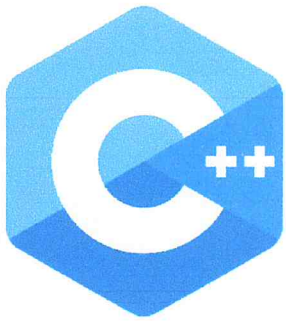
Après avoir exprimé les objectifs de notre travail ainsi que les solutions proposées en utilisant le modèle orienté objet et le langage de modélisation UML, ce chapitre a pour objectif de donner une vue assez complète du travail réalisé en présentant les informations manipulées à travers les sessions d'utilisation développées.

La première partie du chapitre, est réservé à la présentation de notre application logicielle développée en utilisant le langage de programmation Builder C++ et la bibliothèque graphique OpenGL (Open Graphics Library), en mettant en évidence son interface, ainsi que ses différentes fenêtres. La deuxième partie est réservée à la validation de l'approche proposée et développée.

1. Présentation des langages utilisés :

Comme notre système est appelé à être intégré dans l'environnement de production des surfaces complexes développé par l'équipe « CFAO » du « CDTA », les outils de développement utilisés lors de la mise en œuvre de notre application (C++, OpenGL) sont les mêmes que ceux utilisés par l'équipe « CFAO » afin de se conformer à la tendance qui veut que la majorité des systèmes de « CFAO » sont développés en utilisant le C++ et OpenGL.

1.1. Présentation du langage C++ :



Le langage C++, inventé par Bjarne Stroustrup vers 1983, est une évolution orientée objets du langage C de Brian Kernighan et Denis Ritchie. Il s'est enrichi, au cours de la décennie 1980, parallèlement à la stabilisation et la normalisation de C. Ce langage repose sur les mêmes mécanismes d'écriture et de génération. Il apporte notamment la gestion des exceptions, la gestion des références, la surcharge des opérateurs et les Templates, …etc. Enfin, une rétrocompatibilité a été gardée où les programmes en C sont compilés sans difficulté avec un compilateur C++.

Comme tout langage, C++ dispose d'une bibliothèque standard, c'est-à-dire de fonctions et de classes prédéfinies. Elle comporte notamment de nombreux patrons de classes et de fonctions permettant de mettre en œuvre les structures de données les plus importantes (vecteurs dynamiques, listes chaînées, chaînes, …etc.) et les algorithmes les plus usuels.

Parmi les environnements de développement AnjutaDevStudio, C++ Builder, Code :: Blocks (open-source), Dev-C++, Eclipse (open-source), Microsoft Visual C++, …etc.

1.2. Présentation d'OpenGL :

OpenGL (OpenGraphicsLibrary) est un ensemble normalisé de fonctionsde calcul d'images 2Dou 3Dlancé par SiliconGraphicsen 1992. Cette interface de programmationest disponible sur de nombreuses plates-formes. Elle est utilisée pour des applications allant du jeu vidéo jusqu'à la CAO en passant par la modélisation. OpenGL permet à un programme de déclarer la géométrie d'objets sous forme de points, de vecteurs, de polygones et de textures. OpenGL effectue ensuite des calculs de projection en vue de déterminer l'image à l'écran, en tenant compte de la distance, de l'orientation, des ombres, de la transparence et du cadrage. L'interface regroupe environ 250 fonctions différentes qui peuvent être utilisées pour afficher des scènes tridimensionnelles complexes à partir de simples primitives géométriques. Du fait de son ouverture, de sa souplesse d'utilisation et de sa disponibilité sur toutes les plates-formes, elle est utilisée par la majorité des applications scientifiques, industrielles ou artistiques 3D et certaines applications 2D vectorielles. Cette bibliothèque est également utilisée dans l'industrie du jeu vidéo où elle est souvent en rivalité avec la bibliothèque de MicrosoftDirect3D.

1.3. Présentation d'Embarcadero C++ Builder 10 Seattle [23] :

C++ Builder est un logiciel de développement rapide d'applications « RAD » conçu par Borland qui reprend les mêmes concepts, la même interface et la même bibliothèque que Delphien utilisant le langage C++. Il permet de créer rapidement des applicationsWin32 et Win64 ainsi qu'une interface graphique avec son éditeur de ressources. Il est compatible avec la version de norme ISO C++ de 2011.**Embarcadero C ++ Builder 10Seattle** est le moyen le plus rapide de créer et de mettre à jour des applications riches en données, hyper connectées et visuellement engageantes pour Windows 10, Mac, Mobile, IoT et plus encore en utilisant le standard C ++.

2. Présentation de l'application :

2.1. Fenêtre principale :

La fenêtre principale de la plate-forme logicielle développée par l'équipe « CFAO » est composée de deux parties (Figure III.1) :

- **Partie d'affichage :** cette partie est réservée à la visualisation de tous les objets géométriques en 3D (points, surfaces, outils, machine, simulation, trajet d'usinage, …etc.) en

utilisant la bibliothèque graphique « OpenGL ». Dans cette partie, les objets visualisés peuvent subir diverses transformations géométriques telles que translations, rotations, projections, zoom et changement d'échelles.

➢ **Partie de manipulation :** cette partie est composée d'une barre de menu et d'un ensemble de boutons de manipulation des différents paramètres des modèles CAO, des courbes et des surfaces (création d'une surface, suppression d'une surface, modification des points de contrôle, …etc.). Dans cette partie sont lancées toutes les fenêtres intégrées à la plate-forme logicielle.

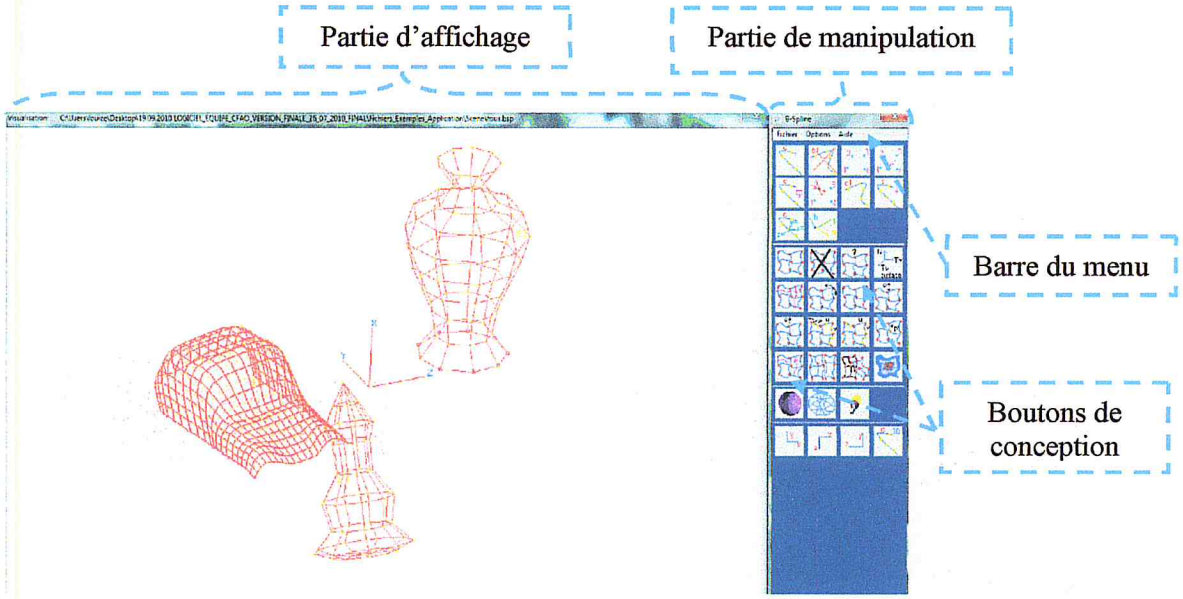


Figure III.1. Fenêtre principale.

2.2. Barre du menu principal :

La barre du menu principal est composée de trois rubriques :

- Rubrique « Fichier » : comporte toutes les fonctionnalités de manipulation des fichiers (ouverture, création, sauvegarde, …etc.).
- Rubrique « Option » : permet la modification des différents paramètres des courbes et des surface ainsi que le lancement des différentes opérations d'usinage. Les fenêtres que nous avons créées sont lancées à partir de cette rubrique.
- Rubrique « Aide » : pour l'affichage de l'aide.

2.3. Rubrique d'ébauchage par Tréflage à partir d'un fichier STL :

La partie contenant les modules développés est lancée à partir du menu au niveau de la rubrique Option, Usinage 5-axes, Tréflage 5-axes (Figure III.2). Le lancement et l'exécution des différentes fonctions de Tréflage 5-axes passe par l'ouverture d'un fichier STL contenant les modèles des surfaces à usiner.

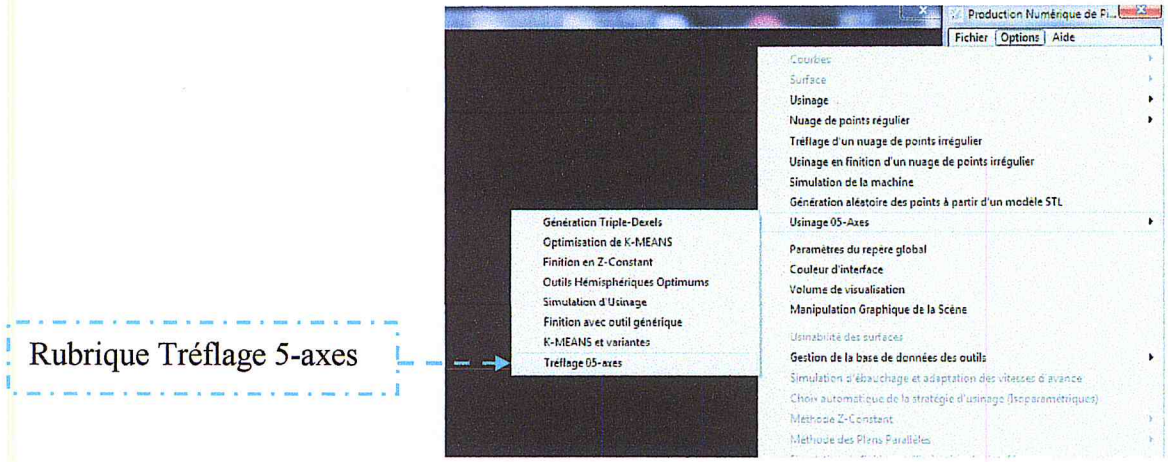


Figure III.2. Rubrique « Tréflage 5-axes ».

2.4. Présentation des onglets :

Le module logiciel développé est composé de 8 onglets (Figure III.3) pour lancer les fonctionnalités intégrées à la plate-forme logicielle.

Figure III.3. Onglets de l'application développée.

La représentation détaillée de chaque onglet est décrite dans les paragraphes suivants.

2.4.1. Visualisation :

Le premier onglet «VISUALISATION » (Figure III.4) permet la lecture du fichier STL contenant l'ensemble des sommets et des triangles de la surface. Avec un simple clic sur le bouton « Lire fichier STL » et sélection du fichier à ouvrir, les limites de brut sont calculées.

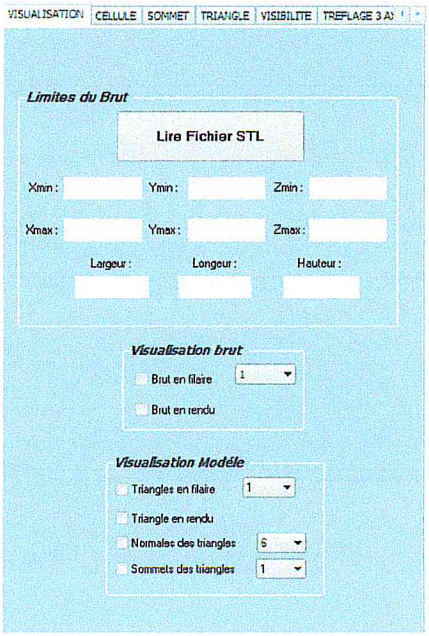


Figure III.4. Onglet « VISUALISATION ».

Une fois le fichier lu, les dimensions minimales du brut sont affichées. En plus de ces informations, il est possible de visualiser :

- Le brut du modèle en deux modes filaire et rendu.
- Les triangles du modèle en deux modes filaire et rendu.
- Les normales des triangles du modèle.
- Les sommets des triangles du modèle.

2.4.2. Création des cellules :

Pour restructurer le modèle STL, l'utilisateur doit effectuer deux étapes importantes (Figure III.5). Un clic sur le bouton «Créer cellules » permet la création des cellules par la spécification de leurs nombres suivant les trois axes X, Y et Z. Une fois les cellules créées et en cliquant sur le bouton « Affecter sommets », les sommets sont affectés aux cellules.

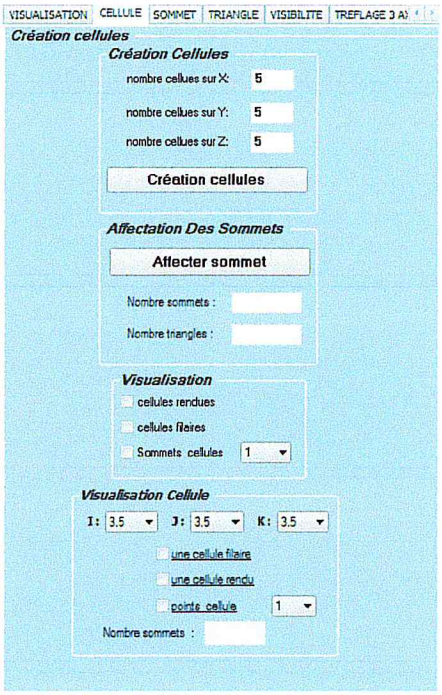


Figure III.5. Onglet « CELLULE».

Après la réalisation des étapes de cet onglet, les options suivantes sont disponibles :

- Récupérer le nombre de sommets.
- Récupérer le nombre de triangles.
- Visualiser les cellules en deux modes filaire et rendu.
- Visualiser les sommets des cellules.
- Visualiser le paramètre d'une cellule sélectionnée.

2.4.3. Paramètres des sommets :

Dans cet onglet, les paramètres des sommets sont calculés (Figure III.6). Le premier paramètre est la normale. Celle-ci est estimée selon trois modes de pondération.

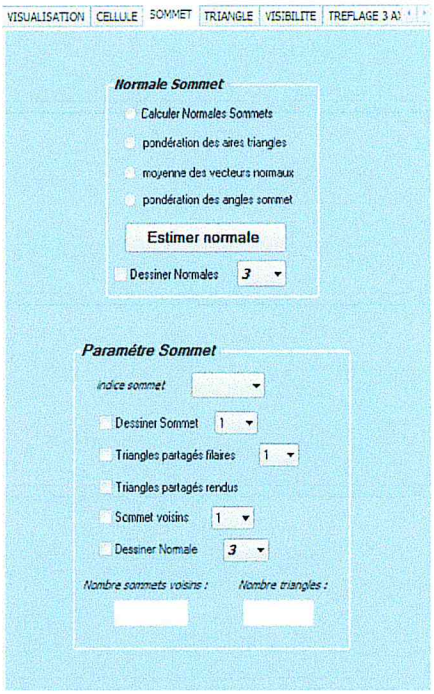


Figure III.6. Onglet « SOMMET».

Une fois les normales sont estimées, le plan tangent est déterminé. Par la suite, les voisins de chaque sommet sont identifiés. A ce niveau, il est possible de visualiser :

- Les normales des sommets des triangles.
- Les différents paramètres d'un sommet sélectionné.

2.4.4. Paramètres des triangles :

Dans cet onglet, les voisins de chaque triangle sont calculés à partir de ses sommets (Figure III.7). A ce stade, les paramètres d'un triangle sélectionné peuvent être visualisés :

- Triangle sélectionné et sa normale.
- Voisins d'un triangle en deux modes filaire et rendu.

La dernière étape consiste à enrichir le modèle STL par l'ajout d'un ensemble de points pour chaque triangle. Pour cela, l'utilisateur fixe le nombre de points à ajouter en fonction des aires minimales et maximales des triangles par un clic sur le bouton « Déterminer points supplémentaires ». Par la suite, il est possible de visualiser :

- Les points supplémentaires ajoutés.
- Les triangles enrichis et non enrichis par l'ajout de points.

2.4.5. Visibilité et accessibilité :

Cet onglet est devisé en deux parties (Figure III.8). La première partie sert à déterminer la visibilité de chaque sommet suivant un axe de visibilité donné. Une fois les sommets visibles sont traités, il est possible de visualiser :

- Les sommets visibles et invisibles.
- Les triangles visibles et invisibles.

La deuxième partie sert à déterminer l'accessibilité de chaque sommet suivant un axe de visibilité donné. Une fois les sommets accessibles sont traités, il est possible de déterminer l'accessibilité de chaque triangle. Après les calculs, nous pouvons visualiser :

- Les sommets accessibles et inaccessibles.
- Les triangles accessibles et inaccessibles.

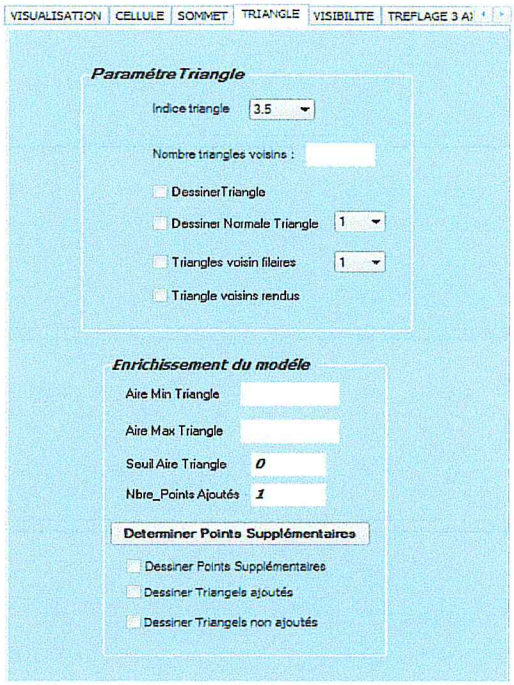


Figure III.7. Onglet « TRIANGLE ».

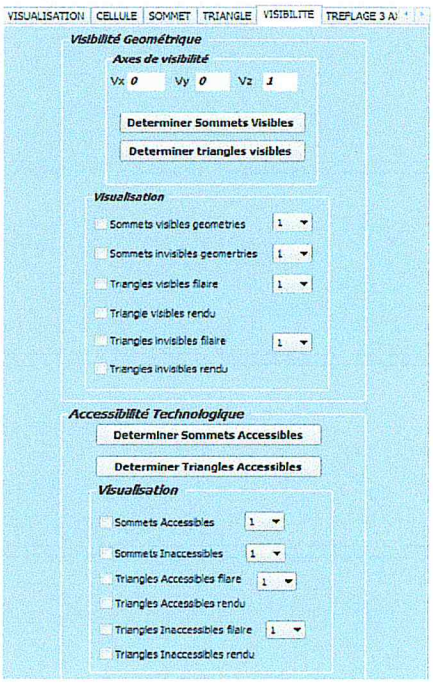


Figure III.8. Onglet « VISIBILITE ».

2.4.6. Tréflage 03-axes :

Cet onglet est subdivisé en trois étapes (Figure III.9). La première étape consiste à déterminer les contours selon une distance d'engament et un engagement radial introduits par l'utilisateur. La deuxième étape sert à créer et à visualiser les points de plongées initiaux à partir d'un pas radial spécifié. Dans la troisième étape, l'utilisateur introduit le rayon d'outil, la surépaisseur et la vitesse d'usinage à utiliser, par la suite, il détermine les points de plongées finaux par clic sur «Déterminer points plongées finales ». Une fois ces traitements sont terminés, l'utilisateur peut visualiser les informations suivantes :

- Positions de plongées et points de plongées finaux.
- Disques de plongées finaux.
- Points de plongées finaux valides.
- Les vecteurs de plongée.

Pour un numéro de point de plongée final sélectionné, l'utilisateur peut visualiser :

- Position de plongée.
- Point de plongée finale.
- Disque de plongée et disque de plongée finaux.
- Vecteur de plongée.

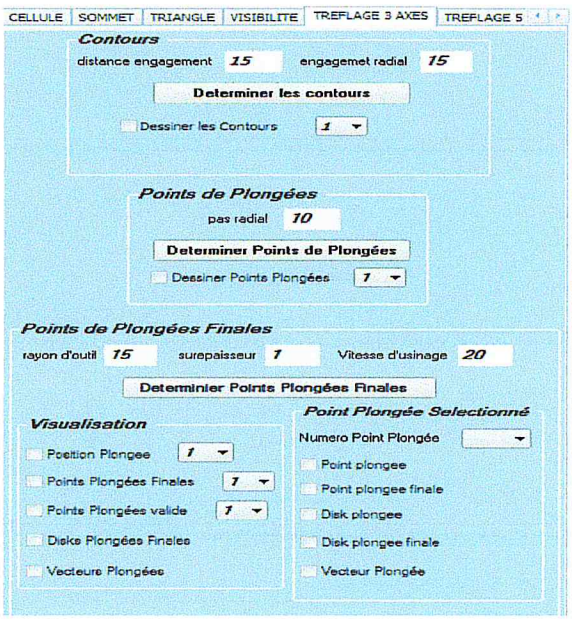


Figure III.9. Onglet « TREFLAGE 03-AXES ».

2.4.7. Tréflage 05-axes :

Cet onglet est divisé en trois parties (Figure III.10).La première partie permet à l'utilisateur de créer un point de plongée initial rotatif en introduisant ses composantes (vx, vy, vz) mettant en évidence les limites maximales du brut affichées. Par la suite, l'utilisateur

clic sur « calculer axes_T1 axes_T2 » et « déterminer point plongée finale rotative » pour calculer le point de plongée finale rotative. À ce stade, on peut visualiser :

- Disque plongée initial.
- Droite de plongée.
- Point de plongée finale rotative.
- Disque de plongée final.

La deuxième partie consiste à créer la stratégie demi sphérique par la spécification du pas Téta, pas Delta, le rayon d'outil, la surépaisseur et la vitesse d'usinage et par un clic sur «Déterminer points demi sphère ». À ce stade il est possible de visualiser :

- La sphère.
- Les points de la demi-sphère.

La troisième partie est réservée aux points de plongées finaux rotatives. L'utilisateur clic sur les boutons « Déterminer les vecteurs », « Calculer les axes_t1 axes_t2 » et « Déterminer points plongées finales rotatifs ». Une fois les calculs terminés, l'utilisateur peut visualiser :

- Disques initiaux.
- Disques plongées finaux.
- Vecteurs de plongées.
- Points de plongées finaux rotatives.

Pour un numéro de point de plongée final sélectionné, l'utilisateur peut visualiser :

- Point plongée initial, point position plongée et le point usinage.
- Disque initial, disque plongée et le disque usinage.
- Vecteur de plongée.
- Outil plongée initial, outil position plongée, outil points plongée finale.
- Axe d'outil, axe T1 et axe T2.

2.4.8. Simulation :

Le dernier onglet a comme but de simuler le travail réalisé en Tréflage 03-axes et en Tréflage 05-axes (Figure III.11).Il est donc possible de choisir parmi les trois simulations 03-axes, 05-axes et 03 et 05-axes. D'autres options sont intégrées à cet onglet tel que l'arrêt, la pause, le recul ou l'avance du point de départ de la simulation. Nous pouvons visualiser :

- Outil ou il est positionné.
- Position de la simulation.
- Trajet de Tréflage.
- Axes de rotations (outil, axe T_1, axe T_2).

Lors de l'exécution de la simulation, les informations suivantes sont affichées :

- Coordonnées du point (x,y,z).
- Coordonnées de l'axe T1 (x,y,z).
- Coordonnées de l'axe T2 (x,y,z).
- Coordonnées du vecteur (x,y,z).
- Vitesse d'usinage.

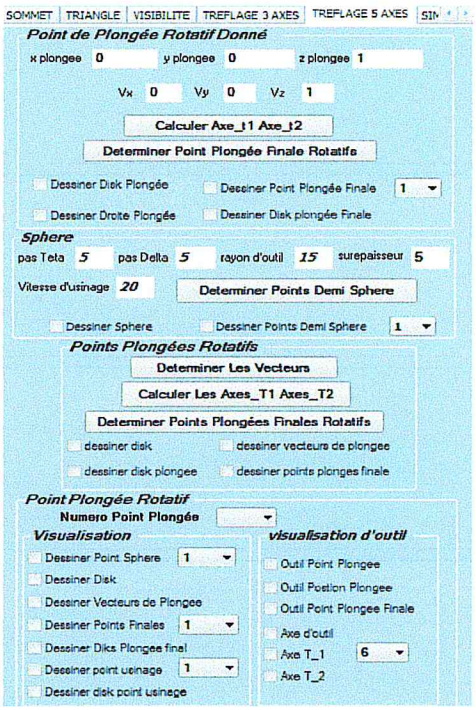


Figure III.10. Onglet « TREFLAGE 05-AXES».

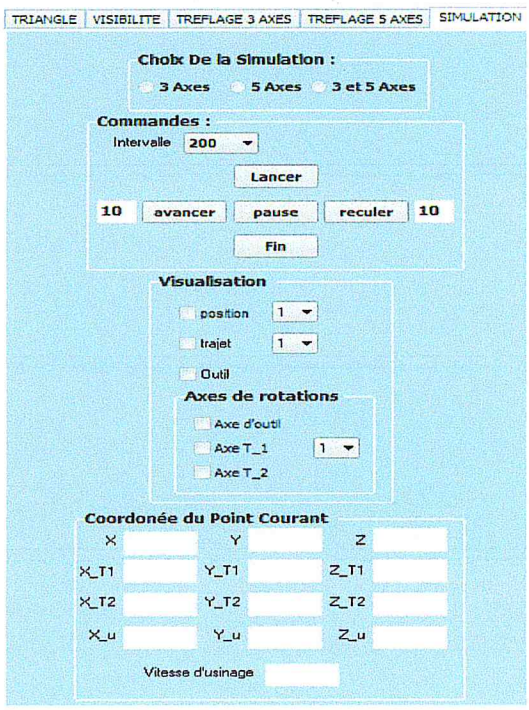


Figure III.11. Onglet « SIMULATION ».

3. Test et validation :

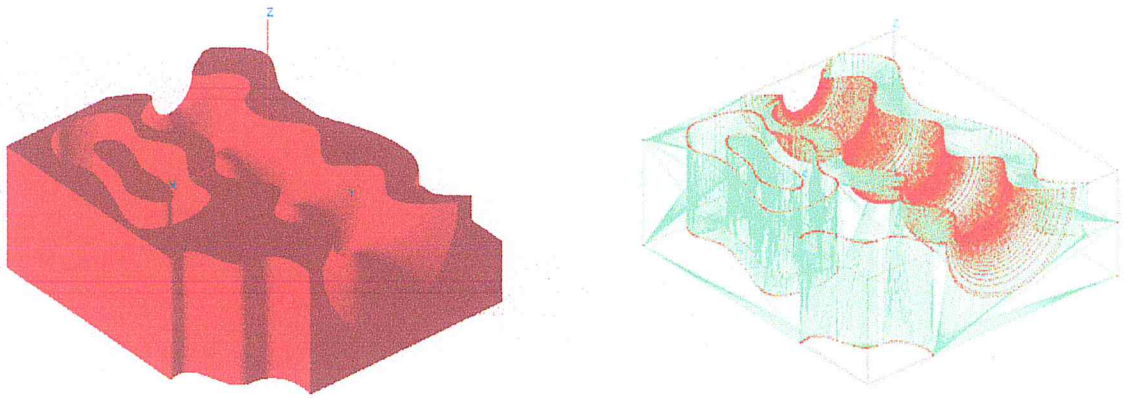
La validation de notre travail est menée sur deux modèles STL générés à partir de leurs modèles CAO. Le premier modèle vise à montrer les différentes étapes de validation de notre approche et le Tréflage en 03-axes. Par contre, le deuxième exemple est composé d'un ensemble de surfaces très complexes adapté pour le Tréflage en 05-axes.

3.1. Premier modèle STL :

L'exemple considéré est représenté par la Figure III.12. Les résultats des différentes étapes présentés dans les paragraphes suivants sont relatifs à ce modèle STL.

- **Etape 1 :** lecture du fichier STL et calcul des limites du brut. Les résultats sont :

Xmin =0.000 mm	Ymin = 0.000mm	Zmin = 0.000mm
Xmax =140.000 mm	Ymax = 150.000mm	Zmax = 50.000mm
Longueur = 140.000mm	Largeur = 150.000mm	Hauteur=50.000mm



a. Triangle en rendu. b. Sommets des triangles et le brut.

Figure III.12. Visualisation du premier modèle STL.

- **Etape 2 :** division du brut en cellules avec un nombre de 05 cellules suivant les trois axes X, Y et Z (Figure III.13). Après l'affectation des sommets aux cellules et aux triangles,les résultas obtenus sont les suivants :

Nombre total des sommets = 16231 Nombre total des triangles =30190

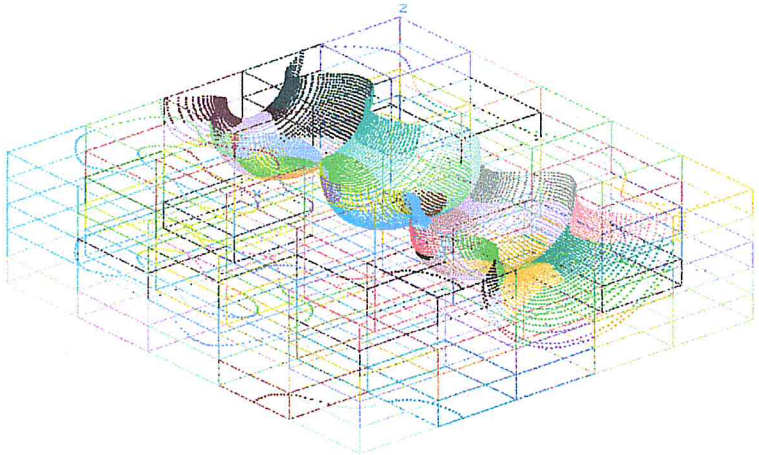


Figure III.13. Cellules en filaire avec les sommets affectés.

❖ **Etape 3 :** pour chaque sommet, le plan tangent et les voisins sont déterminés.La normale est estimée par le choix du mode de calcul « moyenne des vecteurs normaux ». La figure suivante montre les résultats obtenus (Figure.III.14).

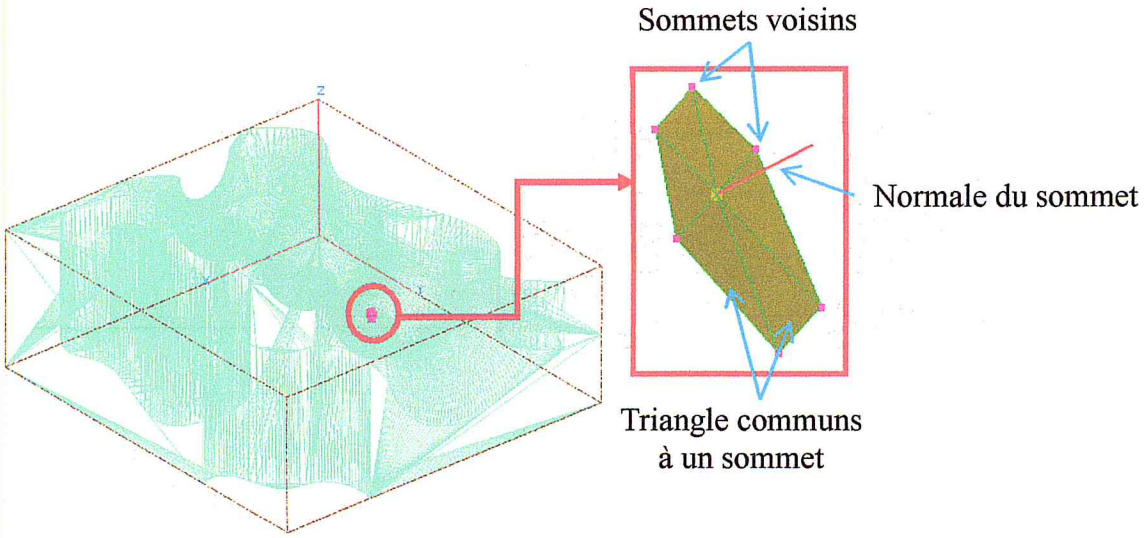


Figure III.14. Paramètres d'un sommet.

❖ **Etape 4** : pour chaque triangle, les voisins sont déterminés et affichées. La figure suivante montre les résultats obtenus (Figure III.15).

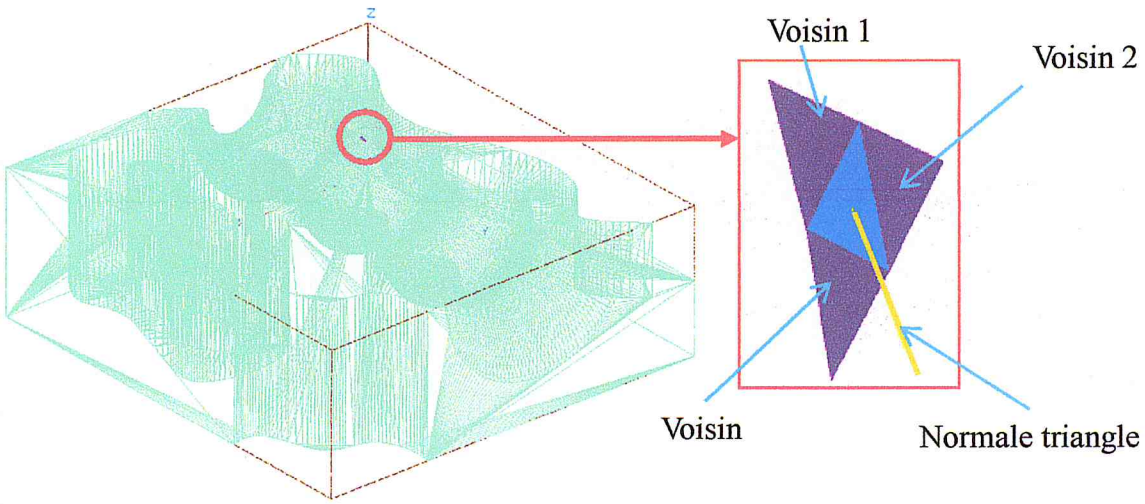


Figure III.15. Paramètres d'un triangle.

L'enrichissement du modèle STL par l'ajout de points supplémentaires. Pour ce modèle, l'aire maximale et minimale des triangles sont 10500mm^2 et 0,0001mm^2.Dans cet exemple, le seuil de l'aire des triangles est pris égal à 5mm^2 (Figure III.16) et les densités des points sont fixées à 1point/mm^2 et à 5points/mm^2. La Figure III.17 montre les résultats obtenus.

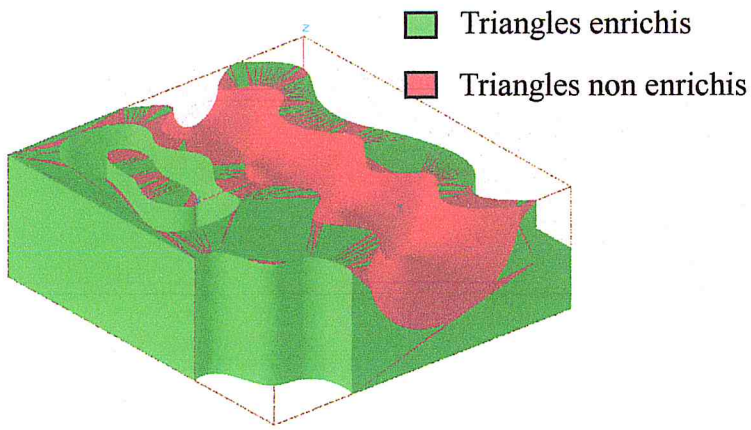
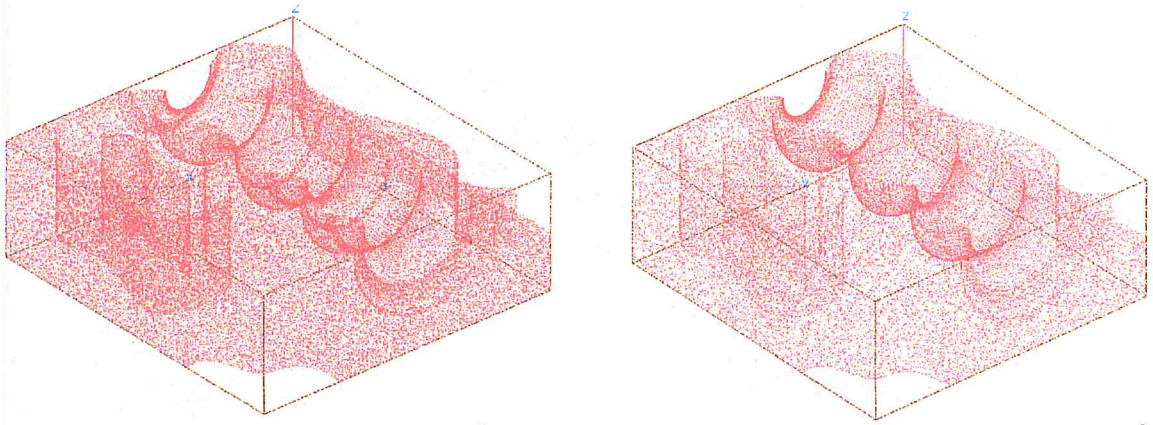


Figure III.16. Triangles à enrichir en rendu.



a. Ajout de cinq (05) points/mm^2. b. Ajout d'un (01) point/mm^2.

Figure III.17. Modèle enrichi.

❖ **Etape 5** : détermination de la visibilité des sommets (Figure III.18) suivie par celle des triangles selon trois axes de visibilité de composantes respectives (0,0,1) (Figure III.19), (0,1,0) (Figure III.20) et (1,0,0) (Figure III.21).

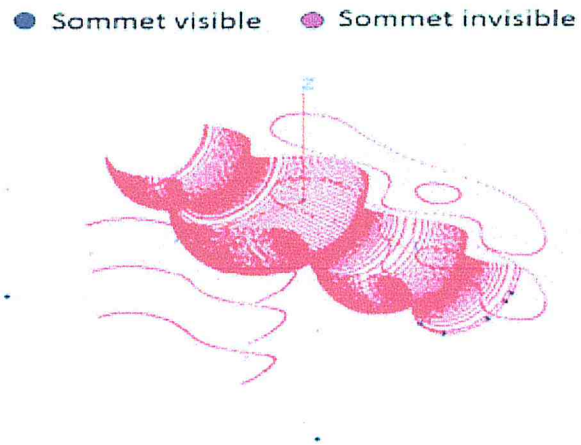
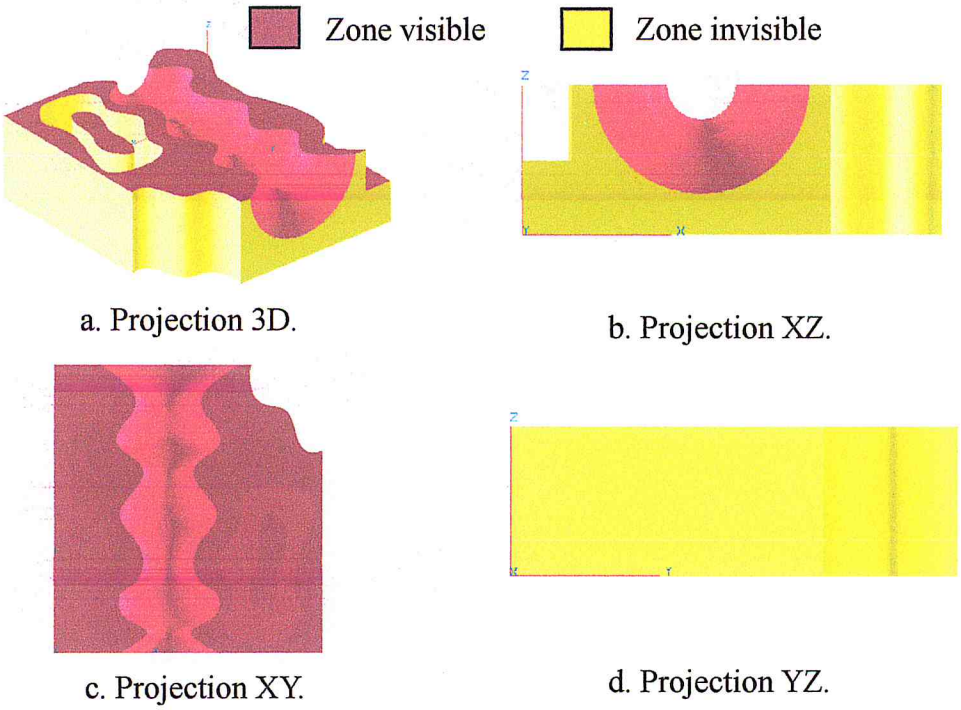


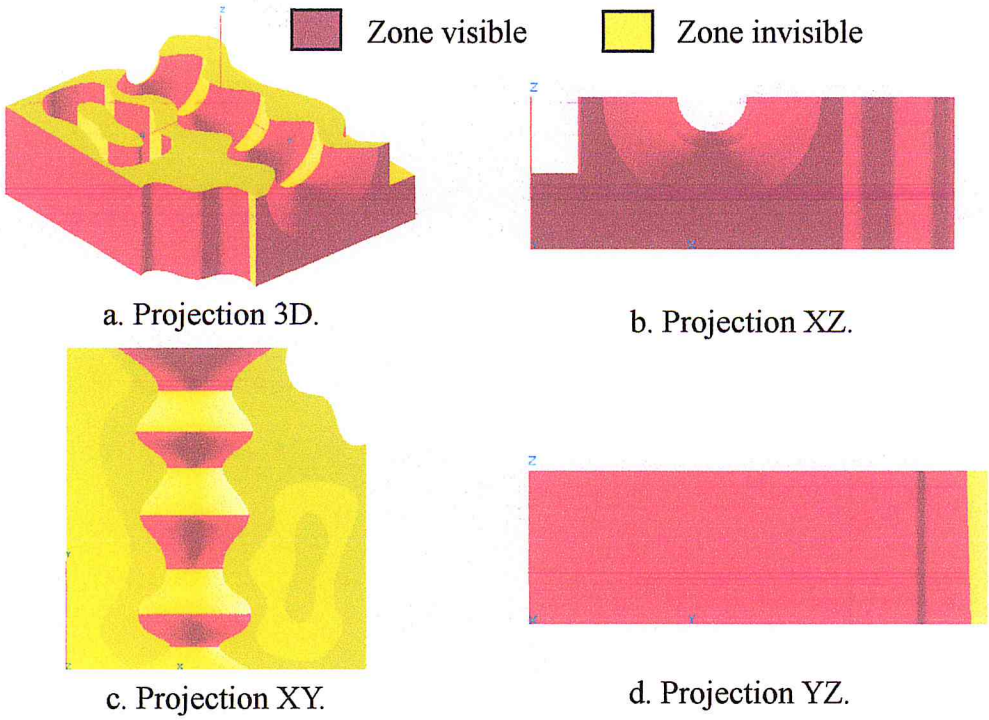
Figure III.18. Sommets visibles et invisibles (0,0,1).



a. Projection 3D. b. Projection XZ.

c. Projection XY. d. Projection YZ.

Figure III.19. Visibilité des triangles selon l'axe (0, 0, 1) en rendu.



a. Projection 3D. b. Projection XZ.

c. Projection XY. d. Projection YZ.

Figure III.20. Visibilité des triangles selon l'axe (0,1,0) en rendu.

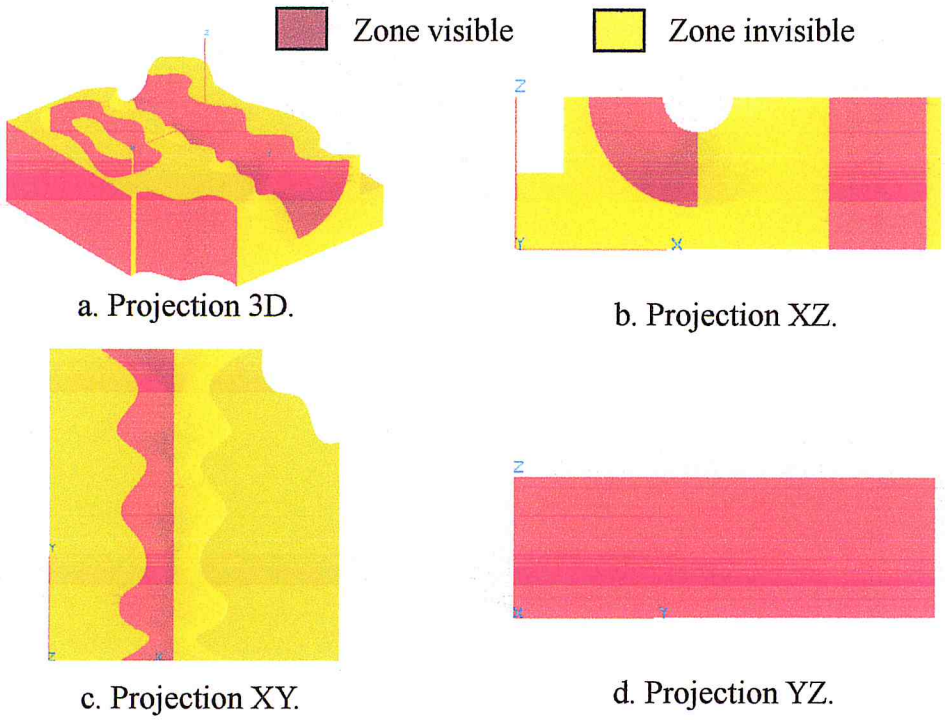


Figure III.21. Visibilité des triangles selon l'axe (1,0,0) en rendu.

Après avoir déterminé la visibilité pour chaque direction, nous passons à la détermination de l'accessibilité des sommets (Figure III.22) et des triangles pour les mêmes directions. La Figure III.23 illustre l'accessibilité pour l'axe de visibilité (0,0,1).

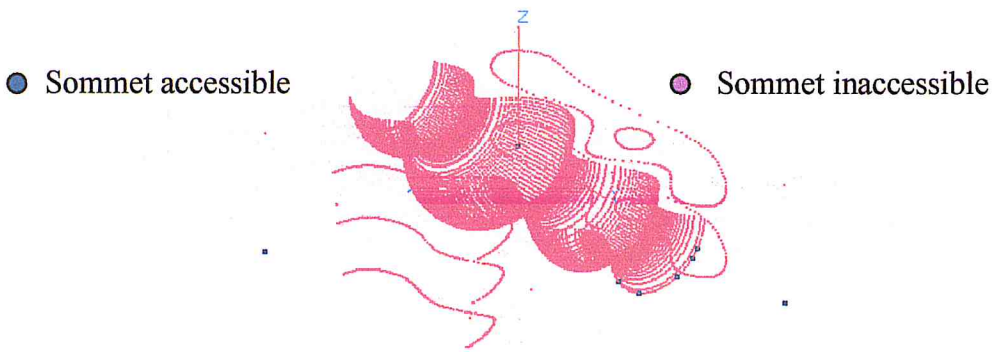
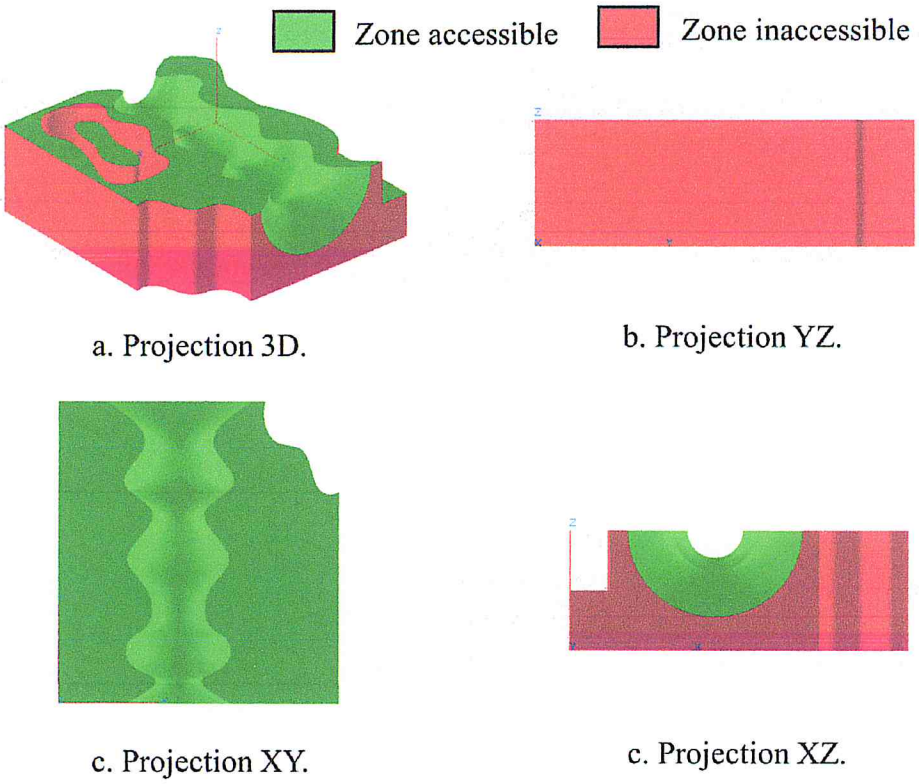


Figure III.22. Sommets accessibles.



a. Projection 3D. b. Projection YZ.

c. Projection XY. c. Projection XZ.

Figure III.23. Accessibilités des triangles.

❖ **Etape 6** : génération des contours décalés avec un engagement radial de 10mm (Figure III.24) et une distance d'engament de 20mm. Par la suite, les points de plongées initiaux sont créés à partir d'un pas radial de 3mm (Figure III.25).

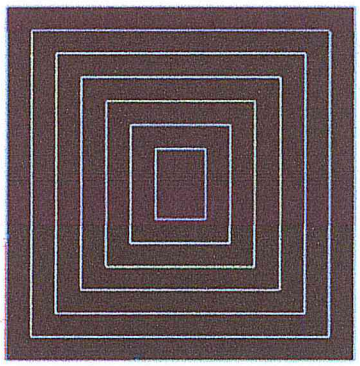


Figure III.24. Création des contours décalés sur le plan XY.

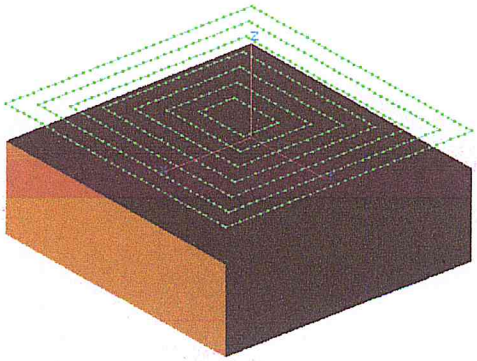
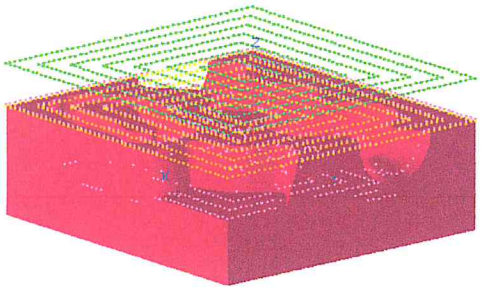


Figure III.25. Génération des points de plongées initiaux en 3D.

La détermination des points de plongées finaux est illustrée dans la Figure III.26 pour un rayon d'outil de 5mmune surépaisseur de 1mm.

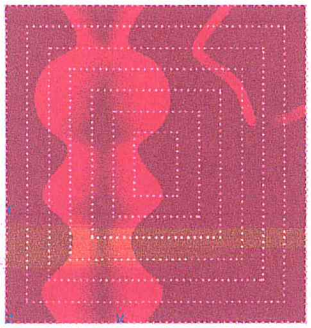


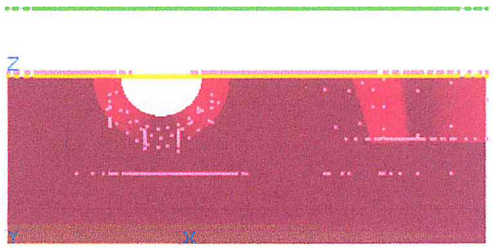
Point de plongée initiale.

Position de plongée.

Point de plongée finale.

a. Projection 3D.



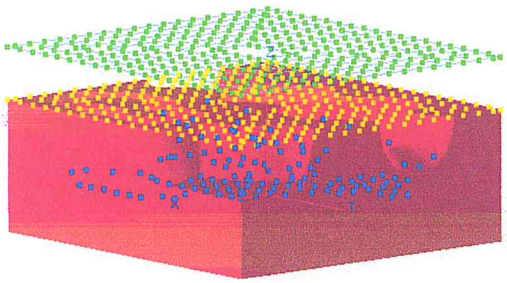


b. Projection XY.

c. Projection XZ.

Figure III.26. Visualisation des points de plongées finaux.

La Figure III.27 montre les points de plongée finaux valides après élimination des points de plongées finaux non valides.



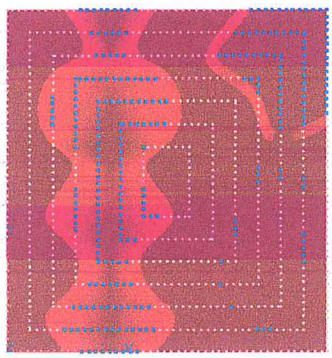
Point de plongée initiale.

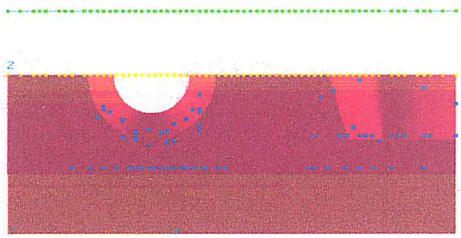
Position de plongée.

Point de plongée finale valide.

Point de plongée finale.

a. Projection 3D.



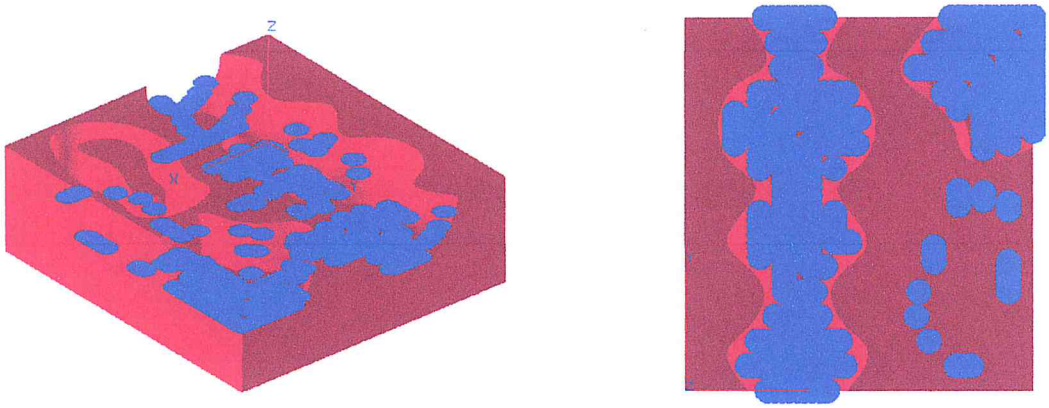


b. Projection XY.

c. Projection XZ.

Figure III.27. Détermination des points de plongées finaux valides.

La Figure III.28 montre les disques de plongée finaux valides.



a. Projection 3D. b. Projection XY.

Figure III.28. Visualisation des disques de plongées finaux valides.

Pour un point de plongée initial, la Figure III.29 montre le disque initial et final.

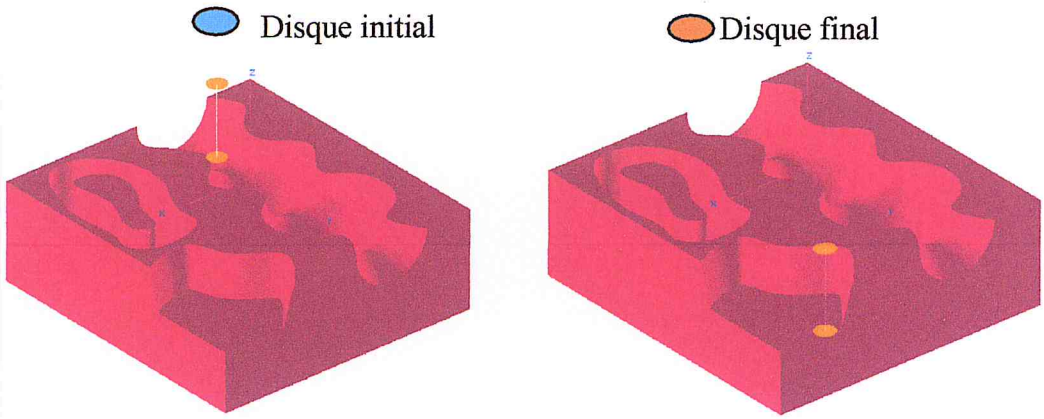
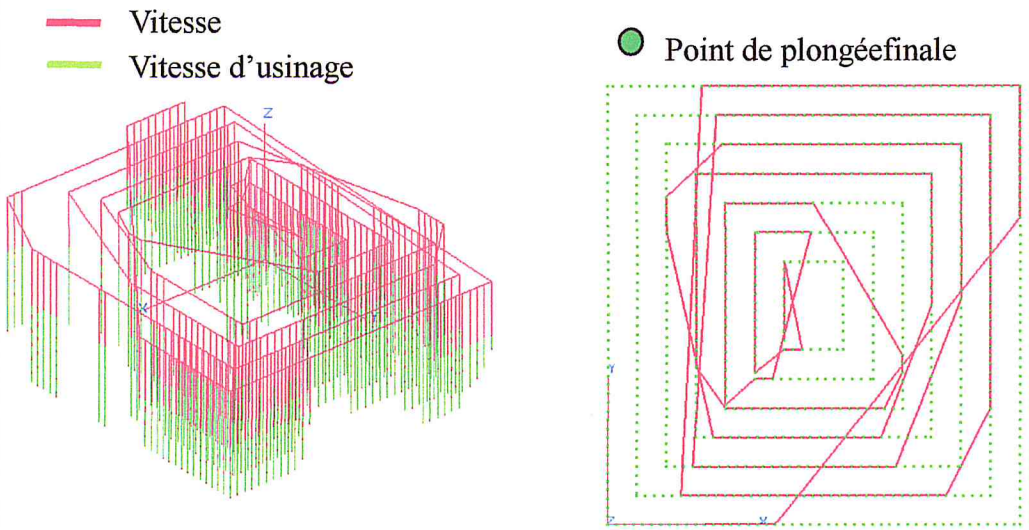


Figure III.29. Visualisation d'un disque de plongée sélectionné.

❖ **Etape 7** : simulation du trajet d'outil pour le Tréflage en 03-axes. La figure III.30 montre le trajet de Tréflage obtenu.



a. Projection 3D. b. Projection XY.

Figure III.30. Trajet de Tréflage en 03-axes.

Lorsque la simulation est en cours de traitement, par un clic sur « pause »à n'importe quel moment, l'outil est visualisé et les informations du point de plongée initial sont récupérées (Figure III.31).Par la suite, l'outil en plongée est visualisé et ses informations sont récupérées (Figure III.32).

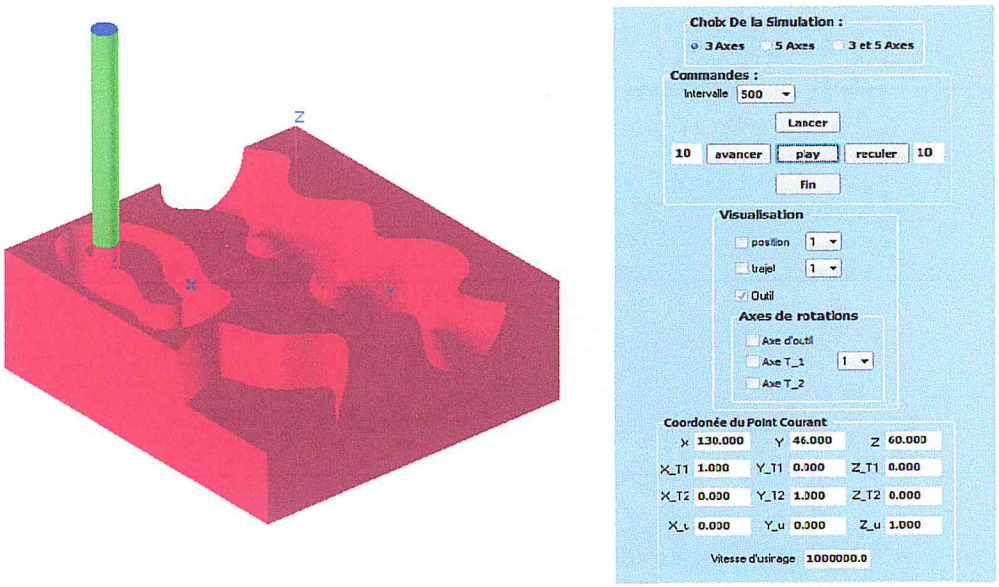


Figure III.31. Visualisation d'outil et coordonnées du point courant.

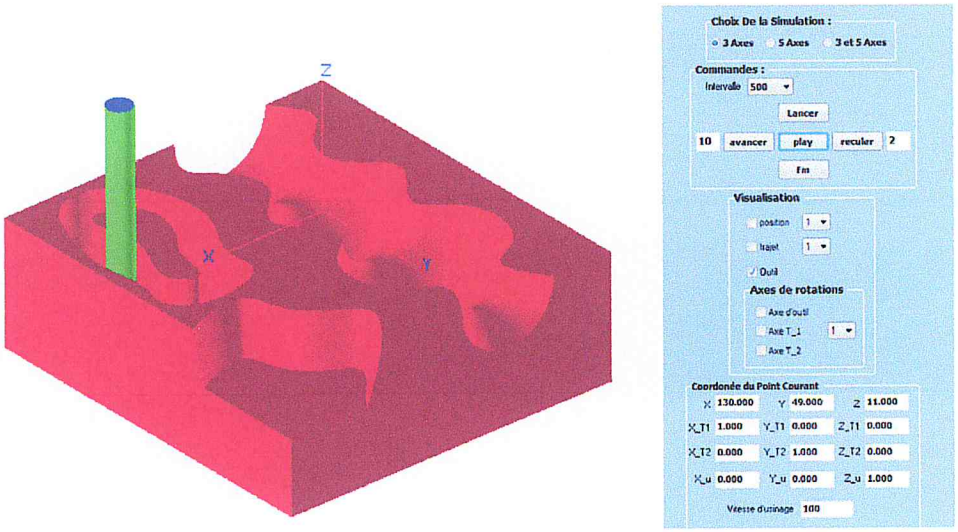
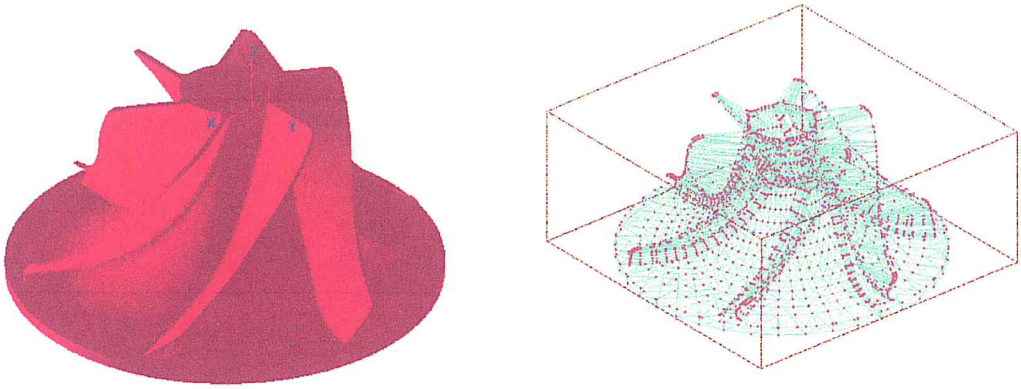


Figure III.32. Visualisation d'outil en plongée et coordonnées du point courant.

3.2. Deuxième modèle STL :

Les résultats des différentes étapes présentés dans les paragraphes suivants sont relatifs au deuxième modèle STL (Figure III.33).



a. Triangles en rendu. b. Sommets des triangles et le brut.

Figure III.33. Visualisation du deuxième modèle STL.

❖ **Etape 1** : détermination de la visibilité des sommets (Figure III.34) suivie par la détermination de la visibilité des triangles suivant trois axes de visibilité de composantes (0, 0,1) (Figure III.35), (0, 1,0) (Figure III.36) et (1, 0,0) (Figure III.37).

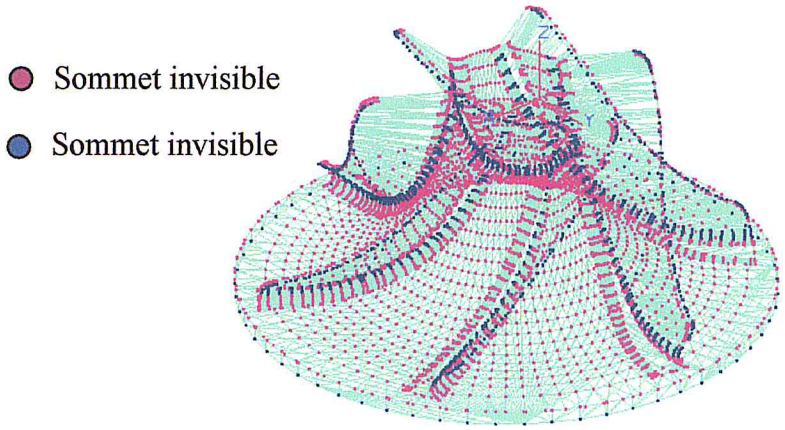
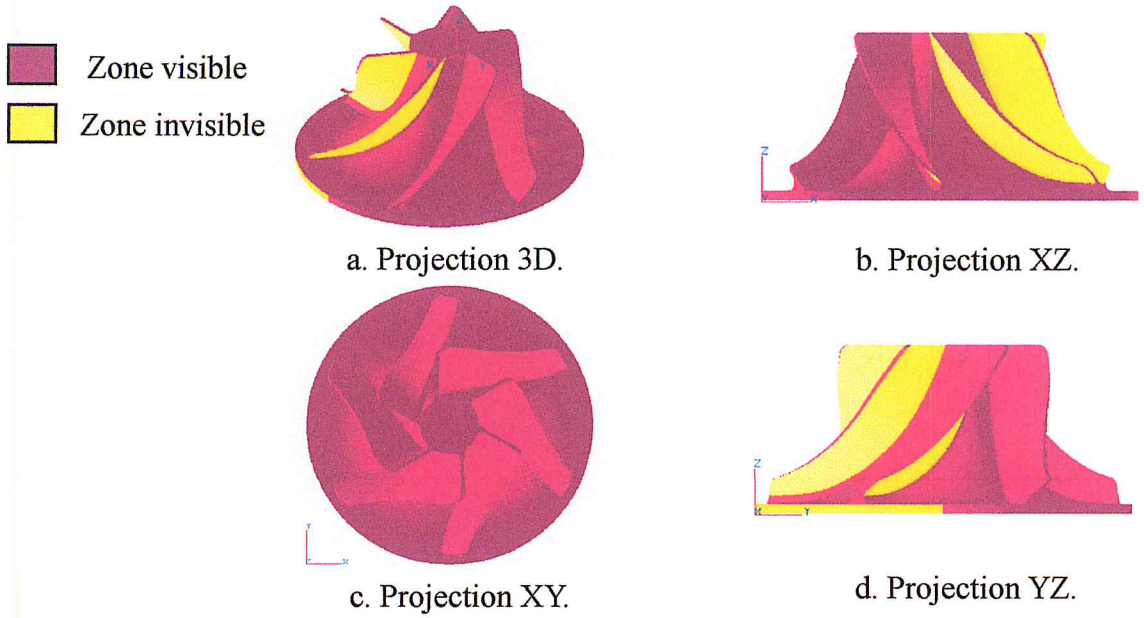


Figure III.34. Détermination des sommets visibles et invisibles.



a. Projection 3D. b. Projection XZ.

c. Projection XY. d. Projection YZ.

Figure III.35. Visibilité des triangles suivant l'axe (0,0,1).

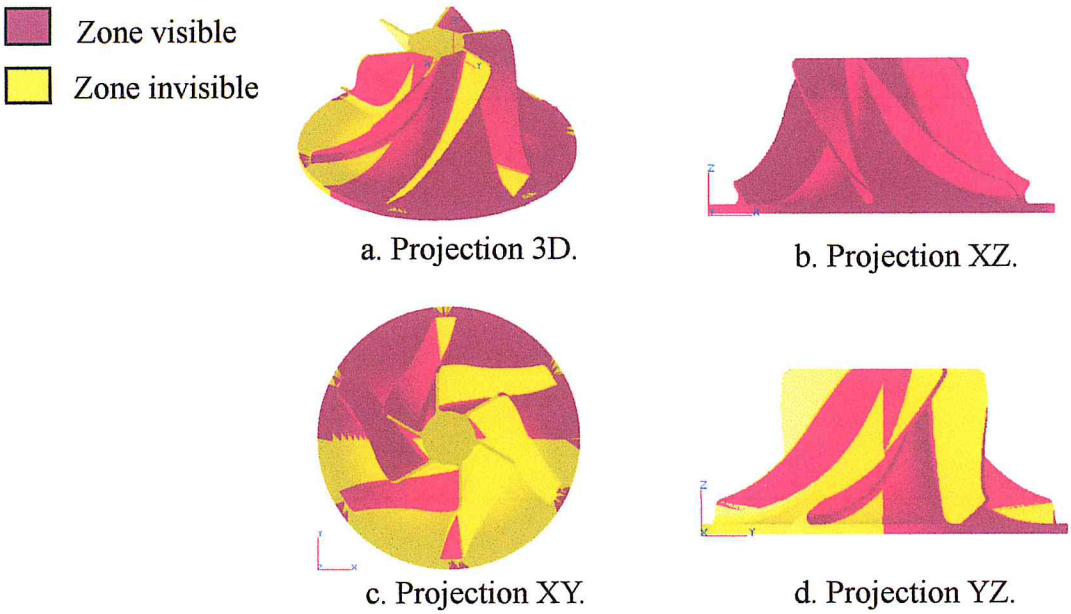


Figure III.36. Visibilité des triangles suivant l'axe (0,1,0).

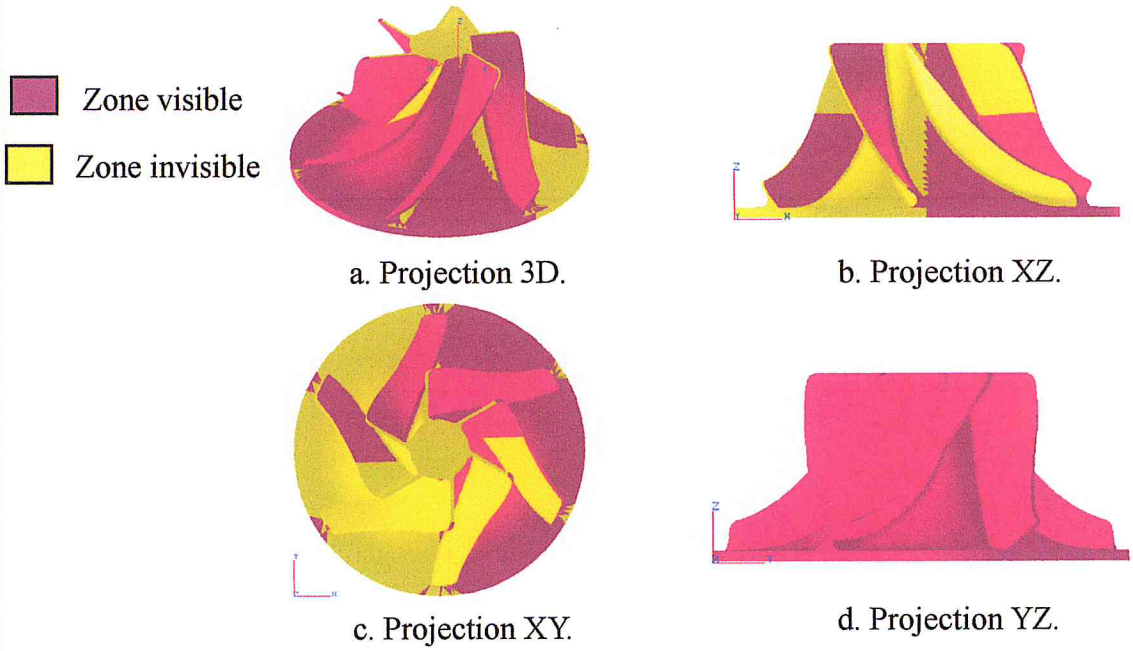


Figure III.37. Visibilité des triangles suivant l'axe (1,0,0).

Après avoir déterminé la visibilité pour chaque direction, la détermination de l'accessibilité des sommets (Figure III.38) ainsi que celle des triangles pour les mêmes directions sont déterminées. Cette accessibilité est établie pour l'axe de visibilité (0,0,1) (Figure III.39).

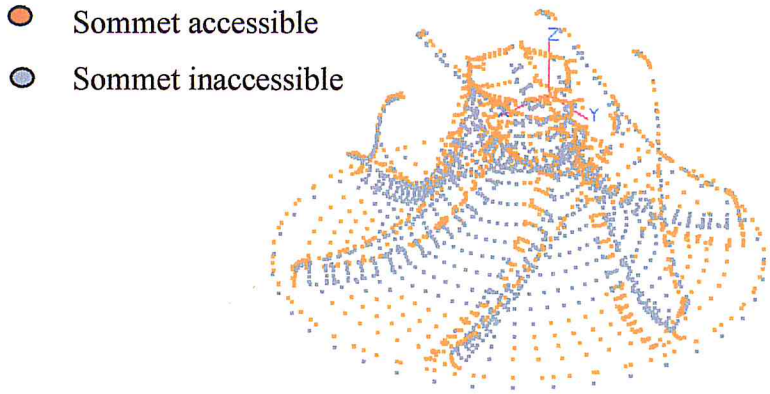
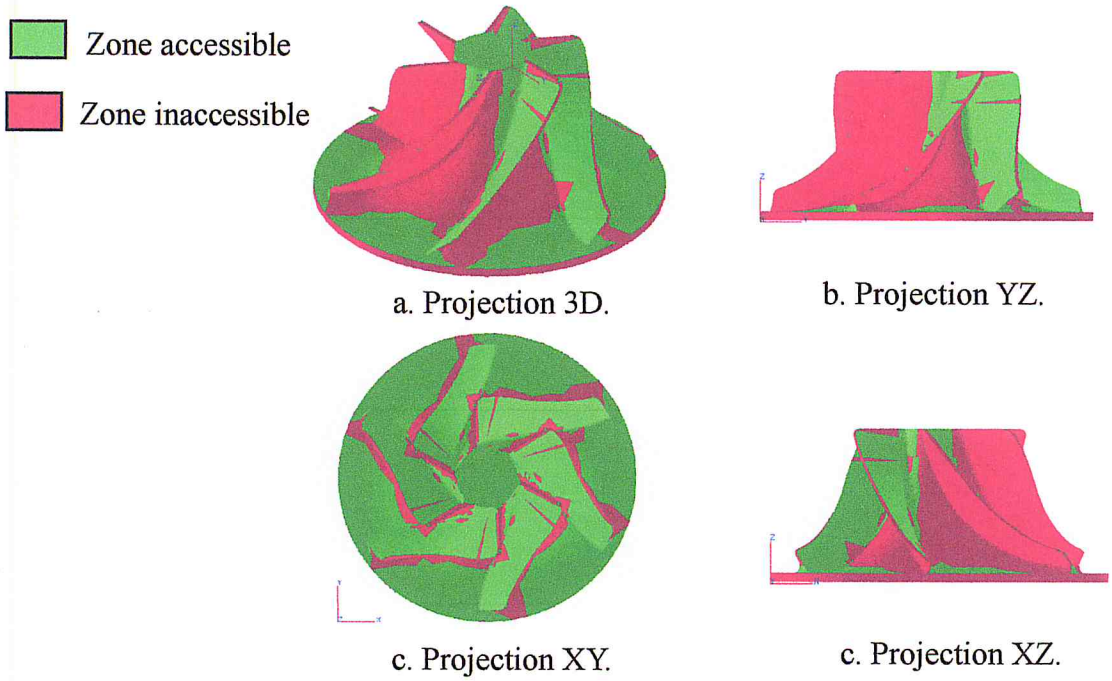


Figure III.38. Accessibilités des sommets suivant l'axe (0, 0, 1).

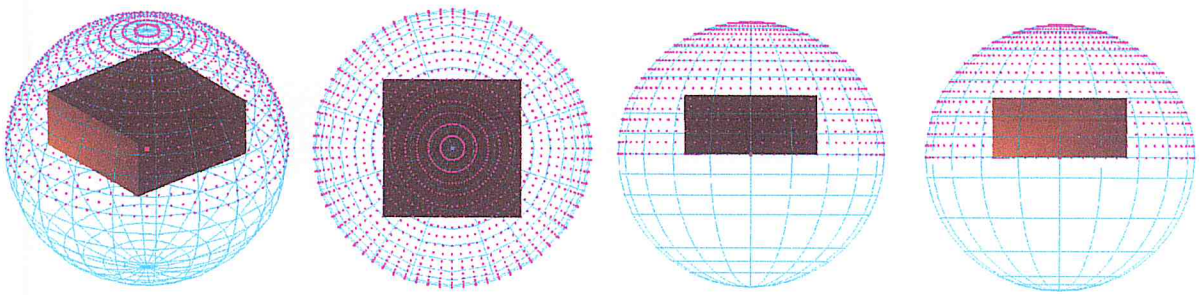


a. Projection 3D. b. Projection YZ.

c. Projection XY. c. Projection XZ.

Figure III.39. Accessibilité des triangles suivant l'axe (0,0,1).

❖ **Etape 2 :** génération de la sphère avec un pas Δ et un pas ∂ de 5° suivie de la création des points de plongées initiaux (Figure III.40).



a. Projection 3D b. Projection XY. c. Projection XZ. d. Projection YZ.

Figure III.40. Génération des points de plongée initiaux sur la demi-sphère.

❖ **Etape 3 :** détermination des points de plongées finaux rotatifs en fixant le rayon d'outil égal à 8mmet la surépaisseur à 1mm. La Figure III.41montre les résultats obtenus.

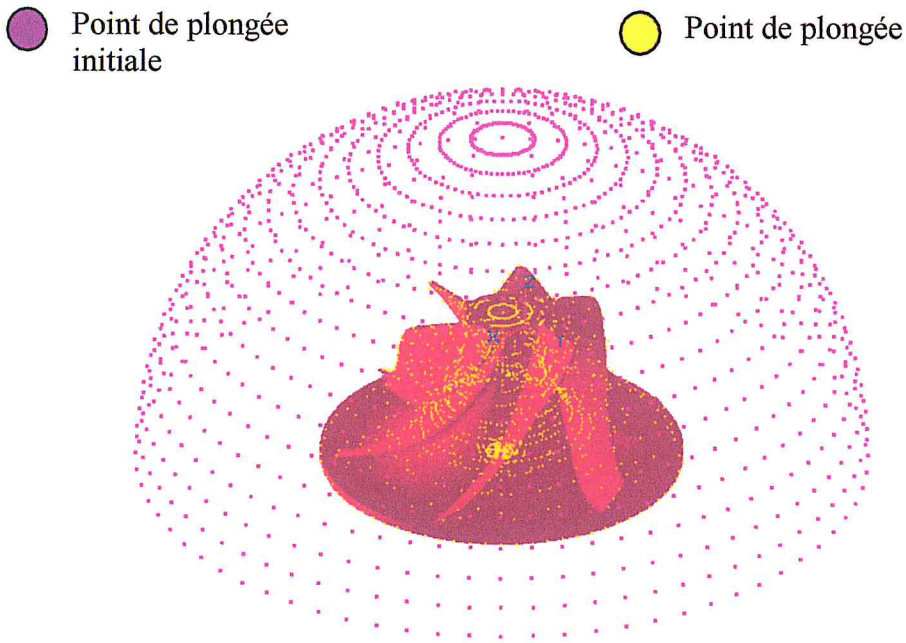


Figure III.41. Détermination des points de plongées finaux rotatifs.

La Figure III.42 montre le disque initial et le disque final pour un point de plongée initial.

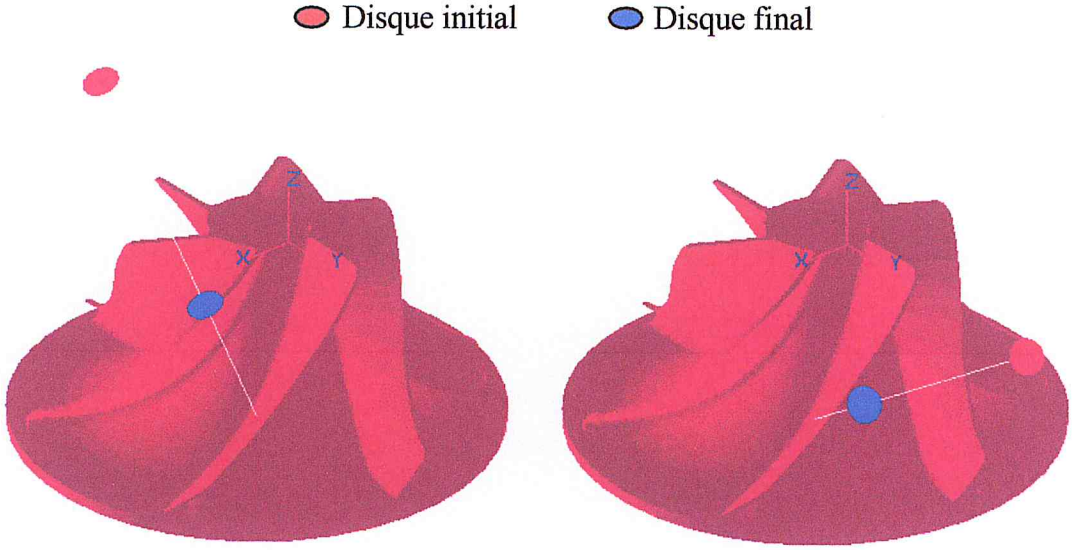
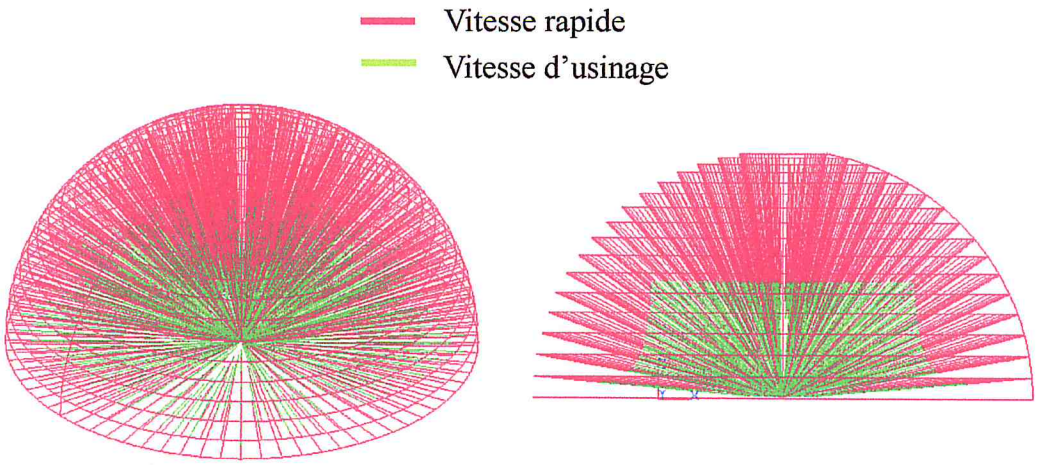


Figure III.42. Visualisation d'un disque de plongée sélectionné.

❖ **Etape 4** : simuler le trajet d'outil pour le Tréflage en 05-axes. La Figure III.43 montre le trajet d'usinage obtenu.



a. Projection 3D.

b. Projection XZ.

Figure III.43. Trajet de Tréflage en 05-axes.

Lorsque la simulation est en cours de traitement, un simple clic sur le bouton « pause »à n'importe quel moment, permet la visualisation de l'outil et la récupération des informations du point de plongée initial (Figure III.44).Par la suite, l'outil en plongée est visualisé et ses informations sont récupérées (Figure III.45).

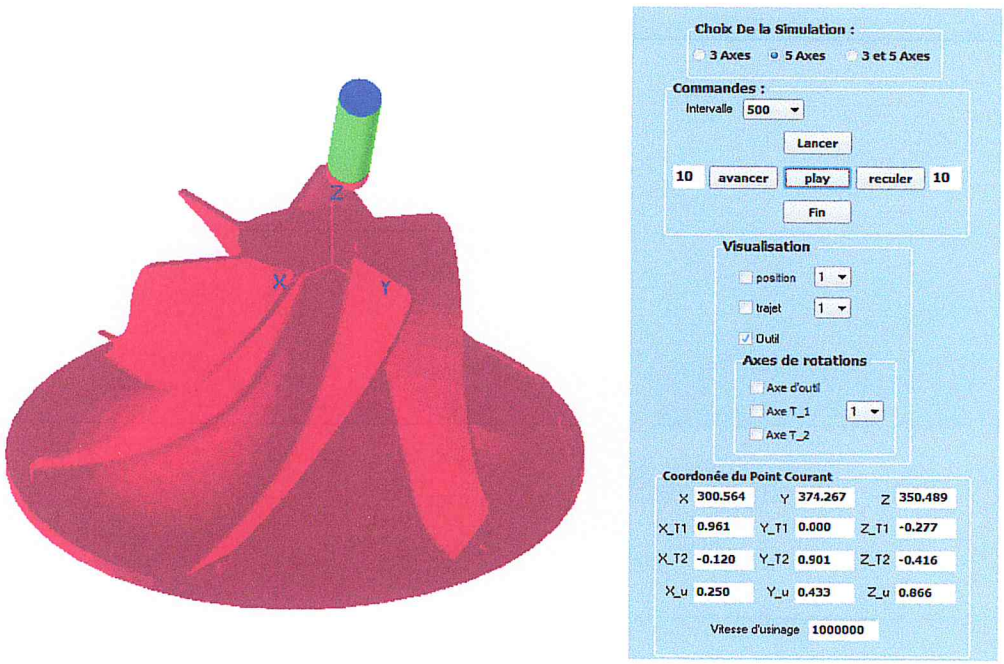


Figure III.44. Visualisation d'outil et coordonnées du point courant.

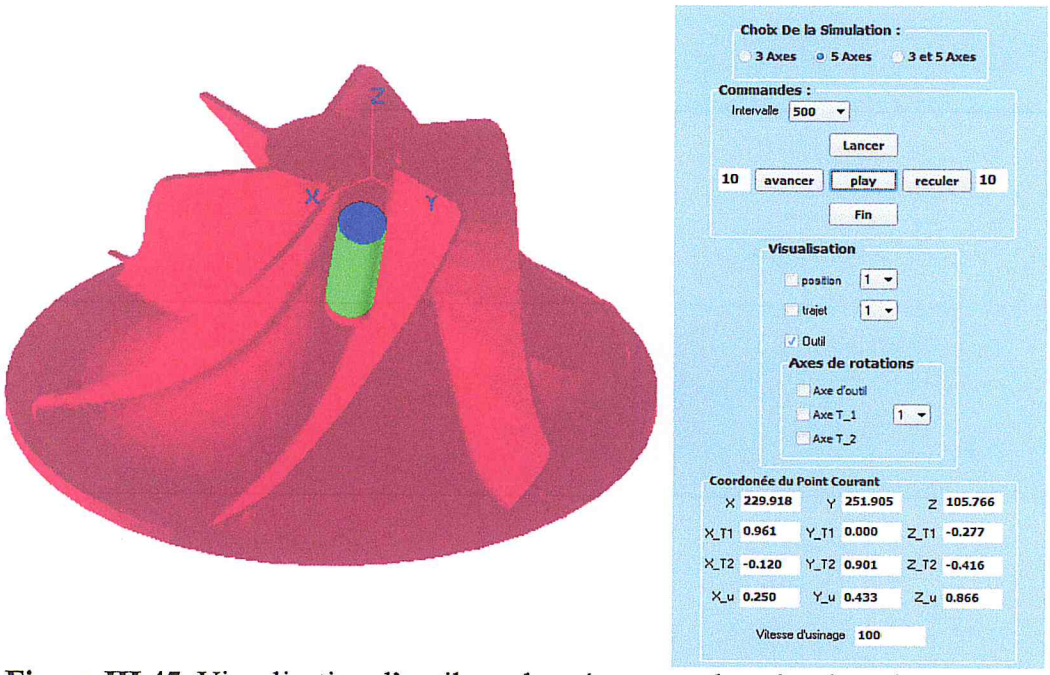


Figure III.45. Visualisation d'outil en plongée et coordonnées du point courant.

Conclusion :

Au cours de ce chapitre, nous avons présenté notre application afin de valider les résultats obtenus. Dans un premier temps, nous avons présenté l'environnement de développement de notre application ainsi que les interfaces pour décrire les différentes fonctionnalités. Nous avons testé les différentes fonctionnalités de notre application à travers deux exemple depuis la lecture du fichier STL jusqu'à la simulation du trajet d'usinage.

Conclusion Générale

CONCLUSION GÉNÉRALE

Le projet que nous avons présenté dans ce mémoire consiste à concevoir et à développer une application logicielle graphique et interactive de FAO permettant l'automatisation de l'opération d'ébauchage des surfaces gauches définies par leurs modèles STL en utilisant la stratégie d'ébauchage « Tréflage » par la nouvelle stratégie « Demi-sphère » sur des fraiseuses à commande numérique à 05-axes. Cette application permet de déterminer les points de plongées finaux et la profondeur de plongée pour chaque point de plongée initial en 03-axes et en 05-axes et le trajet d'outil associé tout en évitant les problèmes d'interférences et de collisions.

Lors de la réalisation de ce projet, nous avons commencé par une étude bibliographique sur les méthodes de conception des surfaces et du format d'échange de données « STL ». Par la suite, nous avons présenté les différentes stratégies d'ébauchage où nous avons mis l'accent sur la stratégie de Tréflage en 03-axes et en 05-axes. A la fin de notre mémoire, nous avons présenté la conception de notre application logicielle, les algorithmes développés ainsi que l'implémentation informatique et les résultats de validation.

Le résultat de l'application développée est l'intégration à la plate-forme logicielle graphique de l'équipe « CFAO » des fonctions permettant de réaliser les tâches suivantes :

- ✓ Lecture du fichier STL et création des limites du brut.
- ✓ Création des cellules.
- ✓ Affectation des sommets des surfaces aux cellules.
- ✓ Détermination des paramètres des triangles et des sommets.
- ✓ Localisation des zones visibles
- ✓ Identification des zones accessibles.
- ✓ Détermination des points de plongées initiaux et finaux en 03-axes et en 05-axes.
- ✓ Génération du trajet de Tréflage en 03-axes.

- ✓ Génération du trajet de Tréflage en 05-axes.
- ✓ Simulation virtuelle des mouvements de l'outil en 03-axes et en 05-axes.

En perspective à notre travail, nous recommandons de traiter les points suivants :

- ➢ Intégration de nouveaux modes de balayage de l'outil pour le Tréflage en 03-axes.
- ➢ Simulation de l'opération d'enlèvement de matière lors du Tréflage en 03-axes et en 05-axes.
- ➢ Génération du trajet d'outil en utilisant d'autres modes de balayage de l'outil pour le Tréflage en 05-axes.
- ➢ Génération du programme d'usinage « G-Code ».
- ➢ Détermination des conditions de coupe optimales.
- ➢ Calcul des cônes de visibilité et d'accessibilité pour accélérer la détermination des zones visibles et des zones accessibles.
- ➢ Génération du trajet de Tréflage à partir d'un nuage de points.
- ➢ Simulation virtuelle de la cinématique des fraiseuses numériques 05-axes.
- ➢ Intégration du calcul parallèle pour accélérer le processus de génération du trajet d'outil.

Références Bibliographiques

REFERENCES BIBLIOGRAPHIQUES

[1] M. Bey, « Modélisation des courbes et des surfaces », Rapport de Recherche, CDTA, Avril 2000.

[2] A. Doneddu. « Géométrie différentielle, intégrales multiples ». Tome 6. Editions Vuibert. 1981.

[3] M. Boutassouna, « Conception et Développement d'une Application de Recons1truction des Surfaces Gauches à Partir d'un Nuage des Points et Recherche des Outils d'Usinage Optimums », mémoire de fin d'études, Département d'Informatique, USDB, 2007.

[4] Y. Zhao. « Problems in Surface Intersection Representation and Construction ». Thèse de Master. Université de Michigan, Août 1998.

[5] Y. Zhao, Y. Zhou, J.L. Lowther and C.K. Shene. « Cross-Sectional Design: A Tool for Computer Graphics and Computer-Aided Design Courses ». 29th ASEE/IEEE Frontiers in Education, 10-13 Novembre, San Juan, Puerto Rico, Vol. II (1999), pp. (12b3-1)-(12b3-6).

[6] M. Peternell, H. Pottmann and B. Ravani. « On the computational geometry of ruled surfaces ». Computer-Aided Design, 1999, Vol 31, pp. 17–32.

[7] H.S Heo, M.S. Kim and G. Elber. « The intersection of two ruled surfaces ». Computer-Aided Design, 1999, Vol 31, pp. 33–50.

[8] En ligne :http://junior.universalis.fr.

[9] En ligne :https://www.futura-sciences.com

[10] En ligne :http://aip-primeca.ups-tlse.fr/themes/fao.html

[11] P. Lefebvre, B. Lauwers, « STL model segmentation for multi axis machining operations planning », Computer-Aided Design and Applications, pages 277-284, 2004.

[12] S. Rezzak, H. Taibi, « Méthode de Z-Constant pour l'usinage des surfaces gauches sur des fraiseuses à commande numérique à 3 axes », PFE, Département d'Informatique, Université Saad Dahlab, Blida, 2005.

[13] E. Duc, « Usinage de formes gauches; contribution à l'amélioration de la qualité des trajectoires d'usinage », Thèse de Doctorat à l'Ecole Normale Supérieure de Cachan, France 1998.

[14] A. Toumine, « Usinage par enlèvement de copeaux », cours de fabrication, INSA, Lyon, France.

[15] B. Furet, « Usinage des formes complexes », Cours de génie mécanique, Ecole Polytechnique de Montréal, Canada.

[16] B. H. Kim, B. K. Choi, « Machining efficiency comparison direction parallel tool path with contour parallel tool path », Computer Aided Design, Vol. 34, pages 89-95, 2002.

[17] Y. S. Lee, B. K. Choi, T. C. Chang, « Cut distribution and cutter selection for sculptured surface cavity machining », International Journal of Production Research, Vol. 30, N°30, pages 1447-1470, 1992.

[18] R. B. Jerard, R. L. Drysdale, K. Hauck, B. Schaudt, J. Magewick, « Methods for detecting errors in numerically controlled machining of sculptured surfaces ». IEEE Computer Graphics and Applications, Volume 9 Issue 1, January 1989.

[19] C. Tournier, « Evaluation des modes de génération des trajets outil pour l'usinage de formes gauches. Etude de cas ». Mémoire de DEA à l'Ecole Normale Supérieure de Cachan, France 1996.

[20] F. Girardin, N. Guillemot, « Maîtrise de l'opération de Tréflage », Rapport de stage, France, Session 2004-2005.

[21] M. Al-Ahmad, « Industrialisation de procédé : contribution à la maîtrise de l'opération de tréflage ou fraisage vertical - approche analytique et expérimentale », Thèse de Doctorat, École Nationale Supérieure d'Arts et Métiers, ENSAM, Metz, France, 2008.

[22] C. Hirano and K. Morishige, «Development of Rough Cutting Method with Plunge Milling Using 5-axis Control Machine Tool», 2007.

[23] En ligne : https://www.embarcadero.com/