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

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département de génie informatique.

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.
Option : Intelligence Artificielle (SI)

Sujet :

Conception et Développement d'une Application d'Optimisation du Trajet d'Usinage - Application à la Méthode Z-Constant -

Présenté par : CHERFI Abdelhak

Promoteur : BEY Mohamed

Organisme d'accueil : CDTA (Centre de Développement des Technologies Avancées).
Division productique et robotique

Soutenue le: 08/10/2006, devant le jury composé de :

Président : BENSTITI

Examinateurs : BENBLIDIA

TOUBALINE

- 2005/2006 -

Remerciements

Nous tenons à adresser au Directeur du Centre de Développement des Technologies Avancées (CDTA) nos sincères remerciements pour nous avoir accueilli dans son centre.

J'avais eu un réel plaisir de travailler avec mon promoteur Mr. M. BEY qui a su rester à mon écoute dans les moments où j'avais besoin de lui.

Je tiens à remercier tous les professeurs de l'USDB qui m'ont accompagnés dans mes études supérieures, ainsi qu'à tous nos enseignants et enseignantes d'étude primaire, moyenne et lycéenne.

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

DEDICACES

Je dédie ce modeste mémoire à :

Mon cher père.

Ma très chère mère.

A mes grands parents, mes frères Amine, Farid ; Adberrahmane, Ridha et Youcef

A ma sœur Romaissa.

Et à toute ma grande famille sans exception.

A tous les voisins.

A tout les gens qui m'ont aimé, à tous mes amis et collègues sans restriction.

Abdelhak

Résumé

Ce travail s'insère dans le cadre de développement d'outils de conception et de fabrication des surfaces gauches initié par l'équipe Conception et Fabrication Assistées par Ordinateur (CFAO) au niveau de la Division Robotique et Productique du Centre de Développement des Technologies Avancées (CDTA).

Dans ce projet on s'intéressera à l'usinage en finition des surfaces gauches sur des fraiseuses à commande numérique à 3 axes par la méthode Z-Constant. Le but de ce travail est le développement d'une application logicielle graphique et interactive sous Windows qui permet en premier point, l'optimisation de la trajectoire d'outil par la prise en compte de différentes contraintes (priorité d'usinage, longueur du trajet, ...etc.) pour les deux stratégies d'usinage One-Way et Zig-Zag. En deuxième point, la détection des collisions et des interférences entre les différentes parties d'outil et les surfaces à usiner. En dernier point, la génération du trajet d'usinage en interpolation polynomiale avec les courbes B-Spline.

This work fits in the setting of development of tools of conception and left surface manufacture initiated by the team Conception and Fabrication Attended by Computer (CFAO) at the level of the Division Robotics and Production of the of Development Technology Advanced Center (CDTA).

In this project we are interested in the machining of the free shape surfaces on tools-machine numeric order to 3 axes by the Z-Constant strategy, The objective of this work is the development of a graphic and interactive software under Windows that permits in the first time ,the optimisation of tool path by the consideration of the different constraints (milling priority, trajectory longer...etc) for the One-Way and Zig-Zag milling strategies; in the second time, the detection of the interference and collision between the different parts of the tool and the surface to milling; in the end the generation of the path milling by the polynomial interpolation with B-Spline curves.

ملخص

إنّ هذا العمل يندرج في إطار تطوير وسيلة, قادرة على تصميم و تصنيع الأسطح ذات الأشكال الحرّة و التي تقوم بتطويرها فرقة البحث الخاصّة بالتّصميم و التّصنيع عن طريق الكمبيوتر على مستوى فرع التصنيع الآلي الخاص بمركز تطوير التكنولوجية المتقدّمة.

في هذا المشروع, نهتم بتصنيع المساحات الحرّة باستعمال آلة التقطيع الرقمية, ذات الثلاث محاور. إنّ الهدف من هذا العمل هو تطوير برنامج آلي تصويري قابل للاستعمال على نظام التشغيل ويندوز و الذي يسمح بما يلي:
اختصار طريق القاطعة مع الأخذ بعين الاعتبار مختلف الشروط مثـل أولوية التقطيع' طول المسار بالنسبة للاستراتيجية ذهاب و ذ هاب- رجوع ثانيا إيجاد إذا كان هناك تداخل بين لمساحة التقطيع و مختلف أطراف القاطعة أو في الأخير تقريب المسار عن طريق منحنيات بـــسبلاين.

SOMMAIRE

CHAPITRE 2: SURFACES B-SPLINE ET NURBS

CHAPITRE 3: ARCHITECTURE ET PROGRAMMATION DES FRAISEUSES A COMMANDE NUMERIQUES

CHAPITRE 4: USINAGE DES SURFACES GAUCHES

CHAPITRE 7: TESTS ET VALIDATIONS

INTRODUCTION GENERALE

I. SITUATION DU PROBLEME

L'utilisation de l'outil informatique a beaucoup apporté à l'industrie mécanique où il est nécessaire d'automatiser tout le processus de réalisation des pièces mécaniques depuis la conception jusqu'à l'usinage sur machine tout en tenant compte de la qualité du produit et des conditions de travail.

Les ingénieurs dans le domaine de la mécanique cherchent à développer et à améliorer des méthodes de conception des pièces de formes libres. L'apparition de la commande numérique a permis d'automatiser le processus de réalisation de ces pièces. Parmi ces pièces, nous citons les moules, les matrices, les formes esthétiques, ...etc. qui sont très rencontrées dans notre vie quotidienne. Ces pièces sont conçues dans le but d'assurer des fonctions inscrites dans le cahier des charges. Ces pièces ne peuvent être usinées que sur des fraiseuses à commande numérique à 03 ou à 05 axes en raison des géométries très complexes. L'usinage de ces surfaces passe généralement par trois étapes : ébauche qui permet d'enlever le maximum de matière, demi finition où on s'approche de la forme finale et finition où on obtient la forme voulue. Toutes ces étapes nécessitent la génération d'un ensemble d'instructions écrites dans un langage propre à la machine appelé programme « G-Code ».

Le travail que nous présentons dans ce mémoire s'inscrit dans le cadre de développement d'outils de conception et de fabrication des surfaces de formes libres (surfaces gauches) initié par l'équipe CFAO de la Division Productique et Robotique du Centre de Développement des Technologies Avancées (CDTA).

Notre projet est une continuité des travaux précédents traitant :

- La modélisation et la conception des courbes et des surfaces B-Spline et NURBS;
- La reconstruction des courbes et des surfaces B-Spline et NURBS par interpolation et par approximation à partir d'un nuage de points;
- Usinage des surfaces gauches par les courbes isoparamétriques en utilisant les six stratégies d'usinage (One-Way, Zig-Zag, Concentrique, Spiral-In, Spiral-Out et Radiale).
- Usinage des surfaces gauches par la méthode des plans parallèles.
- Ebauchage des surfaces gauches.

Notre projet s'intéressera à l'optimisation du trajet d'usinage lors de l'usinage en finition des surfaces gauches avec la stratégie Z-Constant. L'intitulé de notre projet est le suivant :

« Conception et développement d'une application du trajet d'usinage, - application à la méthode Z-Constant - »

Dans le cas de l'usinage des cavités profondes en finition, la méthode la plus adaptée est la méthode Z-Constant. Pour cette méthode, l'usinage s'effectue sur des plans horizontaux, donc, il est nécessaire d'optimiser les déplacements afin de réduire le trajet d'usinage et par conséquent les coûts de fabrication tout en tenant compte de la forme locale de la surface. Un autre problème qu'il faut considérer, c'est les collisions entre les différentes parties de l'outil et les surfaces à usiner.

Dans les programmes d'usinage G-Code, les mouvements de l'outil sont exprimés dans un format d'interpolation linéaire qui est basé sur l'association d'une ligne brisée à la courbe théorique lieu des centres de l'outil. Dans ce format, le mouvement de l'extrémité de l'outil est discontinu en tangence et en courbure ce qui provoque un choc au niveau de l'asservissement engendrant des vibrations et des marques sur la pièce. Pour palier à ce problème, nous devons passer à l'interpolation polynomiale avec les courbes B-Spline pour minimiser les vibrations et améliorer l'état de la surface.

II. OBJECTIF DU TRAVAIL

Dans ce projet, nous nous s'intéresserons à l'optimisation du trajet d'usinage pour la finition des surfaces gauches avec la stratégie d'usinage Z-Constant sur des fraiseuses à commande numérique à 03 axes.

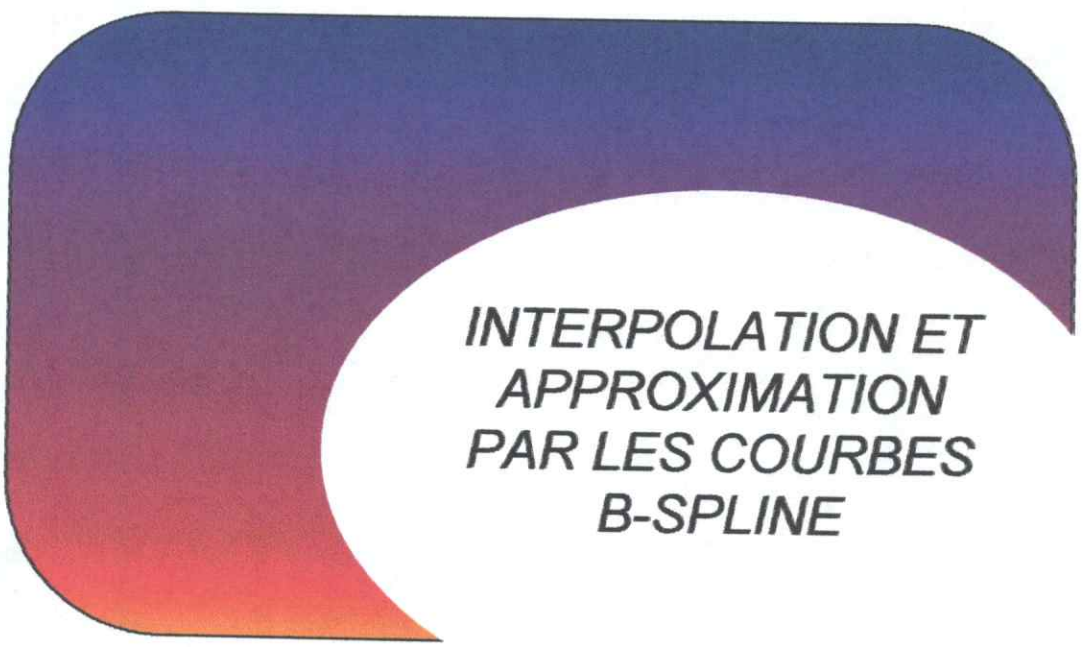
L'objectif de notre projet est de développer une application logicielle graphique et interactive sous Windows permettant la détection des collisions et des interférences pour les différentes parties d'outil, l'approximation du trajet d'usinage par des courbes B-Spline et enfin l'optimisation du trajet d'usinage en considérant les modes de balayage de l'outil en One-Way et en Zig-Zag.

III. DESCRIPTION DU TRAVAIL

Le présent mémoire est composé de sept chapitres :

- Le premier chapitre sera consacré aux définitions de base des courbes B-Spline ainsi qu'à l'approximation d'un nuage de points par ces courbes.
- Dans le deuxième chapitre, nous allons étudier les différentes méthodes de représentation et de conception des surfaces et en particulier les surfaces NURBS et B-Spline.
- Le troisième chapitre sera consacré à la présentation de l'architecture et de la programmation des fraiseuses à commande numérique.
- Dans le quatrième chapitre, nous allons étudier les différentes méthodes et stratégies d'usinage des surfaces gauches.
- Les besoins et la modélisation de notre système seront présentés dans le cinquième chapitre.
- Dans le sixième chapitre, nous allons présenter l'implémentation de notre système ainsi que les fonctions et les algorithmes utilisés.
- Le dernier chapitre présente les tests et la validation des résultats.

CHAPITRE 1



INTERPOLATION ET APPROXIMATION PAR LES COURBES B-SPLINE

I. INTRODUCTION :

La naissance de la conception et de la fabrication assistées par ordinateur (CFAO), ainsi que le développement dans le domaine industriel (fabrication des moule, formes libres, ...etc.), d'autre part les exigences de fabriquer des pièces à haute qualité, ont nécessité le développement des méthodes de modélisation et de conception des courbes et des surfaces. Plusieurs méthodes ont été introduites (Bézier, B-Spline, NURBS) dans le but de faciliter le travail aux ingénieurs concepteurs.

Dans ce premier chapitre, nous allons donner des définitions et des schémas explicatifs concernant les courbes B-Spline en commençant par les méthodes de représentation, et en terminant avec les méthodes de construction des courbes B-Spline par interpolation et par approximation.

II. COURBES PARAMETRIQUES ET NON PARAMETRIQUE [1, 2] :

II.1. Courbes paramétriques :

En CAO, c'est cette méthode de représentation qui est utilisée dans la conception des courbes. Les courbes paramétrique sont définies dans l'espace par :

$$C(u) = (f(u), g(u), h(u)) \qquad (1.1)$$

Où f(u), g(u) et h(u) sont des fonctions réelles. Pour chaque valeur de u correspond un point sur la courbe de coordonnées (f(u),g(u),h(u)). Les fonctions utilisées peuvent être des fonctions polynomiales, rationnelles, logarithmiques, exponentielles ou trigonométriques.

II.1.1. Caractéristiques des courbes paramétriques :

Les courbes paramétriques sont caractérisées par les propriétés suivantes :

- **Vecteur tangent à la courbe :** le vecteur tangent à la courbe t (voir figure 1) est défini par la dérivée de la courbe paramétrique C(u) par rapport à u et il est donné par :

$$\vec{T} = C'(u) = (f'(u), g'(u), h'(u)) \qquad (1.2)$$

Où f'(u),g'(u),h'(u) sont les dérivées de f(u), g(u) et h(u) par rapport à u respectivement.

La ligne tangente à la courbe au point C(u) est donnée par :

$$C(u) + t\frac{C'(u)}{\lfloor C'(u) \rfloor} \quad t \in R \qquad (1.3)$$

- **Vecteur normal à la courbe :** pour définir le vecteur normal à la courbe, il faut d' abord définir le vecteur binormal (voir figure 1) qui est donné par :

$$\vec{b} = \frac{C'(u) * C''(u)}{|C'(u) * C''(u)|} \tag{1.4}$$

Le vecteur normal **n** à la courbe en un point (voir figure1) est perpendiculaire au vecteur tangent **t** et au vecteur binormal **b** au même point et il est donné par :

$$\vec{n} = \frac{b * C(u)}{|b * C(u)|} \tag{1.5}$$

Le plan construit par les vecteurs **n** et **t** est appelé le plan osculateur (voir figure 1) et son équation est donné par :

$$C(u) + pC'(u) + qC''(u) \tag{1.6}$$

Les vecteurs **b** et **n** definissent le plan normal, tandis que les vecteurs **t** et **b** definissent le plan rectifiant (voir figure1). Les vecteurs **t**, **n** et **b** definissent un repère local appelé repère de frenet.

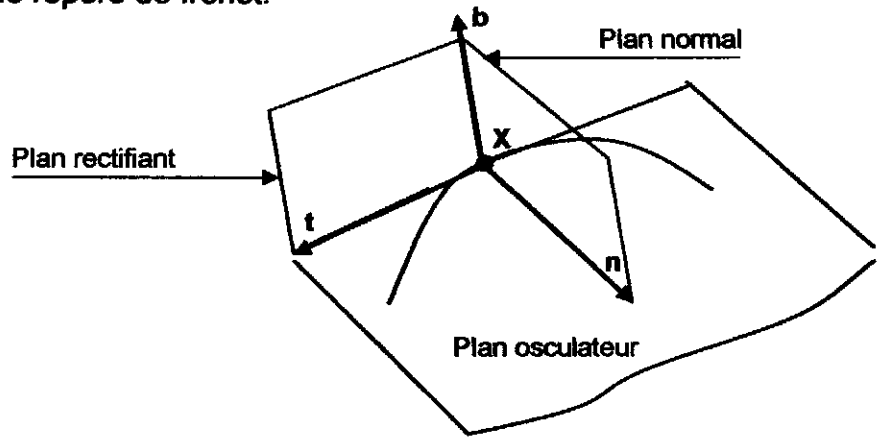


Figure 1. Repère de frenet en un point X.

- **Courbure :** le rayon de courbure R est le rayon de cercle qui approxime mieux la courbe en un point. L'inverse de ce rayon est appelé la courbure (voir figure 2) et il est donné par :

$$K(u) = \frac{C'(u) * C''(u)}{C'(u)^3} \tag{1.7}$$

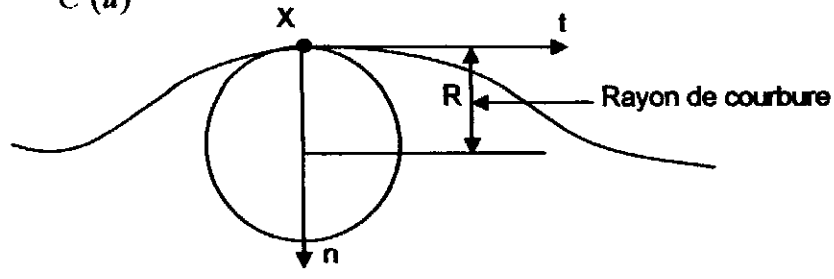


Figure 2. Courbure en un point de la courbe.

II.2. Courbes non paramétriques [2] :

Une autre forme de représentation des courbes est la représentation non paramétrique et qui peut :

Explicite : dans ce cas, la courbe est donnée par :

$$Y = f(x) \tag{1.8}$$

Implicite : dans ce cas, la courbe est donnée par :

$$f(x,y) = 0 \tag{1.9}$$

III. COURBES B-SPLINE :

Parmi les courbes paramétriques les plus utilisées dans la conception des courbes en CAO, nous avons les courbes B-Spline. La conception de ces courbes nécessite :

1. Un ensemble de points de contrôle.
2. Un ensemble de point nodaux (nœuds).
3. Des coefficients, un pour chaque point de contrôle.

III.1. Fonctions base B-Spline [3] :

La définition des fonctions B-Spline nécessite un autre paramètre qui est le degré de ces fonctions base **p**. La ième fonction base B-Spline de degré **p** est donnée par :

$$N_{i,0}(u) = \begin{cases} 1 & \text{si } u_i \le u < u_{i+1} \\ 0 & \text{sinon} \end{cases} \tag{1.10}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{1.11}$$

Si le degré **p** =0 et si $u \in [u_i, u_{i+1}[$, les fonctions base $N_{i,0}(u)$ sont égales à 1 et elles sont nulles ailleurs. La fonction B-Spline Ni,p(u) est une combinaison linéaire de $N_{i,p-1}$ et $N_{i+1,p-1}$ avec deux coefficients linéaires en u dont les valeurs sont comprises entre 0 et 1.

III.2. Propriétés des fonctions base B-Spline [3.2] :

1- $N_{i,p}(u)$ est un polynôme de degré *p* en *u*. et non négative pour tout *i*, *p* et *u*.

2- Support local : les fonctions base $N_{i,p}(u)$ sont non nulles sur l'intervalle $[u_i, u_{i+p+1}[$ et elles sont nulles ailleurs.

3- Partition d'unité : la somme de toutes les fonctions bases non nulles de degré *p* sur l'intervalle $[u_i, u_{i+1}[$ est égale à 1.

4- En un nœud de multiplicité *k*, la fonction base $N_{i,p}(u)$ a une continuité C^{p-k}.

III.3. Définition des courbes B-Spline [3] :

Une courbe B-Spline est définie par les paramètres suivants (voir figure3) :

- Le degré p de la courbe B_Spline.
- *n*+1 points de contrôle $p_0, p_1, \ldots, p_n$ (polygone de contrôle).
- Le vecteur nodal $U= \{u_0, u_1, \ldots, u_m\}$ tel que $u_0=0$ et $u_m=1$.

La courbe B-Spline est donnée par :

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u) p_i \qquad (1.12)$$

La figure suivante montre une courbe B-Spline et ces composantes [2] :

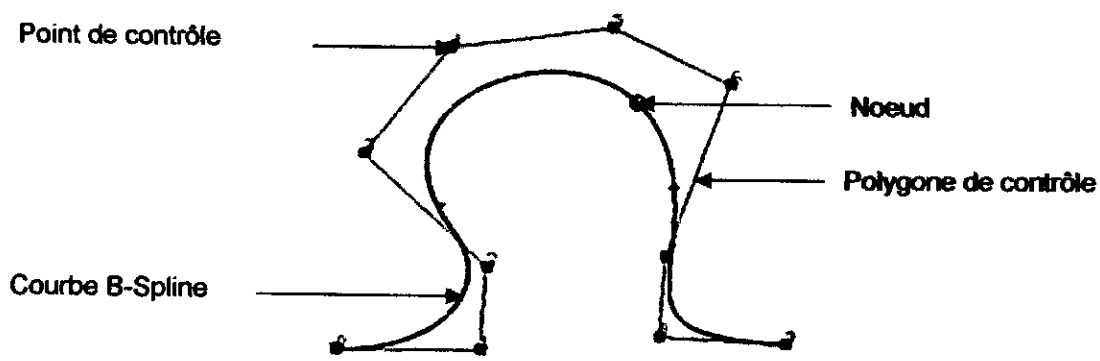


Figure 3. Paramètres d'une courbe B-Spline.

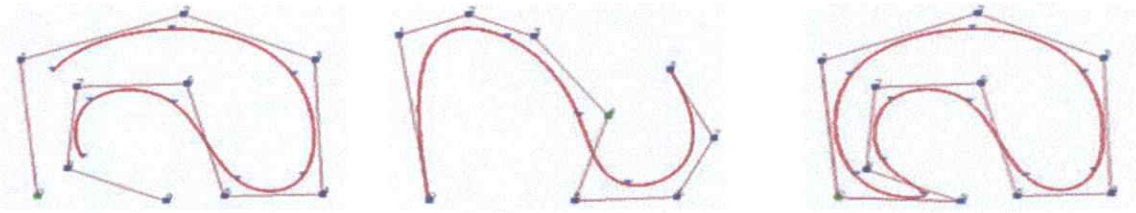
III.4. Types de courbes B-Spline [4,2] :

Les courbes B-Spline peuvent être sous trois formes :

1. **Courbe B-Spline ouverte :** dans ce type de courbes, le vecteur des points nodaux n'a pas une structure particulière, et la courbe ne passe pas par le premier et le dernier segment du polygone de contrôle (voir figure 4.1).

2. **Courbe B-Spline pincée :** dans ce type de courbes, la courbe B-Spline passe par le premier et le dernier point de contrôle et elle est tangente au premier et au dernier segment du polygone de contrôle avec la condition que le premier nœud et le dernier nœud doivent être répétés *p+1* fois (voir figure 4.2).

3. **Courbe B-Spline fermée :** dans ce type de courbes, quelques points de contrôle et quelques nœuds sont répétés et le début et la fin de la courbe sont confondus (voir figure 4.3).



1. Ouverte. 2. Pincée. 3. Fermée.

Figure4. Types de courbes B-Spline.

III.5. Propriétés des courbes B-Spline [3.2] :

Les plus importantes propriétés des courbes B-Spline sont les suivantes :

- La courbe B-Spline **C**(u) est une courbe définie par morceaux où chaque morceau de courbe est une courbe de degré p.
- L'égalité fondamentale suivante doit être satisfaite :

$$m = n + p + 1 \qquad (1.13)$$

- Propriété de l'enveloppe convexe : la courbe B-Spline est contenue dans l'enveloppe convexe de son polygone de contrôle.
- Schéma de modification locale : le changement de la position des points de contrôle $\mathbf{p}_i$ n'influe sur la courbe $\mathbf{C}(u)$ que dans l'intervalle $[u_i, u_{i+p+1}[$.
- La courbe $\mathbf{C}(u)$ est C^{p-k} continue pour un nœud de multiplicité k.
- Propriété de diminution de variation : la courbe construite est plus simple que son polygone de contrôle.
- Invariance affine : si une transformation affine est appliquée à la courbe B-Spline, la courbe résultante peut être construite à partir des images des points de contrôle.
- La courbe est indépendante du système de coordonnées utilisé pour décrire le polygone de contrôle.

Malgré la flexibilité et les avantages des courbes B-Spline, elles ne permettent pas de représenter les formes coniques telles que les cercles, les ellipses et les paraboles puisqu'elles sont des courbes polynomiales.

IV. CONSTRUCTION DES COURBES B-SPLINE PAR INTERPOLATION ET PAR APPROXIMATION :

Dans les applications industrielles nous sommes affrontés à traiter un nuage de points et le représenter par des courbes simples à manipuler. A cet effet, deux méthodes de conversion d'un nuage de points en courbes sont utilisées à savoir l'interpolation et l'approximation.

IV.1. Interpolation :

L'interpolation consiste à générer une courbe qui doit passer par tous les points.

IV.1.1. Interpolation globale :

L'interpolation globale est la méthode la plus simple de joindre un ensemble de points de données. Nous avons n+1 points de données D_k (k=0,.n) et on veut les interpoler par une courbe B-Spline. Supposons que le vecteur nœud U = $\{u_0,u_1,.....u_n\}$ est généré à base d'un ensemble de paramètres $\{t_0,t_1,.....t_n\}$ obtenus par une méthode de calcul des paramètres. L'interpolation consiste à construire une courbe de degré p et à déterminer (n+1) points de contrôle.

$$D_k = C(t_k) = \sum_{k=0}^{n} N_{i,p}(t_k) p_i \qquad (1.14)$$

L'équation précédente peut s'écrire sous la forme :

$$D = N * P \qquad (1.15)$$

On peut écrire l'équation sous la forme matricielle suivante :

$$\begin{bmatrix} d_{0.1} \\ d_{1.1} \\ . \\ . \\ d_{n.1} \end{bmatrix} = \begin{bmatrix} N0,p(t0) & N1,p(t0) & . & . & Nn,p(t0) \\ N0,p(t1) & N1,p(t1) & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ N0,p(tn) & . & . & . & Nn,p(tn) \end{bmatrix} * \begin{bmatrix} p_{0.1} \\ p_{1.1} \\ . \\ . \\ p_{n.1} \end{bmatrix} \qquad (1.16)$$

La résolution du système d'équations linéaires permet de calculer les coordonnées des points de contrôle.

La figure 5 montre un nuage de points et la courbe générée par interpolation.

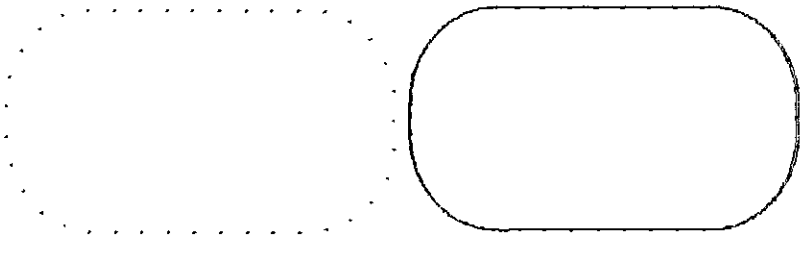


Figure 5. Interpolation d'un nuage de points.

IV.1.2. Interpolation locale :

Dans cette approche, on construit des morceaux de courbes en considérant des sous ensembles des points de données qu'il faut joindre pour construire la courbe

entière. Le changement d'un point n'affecte la courbe que dans le morceau concerné de la courbe.

IV.2. Approximation :

L'approximation consiste à générer une courbe qui ne passe pas nécessairement par tous les points.

IV.2.1. Approximation globale [4] :

Supposons que nous avons n+1 points de données D_k (k=0..n), p le degré de la courbe à approximer, nous voulons construire une courbe B-Spline avec h+1 points de contrôles avec la condition que $n \geq h \geq p$.

Il faut d'abord commencer par la détermination d'un ensemble de (n+1) paramètres et générer un vecteur nodal $U = \{u_o, u_1, \ldots, u_m\}$ et par la suite calculer les (h+1) points de contrôles de la courbe B-Spline avec la condition que la courbe générée doit passer par le premier et le dernier point de donnée ($p_0 = C(0)$ et $p_h = C(1)$) et il reste à calculer p_1, p_2, ...et p_{h-1}.

Pour calculer les points de contrôle, nous utilisons la méthode des moindres carrés et la fonction à minimiser est donnée par :

$$f(p_1, p_2, \ldots, p_{h-1}) = \sum_{k=1}^{n-1} \left| Q_k - \sum_{i=1}^{h-1} N_{i,p}(tk) pi \right|^2 \qquad (1.17)$$

Avec :

$$Q_k = D_k - N_{0,p}(tk) D_0 - N_{h,p}(tk) D_n \qquad (1.18)$$

Après tout les calculs, nous obtenons le système suivant :

$$(N^T N)P = Q \qquad (1.19)$$

Avec :

$$P = \begin{bmatrix} p_1 \\ p_2 \\ . \\ . \\ p_{h-1} \end{bmatrix} \quad Q = \begin{bmatrix} \sum_{k=1}^{n-1} N_{1,p}(t_k) Q_k \\ \sum_{k=1}^{n-1} N_{2,p}(t_k) Q_k \\ . \\ . \\ \sum_{k=1}^{n-1} N_{h-1,p}(t_k) Q_k \end{bmatrix} \quad N = \begin{bmatrix} N_{1,p}(t1) & N_{2,p}(t1) & . & . & N_{h-1,p}(t1) \\ N_{1,p}(t2) & N_{2,p}(t2) & . & . & N_{h-1,p}(t2) \\ . & . & . & . & . \\ . & . & . & . & . \\ N_{1,p}(t_{n-1}) & . & . & . & N_{h-1,p}(t_{n-1}) \end{bmatrix}$$

La résolution du système linéaire permet de trouver les (h-1) points de contrôle. La figure 8 représente un nuage de points la courbe générée par approximation.

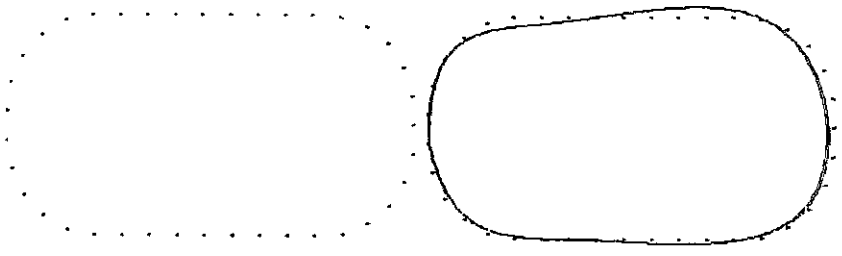


Figure 8. Approximation d'un nuage de points.

IV.2.2. Approximation locale [4, 5] :

C'est le même principe que l'approximation globale, sauf que nous considérons des sous ensembles des points de donnée et nous construisons plusieurs morceaux de courbes qu'il faut joindre pour construire la courbe entière. Le changement d'un point de donnée n'affecte la courbe que dans le morceau concerné.

IV.3. Critères de précision :

Ce critère s'applique essentiellement aux méthodes d'approximations. En effet, dans les méthodes d'interpolation, les seuls écarts mesurables sont dus à d'éventuelles difficultés numériques. Nous allons définir deux mesures de précision :

- L'écart supérieur C_1 qui permet de mesurer l'écart sur le point le moins bien approché, et donc s'assurer que la courbe est globalement proche de tous les points.

$$C1 = \sup_{i=0,\ldots,n} |Q_i - C(u_i)| \qquad (1.20)$$

- L'écart quadrique moyen C_2 qui permet de mesurer l'erreur moyenne effectuée sur un point, et donc de s'assurer que la courbe est globalement proche de tous les points.

$$C2 = \frac{\sqrt{\sum_{i=0}^{n} (Q_i - C(u_i))^2}}{n+1} \qquad (1.21)$$

Les critères de précision C_1 et C_2 sont donc tous les deux nécessaires pour juger la précision du résultat obtenu.

IV.4. Méthodes de paramétrisation [6] :

Le problème essentiel dans l'interpolation et l'approximation est la détermination des valeurs nodales et des vecteurs nodaux nécessaires à la résolution des

systèmes d'équations obtenus. Le choix aléatoire des paramètres peut donner des résultats imprévisibles (oscillations, formation de boucles, ...etc.). Pour éviter ces problèmes, nous allons définir quelques méthodes de paramétrisation.

IV.4.1. Méthode uniformément espacée :

C'est la méthode la plus simple. Elle consiste à diviser l'intervalle en sous intervalles égaux. Supposons que le domaine est [0,1] et que nous avons besoin de **n+1** paramètres, un paramètre pour chaque point de données, alors les paramètres sont calculés par :

$$\begin{cases} t_0 = 0 \\ t_i = \dfrac{i}{n} \\ t_n = 1 \end{cases} \quad \text{Avec } 1 \le i \le n-1 \qquad (1.22)$$

IV.4.2. Méthode de longueur de corde :

La paramétrisation avec cette méthode est basée sur le calcul des longueurs de cordes proches. La longueur L du polygone des points de donnée D_k (k=0..n) est donnée par :

$$L = \sum_{i=1}^{n} |D_i - D_{i-1}| \qquad (1.23)$$

Les paramètres sont donnés par :

$$\begin{cases} t_0 = 0 \\ t_k = \dfrac{1}{L} \sum_{i=1}^{k} |D_i - D_{i-1}| \\ t_n = 1 \end{cases} \qquad (1.24)$$

IV.4.3. Méthode centripète :

La méthode centripète est une extension de la méthode de longueur de corde. Pour ce cas, nous ajoutons une puissance (a) à la distance entre deux points de donnée adjacents. En général, la puissance a est égale à ½. La nouvelle longueur L du polygone des points de donnée D_k (k=0..n) est donnée par :

$$L = |D_i - D_{i-1}|^a \qquad (1.25)$$

Les paramètres sont donnés par :

$$\begin{cases} t_0 = 0 \\ t_k = \dfrac{1}{L}\sum |D_i - D_{i-10}|^a \\ t_n = 1 \end{cases} \qquad (1.26)$$

Si a=1, la méthode centripète sera réduite à la méthode longueur de corde.

IV.5. Génération du vecteur nodal [6,7]:

Après l'obtention des paramètres nodaux t_0, t_1, ...,t_n, nous avons besoin de calculer le vecteur nodal. Ce vecteur peut être calculé par deux méthodes :

IV.5.1. Méthode uniformément espacée :

La méthode uniformément espacée n'exige pas la connaissance préalable des paramètres. Dans cette méthode, les nœuds sont uniformément espacés et sont donnés par :

$$\begin{cases} U_0 = U_1 = ... = U_p = 0 \\ U_{j+p} = \dfrac{j}{n-p+1} \quad \text{pour} \quad j = 1,2,.....,n-p \\ U_{m-p} = U_{m-p+1} = ... = U_m = 1 \end{cases} \qquad (1.27)$$

IV.5.2. Méthode moyenne des paramètres :

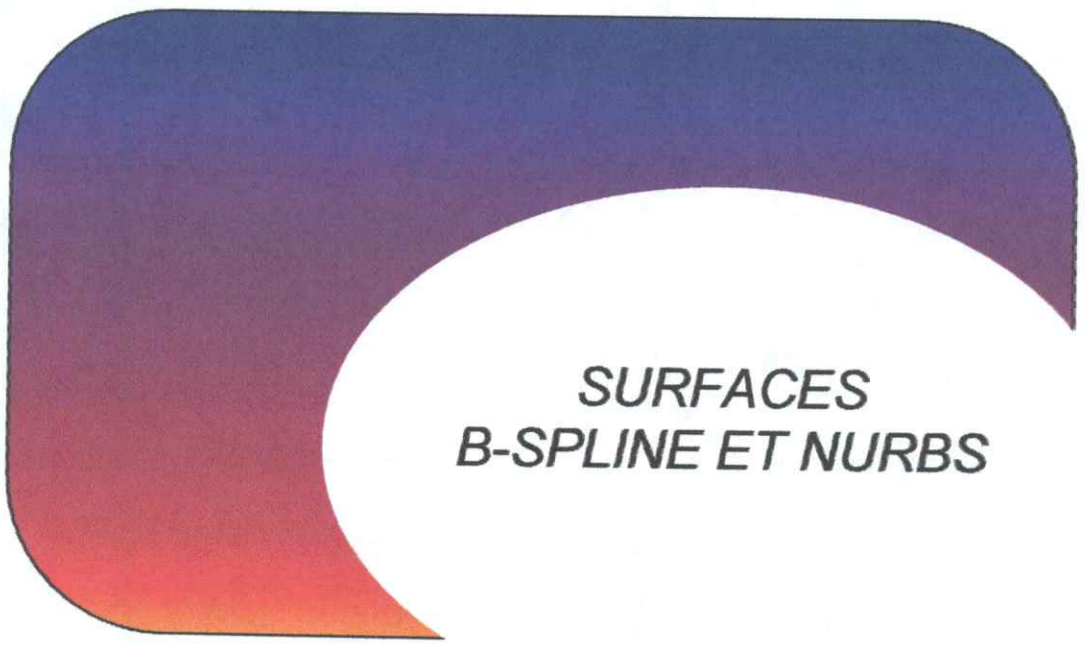
Cette méthode est basée sur le calcul de la moyenne des paramètres obtenus et les valeurs du vecteur nodal sont données par :

$$\begin{cases} U_0 = U_1 = ... = U_p = 0 \\ U_{j+p} = \dfrac{1}{p}\sum_{i=j}^{j+p-1} u_i \quad \text{pour} \quad j = 1,2,.....,n-p \\ U_{m-p} = U_{m-p+1} = ... = U_m = 1 \end{cases} \qquad (1.28)$$

V. Conclusion :

Dans ce chapitre, nous avons étudié les différentes méthodes de représentation des courbes et en particuliers les courbes paramétriques. Ensuite, nous avons présenté les courbes B-Spline et les étapes de reconstruction des courbes B-Spline par interpolation et par approximation et cela à partir d'un nuage de points. L'interpolation nous donne des courbes plus précises, mais elle nécessite un grand nombre de points de contrôle tandis que l'approximation nécessite moins nombre de points de contrôle et elle est plus flexible que l'interpolation.

CHAPITRE 2



SURFACES B-SPLINE ET NURBS

I. INTRODUCTION :

Nous avons étudié dans le premier chapitre les courbes paramétriques et en vérité ce n'est qu'une étape préparatoire dans la conception des surfaces. En pratique, la modélisation des surfaces est nécessaire dans la conception des pièces mécaniques de formes libres (gauches) (moules, matrices, ...etc.). Dans ce chapitre, nous allons étudier les différentes méthodes de représentation des surfaces paramétriques et non paramétriques. Ensuite, nous allons passer aux méthodes de conception des surfaces et en particulier les surfaces B-Spline et NURBS.

II. MODELISATION DES SURFACES [3, 4, 8] :

Il existe deux types de représentation des surfaces de formes libres dans la modélisation des surfaces et qui sont les représentations paramétriques et les représentations non paramétriques.

II.1. Surfaces non paramétriques :

Pour les surfaces non paramétriques, nous avons deux types de représentation :

- **Surfaces implicites** : ces surfaces sont définies par une fonction de trois variables :

$$f(x,y,z)=0 \qquad (2.1)$$

Ce type de représentation ne permet pas de représenter des surfaces de formes quelconques ainsi que la manipulation interactive est difficile.

- **Surfaces explicites** : ces surfaces sont données par l'équation suivante :

$$Z=f(x,y) \qquad (2.2)$$

Pour chaque (x, y), il correspond un z unique. Cette représentation est limitée et elle ne permet pas de représenter les surfaces fermées.

II.2. Surfaces paramétriques :

II.2.1. Définition des surfaces paramétriques :

Une surface paramétrique est définie par trois fonctions une pour chaque coordonnée. Chaque fonction dépend de deux paramètres u et v tel que u, v $\in$ [0,1]. La surface paramétrique est donnée par :

$$S(u,v)=(x(u,v),y(u,v),z(u,v)) \qquad (2.3)$$

Pour chaque point dans le plan paramétrique de coordonnées (u,v) lui correspond un point de la surface (voir figure 1).

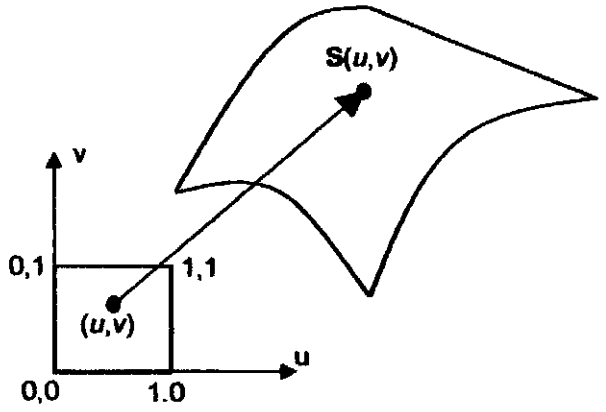


Figure 1. Localisation d'un point sur une surface paramétrique.

II.2.2. Vecteurs tangents à la surface paramétrique en un point :

Les vecteurs tangents à la surface paramétrique au point **S(u, v)** sont donnés par :

$$T_u = S'_u = \frac{\partial S}{\partial u} = \left(\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u}\right) \quad (2.4)$$

$$T_v = S'_v = \frac{\partial S}{\partial v} = \left(\frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v}\right) \quad (2.5)$$

Avec T_u est le vecteur tangent dans la direction u, T_v est le vecteur tangent dans la direction v. T_u et T_v définissent le plan tangent à la surface (voir figure 2).

II.2.3. Vecteur normal à la surface paramétrique en un point :

Le vecteur normal unitaire **n(u,v)** à la surface au point **S(u,v)** (voir figure 2) est le produit vectoriel des deux vecteurs tangents et il est donné par :

$$n = \frac{\frac{\partial C}{\partial u} \times \frac{\partial C}{\partial v}}{\left|\frac{\partial C}{\partial u} \times \frac{\partial C}{\partial v}\right|} \quad (2.6)$$

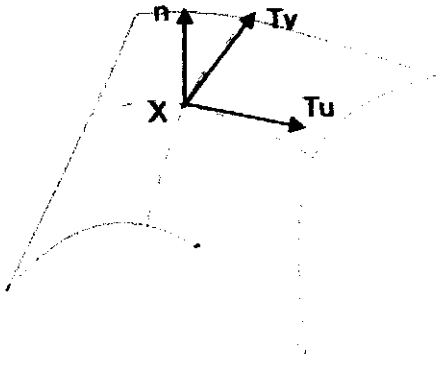


Figure 2. Vecteurs tangents et vecteur normal à la surface.

II.2.4. Courbes isoparamétriques :

Pour la surface paramétrée **S(u,v)**, si la valeur de *u* est fixée à u_1 et en laissant *v* varier, cela génère une courbe sur la surface appelée courbe isoparamétrique dans la direction v. Similairement, en fixant la valeur de *v* à v_1 et en laissant u varier, on

obtient une courbe isoparamétrique dans la direction u (voir figure 3). La surface paramétrique peut être vue comme l'union d'un nombre infini de courbes isoparamétriques.

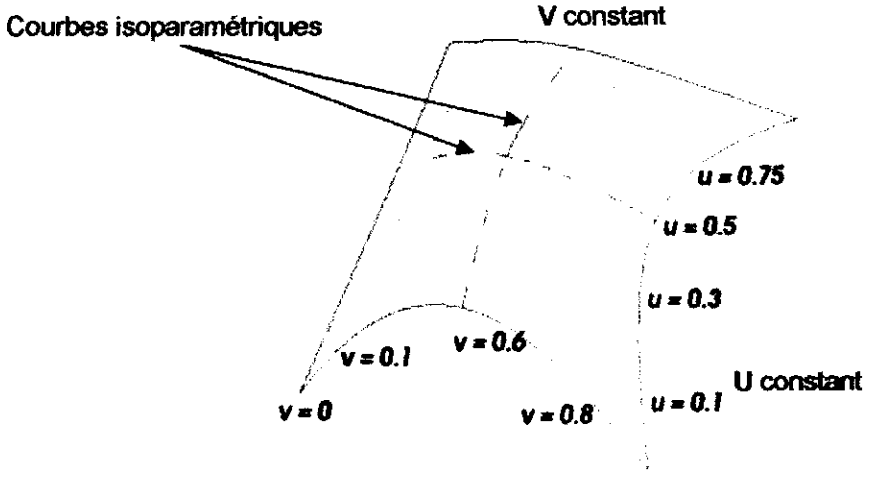


Figure 3. Courbes isoparamétriques d'une surface.

II.2.5 Courbures principales [2]:

Pour les courbes, il y a une seule courbure, par contre les surfaces ont plusieurs types de courbure. Le calcul de ces courbures permet de détecter les erreurs sur la surface. Par un point quelconque sur la surface, un nombre infini de courbes passent par ce point. Par conséquent, il y a un nombre infini de courbures à calculer. Pour surmonter ce problème, deux courbures principales sont définies, la courbure minimale K_1 et la courbure maximale K_2. Ces deux courbures sont les solutions de l'équation suivante :

$$(LN - M^2)R^2 + (2MF - GL - EN)R + EG - F^2 = 0 \qquad (2.7)$$

Avec *L, N, M, F, G et E* sont donnés par :

$$E = \frac{\partial P}{\partial u} \times \frac{\partial P}{\partial u} \qquad (2.8)$$

$$F = \frac{\partial P}{\partial u} \times \frac{\partial P}{\partial v} \qquad (2.9)$$

$$G = \frac{\partial P}{\partial v} \times \frac{\partial P}{\partial v} \qquad (2.10)$$

$$L = n \times \frac{\partial^2 P}{\partial u^2} \qquad (2.11)$$

$$M = n \times \frac{\partial^2 P}{\partial u \partial v} \qquad (2.12)$$

$$N = n \times \frac{\partial^2 P}{\partial v^2} \qquad (2.13)$$

La courbure moyenne de la surface en un point S (u, v) est donnée par :

$$H = \frac{K_1 + K_2}{2} \qquad (2.14)$$

La courbe gaussienne est donnée par :

$$K = K_1 \times K_2 \qquad (2.15)$$

Selon les valeurs des courbures H et *K* en un point p sur la surface, on peut déduire les propriétés suivantes :

- Si K = 0 alors p est un point parabolique (développable).
- Si K < 0 alors p est un point hyperbolique (selle de cheval).
- Si K> 0 alors p est un point elliptique,

  Si K1 > 0 et K2 > 0 alors p est un point concave.
  Si K1 < 0 et K2 < 0 alors p est un point convexe.

III. METHODES DE CONCEPTION DES SURFACES [5] :

Pour concevoir des surfaces de formes libres (surfaces gauches), différentes méthodes de conception peuvent être utilisées. Il existe deux méthodes qui peuvent aider le concepteur dans la conception des surfaces : les méthodes basées sur les points et les méthodes basées sur les courbes. La figure 4 montre les différentes méthodes de conception.

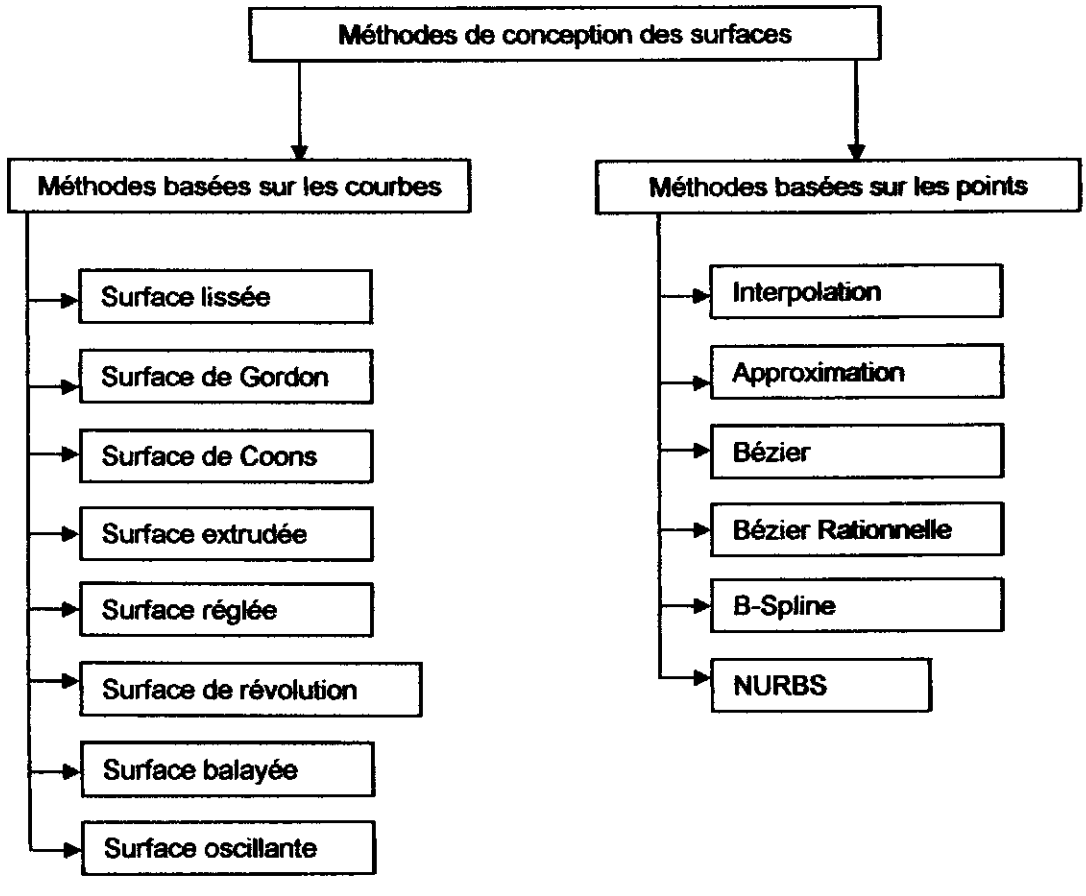


Figure 4. Méthodes de conception des surfaces.

III.1. Surfaces B-Spline [5]:

Parmi les surfaces paramétriques les plus utilisées en CAO, nous avons les surfaces B-Spline. Une surface B-Spline est définie par (voir figure 5) :

- Un réseau de (m+1)×(n+1) points de contrôle $\mathbf{p}_{i,j}$, où $0 \le i \le m$ et $0 \le j \le n$.
- Un vecteur nodal de h+1 nœuds dans la direction u, $U = \{0 = u_0, u_1, ..., u_h = 1\}$.
- Un vecteur nodal de k+1 nœuds dans la direction v, $V = \{0 = v_0, v_1, ..., v_k = 1\}$.
- Le degré *p* de la surface dans la direction u.
- Le degré *q* de la surface dans la direction v

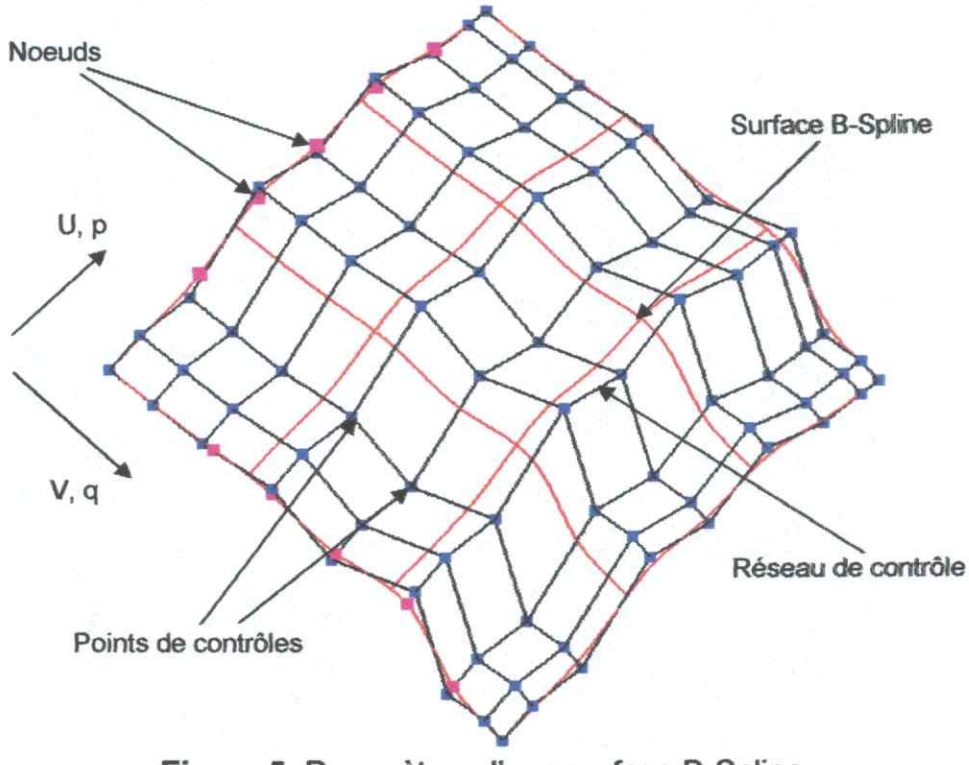


Figure 5. Paramètres d'une surface B-Spline.

La surface B-Spline est donnée par :

$$p(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u) N_{j,q}(v) p_{i,j} \qquad (2.16)$$

$N_{i,p}(u)$ et $N_{j,q}(v)$ sont les fonctions base B-Spline de degré p et q respectivement et sont données par :

$$N_{i,0}(u) = \begin{cases} 1 & \text{si} \quad u_i \le u < u_{i+1} \\ 0 & \text{sinon} \end{cases} \qquad (2.17)$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \quad (2.18)$$

$$N_{j,0}(v) = \begin{cases} 1 & si \quad v_i \le v < v_{i+1} \\ 0 & \text{sinon} \end{cases} \quad (2.19)$$

$$N_{j,q}(v) = \frac{v - v_j}{v_{j+p} - v_j} N_{j,q-1}(v) + \frac{v_{j+q+1} - v}{v_{j+q+1} - v_{j+1}} N_{j+1,q-1}(v) \quad (2.20)$$

On dit que la surface B-Spline est de degré (p,q). La fonction base du point de contrôle $\mathbf{p}_{i,j}$ est le produit de deux fonctions base B-Spline unidimensionnelles $\mathbf{N}_{i,p}(u)$ dans la direction *u* et $\mathbf{N}_{j,q}(v)$ dans la direction *v*.

III.1.1. Types des surfaces B-Spline [6]:

Les surfaces B-Spline peuvent être de trois types :

- Surface B-Spline pincée : passe par les points de contrôle $\mathbf{p}_{0,0}$, $\mathbf{p}_{m,0}$, $\mathbf{p}_{0,n}$, $\mathbf{p}_{m,n}$ et la surface est tangente au 8 segments du réseau de contrôle avec : ($u_0 = u_1 = \ldots = u_p = 0$ et $u_{h-p} = u_{h-p+1} = \ldots = u_h = 1$) et ($v_0 = v_1 = \ldots = u_q = 0$ et $u_{k-q} = u_{k-q+1} = \ldots = u_k = 1$).
- Surface B-Spline fermée : une surface B-Spline est fermée dans une direction si toutes les courbes isoparamétriques dans cette direction sont fermées.
- Surface B-Spline ouverte : la surface ne passe pas par les points $\mathbf{p}_{0,0}$, $\mathbf{p}_{m,0}$, $\mathbf{p}_{0,n}$, $\mathbf{p}_{m,n}$

III.1.2. Propriétés des surfaces B-Spline :

Les importantes propriétés des surfaces B-Spline sont les suivantes :

- Non négativité : la fonction $\mathbf{N}_{i,p}(u)\ \mathbf{N}_{j,q}(v)$ est non négative pour tout p, q, i, j et u et v dans l'intervalle [0, 1].
- Partition de l'unité : la somme de toutes les fonctions $\mathbf{N}_{i,p}(u)\ \mathbf{N}_{j,q}(v)$ est égale à 1 pour tout *u* et *v* dans l'intervalle [0, 1].

$$\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u) N_{j,q}(v) = 1 \quad (2.21)$$

- Propriété de l'enveloppe convexe : si (u,v) est dans $[u_i,u_{i+1}[\times[v_j,v_{j+1}[$, alors $\mathbf{S}(u,v)$ est contenue dans l'enveloppe convexe définie par les points de contrôle $p_{h,k}$, où $i-p \le h \le i$ et $j-q \le k \le j$.
- Schéma de modification locale : $\mathbf{N}_{i,p}(u)\mathbf{N}_{j,q}(v)$ est nulle si (u,v) est en dehors de l'intervalle $[u_i,u_{i+1}[\times[v_j,v_{j+q+1}[$.
- $\mathbf{S}(u,v)$ a une continuité C^{p-s} (respectivement C^{q-t}) dans la direction *u* (respectivement v direction) si u (resp., v) est un nœud de multiplicité *s* (respectivement t).
- Les courbes isoparamétriques $\mathbf{S}(0,v)$, $\mathbf{S}(1,v)$, $\mathbf{S}(u,0)$ et $\mathbf{S}(u,1)$ sont les courbes de frontières de la surface.

- Si m=p, n=q et U = { 0,0,0,.....,0.,1,1,.....,1} et V = { 0,0,0,.....,0.,1,1,.....,1}, la surface B-Spline devient une surface de Bézier.
- Invariance affine.
- La surface est indépendance du système de coordonnées.
- Les identités fondamentales suivantes doivent être satisfaites :

$$h = m + p + 1 \tag{2.22}$$

$$K = n + q + 1 \tag{2.23}$$

III.2. Surfaces NURBS [7, 1]:

Les surfaces B-Spline sont des surfaces polynomiales, et par conséquent il est impossible de représenter les surfaces coniques (sphère, ellipsoïde, ...etc.). D'où la nécessite de passer à une autre représentation, c'est les NURBS.

Une surface NURBS est définie par les paramètres suivants (voir figure 6) :

- Un réseau de (m+1)×(n+1) points de contrôle $\mathbf{p}_{i,j}$, où $0 \le i \le m$ et $0 \le j \le n$,
- Pour chaque point de contrôle est associé un poids $w_{i,j} \ge 0$,
- Un vecteur nodal de h+1 nœuds dans la direction u, $U = \{0 = u_0, u_1, ..., u_h = 1\}$,
- Un vecteur nodal de k+1 nœuds dans la direction v, $V = \{0 = v_0, v_1, ..., v_k = 1\}$,
- Le degré p de la surface dans la direction u,
- Le degré q de la surface dans la direction v.

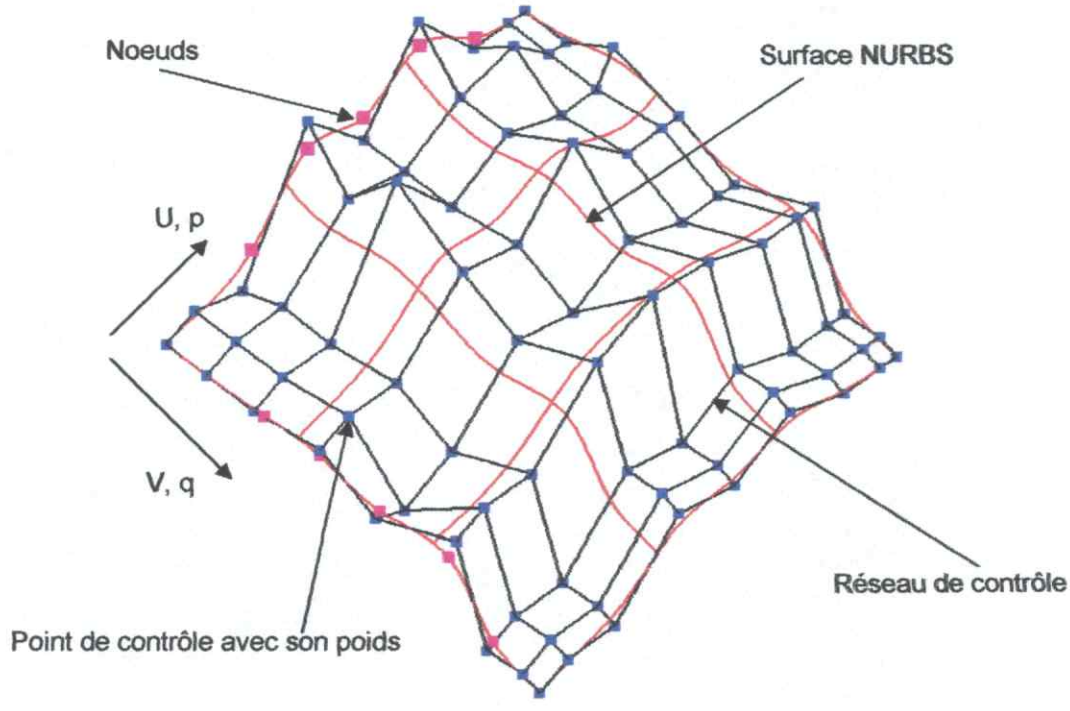


Figure 6. Paramètres d'une surface NURBS.

La surface NURBS est donnée par:

$$p(u,v) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)w_{i,j}p_{i,j}}{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \qquad (2.24)$$

$N_{i,p}(u)$ et $N_{j,q}(v)$ sont les fonctions base B-Spline de degré p et q respectivement .

III.2.1. Propriétés des surfaces NURBS :

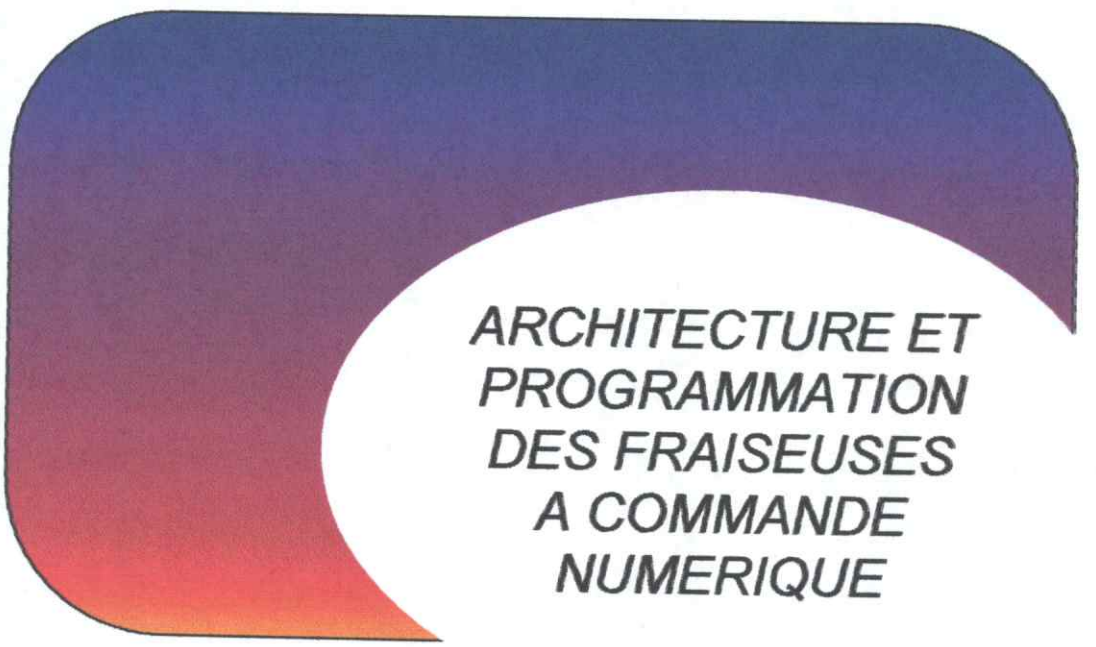
En plus des propriétés des surfaces B-Spline, les NURBS ont les propriétés suivantes :

- L'augmentation de la valeur du poids $w_{i,j}$ du point de contrôle $\mathbf{p}_{i,j}$ attire une portion de la surface vers le point de contrôle $\mathbf{p}_{i,j}$, la diminution du poids éloigne la surface de ce point.
- N'importe quelle courbe isoparamétrique sur la surface NURBS est une courbe NURBS définie par un ensemble de points de contrôle.
- Les courbes isoparamétriques **S**(0,v), **S**(1,v), **S**(u,0) et **S**(u,1) sont les courbes de frontières.
- Si tous les poids $w_{i,j}$ sont égaux, la surface NURBS devient une surface B-Spline, si en plus m=p, n=q , U = {0,0,.....,0,1,1,.....1} et V = {0,0,.....,0,1,1,.....1}, la surface NURBS devient une surface de Bézier.
- Invariance projective.
- Représentation des surface coniques (sphère, ellipsoïde, ... etc.).

IV.CONCLUSION :

Dans ce chapitre nous avons étudié les différentes méthodes de conception des surfaces paramétriques ainsi que les différentes méthodes de représentation des surfaces. Nous avons considéré en particulier les surfaces B-Spline et NURBS. Cette dernière est la plus puissante et la plus générale.

CHAPITRE 3


ARCHITECTURE ET
PROGRAMMATION
DES FRAISEUSES
A COMMANDE
NUMERIQUE

I. INTRODUCTION :

Après l'étape de conception et d'analyse des surfaces gauches conçues, il faut choisir les machines à utiliser pour usiner ces surfaces et qui sont en général des fraiseuses. En raison de la complexité géométrique de ces surfaces, les machines outils utilisées doivent être commandées numériquement.

Dans ce chapitre, nous allons considérer l'architecture et la programmation des fraiseuses à commande numériques à 3 axes puisque ces machines sont très utilisées dans le vécu industriel et permettent d'obtenir des pièces simples et des pièces complexes.

II. DEFINITION D'UNE MACHINE-OUTIL [9]:

La machine-outil est une machine fixe actionnée par un moteur, servant à façonner les matières solides, et particulièrement les métaux. Le façonnage est réalisé par l'enlèvement de la matière d'une pièce, ou par la déformation d'une pièce sinon par l'ajout de la matière à une pièce. Les machines-outils sont à la base de l'industrie moderne et sont utilisées dans la fabrication des machines et des pièces.

III. DEFINITION D'UNE MACHINE-OUTIL A COMMANDE NUMERIQUE [10] :

Une machine outil à commande numérique (MOCN) est une machine d'usinage à cycle automatique programmable. La machine est commandée par des instructions numériques fournies par un calculateur. Les organes mobiles de la machine sont motorisés et gérés par un automatisme qui contrôle les mouvements de ces organes. La machine outil est constituée principalement par une partie opérative (usinage) et une partie commande (commande d'usinage). Les deux compartiments se réunissent pour composer la machine outil à commande numérique. Les principales machines outils à commande numérique sont :

- **La fraiseuse** : utilisée pour la réalisation des pièces prismatiques et sur laquelle on peut réaliser des opérations de fraisage, de perçage et de gravure.
- **Le tour** : utilisé pour la réalisation des pièces de révolution et sur lequel on peut réaliser des opérations de tournage et de perçage.

Le rôle principal de ces machines est de réaliser physiquement des mouvements de coupe nécessaires à l'obtention des surfaces désirées, ainsi que les mouvements d'avance de l'outil par rapport à la pièce dans le but d'obtenir des surfaces précises.

III.1. Classification des MOCN [11] :

Les différents types de déplacements des MOCN sont les suivants :

- **Point à point** : le contrôle se fait sur la position à atteindre seulement et les déplacements ne sont pas contrôlés.
- **Paraxial** : un seul axe de déplacement est asservi en position et en vitesse et la vitesse est contrôlée dans des directions parallèles aux axes de la machine outil.

- **Contournage :** plusieurs axes sont asservis simultanément en vitesse et en position. Ces MOCN supportent généralement l'interpolation linéaire et circulaire. Le contournage peut être dans plan (2D) ou dans l'espace (3D). Le domaine d'utilisation de ce type de machines est l'usinage des surfaces complexes.

III.2. Caractéristiques des machines-outils à commande numérique :

Une machine-outil à commande numérique doit satisfaire certaines caractéristiques permettant de réaliser des cycles d'usinage compétitifs. Ces caractéristiques sont :

- La puissance et la vitesse doivent être élevées.
- La machine doit être robuste et elle doit avoir une bonne résistance à l'usure.
- Déplacement rapide, précis, accélération et décélération rapide.
- Frottement et jeu très faibles.
- Peu de vibration.
- Faible échauffement.

III.3. Choix de la MOCN appropriée à l'usinage [12] :

Les MOCN permettent d'usiner des surfaces de plusieurs formes différentes, un classement par famille des pièces permet de choisir la machine nécessaire à l'usinage. Pratiquement, on cherche à faire un maximum d'usinage sans démontage de la pièce afin de réduire les temps d'usinage. La sélection finale de la MOCN appropriée doit tenir compte de la forme et des dimensions de la pièce à usiner, de la puissance nécessaire à la broche ainsi que de la capacité de la machine.

IV. ARCHITECTURE DES FRAISEUSES A COMMANDE NUMERIQUE [12] :

Les fraiseuses à commande numérique peuvent réaliser des opérations de contournage. L'outil est fixé à une broche qui le fait tourner (mouvement de coupe), l'outil peut se déplacer sur la pièce à usiner suivant 3 axes (la broche n'est pas comprise) (mouvement d'avance). La fraiseuse à commande numérique est constituée principalement par deux parties complémentaires la partie commande et la partie opérationnelle (voir figure 1).

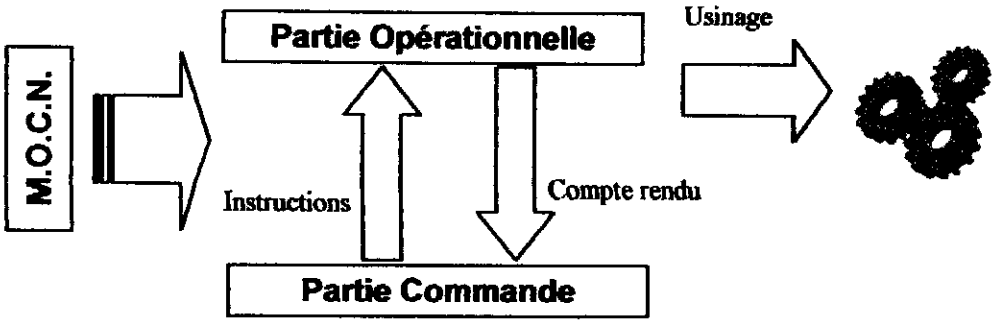


Figure 1. Réaction entre partie opérationnelle et commande.

IV.1. Partie opérationnelle :

Cette partie englobe les éléments mécaniques de la fraiseuse qui agissent directement sur la pièce durant l'usinage. Cette partie est constituée par :

Bâti : c'est une plateforme qui assure le guidage des axes des mouvements. Le bâti doit être rigide et doit supporter la chaleur et les vibrations.

Broche : elle supporte l'outil d'usinage. La broche crée le mouvement de coupe donné à l'outil nécessaire à l'usinage par un mouvement de rotation [5].

Porte outil : assurer la liaison entre l'outil et la broche.

IV.1.1. Axes de la machine [13,14] : les axes de déplacement mettent en mouvement les parties mobiles des fraiseuses avec de fortes accélérations par des glissières, Il existe deux types d'axes, axes de translation et axes de rotation.

IV.1.1.1. Mouvements de translation :

- **Axe du mouvement Z :** cet axe est parallèle à la broche principale.
- **Axe du mouvement X :** cet axe est horizontal et il représente le mouvement longitudinal et il est perpendiculaire aux axes Z et Y.
- **Axe du mouvement Y :** cet axe représente le mouvement transversal et il est perpendiculaire aux axes Z et X et forme un trièdre direct.

Les axes X, Y et Z forment un trièdre direct. On peut construire ce trièdre avec la règle de la main droite (voir figure 2).

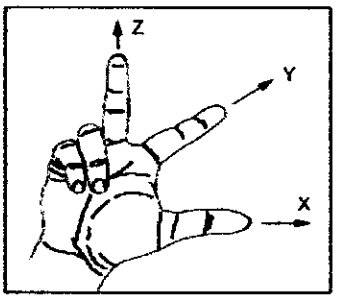


Figure 2. Axes de translation de la machine.

IV.1.1.2. Mouvements de rotation :

Les mouvements de rotation A, B et C sont les mouvements de rotations effectués autour des axes X, Y et Z respectivement. L'orientation positive d'un axe de rotation est celle du sens de vissage d'une vis (pas à droite) (voir figure 3).

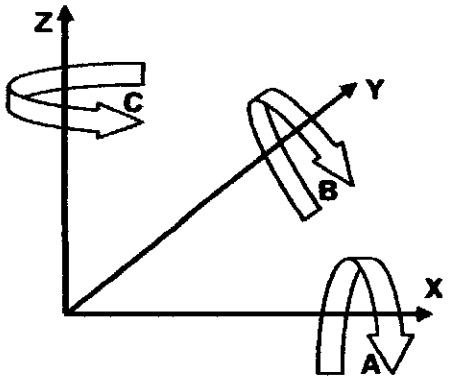


Figure.3 : Axes rotatifs autour de X, Y et Z.

IV.1.2. Types d'axes des fraiseuses :

Les axes d'une MOCN peuvent être :

- **Axe numérique :** c'est un axe sur lequel le mouvement est contrôlé en position et en vitesse et par conséquent une infinité de positions sont possibles.

- **Demi axe numérique :** c'est un axe sur lequel le mouvement est contrôlé en position ou en vitesse et par conséquent un ensemble fini de positions sont possibles.

- **Axe indexé :** c'est un axe autorisant un ensemble très réduit de positions.

IV.2. Partie commande :

Le déplacement de l'outil sur la trajectoire d'usinage est décrit par l'opérateur à l'aide d'un programme qui est constitué des coordonnées des différents points de passage de l'outil par rapport à la pièce. La partie commande de la machine s'occupe de contrôler et de gérer les mouvements à partir du programme d'usinage qui est transmis à la partie opérationnelle soit par une ligne externe (ordinateur, serveur) ou par le pupitre de commande. La partie commande est constituée principalement par :

IV.2.1. Pupitre de commande : est un écran doté d'un clavier pour commander et contrôler la machine. Il sert aussi à dialoguer avec la machine (envoi des instructions et reçoit des comptes rendus). L'écran sert à visualiser le programme d'usinage et la pièce à usiner.

IV.2.2. Armoire électronique : elle relie le DCN et la machine et elle englobe tous les composants électroniques (câbles, transistors, fusibles,...etc.).

IV.2.3. Directeur de commande numérique (DCN) [13,14] : à partir d'un programme d'usinage, le directeur de commande numérique donne des ordres aux servocommandes des axes de la machine et contrôle les organes mobiles de la machine en position et en vitesse (contrôle des axes). Il est doté d'un microprocesseur préprogrammé pour exécuter les fonctions de la commande numérique (CN) ainsi d'une mémoire où les programmes sont chargés. Le DCN réalise les tâches suivantes :

- Interprétation du programme d'application.
- Détermination des phases du travail.
- Calcul et obtention des trajectoires.
- Gestion des données et des mesures.
- Surveillance et détection des erreurs.
- Correction nécessaire des erreurs.
- Commande des actionneurs auxiliaires.
- Traitement des informations de sécurité.

IV.3. Outil d'usinage :

L'outil d'usinage (la fraise) est un cylindre avec une extrémité dotée de dents coupantes. La fraise est produite par une matière solide, elle est caractérisée par :

- Le sens de coupe.
- Le nombre et le type de dents.
- La géométrie de coupe.
- La matière de l'outil.

IV.4. Jauge d'outil [13] :

La jauge d'outil est une distance entre l'arrête coupante de l'outil et le point de référence de la broche. La figure 4 montre les jauges d'outil avec : (L) la longueur de l'outils, (R) le rayon de la fraise, (@) le rayon de bout de la fraise.

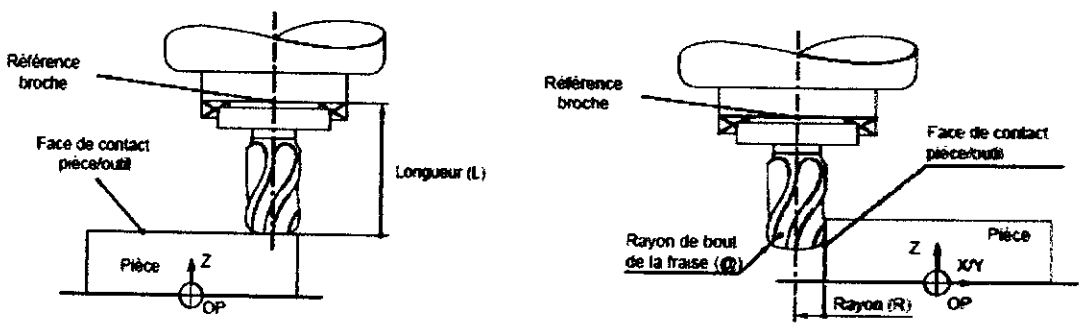


Figure 4. Jauges d'outil.

IV.5. Types de fraiseuses :

Il existe deux types de fraiseuses :

- **Fraiseuse verticale [14]** : pour ce type de fraiseuses, la broche est verticale et l'axe Z est parallèle à la broche, l'axe X et l'axe Y sont horizontaux (voir figure 5).

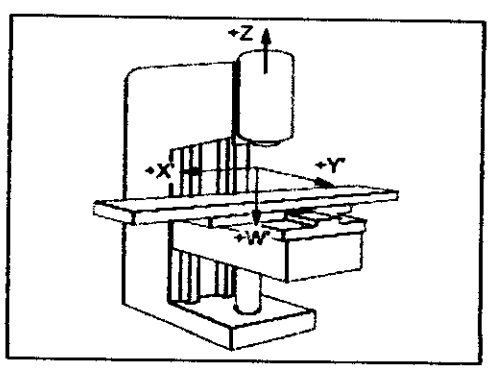


Figure 5. Fraiseuse verticale.

- **Fraiseuse horizontale [14]** : pour ce type de fraiseuses, la broche est horizontale et l'axe Z est parallèle à la broche.

V. PROGRAMMATION DES FRAISEUSES [12] :

Un programme est l'interprétation des algorithmes dans un langage compréhensible par le directeur de commande numérique d'une machine, et ce dernier envoi des commandes d'usinage à la partie opérationnelle. Le langage standard de la programmation des fraiseuses est le langage ISO (International Standardization Organization). Il existe plusieurs modes de programmation des fraiseuses.

V.1. Langage ISO :

La programmation se déroule soit au niveau du pupitre de commande (la majorité des systèmes offrent cette possibilité par commande directe de la fraiseuse) soit par la programmation sur un poste externe et ensuite le programme est transmis à la machine (le poste de programmation peut être une console spécialisée reproduisant l'interface utilisateur de la CN ou un ordinateur standard utilisant un logiciel spécialisé ou un éditeur de texte).

V.2. Langage conversationnel :

Le langage conversationnel réside au niveau du pupitre, il permet de créer, de modifier, de visualiser et de simuler un programme pièce pendant l'usinage d'une autre pièce. Le programmeur doit être présent sur le site au cours d'usinage.

V.3. Langage de haut niveau :

Ce langage normalisé facilite la programmation, il présente l'avantage d'être indépendants de la CN. Un traducteur spécifique pour chaque directeur de commande numérique est mis en pratique pour convertir le programme écrit dans un langage évolué en un langage compris par le DCN.

V.4. Programmation par FAO :

La programmation dans ce mode est faite par la sélection de la surface à usiner et le choix des outils appropriés ainsi que la définition de la méthode d'usinage et les paramètres de coupe. Ce mode présente un gain de temps très important particulièrement pour les surfaces complexes car il exploite l'aspect géométrique des pièces.

V.5. Différents éléments d'un programme ISO (NF Z 68-010) [11,12] :

Un programme d'usinage est une suite d'instructions écrites dans un langage codé propre à la commande numérique. Le DCN interprète les instructions en une forme électronique. Le programme est une suite de blocs, chaque bloc représente une séquence d'usinage, de son coté le bloc est constitué par un ensemble de mots. Chaque programme doit contenir un caractère de début et un caractère de fin du programme (voir figure 6).

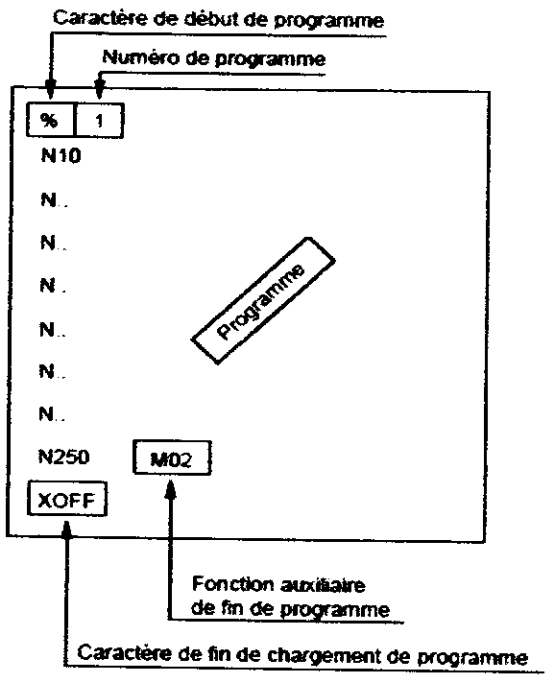


Figure 6. Format général d'un programme.

Les principaux paramètres d'un programme sont :

- Le début du programme est caractérisé par % suivi du numéro du programme et éventuellement un commentaire entre parenthèse.
- Des commentaires entre parenthèse.
- La fin du chargement du programme est caractérisée par XOFF.
- La fin du programme est caractérisée par code M02.
- Ensemble d'instructions à exécuter.

V.5.1. Format du bloc : [12]

Un bloc est un ensemble de mots écrits dans une syntaxe. Chaque bloc à un numéro identifiant ce bloc (voir figure 7)

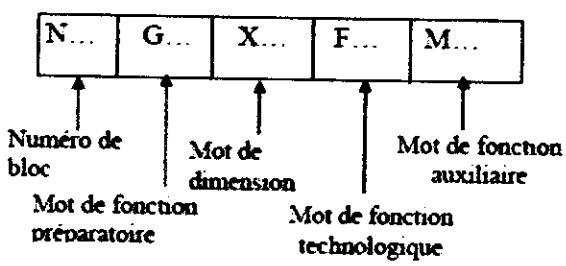


Figure 7. Format du bloc.

V.5.2. Format du mot [12] :

Un mot dans un programme (voir figure 8) est une instruction ou une fonction à exécuter par la machine. Les fonctions employées peuvent être représentées par :

- **G** : fonction préparatoire.
- **F** : fonction vitesse d'avance (« Feederate » = Avance).
- **S** : fonction vitesse de la broche (« Speed »= Vitesse).

- **T** : fonction outil (« Tool »=outil).
- **M** : fonction auxiliaire («Miscellaneous »=divers).
- **X** : mouvement de translation suivant l'axe X.
- **Y** : mouvement de translation suivant l'axe Y.
- **Z** : mouvement de translation suivant l'axe Z.
- **A** : mouvement de rotation suivant l'axe X.
- **B** : mouvement de rotation suivant l'axe Y.
- **C** : mouvement de rotation suivant l'axe Z.

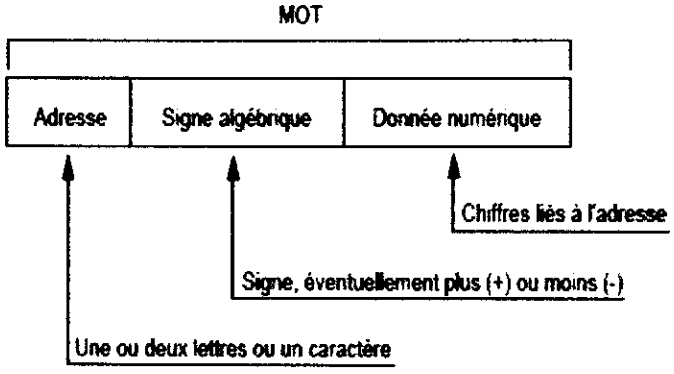


Figure 8. Format du mot.

V.6. Types de fonctions :

Il existe deux types de fonctions, préparatoires G et auxiliaires **M**. Les fonctions préparatoires sont de deux types :

1. **Fonction G modale :** la validité de ces fonctions est maintenue jusqu'à ce qu'une autre fonction de même famille interrompe leur validité.
2. **Fonction G non modale :** fonction valide seulement dans le bloc qu'il l'inclut. La fonction sera détruite en fin du bloc.

Les fonctions auxiliaires **M** sont de deux types :

1. **Fonction M modale :** la validité de ces fonctions est maintenue jusqu'à ce qu'une autre fonction de même famille interrompe leur validité.
2. **Fonction M non modale :** fonction valide seulement dans le bloc qu'il inclut. La fonction sera détruite en fin du bloc.

Les fonctions auxiliaires **M** peuvent être :

- **Fonction M avant :** ces fonctions sont exécutées avant le déplacement sur les axes programmés.
- **Fonction M après :** ces fonctions sont exécutées après le déplacement sur les axes programmés.

VI. CONCLUSION :

Dans ce chapitre, nous avons analysé la constitution des machines outils à commandes numérique et particulièrement l'architecture des fraiseuses à commande numérique. Ensuite, nous sommes passé à la programmation de ces fraiseuses en commençant par les différents langages de programmation et en terminant par les principales fonctions du programme d'usinage « G-Code ».

CHAPITRE 4

USINAGE DES SURFACES GAUCHES

I. INTRODUCTION :

Les pièces de formes gauches sont devenues des pièces courantes dans notre vie quotidienne en raison de l'évolution des styles et des techniques d'usinage de ces pièces. L'arrivé de la commande numérique et de la CAO à remis en cause le processus global de fabrication des pièces de formes gauches. La mécanique moderne nous offre la possibilité d'usiner des surfaces complexes via le calcul des trajets d'usinage et la réalisation de la pièce à partir d'un modèle géométrique.

Dans ce chapitre, nous allons analyser le processus d'usinage des surfaces gauches sur des fraiseuses à commande numérique à trois axes.

II. PROCESSUS DE REALISATION DES FORMES GAUCHES [15] :

Le processus de réalisation des pièces de formes gauches est un processus industriel qui a subi une grande évolution dans le temps selon les besoins. Ces évolutions ont données l'introduction de la Conception et Fabrication Assistées par Ordinateur (CFAO) et de la commande numérique. Le processus de réalisation des pièces des formes gauches est une suite d'activités dont le but est d'obtenir des formes répondant aux exigences inscrites dans le cahier des charges. Ce processus passe par deux étapes essentielles.

1. **Conception :** cette étape exprime l'ensemble des contraintes fonctionnelles écrites dans un cahier de charge sous forme de surfaces ainsi que d'autres formes géométriques conçues à l'aide d'un system de CAO.

2. **Fabrication :** cette étape est l'usinage effectif et qui est un résultat des transformations des formes conçues dans la première étape en des formes réelles répondant aux contraintes décrites. Entre ces deux étapes, la génération des trajectoires d'usinage est une étape intermédiaire. Le générateur des trajectoires d'usinage décrit la géométrie de la pièce sous forme de déplacements d'un outil dans un format adapté aux DCN. La figure 1 montre le processus de réalisation des pièces de formes gauches.

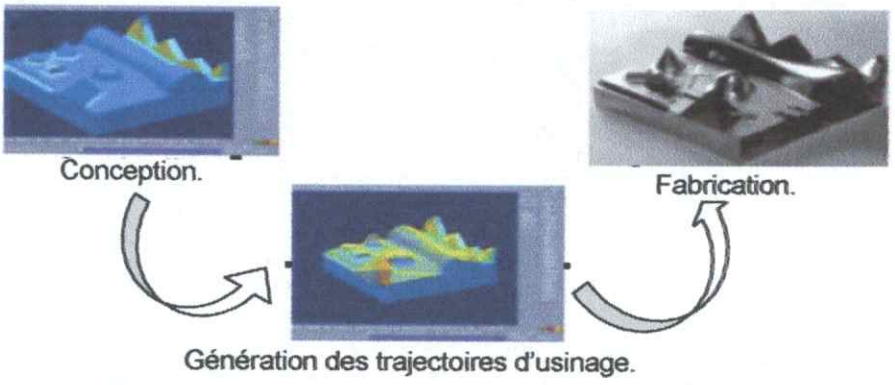


Figure 1. Processus de réalisation des pièces de formes gauches.

III. TYPES D'USINAGE EN FRAISAGE [15, 16, 17] :

Les opérations de fraisage regroupent les opérations de fraisage en bout et les opérations de fraisage en roulant et dont le choix dépend du type d'usinage et de la forme à usiner.

- **Fraisage en bout :** dans ce type de fraisage, l'axe Z de la fraise est perpendiculaire à la surface à usiner (voir figure 2).

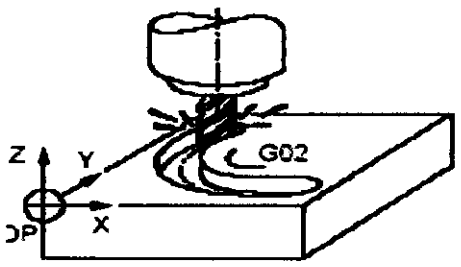


Figure 2. Fraisage en bout.

- **Fraisage en roulant (de profil) :** dans ce type de fraisage, la surface à usiner est tangente à la génératrice de la fraise (voir figure 3), c'est donc le profil de la fraise qui va faire l'enlèvement de la matière.

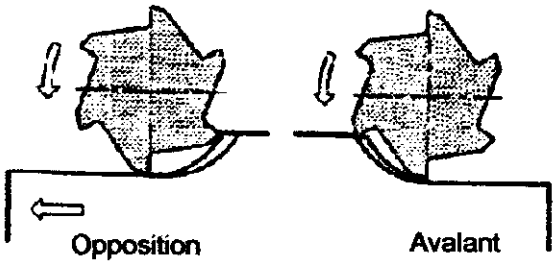


Figure 3. Fraisage de profil.

Lors de l'usinage, nous avons le choix entre deux modes d'usinage.

- **Usinage en avalant :** dans ce type d'usinage, les dents de la fraise sont prises avec la matière avançant dans le sens inverse du sens d'usinage, l'outil attaque la pièce par une épaisseur de copeau maximale et il termine le travail avec une épaisseur nulle du copeau (voir figure 4).

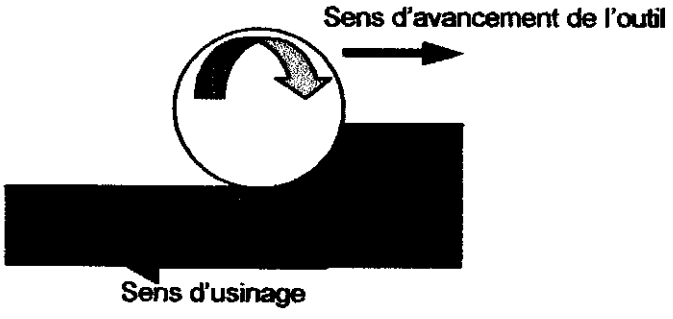


Figure 4. Usinage en avalant.

- **Usinage en opposition :** dans ce type d'usinage, les dents de la fraise sont prises avec la matière avançant dans le même sens d'usinage, l'outil attaque la pièce par une épaisseur du copeau nulle et il termine le travail avec une épaisseur de copeau maximale (voir figure 5).

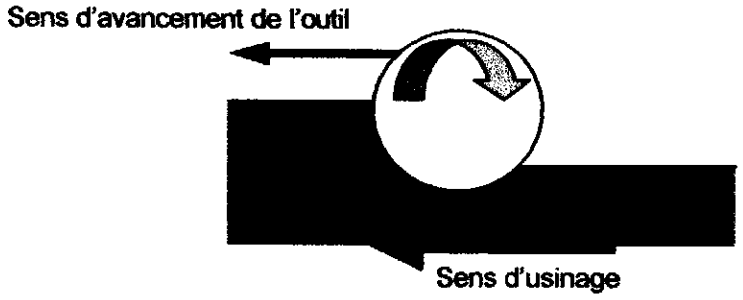


Figure 5. Usinage en opposition.

IV. USINAGE A 3 AXES ET A 5 AXES :

Les surfaces gauches peuvent être usinées sur des fraiseuses à 3 axes ou à 5 axes en fonction des possibilités d'accès. L'usinage à 5 axes offre deux degrés de libertés de plus par rapport à l'usinage à 3 axes. En plus, dans l'usinage à 3 axes il y a une seule position de tangence entre l'outil et la surface, par contre, dans l'usinage à 5 axes, il y a une infinité de positions de tangence entre l'outil et la surface (voir figure 6). Les deux axes supplémentaires permettent :

- D'usiner des surfaces par orientation des axes d'outil de façon à éliminer les collisions entre l'outil et la surface.
- D'optimiser la vitesse de coupe en cas d'utilisation des fraises hémisphériques.
- De changer la géométrie de la fraise.

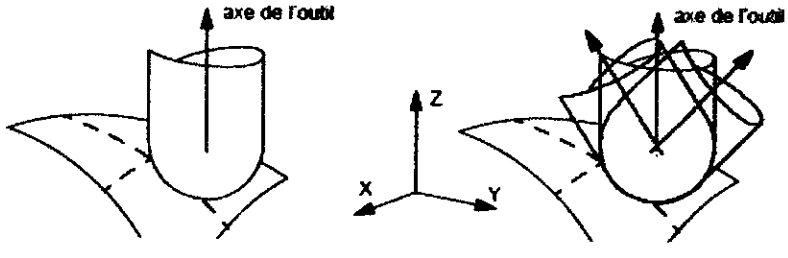


Figure 6. Usinage à trois axes et usinage à cinq axes.

V. CHOIX ET TYPES DE FRAISES :

Dans la mécanique moderne, il n'existe pas de forme d'outil qui permet d'obtenir une surface gauche en un seul mouvement élémentaire. Donc, le choix de l'outil est le résultat d'un compromis entre la rigidité de l'outil, la cinématique de la machine et la forme de la pièce à usiner. Les fraises utilisées pour l'usinage des surfaces gauches sont les suivantes :

- **Fraise cylindrique** : l'arête coupante prend la forme d'un cylindre (voir figure 7.b).
- **Fraise hémisphérique** : l'arête coupante prend la forme d'une sphère et elle est caractérisée par le rayon R (voir figure 7.b).
- **Fraise torique** : l'arête coupante prend la forme d'un tore et elle caractérisé par R le rayon de la fraise et r le rayon de bout de la fraise (voir figure 7.c).

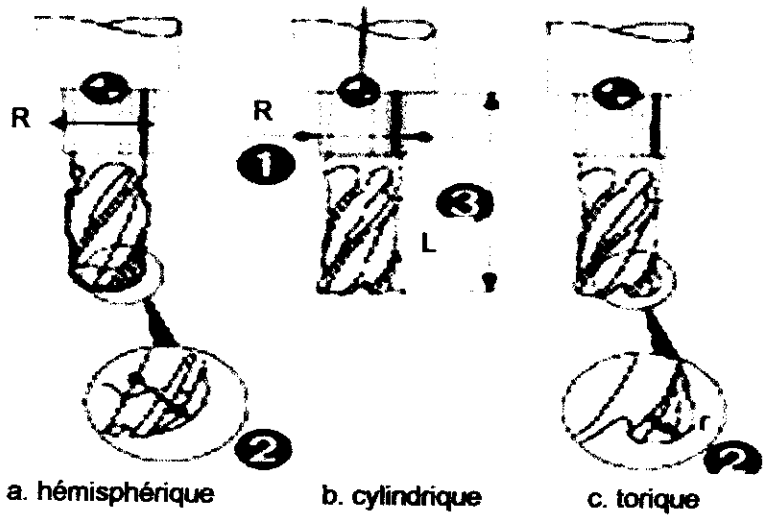


Figure 7. Types de fraises.

VI. CALCUL DE LA POSITION DU CENTRE D'OUTIL :

Généralement, lors de l'usinage des surfaces gauches sur des fraiseuses à commande numérique à trois axes, l'outil utilisé est hémisphérique. La condition principale pour le calcul de la position d'outil est la tangence entre l'outil et la surface à usiner. Les paramètres suivants permettent le calcul de la position d'outil (voir figure 8).

- CC (cutter contact) : le point de contact entre l'outil et la surface.
- CE : le point centre de l'outil.
- CL (cutter location) : le point extrémité de l'outil.
- Le vecteur $\vec{n}$: c'est le vecteur normale à la surface au point de contacte.
- Le vecteur $\vec{u}$: c'est l'axe de l'outil.
- r : le rayon de l'outil hémisphérique.

La position d'outil est donnée par :

$$\vec{OC_E} = \vec{OC_C} + r\vec{n} \qquad (4.1)$$

$$\vec{OC_L} = \vec{OC_E} - r\vec{u} = \vec{OC_E} = \vec{OC_C} + r\vec{n} - r\vec{u} \qquad (4.2)$$

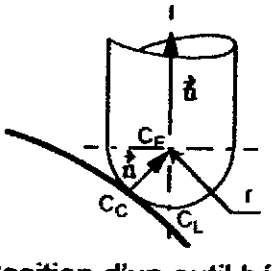


Figure 8. Position d'un outil hémisphérique.

Ces équations permettent de calculer les coordonnées du centre de l'outil à partir du point de contact ou bien les coordonnées du point de contact à partir du centre d'outil.

VI.1. Méthodes du calcul de la position d'outil :

VI.1.1. Calcul par offset de la forme :

Cette méthode est basée sur le calcul de l'offset de la surface à usiner avec une valeur égale à la valeur du rayon de l'outil. La position d'outil tangente à la surface à usiner appartient à la surface offset. L'avantage de cette méthode est qu'elle évite les interférences locales, mais cela n'interdit pas de rencontrer des problèmes tels que la surface offset peut interférer avec elle-même et générer des boucles (voir figure 9).

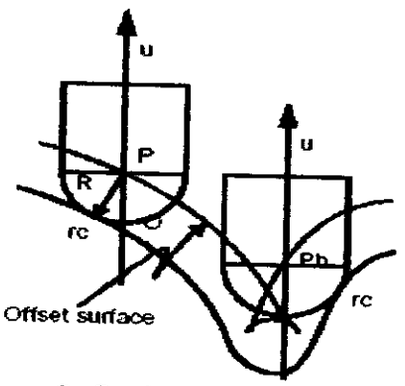


Figure 9. Surface offset d'une surface.

IV.1.2. Calcul par la méthode de copiage informatique :

C'est une méthode de posage par le centre d'outil. L'outil glisse le long d'une droite jusqu'à ce qu'il touche la surface à usiner (voir figure 10). Ce glissement est traduit par un algorithme de balayage d'une direction. Lors d'une opération du copiage classique, l'outil suit la surface et il tombe sur la surface jusqu'à ce qu'il la touche.

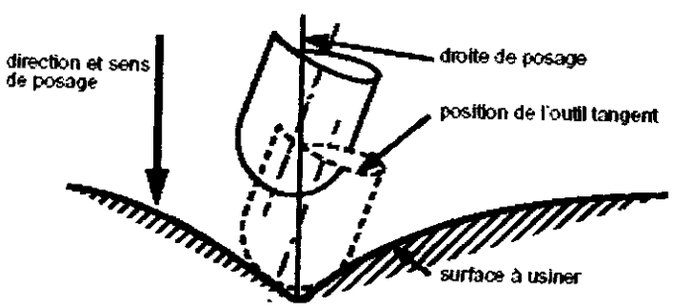


Figure 10. Copiage informatique.

VI.2. Interférences et erreurs d'usinage [15,18] :

VI.2.1. Problème d'interférence :

L'interférence est un problème qu'on peut rencontrer lors d'usinage des surfaces gauches. Deux cas d'interférence peuvent être rencontrés.

- **Interférence locale :** lorsque l'outil usine une partie concave où le rayon de courbure de cette partie est plus petit que le rayon de l'outil, alors on est dans une situation d'interférence locale (voir figure 11). Ce type d'interférence est lié à la partie active de l'outil.

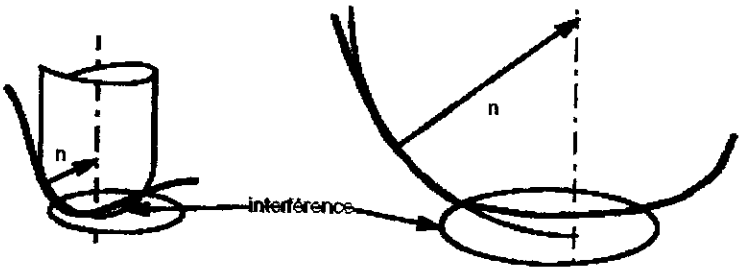


Figure 11. Interférence locale.

- **Interférence globale (collision) :** ce type d'interférences concerne le corps de l'outil ainsi que le porte outil et l'environnement d'usinage et dépend de la longueur de l'outil ainsi que de la forme de la pièce (voir figure 12).

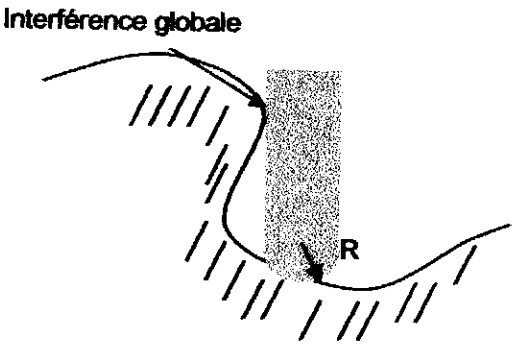


Figure 12. Interférence globale (collision).

VI.2.2. Erreurs d'usinage :

Pour usiner une surface gauche, il faut spécifier deux paramètres d'usinage. Le premier est la hauteur de crête (pas transversal) et le second est l'erreur de flèche (pas longitudinal).

- **Erreur de crête :** le pas d'usinage transversal peut être calculé en respectant soit un critère de distance maximale entre deux passes adjacentes ou bien une hauteur de crête h_c (voir figure 15). La distance peut être donnée entre les points de contact ou entre les axes de l'outil.

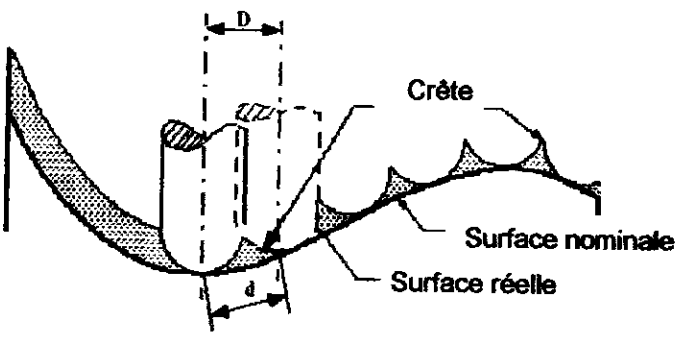


Figure 13 : Erreur de crête.

- **Erreur de flèche :** le trajet d'outil est décrit par des interpolations linéaires (un ensemble de segments de droites liés). Les courbes à interpoler doivent

respecter une certaine tolérance d'usinage qui est la valeur maximale d'erreur de flèche (voir figure 16). Cette erreur est généralement, calculée entre le segment de droite, entre les points de contact et la surface nominale.

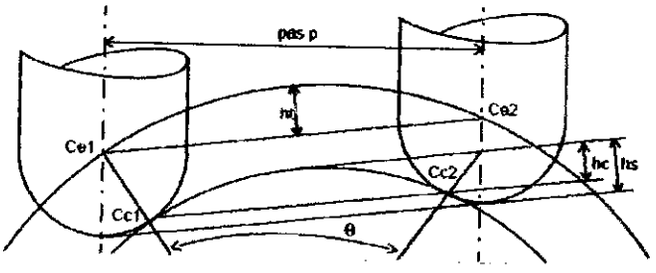


Figure 14. Erreur de flèche.

VI.3. Génération du trajet d'usinage :

Généralement, l'usinage des formes gauches est un usinage par balayage. La génération du trajet d'usinage est une étape importante puisqu'elle sert à obtenir des chemins bien calculés dans le but de construire une surface de haute qualité. Plusieurs éléments sont nécessaires dans le calcul de la trajectoire :

- La définition géométrique de la surface à usiner.
- La géométrie de l'outil.
- Une stratégie d'usinage (tolérances d'usinage et direction d'usinage).

VII. STRATEGIES D'USINAGE DES SURFACES GAUCHES [17, 18, 19] :

Il existe plusieurs stratégies d'usinage des surfaces gauches et nous pouvons citer trois types de stratégies principales : stratégies isoparamétriques, stratégies d'usinage par plans parallèles, et stratégies d'usinage par courbes de niveaux (Z-Constant).

VII.1. Stratégies isoparamétriques :

Ces stratégies sont basées sur les courbes isoparamétriques suivant u ou suivant v ou sur des portions de courbes isoparamétriques. Plusieurs modes de balayage peuvent être utilisés pour cette stratégie (One-Way, Zig-Zag, Concentrique, Spiral-In, Spiral-Out, Radiale) (voir figure 15).

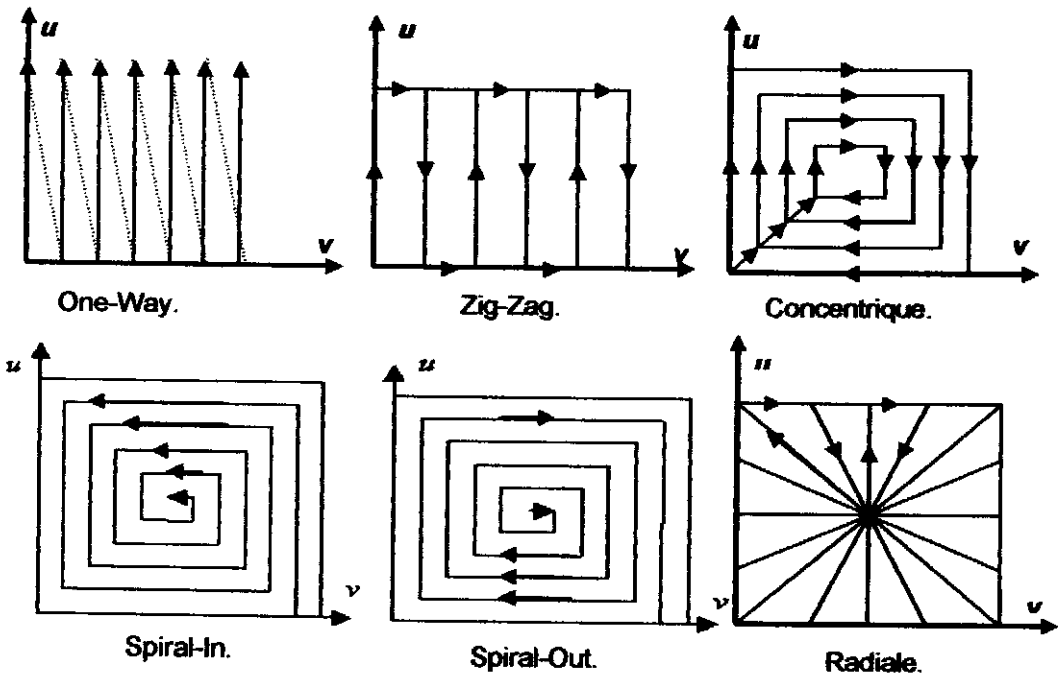


Figure 15. Stratégies d'usinage en isoparamétriques.

VII.2. Usinage par plans parallèles :

L'usinage avec cette stratégie consiste à construire des plans parallèles verticaux formant n'importe quel angle avec le plan XZ (voir figure 16). Les courbes obtenues de l'intersection de ces plans avec la surface à usiner sont utilisées pour générer la trajectoire d'usinage.

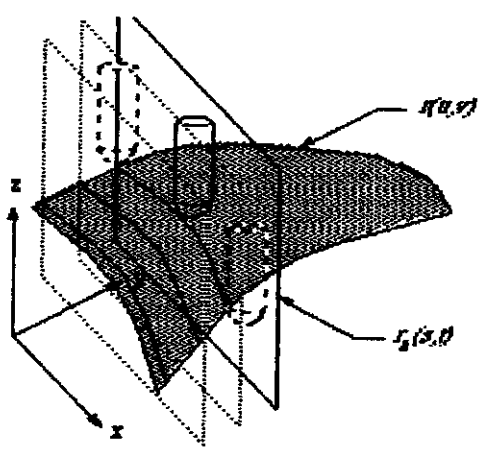


Figure 16. Usinage par plans parallèles.

VII.3. Usinage par des courbes de niveaux (Z-Constant) :

Ce type d'usinage consiste à construire des plans parallèles à l'axe Z (plans horizontaux) (voir figure 17). Les intersections de la surface avec ces plans (les contours) sont utilisées pour générer la trajectoire d'usinage.

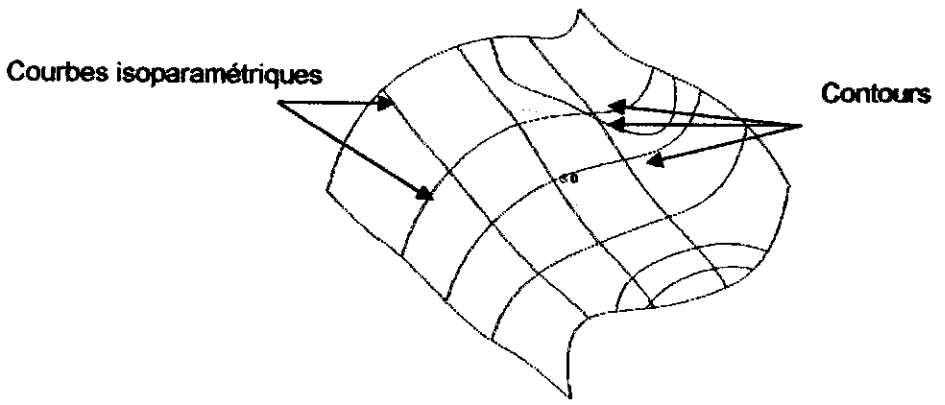


Figure 17. Usinage par Z-Constant.

VIII. ETAPES D'USINAGE D'UNE SURFACE GAUCHES [19,20] :

La complexité géométrique des surfaces gauches et la difficulté d'usiner des surfaces à haute qualité, ainsi que les risques d'obtenir des défauts d'usinage ont données la nécessité de diviser l'opération d'usinage en plusieurs étapes dans le but d'élever la qualité et de minimiser le temps d'usinage. L'usinage en réalité est divisé en trois étapes essentiels, ébauchage, semi finition, et finition.

VIII.1. Ebauchage (dégrossissage) :

En vérité l'ébauchage se n'est pas le vrai usinage de la surface désirée, mais c'est une étape préparatoire de l'usinage. Cette opération nous donne une surface proche de la surface à usiner avec l'enlèvement d'une grande quantité de la matière dans un temps très réduit avec l'utilisation de grands outils cylindriques ou hémisphériques. L'ébauchage est une série de réalisation de poches dans différents plans d'usinage. Les profondeurs de ces plans sont fixées par l'utilisateur. On commence par les plans supérieurs et on descend vers le bas. L'ébauchage avec un outil cylindrique et un outil hémisphérique est représenté respectivement par la figure 18 et la figure 19.

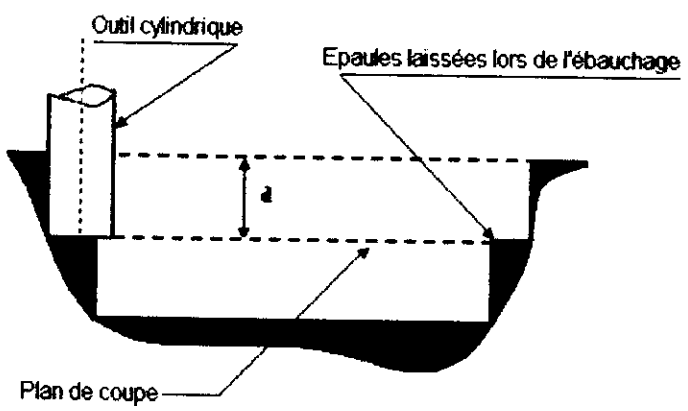


Figure 18. Ebauchage avec outil cylindrique.

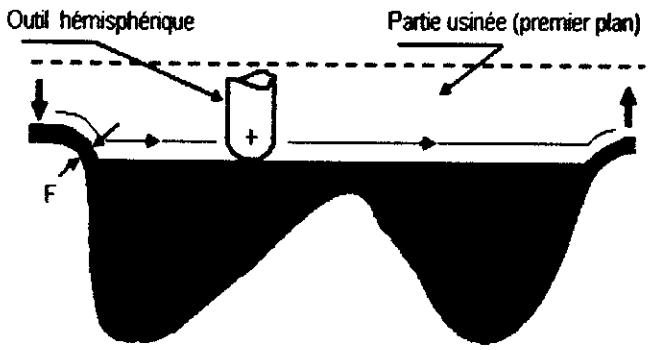


Figure 19. Ebauchage avec outil hémisphérique.

VIII.2. Semi finition :

La semi finition est une étape intermédiaire entre l'ébauchage et la finition. Lorsqu'on ébauche avec des outils cylindriques, beaucoup d'épaules sont produites sur la surface, la semi finition s'occupe de les éliminer avec des outils hémisphériques plus ou moins larges pour s'approcher de la forme finale.

VIII.3. Finition :

C'est la dernière étape de l'usinage, c'est l'usinage réel de la surface en éliminant la matière restante des deux étapes précédentes. Cette étape nécessite une grande précision avec l'utilisation des outils hémisphériques de petits diamètres.

IX. USINAGE AVEC LA STRATEGIE Z-CONSTANT :

La stratégie Z-Constant est utilisée pour l'usinage des cavités profondes afin d'éviter les longues descentes et montées d'outil. Cela est réalisé par le découpage de la surface en plusieurs plans parallèles à l'axe Z et l'intersection de ces plans avec la surface théorique produit ce qu'on appelle les contours. La stratégie Z-Constant suit le processus suivant :

- Calcul des contours.
- Liaison de ces contours pour construire le trajet d'usinage.

IX.1. Construction du trajet d'usinage avec la stratégie Z-Constant :

La génération du trajet d'usinage en utilisant la stratégie Z-Constant passe par les étapes suivantes :

- Triangulation de la surface à usiner.
- Calcul des intersections entre les plans de Z et les triangles.
- Fusion des segments d'intersection et la génération des contours.
- Calcul de la trajectoire d'usinage pour chaque plan Z.
- Réunion des différentes trajectoires pour construire le trajet total.

IX.2. Triangulation [17]:

La triangulation consiste à approximer la surface théorique avec un ensemble de triangles puisqu'il simplifient les calculs et permettent de représenter des plans. La triangulation peut être faite de deux manières différentes :

- **Triangulation uniforme** : dans ce type de triangulation, le nombre de triangle est fixé dans les deux directions **u** et **v** (voir figure 20). La triangulation est faite sur le plan paramétrique et par suite dans l'espace. Le calcul avec cette méthode est simple, mais il faut avoir un nombre très important de triangles pour obtenir une bonne approximation.

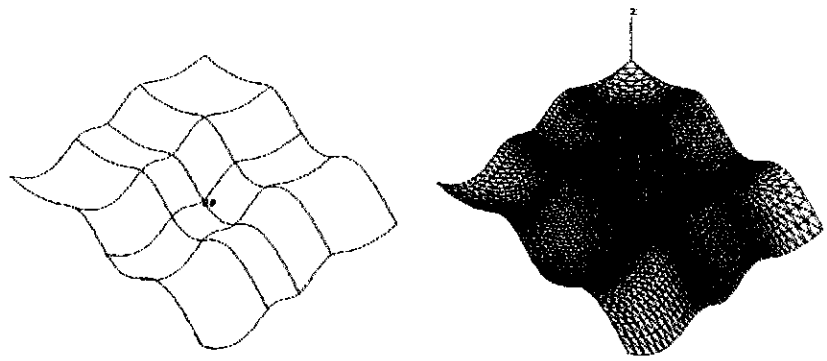


Figure 20. Triangulation uniforme.

- **Triangulation adaptative** : dans ce type de triangulation, les triangles sont créés d'une manière dynamique et adaptative selon le besoin à partir des critères de précision fournis par l'utilisateur. Les parties planes de la surface nécessitent moins de triangles que les parties courbées. Les conditions à respecter sont les suivantes :

1- La longueur maximale de chaque coté du triangle doit être inférieure à une tolérance « d » spécifiée par l'utilisateur.

2- La distance entre le milieu de chaque coté du triangle et la surface doit être inférieure à une tolérance « e » spécifiée par l'utilisateur.

3- La distance entre le centre de gravité du triangle et la surface doit être inférieure à une tolérance « e » spécifiée par l'utilisateur.

La figure 21 montre deux triangulations adaptatives pour la même surface.

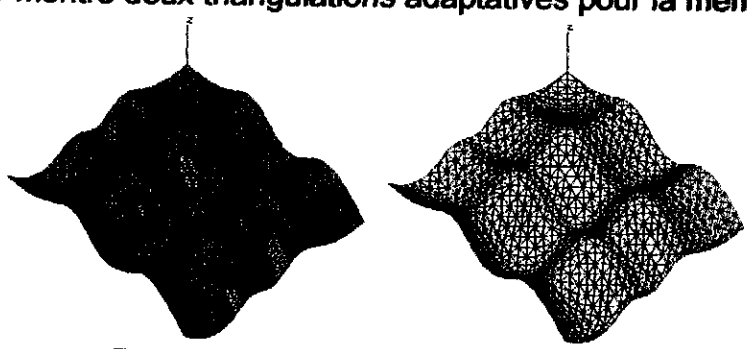


Figure 21. Triangulation adaptative.

Si une des conditions n'est pas vérifié, alors le triangle sera triangulé de nouveau. La nouvelle triangulation est faite comme suit :

1- Si un coté du triangle ne vérifie pas une condition « d » ou « e » alors le triangle est subdivisé en deux triangles (voir figure 22.A).

2- Si deux cotés du triangle ne vérifient pas une condition « d » ou « e » alors le triangle est subdivisé en trois tringles (voir figure 22.B).

3- Si trois cotés du triangle ne vérifient pas une condition « d » ou « v » alors le triangle est subdivisé en quatre triangles (voir figure 22.C).

4- Si le centre de gravité ne vérifie pas la condition « e » alors le triangle sera subdivisé en quatre triangles (voir figure 22.D).

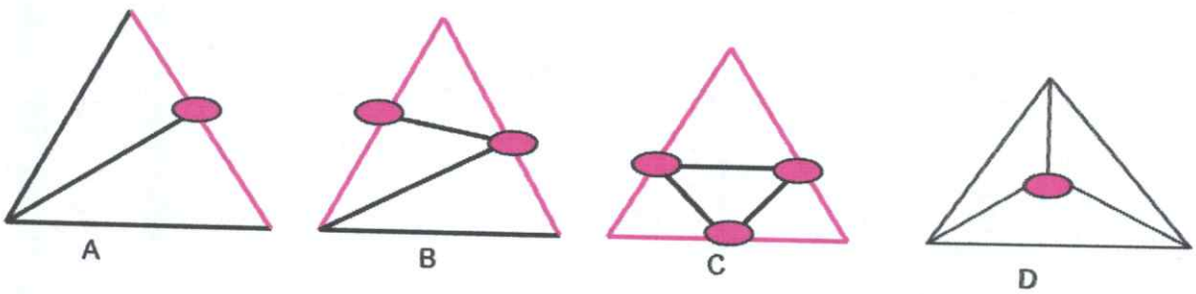
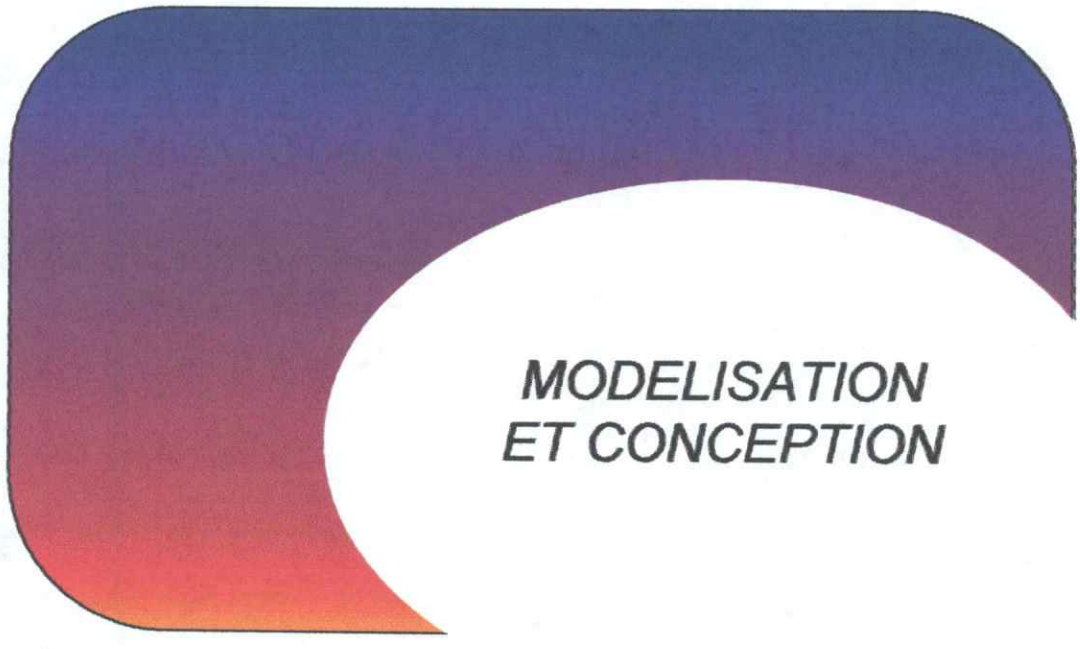


Figure 22. Subdivision d'un triangle.

X. CONCLUSION :

Dans ce chapitre, nous avons présenté les différentes stratégies d'usinage des surface gauches ainsi les types de fraises utilisées dans l'usinage. Ensuite, nous sommes passé au calcul de la position d'outil par rapport aux surfaces et nous avons terminé avec le détail de la stratégie d'usinage Z-Constant.

CHAPITRE 5



I. INTRODUCTION :

Dans les chapitres précédents nous avons présenté les notions fondamentales relatives à l'usinage des surfaces gauches ainsi que les aspects mécaniques liés à l'usinage de ces surfaces et en particulièrement le fraisage à trois axes. Nous allons passer maintenant à une étape très importante dans notre projet, c'est l'étape d'analyse et spécification des besoins. Dans cette étape, nous allons présenter la problématique et la solution proposée ainsi la définition des objectifs du travail.

II. PROBLEMATIQUE :

Lors de l'usinage des surfaces gauches, l'utilisateur peut tomber dans des situations indésirables (interférences et collisions) suite à un mauvais choix des outils ce qui déforme la surface à usiner. Pour cela, il est nécessaire de trouver un moyen pour surmonter ce problème.

Un autre problème, c'est le format utilisé pour exprimer les déplacements des outils lors de l'usinage des surfaces gauches et qui est généralement l'interpolation linéaire. Dans l'interpolation linéaire, entre chaque deux points consécutifs, l'outil fait trois types de mouvements différents, il commence par un mouvement d'accélération, ensuite un mouvement uniforme et enfin un mouvement de décélération. Ces mouvements augmentent considérablement les temps d'usinage et détériorent l'état de la surface. Pour cela, il est nécessaire de trouver un moyen pour éliminer ce problème. Un autre problème très important à traiter, c'est l'optimisation du trajet d'usinage pour la stratégie Z-Constant. Il arrive où nous avons à usiner des surfaces qui comportent plusieurs parties concaves et convexes éloignées, ainsi que des parties ouvertes et des parties fermées éloignées, ce qui donne plusieurs mouvements inutiles de l'outil hors de la surface à usiner ce qui augmente les temps d'usinage.

III. OBJECTIFS VISE :

Les objectifs de notre travail sont les suivants :

1. Détection des collisions et des interférences. Pour atteindre cet objectif, nous devons passer par les étapes suivantes :

- Création des sommets et des triangles,
- Création des régions de points suivant les deux axes X et Y.

2. Approximation des contours par des courbes B-Spline pour minimiser les vibrations et réduire la taille du programme d'usinage.

3. Optimisation du trajet d'usinage pour la stratégie d'usinage Z-Constant. Cet objectif peut être aussi divisé en sous objectifs suivants :

- Construction des contours par plan.
- Construction des hilles sides (contraintes d'usinage).
- Construction des plans des hilles sides.
- Construction des sous chemins à partir des hilles sides.

- Génération du trajet final par deux stratégies différentes d'usinage.
- Simulation d'usinage.
- Génération du programme d'usinage G-Code.

IV. SOLUTION PROPOSEE :

Pour résoudre les trois problèmes cités précédemment, nous allons développer une application logicielle graphique et interactive sous un environnement Windows. Pour atteindre notre objectif, nous allons procéder comme suit :

- Développement des algorithmes de détection des collisions et des interférences entre outil d'usinage et n'importe quelle surface.
- Développement des algorithmes pour l'approximation des trajets d'outils par des courbes B-Spline.
- Développement des algorithmes d'optimisation du trajet d'usinage pour la stratégie d'usinage Z-Constant et trouver le chemin optimum des outils.

V. SPECIFICATION DES BESOINS [21, 22] :

Pour le développement de l'application logicielle, nous utilisons une méthode de modélisation très récente, c'est la méthode OMT (Object Modeling Technique) (voir Annexe A)et le langage UML (voir Annexe B). Ce dernier présente plusieurs types de diagrammes pour pouvoir définir la structure des système (vue statique) ainsi que le comportement du système (vue dynamique). Nous allons utiliser comme cycle de vie du système un cycle de vie classique, c'est le cycle de vie en cascade commençant par l'analyse ensuite la conception, implémentation, test et validation et maintenance. Pour définir les besoins de notre système (cahier de charges), nous allons utiliser deux types de diagrammes, diagramme de cas d'utilisation et diagramme de séquence.

V.1. Diagrammes cas d'utilisation :

Un diagramme cas d'utilisation c'est la description du comportement du système du point de vue de son utilisateur (facilite l'expression des besoins), ainsi que la définition des objectifs correspond à un système.

Un Comportement = {action} + {réaction}

Le diagramme cas d'utilisation est caractérisé par :

Acteur : c'est une entité externe qui agit sur le système. Dans notre système, l'acteur est un utilisateur mécanicien qui commande le système.

Cas d'utilisation : c'est un ensemble d'actions réalisées par le système en réponse à une action d'un acteur.

Dans ce qui suit, nous allons présenter tous les cas d'utilisation du système en commençant par le diagramme cas d'utilisation global. Le système est divisé en trois grands axes :

- Détection des collisions et des interférences.
- Approximation des trajets par des courbes B-Spline.
- Optimisation du chemin d'outils.

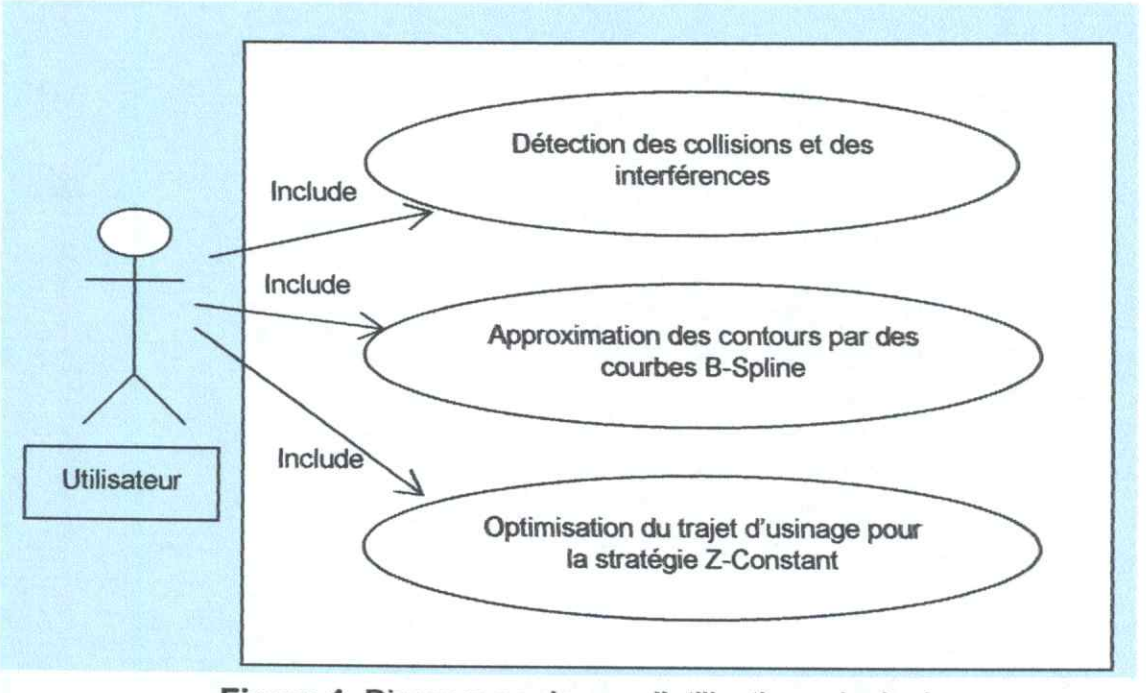


Figure 1. Diagramme de cas d'utilisation principal.

Nous allons passer maintenant aux diagrammes de chaque module (diagrammes élémentaires) et nous commençons par la détection des collisions et des interférences.

Le diagramme cas d'utilisation détection des collisions et des interférences passe par deux étapes :

- Triangulation.
- Détection des collisions et des interférences.

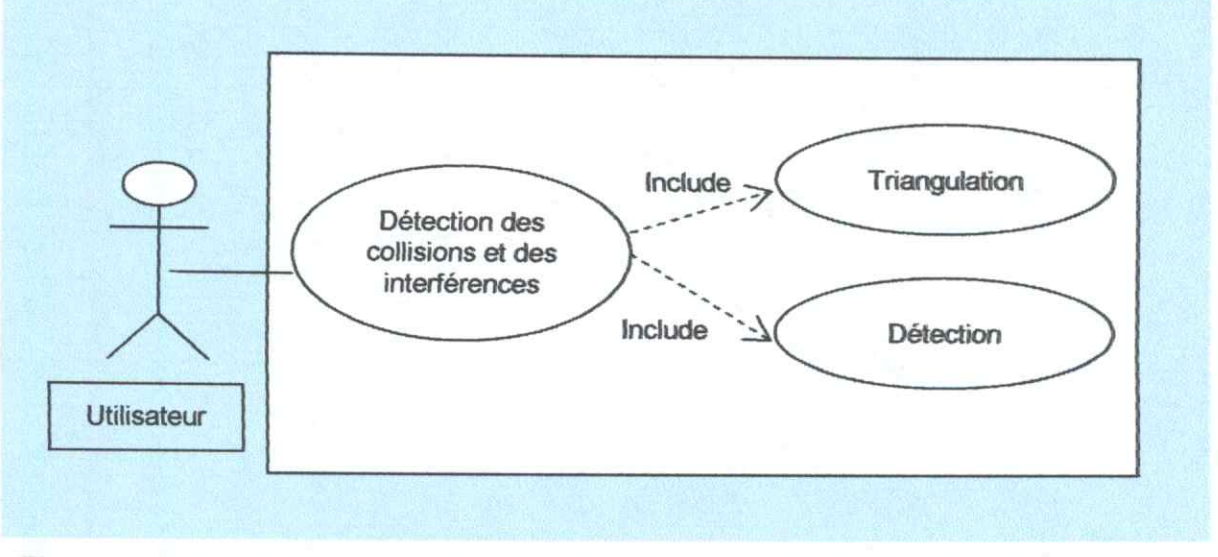


Figure 2. Diagramme cas d'utilisation détection des collisions et des interférences

Nous allons détailler maintenant les modules de triangulation et de détection. Nous pouvons représenter le module de triangulation comme suit :

- Triangulation.
- Division de la surface en régions.

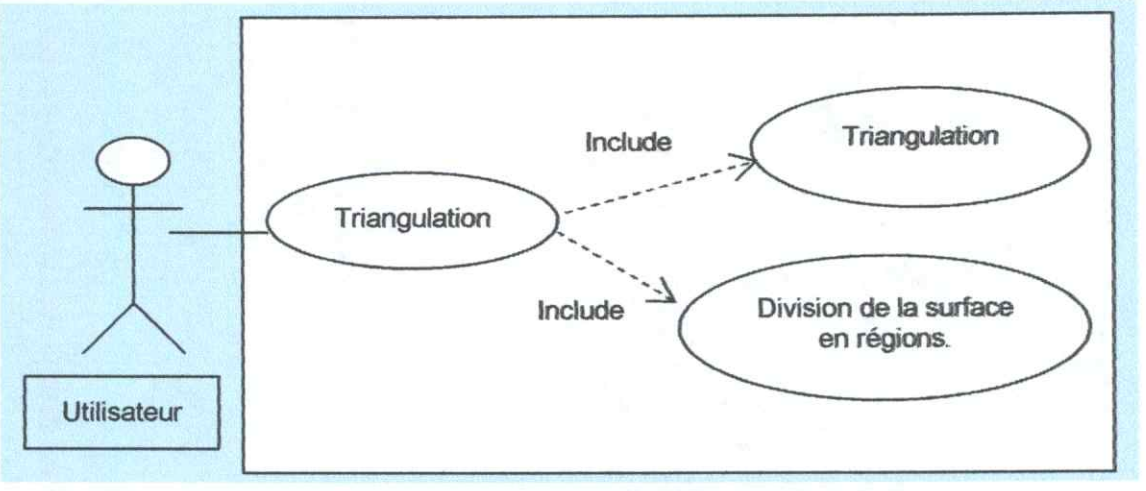


Figure 3. Diagramme cas d'utilisation triangulation.

Le diagramme cas d'utilisation détection est décrit comme suit :

- Choix du mode d'introduction.
- Paramétrisation de l'outil.
- Choix de la partie d'outil concernée.
- Temporisation
- Lancer l'usinage.
- Calcul et visualisation des collisions et des interférences.

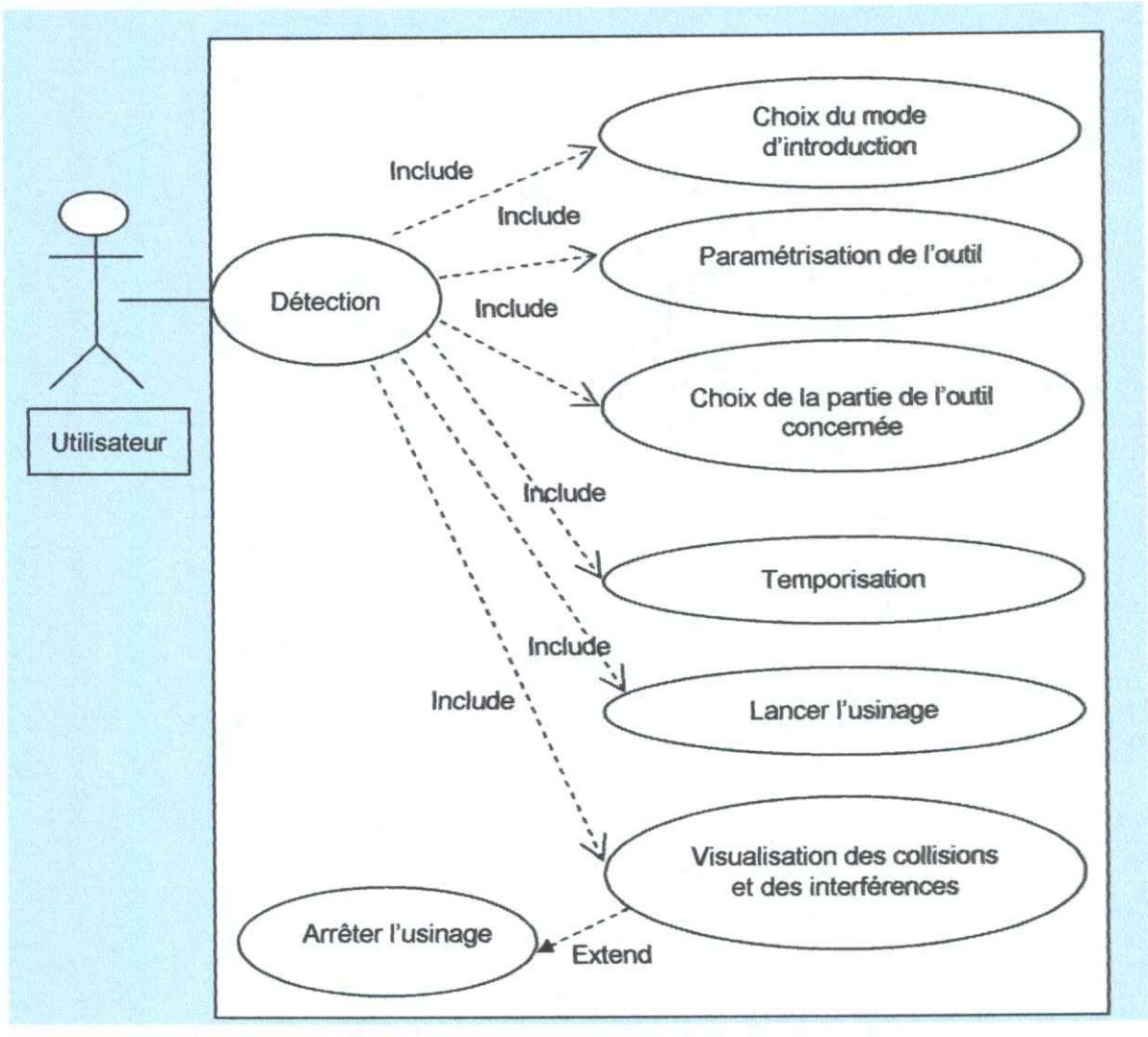


Figure 4. Diagramme cas d'utilisation détection.

Maintenant, nous allons passer à une autre partie, c'est la partie approximation des contours par des courbes B-Spline. La vue générale de l'approximation est comme suit :

- Récupération des contours.
- Approximation des contours par les courbes B-Spline.

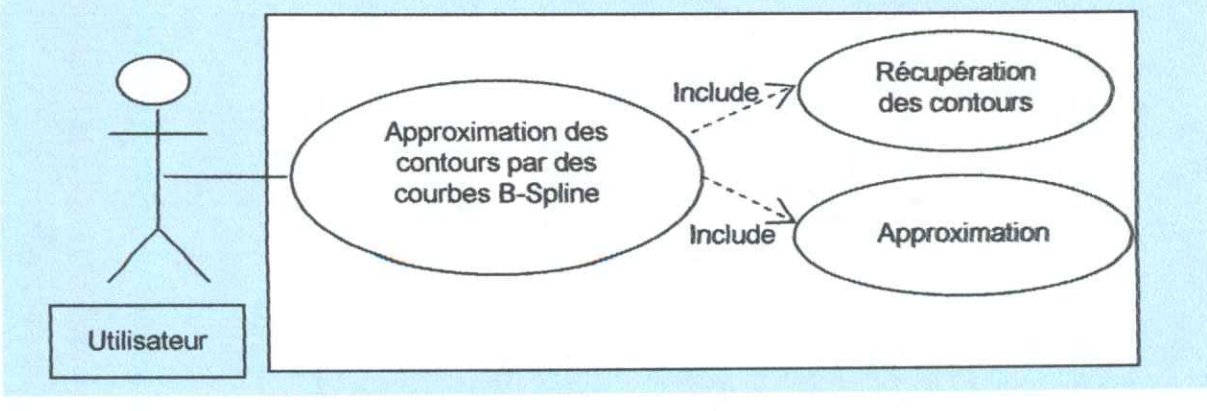


Figure 5. Diagramme cas d'utilisation approximation par des courbes B-Spline.

Le module approximation est décrit comme suit :

- Paramétrisation de l'approximation.
- Génération et visualisation des courbes.

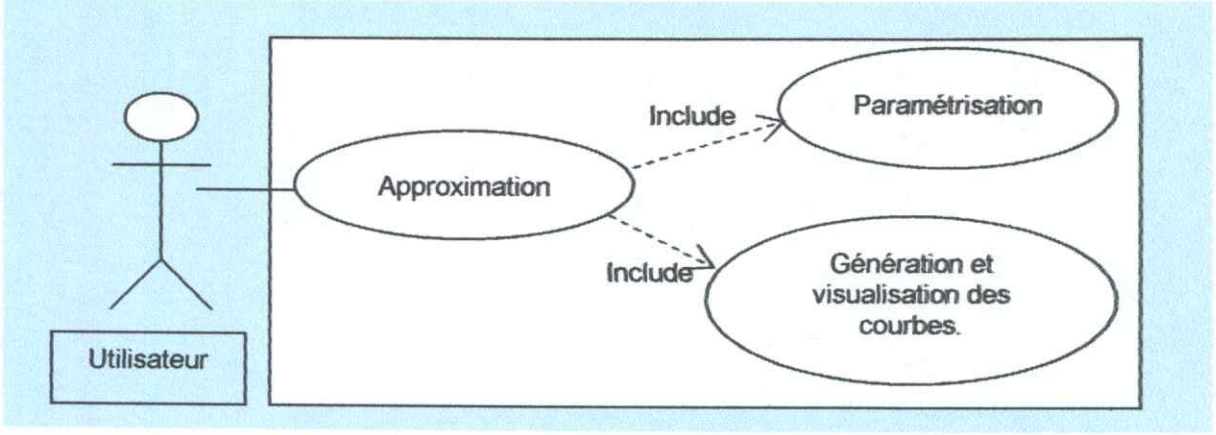


Figure 6. Diagramme cas d'utilisation approximation.

La partie optimisation est décrite comme suit :

- Construction des contours par plan.
- Création des hille_sides.
- Création des sous chemins.
- Génération du trajet d'usinage.

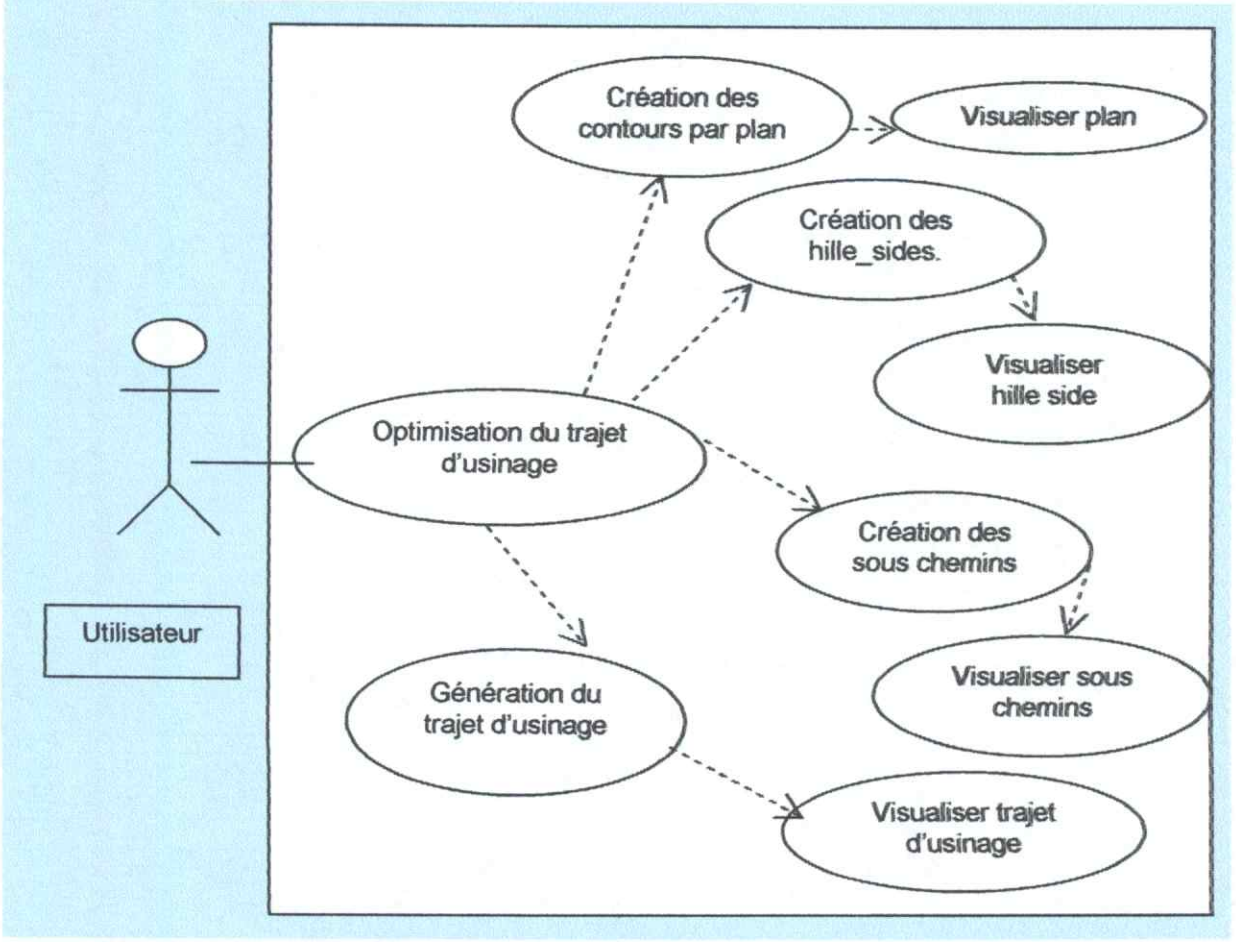


Figure 7. Diagramme cas d'utilisation optimisation du trajet d'usinage.

Nous allons maintenant décrire le module génération du trajet d'usinage. Nous avons deux types d'usinage possibles :

- Usinage en One-Way.
- Usinage en Zig-Zag.

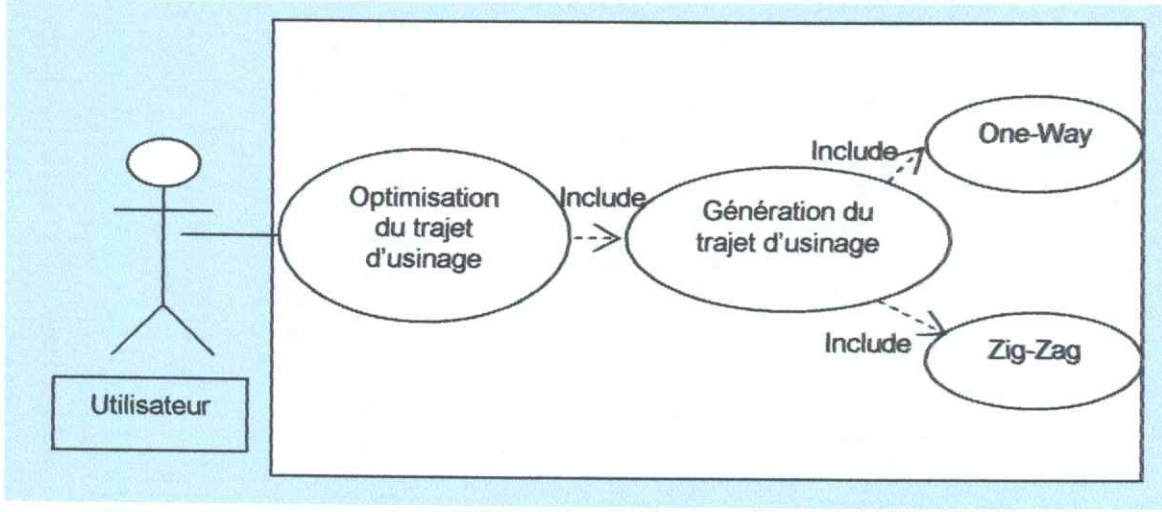


Figure 8. Diagramme cas d'utilisation génération du trajet d'usinage.

La visualisation d'optimisation du trajet d'usinage :

- Type d'affichage.
- Méthode de visualisation.

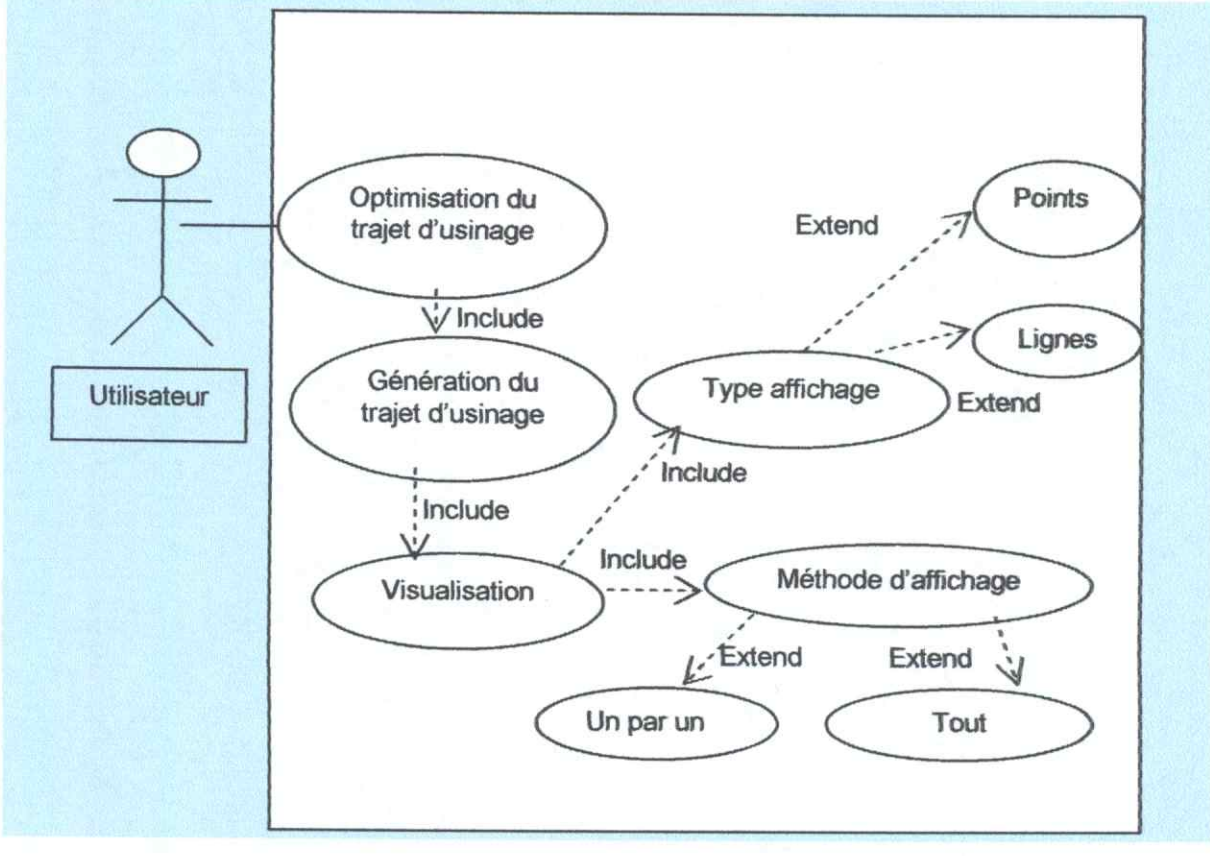


Figure 9. Diagramme cas d'utilisation visualisation.

V.2. Diagrammes de séquence :

Le diagramme de séquences est une spécification dynamique du système sous forme des scénarios et qui décrit un échange entre un acteur ou plusieurs acteurs et le système.

La triangulation et la division de la surface passent par les étapes suivantes :

- L'utilisateur demande l'ouverture de la surface.
- Le système ouvre la surface.
- L'utilisateur demande la triangulation.
- Le système ouvre la fenêtre de la triangulation.
- L'utilisateur introduit les champs de la triangulation.
- Le système calcule la triangulation.
- L'utilisateur introduit les champs de la division des surfaces.
- Le système calcule les surfaces divisées.

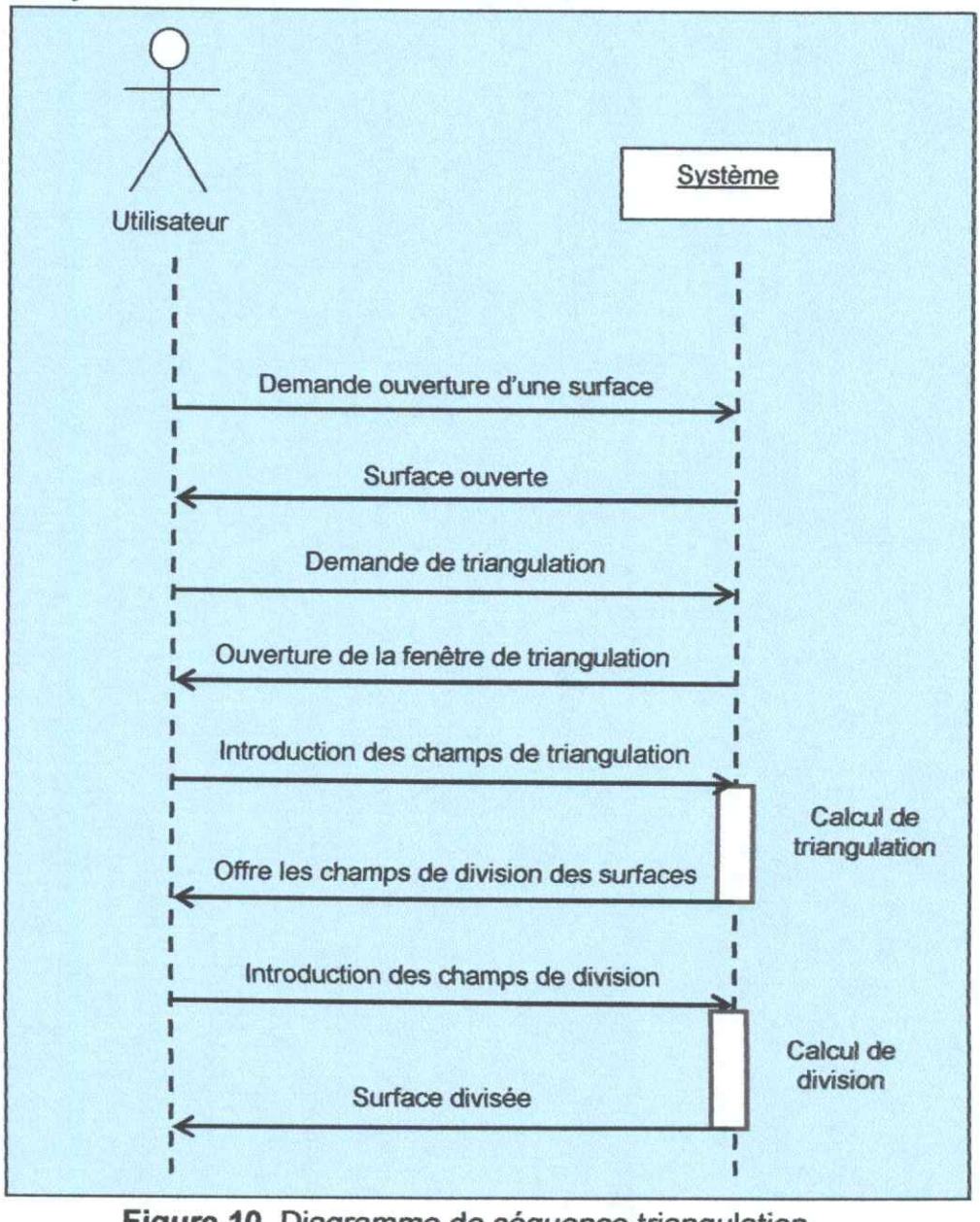


Figure 10. Diagramme de séquence triangulation.

La détection des collisions et des interférences passe par les étapes suivantes :

- L'utilisateur demande la détection des collisions et des interférences.
- Le système demande le choix du mode d'introduction.
- L'utilisateur sélectionne le mode.
- Le système demande les paramètres d'outil.
- L'utilisateur introduit les paramètres d'outil et lance le calcul.
- Le système calcule les collisions et les interférences.

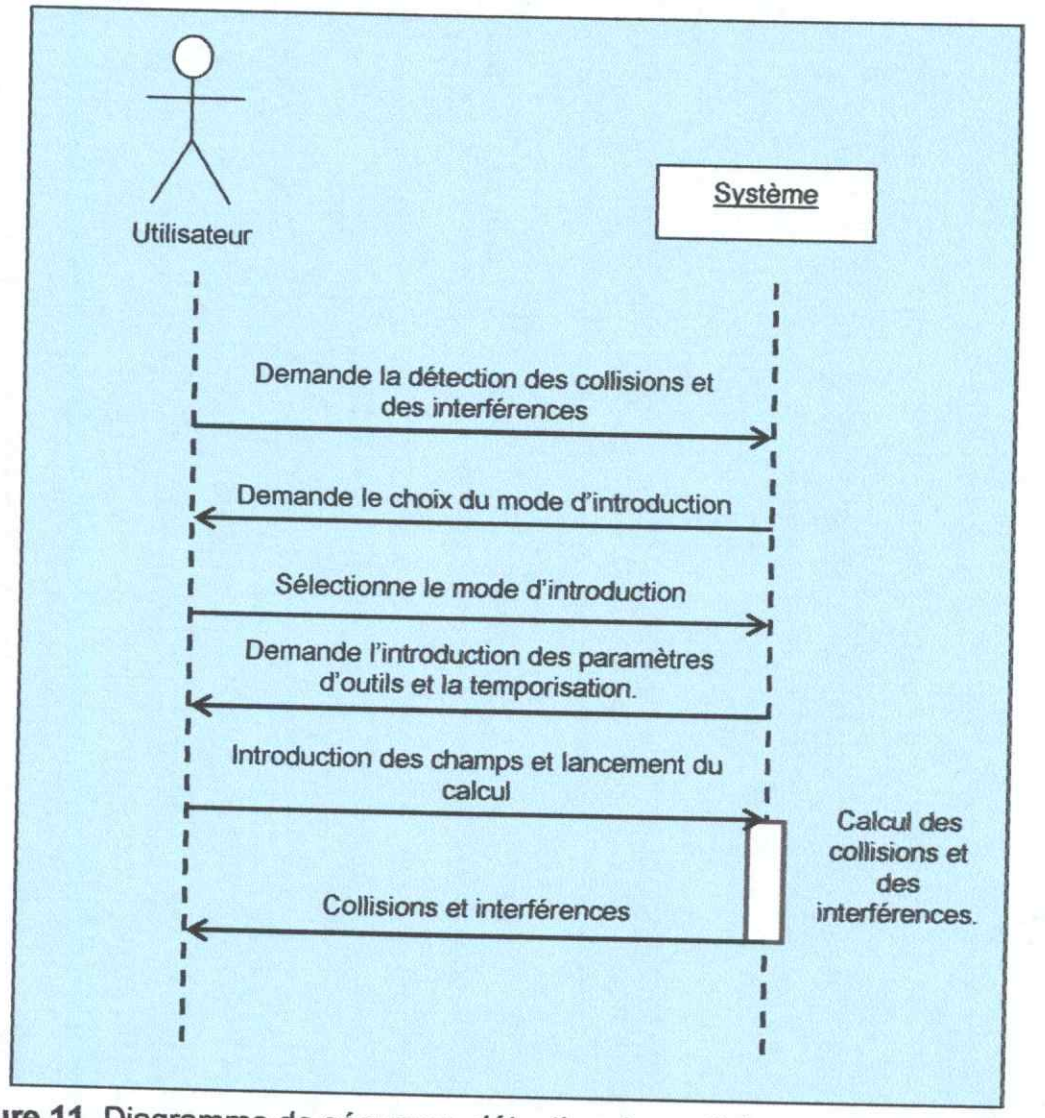


Figure 11. Diagramme de séquence détection des collisions et des interférences.

L'approximation des contours par des courbes B-Spline passe par les étapes suivantes :

- L'utilisateur demande la récupération des contours.
- Le système récupère les contours.
- L'utilisateur demande l'approximation des contours par des courbes B-Spline.
- Le système demande les paramètres d'approximation.
- L'utilisateur introduit les champs et lance le calcul.
- Le système calcule et visualise les courbes approximées.

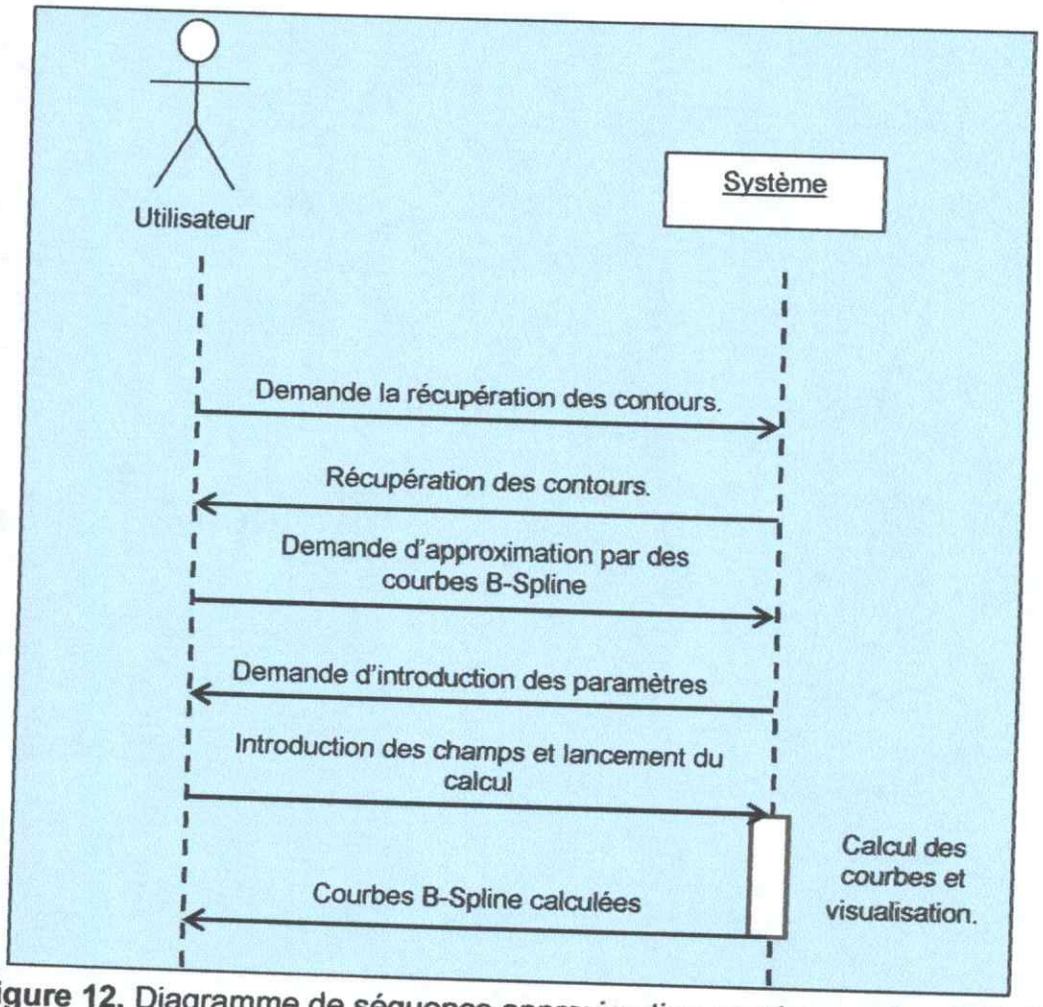


Figure 12. Diagramme de séquence approximation par des courbes B-Spline.

L'optimisation du trajet d'usinage pour la stratégie Z-Constant passe par les étapes suivantes :

- L'utilisateur demande l'optimisation du trajet d'usinage.
- Le système ouvre la fenêtre d'optimisation.
- L'utilisateur demande la création des plans des contours.
- Le système crée les plans.
- L'utilisateur demande la création des plans hille_sides
- Le système crée les plans hille_sides.
- L'utilisateur demande la création des sous chemins.
- Le système crée les sous chemins.
- Le système demande le choix de la stratégie d'usinage.
- L'utilisateur sélectionne la stratégie d'usinage et lance la génération des trajectoires.
- Le système calcule et génère les trajectoires.
- L'utilisateur demande la simulation.
- Le système calcule et visualise la simulation.
- L'utilisateur demande la génération du programme G-Code.
- Le système génère le G-Code.

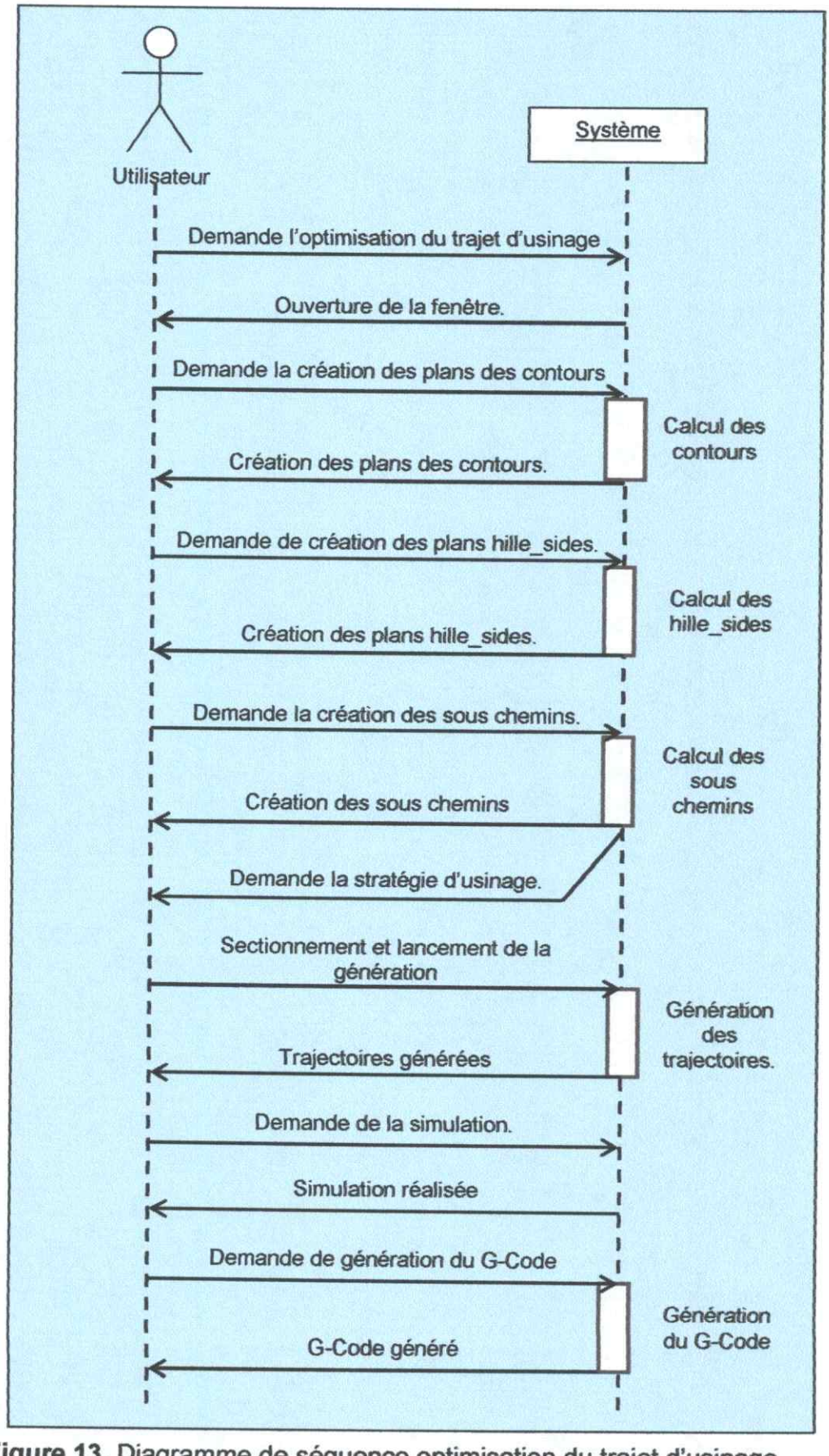


Figure 13. Diagramme de séquence optimisation du trajet d'usinage.

V.3. Diagrammes d'activité :

Le diagramme d'activité met l'accent sur les activités, leurs relations et leurs impactes sur les objets.

Le diagramme d'activité détection des collisions et des interférences et défini comme suit :

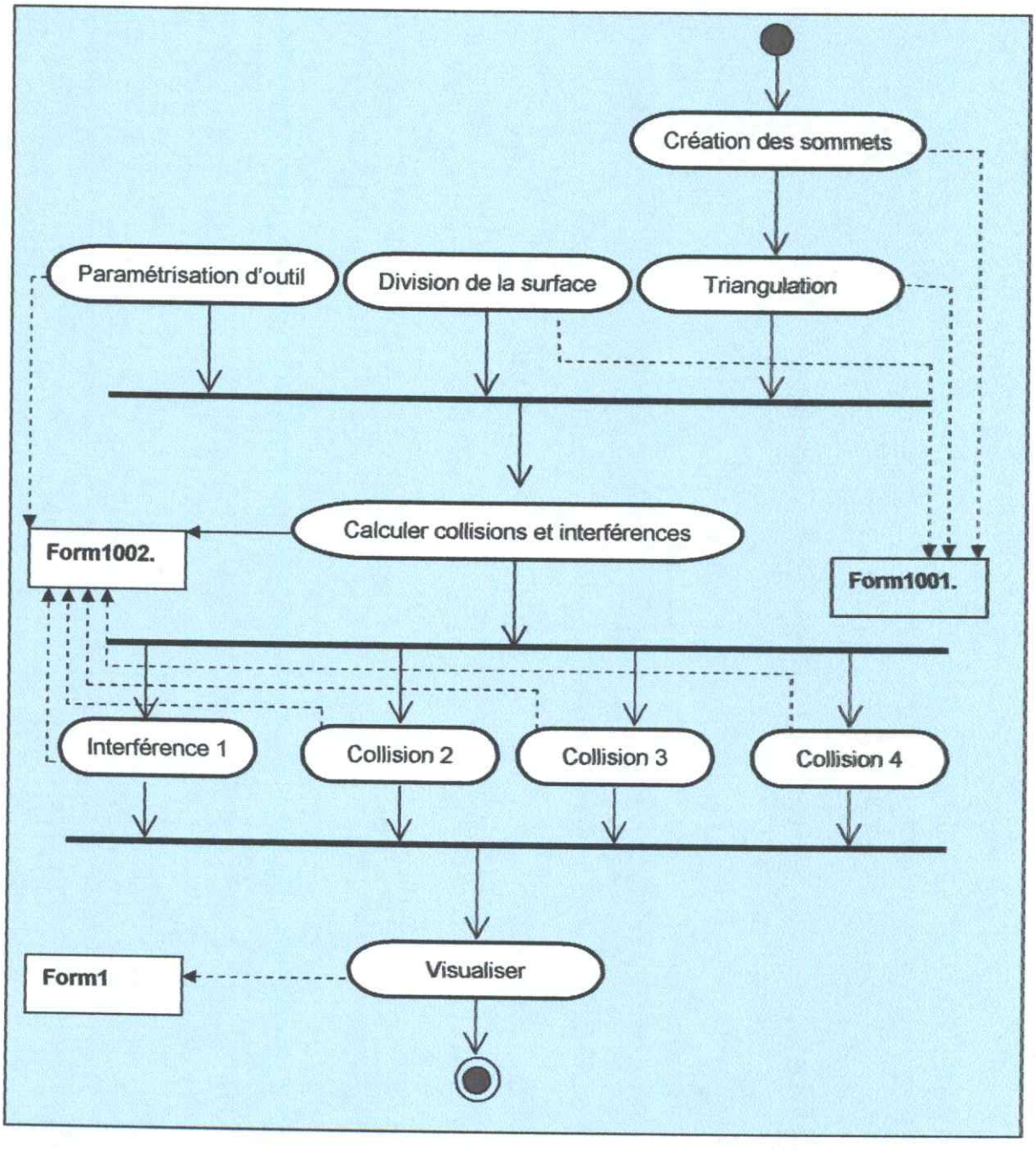


Figure 14. Diagramme d'activité détection des collisions et des interférences.

Nous allons passer au diagramme d'activité approximation des contours par des courbes B-Spline.

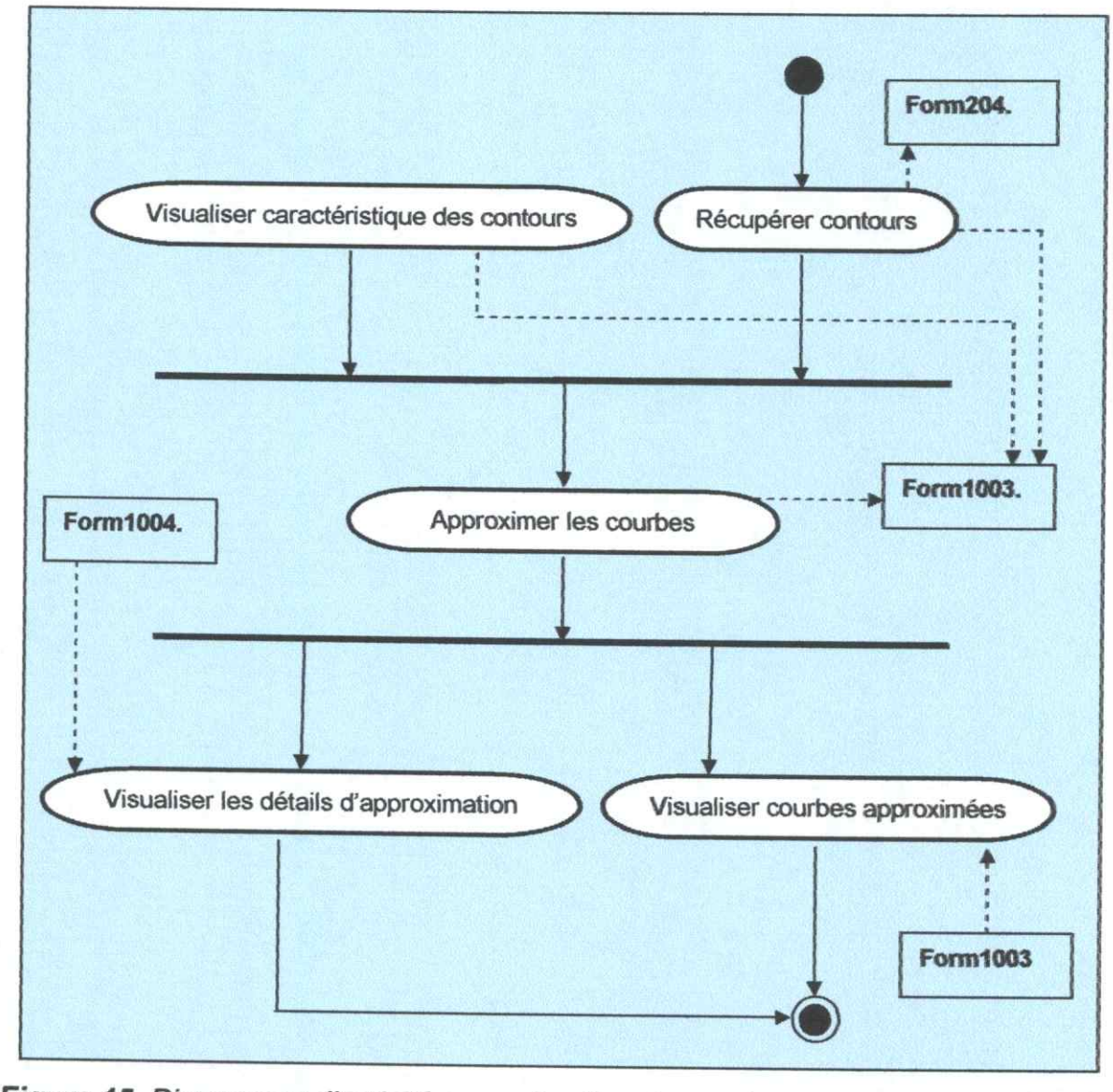


Figure 15. Diagramme d'activité approximation des contours par de courbe B-Spline

Le diagramme d'activité optimisation du trajet d'usinage pour la stratégie Z-Constant est donné comme suit :

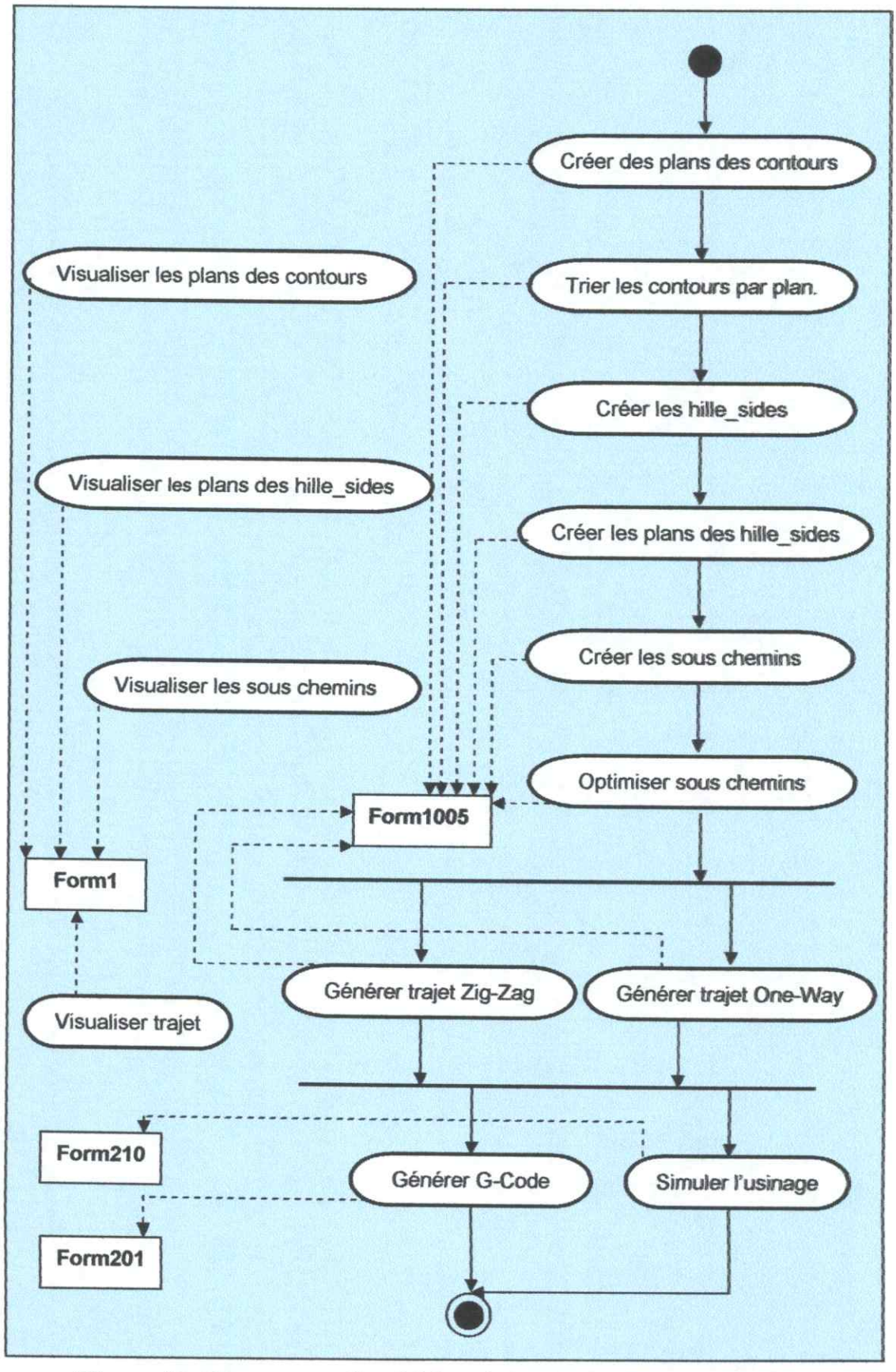


Figure 16. Diagramme d'activité optimisation du trajet d'usinage.

Après la définition et l'analyse des besoins, l'étape logique suivante est la conception, c'est à dire comment réaliser des solutions avec la prise en considération de tous les besoins définis précédemment.

Dans cette étape, nous allons décrire l'architecture du système ainsi que les relations entre les différents composants du système. Pour cela, nous allons utiliser le diagramme de class.

V.4. Diagramme de class :

C'est un diagramme qui permet une description architecturale du système par la définition de tous les objets du système et leurs comportements ainsi que les relations entre les objets du système.

Définitions des classes.

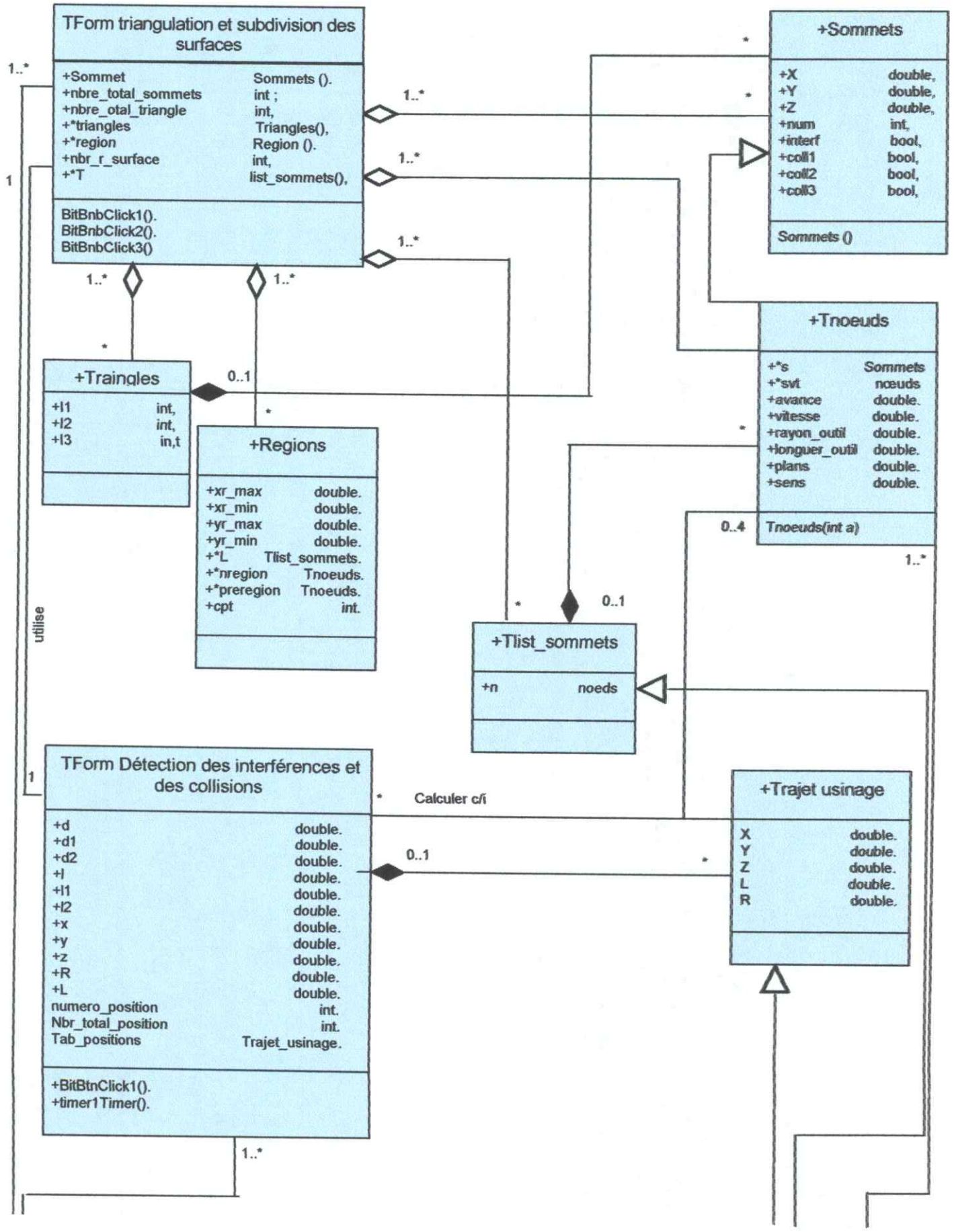
- **La classe triangulation et subdivision de la surface [TForm1001] :** dans cette classe, nous allons à partir d'une surface quelconque ou plusieurs surfaces approximer la surface ou l'ensemble des surfaces par un ensemble finis du triangles (triangulation uniforme) et par suite nous allons diviser les surfaces en des régions dans le but de minimiser le temps du calcul de la détection des collisions et des interférences.

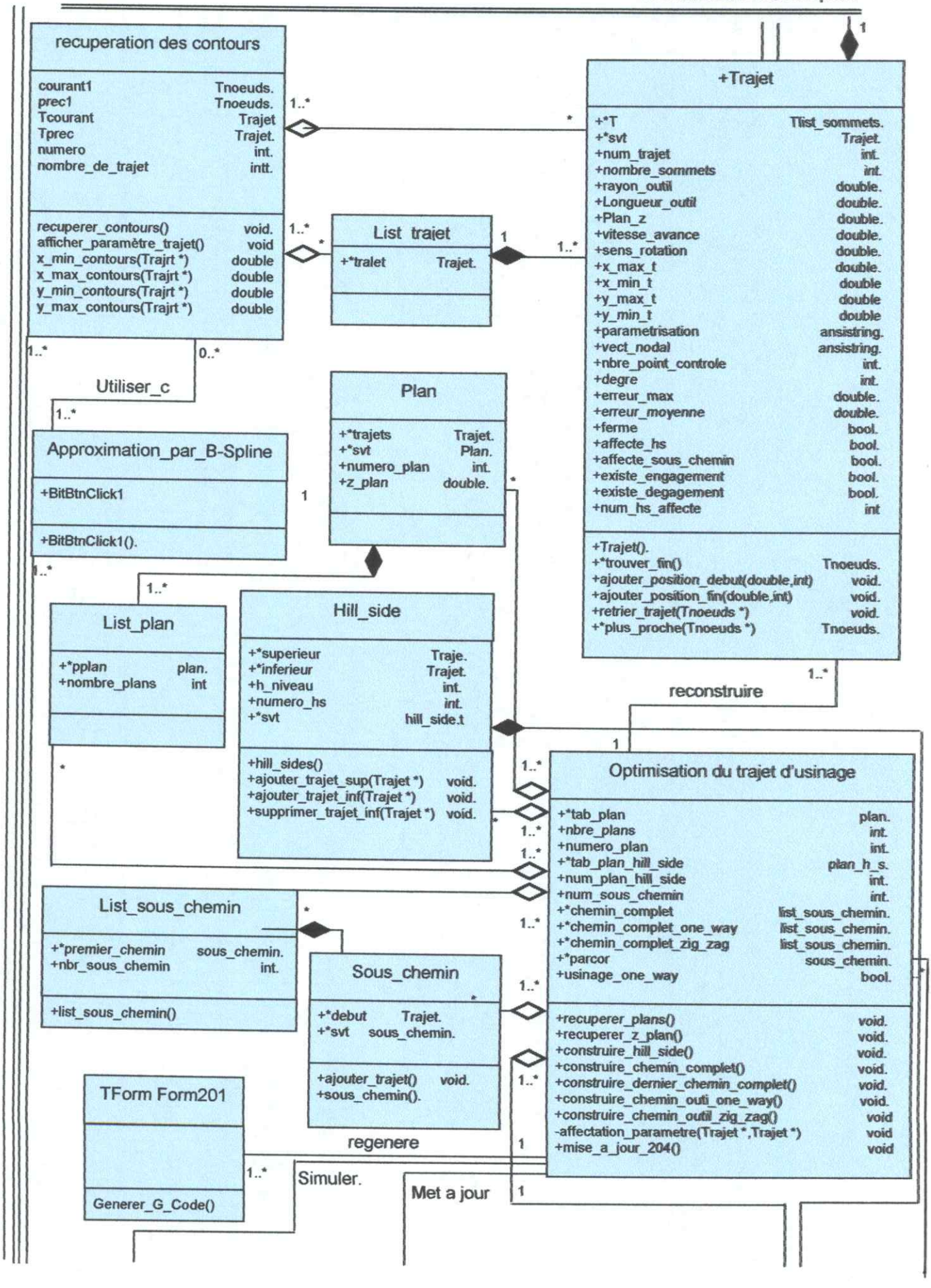
- **La classe détection des collisions et des interférences [TForm1002] :** dans cette classe, nous allons calculer et visualiser les collisions et les interférences pour chaque partie de l'outil.

- **La classe récupération des contours [TForm1003] :** dans cette classe, nous allons récupérer les trajets d'outils (positions d'outils) à partir de la classe 204 dans le but de les approximer par la suite, ainsi que le calcul des propriétés des trajets tels que :Xmin, Xmax, Ymin, Ymax, nombre de trajets, rayon d'outil, plans d'intersection, nombre de points pour chaque trajet et la longueur d'outil.

- **La classe approximation des contours par des courbes B-Spline [TForm1004] :** dans cette classe, nous allons calculer les courbes B-Spline à partir d'un nuage de points (dans notre cas c'est l'ensemble des positions d'outil) après l'introduction du degré de la courbe et l'erreur permise.

- **La classe optimisation du trajet d'usinage pour la méthode Z-Constant [TForm1005] :** dans cette classe, nous allons trier les contours récupérés par plans, ensuite nous allons utiliser ces plans pour construire les hille_sides en considérant les contraintes d'usinage, et trier les hille_sides par plans dans le but d'extraire des sous chemins optimisés. La dernière étape de cette optimisation est la génération du trajet d'usinage par une stratégie choisie. A la fin, nous lançons la simulation et nous régénérons le programme G-Code.


TForm triangulation et subdivision des surfaces
+Sommet Sommets ().
+nbre_total_sommets int ;
+nbre_otal_triangle int,
+*triangles Triangles(),
+*region Region ().
+nbr_r_surface int,
+*T list_sommets(),
BitBnbClick1().
BitBnbClick2().
BitBnbClick3()
+Sommets
+X double,
+Y double,
+Z double,
+num int,
+interf bool,
+coll1 bool,
+coll2 bool,
+coll3 bool,
Sommets ()
+Tnoeuds
+*s Sommets
+*svt nœuds
+avance double.
+vitesse double.
+rayon_outil double.
+longuer_outil double.
+plans double.
+sens double.
Tnoeuds(int a)
+Traingles
+I1 int,
+I2 int,
+I3 in,t
+Regions
+xr_max double.
+xr_min double.
+yr_max double.
+yr_min double.
+*L Tlist_sommets.
+*nregion Tnoeuds.
+*preregion Tnoeuds.
+cpt int.
+Tlist_sommets
+n noeds
TForm Détection des interférences et des collisions
+d double.
+d1 double.
+d2 double.
+I double.
+I1 double.
+I2 double.
+x double.
+y double.
+z double.
+R double.
+L double.
numero_position int.
Nbr_total_position int.
Tab_positions Trajet_usinage.
+BitBtnClick1().
+timer1Timer().
+Trajet usinage
X double.
Y double.
Z double.
L double.
R double.
Calculer c/i
utilise
1..*
1
0..1
0..4
*


recuperation des contours
courant1 Tnoeuds.
prec1 Tnoeuds.
Tcourant Trajet
Tprec Trajet.
numero int.
nombre_de_trajet intt.
recuperer_contours() void.
afficher_paramètre_trajet() void
x_min_contours(Trajrt *) double
x_max_contours(Trajrt *) double
y_min_contours(Trajrt *) double
y_max_contours(Trajrt *) double
+Trajet
+*T Tlist_sommets.
+*svt Trajet.
+num_trajet int.
+nombre_sommets int.
+rayon_outil double.
+Longueur_outil double.
+Plan_z double.
+vitesse_avance double.
+sens_rotation double.
+x_max_t double.
+x_min_t double
+y_max_t double
+y_min_t double
+parametrisation ansistring.
+vect_nodal ansistring.
+nbre_point_controle int.
+degre int.
+erreur_max double.
+erreur_moyenne double.
+ferme bool.
+affecte_hs bool.
+affecte_sous_chemin bool.
+existe_engagement bool.
+existe_degagement bool.
+num_hs_affecte int
+Trajet().
+*trouver_fin() Tnoeuds.
+ajouter_position_debut(double,int) void.
+ajouter_position_fin(double,int) void.
+retrier_trajet(Tnoeuds *) void.
+*plus_proche(Tnoeuds *) Tnoeuds.
List trajet
+*tralet Trajet.
Utiliser_c
Plan
+*trajets Trajet.
+*svt Plan.
+numero_plan int.
+z_plan double.
Approximation_par_B-Spline
+BitBtnClick1
+BitBtnClick1().
List_plan
+*pplan plan.
+nombre_plans int
Hill_side
+*superieur Traje.
+*inferieur Trajet.
+h_niveau int.
+numero_hs int.
+*svt hill_side.t
+hill_sides()
+ajouter_trajet_sup(Trajet *) void.
+ajouter_trajet_inf(Trajet *) void.
+supprimer_trajet_inf(Trajet *) void.
reconstruire
Optimisation du trajet d'usinage
+*tab_plan plan.
+nbre_plans int.
+numero_plan int.
+*tab_plan_hill_side plan_h_s.
+num_plan_hill_side int.
+num_sous_chemin int.
+*chemin_complet list_sous_chemin.
+*chemin_complet_one_way list_sous_chemin.
+*chemin_complet_zig_zag list_sous_chemin.
+*parcor sous_chemin.
+usinage_one_way bool.
+recuperer_plans() void.
+recuperer_z_plan() void.
+construire_hill_side() void.
+construire_chemin_complet() void.
+construire_dernier_chemin_complet() void.
+construire_chemin_outi_one_way() void.
+construire_chemin_outil_zig_zag() void
-affectation_parametre(Trajet *,Trajet *) void
+mise_a_jour_204() void
List_sous_chemin
+*premier_chemin sous_chemin.
+nbr_sous_chemin int.
+list_sous_chemin()
Sous_chemin
+*debut Trajet.
+*svt sous_chemin.
+ajouter_trajet() void.
+sous_chemin().
TForm Form201
Generer_G_Code()
regenere
Simuler.
Met a jour
1..*
0..*
1

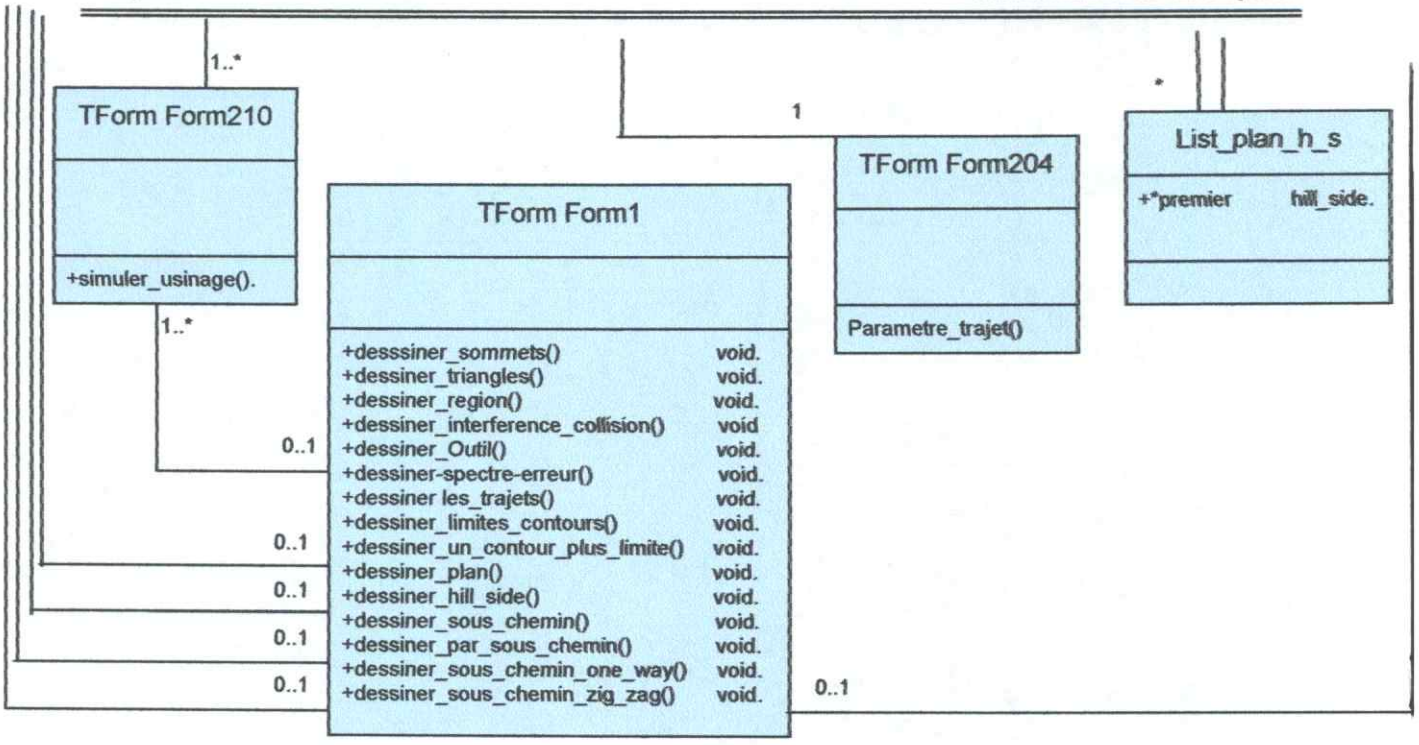


Figure 17. Diagramme de classes.

V.5. Diagramme de collaboration :

Le diagramme de collaboration (voir figure 18) est une extension du diagramme d'objets, c'est une vue dynamique du système de telle façon qu'il présente la réalisation des opérations ainsi que l'inscription des interactions entre objets du système par envoi et réception de messages.

Dans notre système, la forme de triangulation et subdivision (TForm **Form1001**) des surfaces s'occupe du calcul des triangles et des régions et envoie les résultats à la forme de détection des collisions et des interférences (TForm **Form1002**), et cette dernière fait le calcul. La forme (Form1001) envoie à la forme principale d'affichage (TForm **Form1**) des messages pour la visualisation des points de la surface, les triangles et les régions. La forme (Form1002) envoie des messages d'affichage à la forme (Form1).

La forme de récupération des trajets (TForm **Form1003**) envoie des messages à la forme (TForm **Form1000**) pour récupérer les trajets de la forme (TForm **Form204**) et retourne les trajets à la forme (Form1003). Après la récupération des trajets, la forme (Form1003) envoi des messages à la forme (TForm **Form1004**) pour calculer les trajets approximés par des courbes B-Spline et elle envoie des messages d'affichage à la forme (Forme1). La forme (Form1004) fait les calculs des trajets approximés et elle envoie des messages d'affichage à la forme (Form1).

La forme d'optimisation du trajet d'usinage (TForm **Form1005**) utilise les résultats des contours récupérés dans la forme (Form1003). Après application des fonctions d'optimisation sur ces contours, le trajet optimisé est affiché dans la forme (Form1).

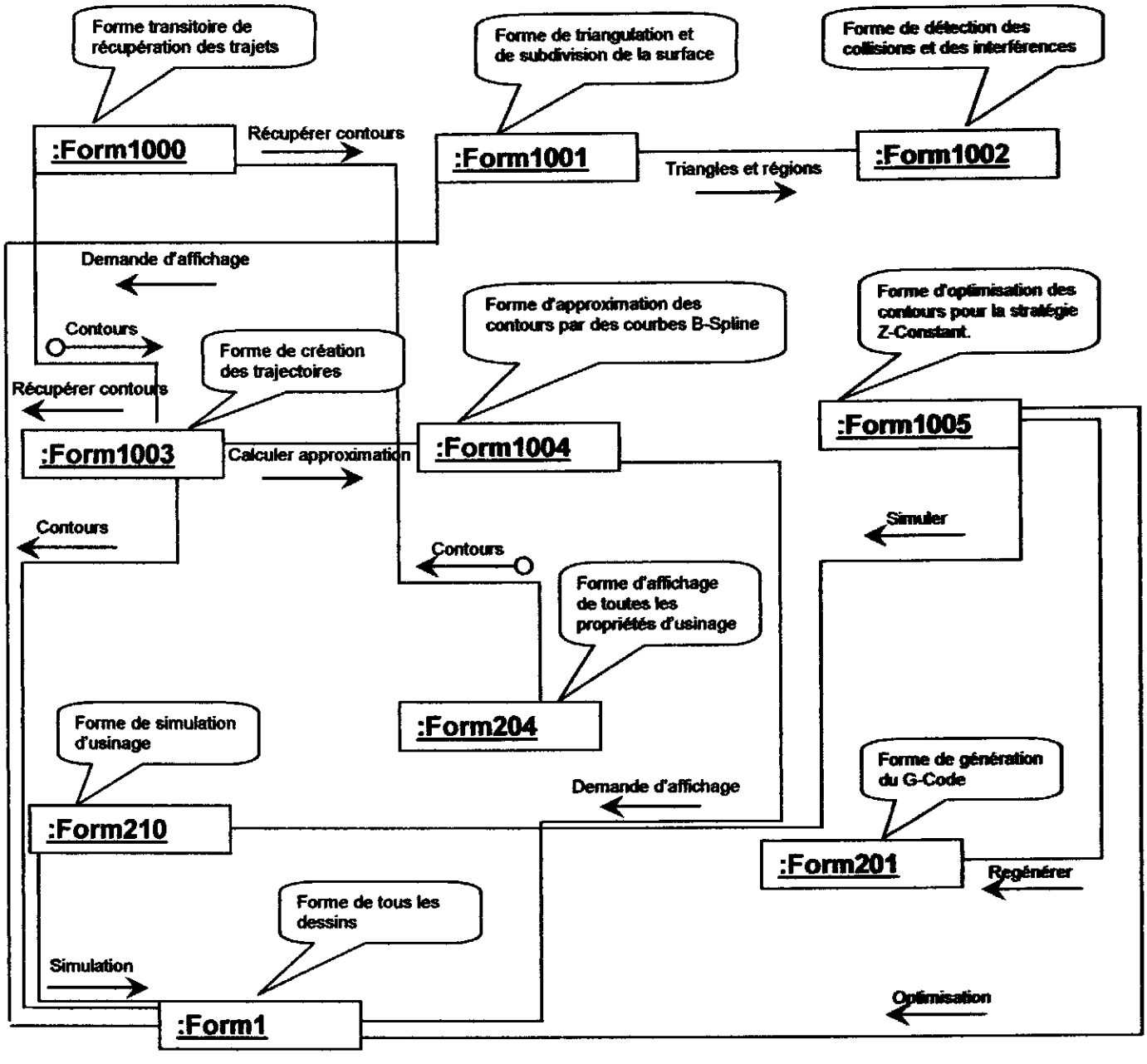


Figure 18. Diagramme de collaboration.

VI. CONCLUSION :

Dans ce chapitre, nous avons présenté les différentes classes de notre système ainsi que l'architecture détaillée du système. Nous avons commencé par la problématique et ensuite nous sommes passé à la solution proposée et nous avons terminé par les différents diagrammes d'UML.

CHAPITRE 6

IMPLEMENTATION

I. INTRODUCTION :

Dans le chapitre précédent, nous avons étudié les différents objectifs de l'application ainsi que les solutions proposées, ce qui nous a donné une idée claire sur l'étape suivante. Dans ce qui suit, nous allons présenter l'interface de notre application ainsi que les principaux algorithmes adoptés et l'implémentation dans un langage évolué.

II. ENVIRONNEMENT DE TRAVAIL :

Nous avons implémenté notre application sur un micro-ordinateur doté d'un processeur de 2.4 GHZ de fréquence, une carte graphique de 32 MO minimum. L'application est réalisée sur une plate forme Windows XP avec l'utilisation du langage C++Builder 6 car il accepte l'orienté objet et il est doté d'une interface qui permet l'interaction de l'utilisateur avec la machine.

Notre application est une application de CFAO graphique et interactive. Puisque l'utilisateur a la possibilité de visualiser tous les résultats intermédiaires et finaux, nous étions obligé d'utiliser des bibliothèques graphiques pour la visualisation de tous les objets. Dans notre cas, nous avons utilisé la bibliothèque **OpenGL**. Cette bibliothèque est une bibliothèque graphique standard qui regroupe un ensemble de primitives spécialisées dans la création des formes géométriques ainsi que les différentes manipulations associées.

III. IMPLEMENTATION :

III.1. Fenêtre principale :

Cette fenêtre est composée de deux parties, une partie d'affichage et une autre pour la manipulation. La partie de manipulation est composée d'un ensemble de boutons pour la création des formes géométriques et la modification de ces formes. De même, cette partie est composée d'un menu pour la manipulation CAO et pour le lancement des différentes fonctions de FAO (voir figure 1).

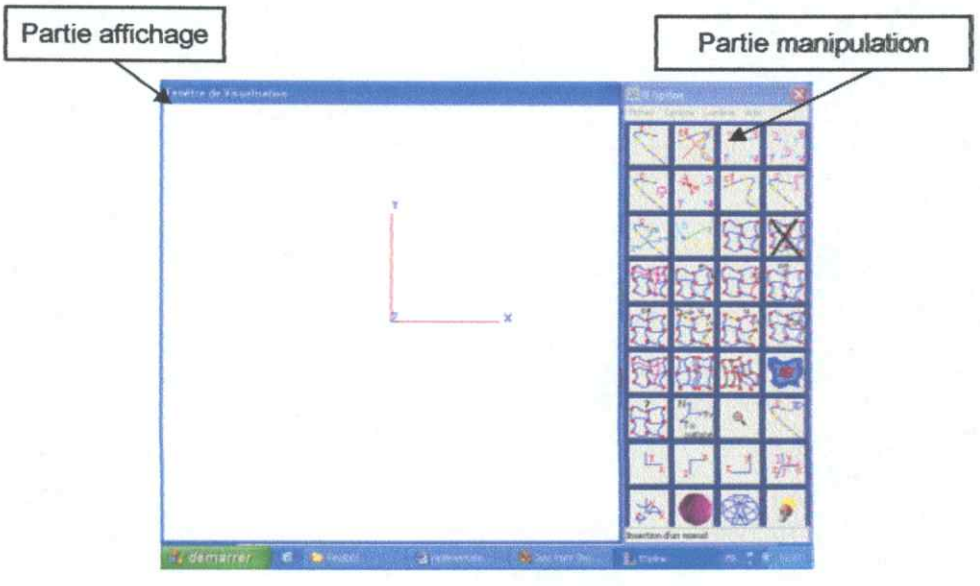


Figure 1. Fenêtre principale.

III.2. Rubrique triangulation et subdivision de la surface :

Dans cette partie, nous allons trianguler les surfaces et par suite diviser la surface ou l'ensemble des surfaces en des régions (voir figure 2). Le but de la division est de minimiser les temps de calcul des interférences et des collisions car si nous avons un nombre très important de sommets, pour chaque position d'outil donnée, nous devons considérer tous les sommets des surfaces pour calculer s'il y a interférence ou collision.

Pour réaliser cette fonction, nous allons utiliser les classes suivantes :

- La classe TSommets : elle est définie par trois réels x, y et z.
- La classe Tnoeuds : elle est définie par un sommet de type TSommets et un pointeur sur le nœud suivant.

La classe Tnoeuds est définie par :

```
Class Tnoeuds
  {
  TSommets *S ;
  Tnoeuds *svt ;
  } ;
```

- La classe Tlist_sommets : elle est définie par un pointeur sur le premier nœud de la liste.
- La clase Triangle : elle est définie par trois entiers qui sont les adresses des nœuds.
- La classe Region : elle est définie par quatre réels et une liste de sommets, ainsi que deux pointeurs sur nœuds et un compteur de type entier.

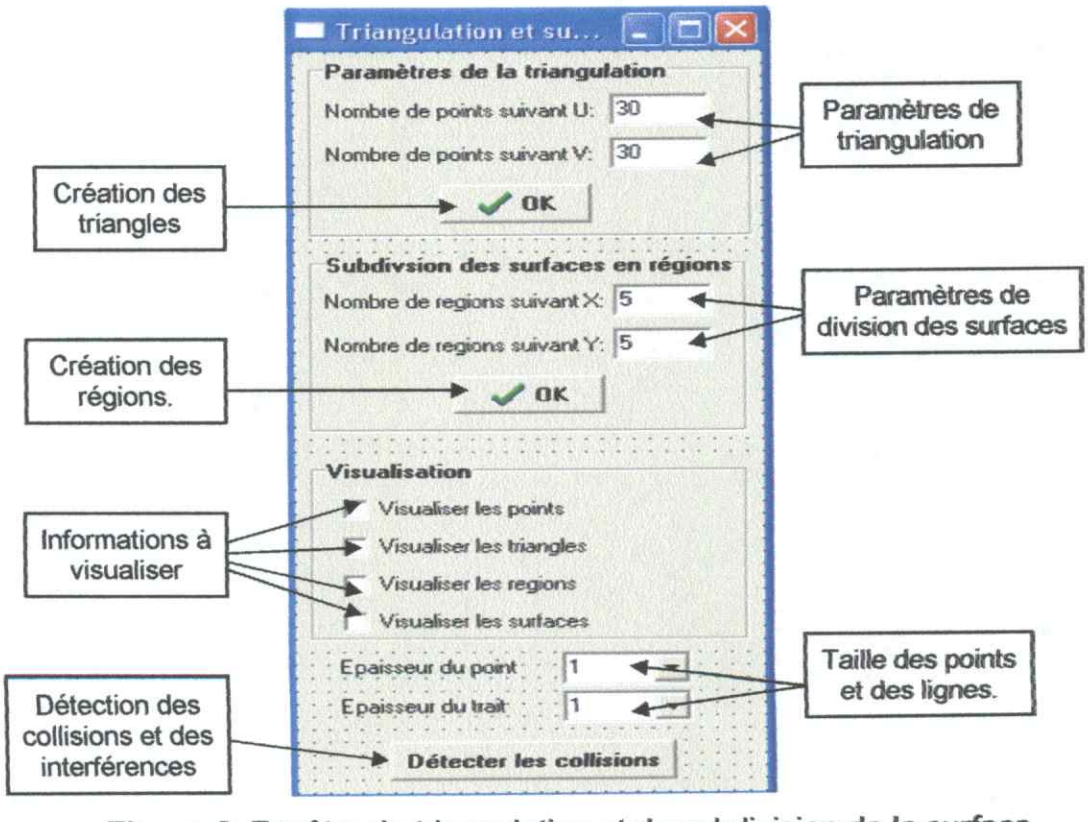


Figure 2. Fenêtre de triangulation et de subdivision de la surface.

III.3. Rubrique calcul des collisions et des interférences [23]:

Cette partie permet de calculer les interférences et les collisions pour chaque partie de l'outil hémisphérique utilisé dans l'opération de finition. Cet outil se compose de quatre parties, une partie sphérique et trois parties cylindriques (voir figure 3). La rubrique de détection s'occupe aussi de la visualisation des interférences et des collisions en temps réel avec la possibilité de visualisation de la surface, les régions ainsi que le spectre des erreurs. La détection des interférences et des collisions est faite par l'algorithme suivant :

Algorithme : détection des collisions et des interférences.

Entrée :
Cl(x0, y0, z0) : position de l'extrémité de l'outil.
r^1 : rayon de la partie sphérique et la première partie cylindrique.
r_2 : rayon de la deuxième partie cylindrique.
r_3 : rayon de la troisième partie cylindrique.
l_1 : hauteur du premier cylindre.
l_2 : hauteur du deuxième cylindre.
l_3 : hauteur du troisième cylindre.
Sortie : les collisions et les interférences.

Debut :
{
Pour chaque position de l'outil. (x_i, y_i, z_i)

Si $(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2 < r_1^2$ && $(z_0 < z_i < z_0 + r_1)$
L'outil entre en interférence avec la surface dans la partie sphérique.
FSi

Si $(x_i - x_0)^2 + (y_i - y_0)^2 < r_1^2$ && $(z_0 + r_1 \leq z_i \leq z_0 + r_1 + l_1)$
L'outil entre en collision avec la surface dans la première partie cylindrique.
FSi.

Si $(x_i - x_0)^2 + (y_i - y_0)^2 < r_2^2$ && $(z_0 + r_1 + l_1 \leq z_i \leq z_0 + r_1 + l_1 + l_2)$
L'outil entre en collision avec la surface dans la deuxième partie cylindrique.
FSi

Si $(x_i - x_0)^2 + (y_i - y0)^2 < r_3^2$ && $(z_0 + r_1 + l_1 + l_2 \leq z_i \leq z_0 + r_1 + l_1 + l_2 + l_3)$
L'outil entre en collision avec la surface dans la troisième partie cylindrique.
FSi
}
Fin

La fenêtre de détection des interférences et des collisions est présentée par la figure 3.

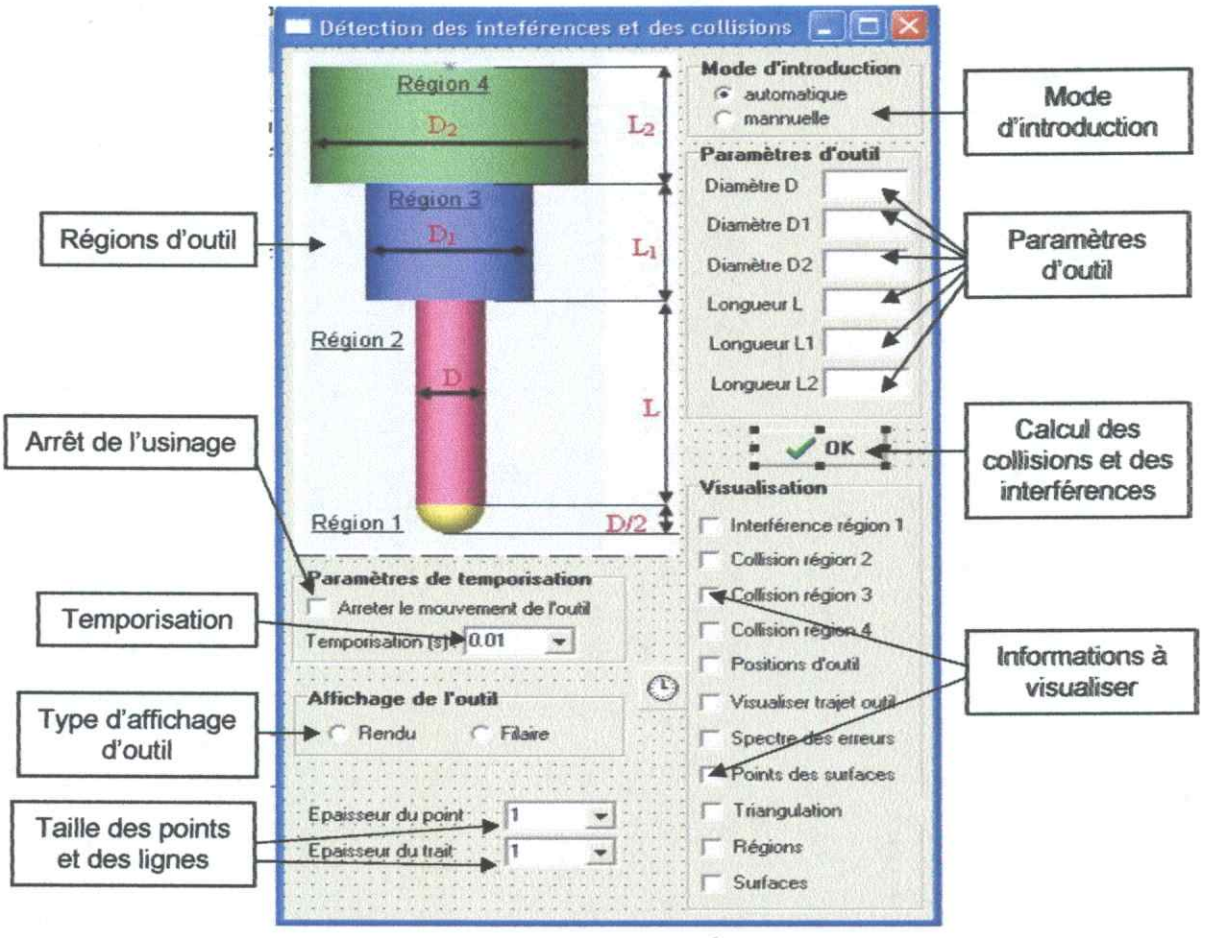


Figure 3. Fenêtre de calcul des collisions et des interférences.

L'algorithme de dessin de l'outil est le suivant :

Algorithme : dessin outil :
Entrée : r_1, r_2, r_3, l_1, l_2, l_3;
Sortie : outil hémisphérique.
Debut :
{
Créer objet de type sphère de rayon r_1 ;
Créer objet de type cylindre de rayon r_1 et de hauteurl_1 ;
Créer objet de type cylindre de rayon r_2 et de hauteurl_2 ;
Créer objet de type cylindre de rayon r_3 et de hauteurl_3 ;
Créer deux objets de type disque de rayon r_3 ;
Créer deux objets de type disque de rayon r_2.
}
Fin

III.4. Rubrique intermédiaire de récupération des contours :

Cette forme est utilisée pour récupérer les différentes positions de l'outil (nœuds) constituant le trajet d'usinage ainsi que les paramètres de l'outil (rayon et longueur).

Cette fenêtre est représentée par la figure 4.

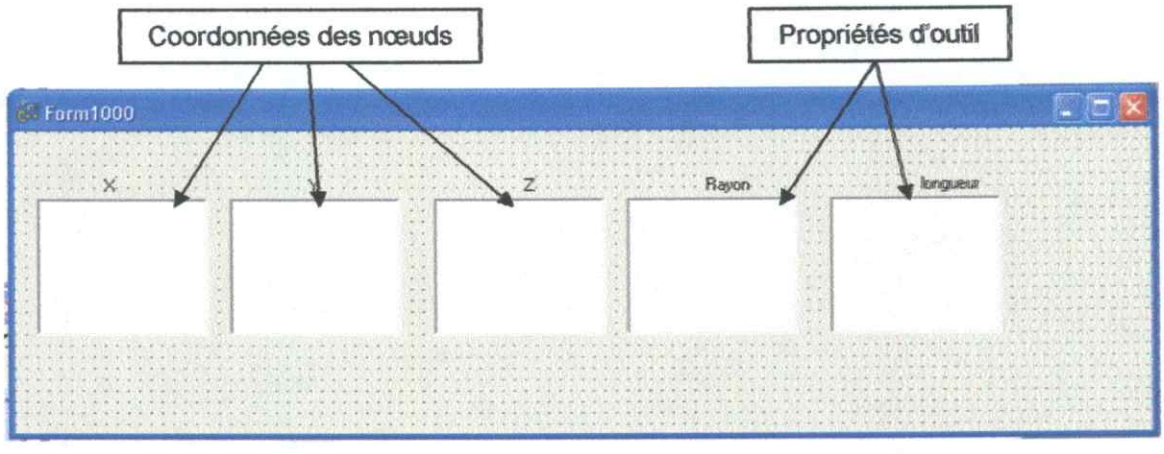


Figure 4. Fenêtre intermédiaire de récupération des nœuds.

III.5. Rubrique de création et de récupération des contours :

Cette très importante rubrique est caractérisée par la classe trajet qui englobe plusieurs propriétés. La classe trajet et définie comme suit :

```
Class Trajet
{
   Tlist_sommets      *T ;
   Trajet             *svt ;
   int                num_trajet ;
   int                nombre de sommets ;
   double             Rayon_outil ;
   double             longueur_outil ;
   double             plan_z ;
   double             vitesse_avance ;;
   double             sens_rotation;
   double             x_min_T, x_max_T, y_min_T, y_max_T, ;
   Ansistring         parametrisation,vect_nodal ;
   int                nombre_epoint_controle ;
   int                degre ;
   double             erreur_max ;
   double             erreur_moyenne ;
   bool               ferme;
   bool               affecte_sous_chemin ;
   bool               existe_engagement ;
   bool               existe_degagement
   int                num_h_s_affecte ;
   Tnoeuds            *trouver_fin() ;
   Void               ajouter_position_debut(double,int) ;
```

```
    Void            ajouter_position_fin(double,int) ;
    Void            retrier_trajet(Tnoeuds *) ;
    Tnoeuds         *plus_proche(Tnoeuds)
} ;
```

Maintenant, nous allons utiliser cette classe pour recréer les contours d'usinage. Cette nouvelle implémentation permet d'approximer ces contours par des courbes B-Spline et d'optimiser le chemin d'outil pour la stratégie d'usinage Z-Constant.

La fenêtre de récupération des contours est représentée par la figure 5.

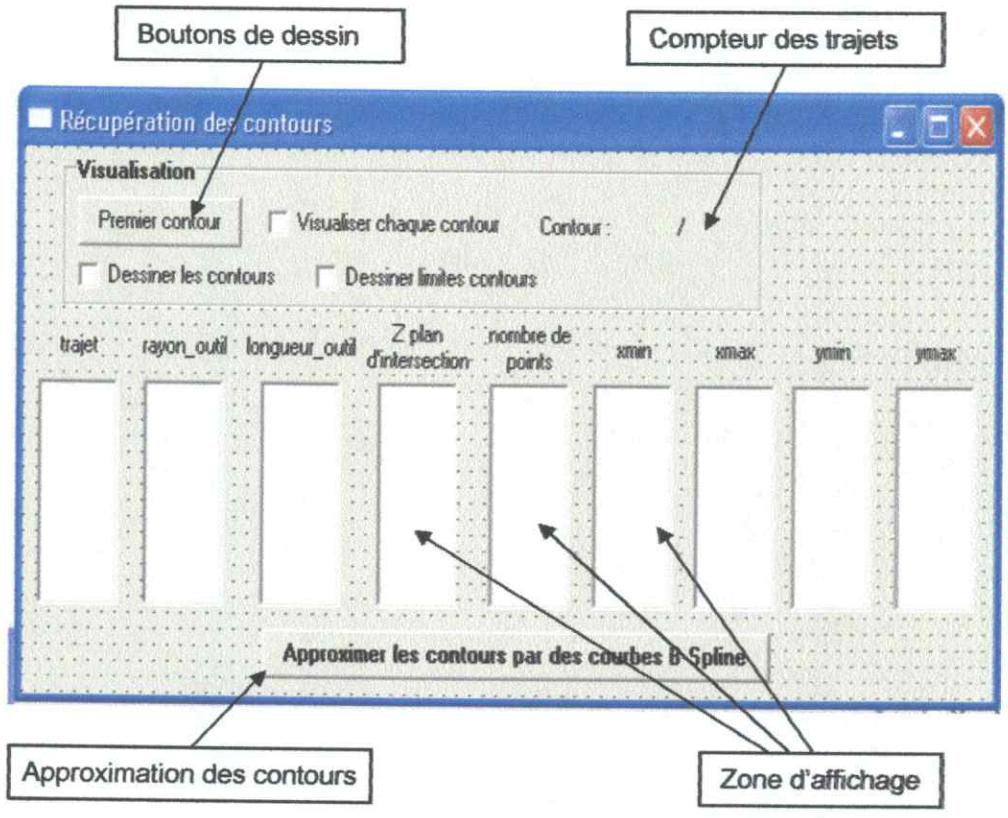


Figure 5. Fenêtre de récupération des contours.

III.6. Rubrique d'approximation :

Dans cette rubrique, nous allons utiliser des fonctions mathématiques pour approximer les contours par des courbes B-Spline. L'algorithme de récupération est le suivant :

Algorithme : approximation

Entrées : degre_courbe, precision,erreur_max, erreur_permise; Respecte ;
Sortie : les courbes approximées.

Debut :
{
Pour toutes les surfaces, récupérer
Le degré des courbes,

La précision des courbes,
Appeler_fonction d'approximation,

Tant que la Respecte==false.
{

1er cas :
Paramétrisation=uniforme
vect_nodal=uniforme.
afficher erreur max ; erreur permise ;
Si (erreur max ≤ erreur permise)
Respecte=true ;
Sortie de la boucle ;
FSi ;

2ème cas :
Paramétrisation=uniforme
vect_nodal=moyenne.
afficher erreur max ; erreur permise ;
Si (erreur max ≤ erreur permise)
Respecte=true ;
Sortie de la boucle ;
FSi ;

3ème cas :
Paramétrisation=centripète ;
vect_nodal= uniforme ;
afficher erreur max ; erreur permise ;
Si (erreur max ≤ erreur permise)
Respecte=true ;
Sortie de la boucle ;
FSi ;

4ème cas :
Paramétrisation= centripète ;
vect_nodal=moyenne.
afficher erreur max ; erreur permise ;
Si (erreur max ≤ erreur permise)
Respecte=true ;
Sortie de la boucle ;
FSi ;

5ème cas :
Paramétrisation=corde ;
vect_nodal= uniforme ;
afficher erreur max ; erreur permise ;
Si (erreur max ≤ erreur permise)
Respecte=true ;
Sortie de la boucle ;
FSi ;

6ème cas :

Paramétrisation=corde ;
vect_nodal=moyenne.
afficher erreur max ; erreur permise ;
Si (erreur max ≤ erreur permise)
Respecte=true ;
Sortie de la boucle ;
FSi ;

7ème cas :

Si encore Respecte==false
Nbre poitn de contrôle+5 ;
}
Fin tant que ;
Afficher approximation ;
}
Fin

La figure 6 représente la fenêtre d'approximation.

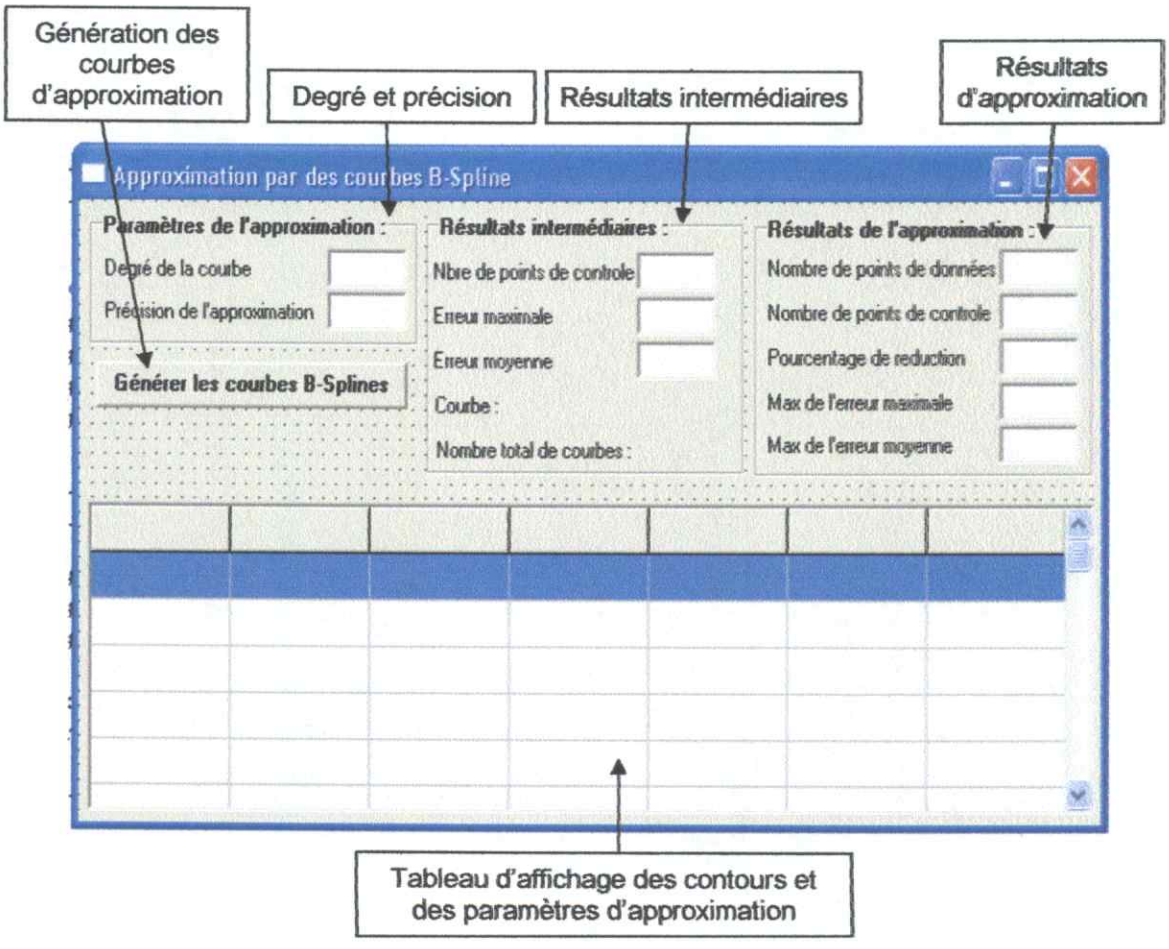


Figure 6. Fenêtre de génération des courbes B-Spline par approximation.

III.7. Rubrique d'optimisation :

Cette partie permet de trouver un chemin d'outil optimum. Le travail dans cette partie est divisé en cinq étapes. La première étape est le classement des trajets par plans, la deuxième est la création des hille sides et le classement des hille sides par plan, la troisième est l'utilisation des hille sides pour générer les sous chemins, ensuite choisir une méthode d'usinage pour générer le chemin d'outil total (One-Way ou Zig-Zag), la quatrième est la simulation d'usinage et la dernière étape c'est la génération du programme d'usinage « G-Code ».

L'algorithme de classement des trajets par plan est comme suit :

Algorithme : classement des trajets par plan

Entrée : les trajets, nombre de plans,
Sortie : les trajets triés par plan, nombre de plans int,

Debut :
{
Pour i=1 jusqu'au nombre de trajets
Trouver le nombre des plans et le Z de chaque plan,
Fin pour,
Pour j=1 jusqu'au nombre de plans
Pour k=1 jusqu'au nombre de trajets
Si (plan Z de trajet==plan Z de plan)
ajouter trajet à la liste des plans.
FinSi ;
Fin pour ;
Fin pour ;
}
Fin ;

La notion des hille sides [24] : le hille side est une zone entre deux plans consécutifs (voir figure 7), elle permet d'extraire les conditions d'usinage entre plan.

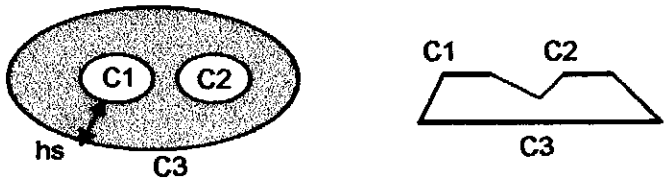


Figure 7. Structure des hille sides.

Nous allons proposer la structure de données hille sides comme suit :

Class hille sides
{
Une liste des trajets supérieurs ;
Une liste des trajets inférieurs,

Le hille side suivant,
},

L'algorithme de création des plans hille sides est présenté comme suit :

Algorithme : créer hille sides ;

Entrée : les plans des trajets ;
Sortie : les plans des hille sides

```
Debut :
{
   Pour  i=1 jusqu'au nombre de plans
         J=i+1 ;
         Parc1=pointer sur le premier trajet du plan i
         Tant que parc1 différent de NULL ;
               Parc2=pointer sur le premier trajet du plan j
               Tant que parc2 différent de NULL ;
                     Si (parc1 se chevauche avec parc2)
                           Si parc1non affecté et parc2 non affecté
                           Créer hille sides
                           Ajouter parc1 à la liste supérieur ;
                           Ajouter parc2 à la liste inférieur ;
                           Ajouter hille sides à la liste i des hille sides ;
                           FinSi
                           Si parc1 affecté et parc2 non affecté
                           Ajouter parc2 à la liste inférieur où parc1 existe ;
                           FinSi ;
                           Si parc1non affecté et parc2 affecté
                           Ajouter parc1 à la liste inférieur où parc2 existe ;
                           FinSi ;
                     FinSi ;
               Fin Tant que ;
         Fin Tant que ;
   Fin pour ;
}
Fin ;
```

Après la construction des plans hille sides, nous allons maintenant utiliser ces plans pour construire les sous chemins. Un sous chemin est une liste de trajets qui contient les trajets qui peuvent être usinés dans une même séquence.

On peut caractériser le sous chemin par un pointeur sur le premier trajet du sous chemin. La classe sous chemin est présenté comme suit :

```
Class sous chemin
{
Trajet              *debut ;
sous chemin         *svt ;
sous chemin()        { debut=NULL ;}
```

```
void                    ajouter_trajet(Trajet *) ;
}
```

La construction des sous chemins à partir des hille sides est présentée par l'algorithme suivant :

Algorithme : construire sous chemin ;

Entrée : les plans hille sides ;
*parc1,*parc2 hille sides ;
*courant,*courant1, *superieur, *inferieur Trajet ;
Sortie : les sous chemin ;

Debut :
{
i=1
J=i+1 ;
Parc1=pointer sur le premier hille sides du plan i ;
Parc2=pointer sur le premier hille sides du pan j ;
Si (parc1→superieur→svt ==NULL)
Et (parc1 → inferieur→svt== NULL)
Créer nouveau sous chemin ;
Chainer sous chemin ;
Ajouter parc1→ superieur ;
Ajouter parc1→ inferieur ;
Si (parc1 → inferieur ==parc2→ suprieur)
et(parc2 →superieur→ svt ==NULL)
Ajouter parc2 → superieur ;
Ajouter parc2 → inferieur ;
Faire le meme travail pour les plans restants ;
FinSi ;
FinSi ;
Si (parc1→superieur→svt != NULL)
et (parc1 → inferieur→svt== NULL)
Courant=pointer sur le premier trajet de parc1
Tant que courant →svt!=NULL
Créer nouveau sous chemin ;
Chainer sous chemin ;
Ajouter courant ;
courant= courant→svt ;
Fin Tant que ;
Créer nouveau sous chemin ;
Chainer sous chemin ;
Ajouter courant;
Ajouter parc1→inferieur ;
Parcourir les plans restant et faire le même travail ;
FinSi ;
Si (parc1→superieur→svt == NULL)
et (parc1 → inferieur→svt != NULL)
créer nouveua sous chemin ;

```
                chainer sous chemin ;
                ajouter parc1→superieur ;
                ajouter parc1→ inferieur ;
                courant=parc1→ inferieur→svt ;
                tant que courant !=NULL
                        créer nouveua sous chemin ;
                        chainer sous chemin ;
                        ajouter courant ;
                        parcourrir les plans restants ;
                        courant= courant→svt ;
                FinTantQue ;
        FinSi ;
        Si (parc1→superieur→svt != NULL)
        et (parc1 → inferieur→svt != NULL)
        Courant=pointer sur le premier trajet de parc1
                Tant que courant →svt!=NULL
                        Créer nouveau sous chemin ;
                        Chainer sous chemin ;
                        Ajouter courant ;
                        courant= courant→svt ;
                Fin Tant que ;
                Créer nouveau sous chemin ;
                Chainer sous chemin ;
                Ajouter courant;
                Ajouter parc1→inferieur ;
                Parcourir les plans restant et faire le même travail ;
                courant1= parc1→inferieur →svt ;
                Tant que courant 1 !=NULL
                        Créer nouveau sous chemin ;


                        Ajouter courant1 ;
                        Parcourir les plans restants ;
                        courant1= courant1→svt
                FinTantQue ;
        FinSi ;
}
Fin
```

Après la création des sous chemins, nous allons optimiser encore une autre fois ces sous chemin en utilisant l'algorithme suivant :

Algorithme : optimiser sous chemins ;

Entrée : la liste des sous chemins ;
*s_courant1, *s_courant2 sous chemin ;
*fin Trajet ;
Sortie : une liste de sous chemins optimisée ;

```
Debut :
{
   Tant que s_courant1 != NULL
   s_courant1=pointer sur le premier sous chemin ;
   s_courant2=pointer sur s_courant1→svt ;
   chercher le dernier trajet( fin) dans courant1 ;
   tant que s_courant2 !=NULL
   Si ((fin) == s_courant2)
         fin→svt= s_courant2→premier ;
   FinSi ;
   FinTantQue ;
   FinTantQue ;
}
Fin
```

Après la construction des sous chemins optimisés, nous allons construire le trajet final d'usinage pour deux modes de balayage One-Way et Zig-Zag

III.7.1. Usinage en One-Way :

L'usinage en One-Way se fait dans un seul sens (aller). Lorsque l'outil termine l'usinage d'un contour quelconque, il quitte la matière et recommence l'usinage à partir du trajet suivant. L'algorithme permettant la génération du trajet final par la méthode One-Way est le suivant :

Algorithme : chemin One-Way ;

Entrée : la liste des sous chemin ;
Sortie : le trajet One-Way ;

```
Debut :
{
   Parc=pointer sur le premier sous chemin
   Tant que parc!=NULL
         Courant=pointer sur le premier trajet de parc ;
         Tant que courant !=NULL
               Ajouter position du premier engagement ;
               Ajouter position du deuxième engagement ;
               Ajouter position du premier dégagement ;
               Ajouter position du deuxième dégagement ;
         FinTantQue ;
   FinTantQue ;
}
Fin
```

III.7.2. Usinage en Zig-Zag:

L'usinage se fait en aller retour. Si on usine un trajet fermé et le trajet suivant est fermé alors lorsque l'outil termine l'usinage du premier trajet, commence l'usinage du trajet suivant sans quitter la surface. Dans cette stratégie d'usinage, on corrige les

trajets par le calcul du nœud le plus proche. L'algorithme permettant la génération du trajet final par la méthode Zig-Zag est le suivant :

Algorithme : chemin Zig-Zag ;

Entrée : la liste des sous chemin ;
Sortie : le trajet Zig-Zag ;

Debut :
{
Parc=pointer sur le premier sous chemin
Tant que parc!=NULL
Courant=pointer sur le premier trajet de parc ;
Ajouter position du premier engagement ;
Ajouter position du deuxième engagement
Tant que courant !=NULL
1er cas :
Si (courant = fermé et courant→svt=fermé)
chercher la fin du courant, ;
chercher le nœuds le plus proche dans courant→svt ;
retrier le trajet courant→svt ;
FinSi ;
2ème cas
Si (courant = fermé et courant→svt=ouvert)
Ajouter position du premier dégagement à courant;
Ajouter position du deuxième dégagement à courant ;
Ajouter position du premier engagement à courant→svt;
Ajouter position du deuxième engagement à courant→svt ;
FinSi ;
3ème cas
Si (courant = ouvert et courant→svt= fermé)
Ajouter position du premier dégagement à courant;
Ajouter position du deuxième dégagement à courant ;
Ajouter position du premier engagement à courant→svt;
Ajouter position du deuxième engagement à courant→svt ;
FinSi ;
4ème cas
Si (courant = ouvert et courant→svt=ouvert)
Ajouter position du premier dégagement à courant;
Ajouter position du deuxième dégagement à courant ;
Ajouter position du premier engagement à courant→svt;
Ajouter position du deuxième engagement à courant→svt ;
FinSi ;
5ème cas
Si (un seul trajet)
Ajouter position du premier engagement à courant;
Ajouter position du deuxième engagement à courant ;
FinTantQue ;
FinTantQue ;
}
Fin

Après la génération de chemin total d'outil, il ne reste que la simulation d'usinage et la génération du programme d'usinage « G-Code ». La simulation en réalité est la visualisation du trajet avec un usinage virtuel d'outil. Cette simulation permet de vérifier les résultats de tous nos travaux.

La figure 8 représente la fenêtre d'optimisation du trajet d'usinage.

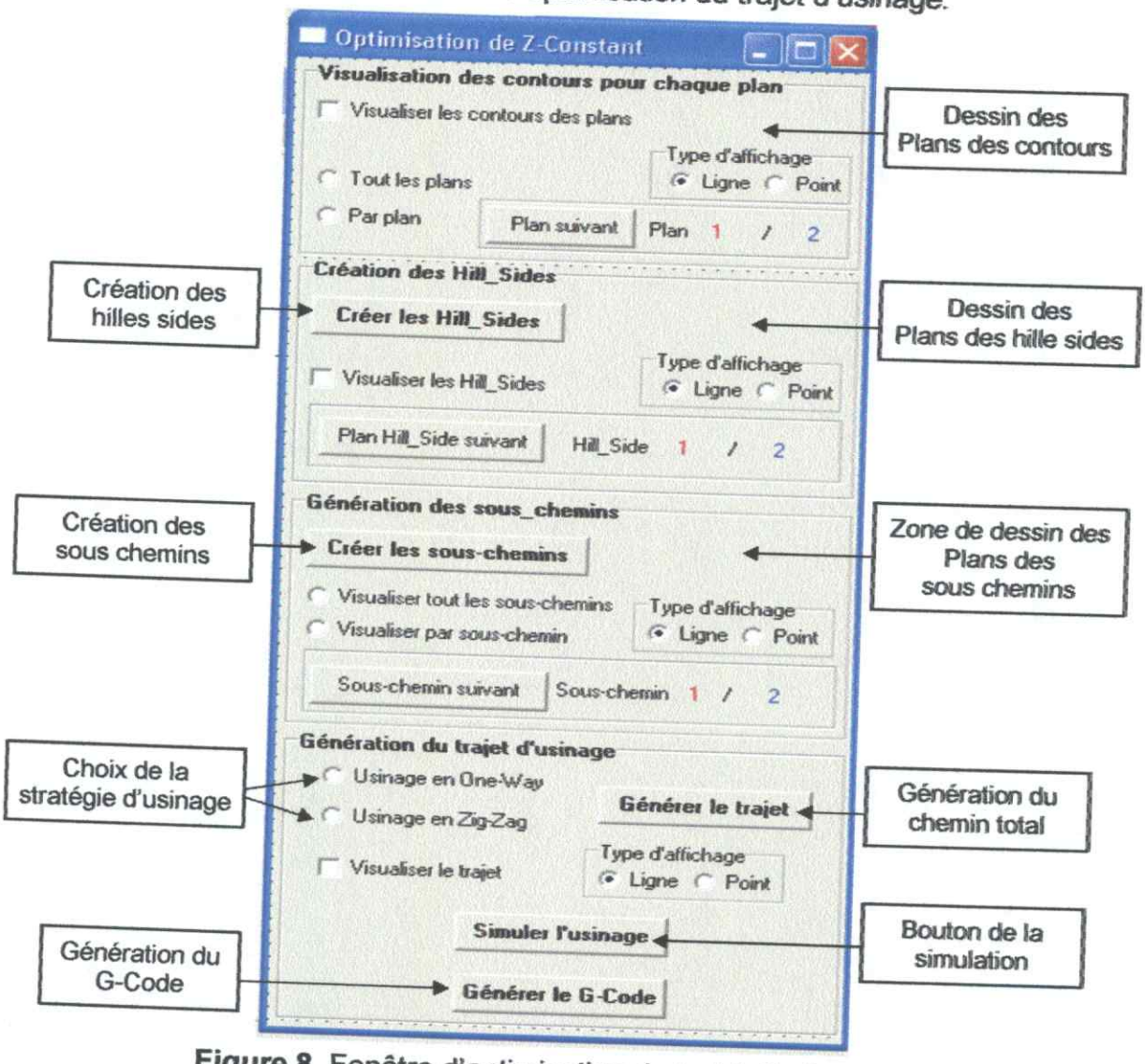


Figure 8. Fenêtre d'optimisation du trajet d'usinage.

La figure 9 représente les instructions du « G-Code » qu'il faut sauvegarder.

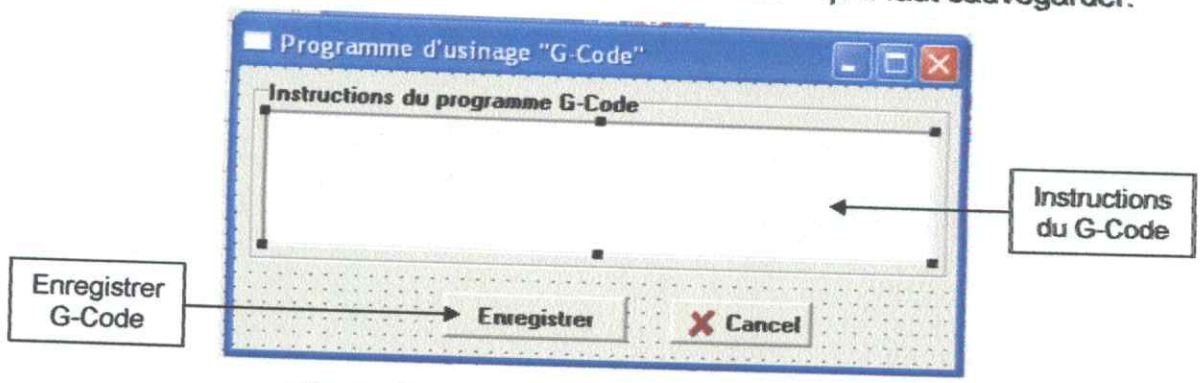


Figure 9. Fenêtre d'affichage du G-Code.

La figure 10 représente la fenêtre de la simulation d'usinage.

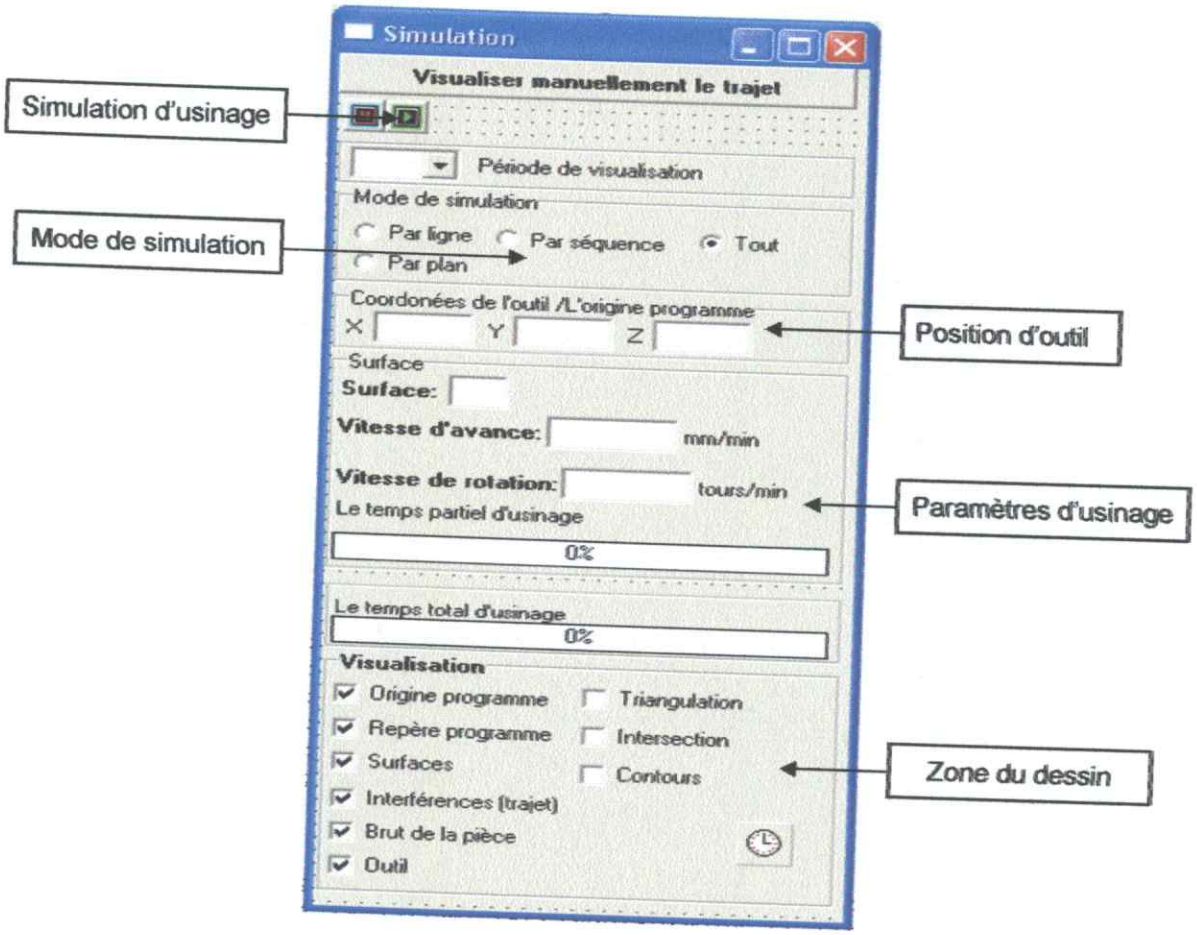
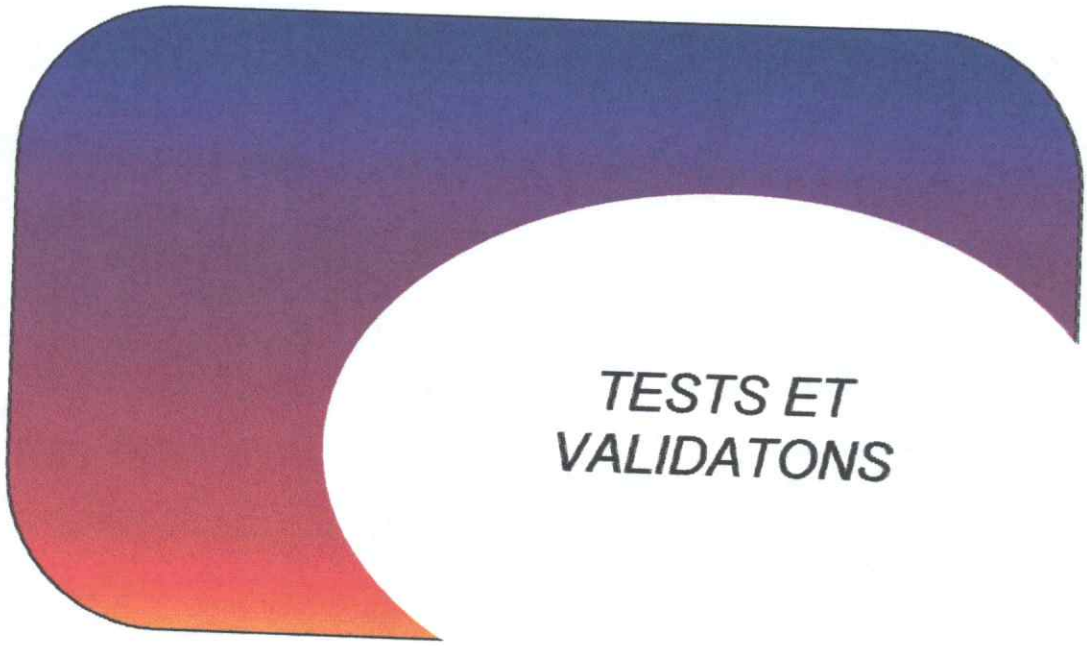


Figure 10. Fenêtre de simulation.

IV. CONCLUSION :

Dans ce chapitre, nous avons présenté l'environnement de travail, les principaux algorithmes utilisés et les différentes fenêtres de notre application logicielle permettant l'interaction avec l'utilisateur.

CHAPITRE 7



I. INTRODUCTION :

Dans le chapitre précédent nous avons présenté l'implémentation de notre application logicielle. Dans ce dernier chapitre, nous allons tester les différents cas d'utilisation cités précédemment et faire les corrections nécessaires pour finaliser l'application.

II. TESTS ET VALIDATIONS :

II.1. Détection des collisions et des interférences:

Nous allons tester le premier cas d'utilisation, c'est la triangulation et la division de la surface (voir figure1).

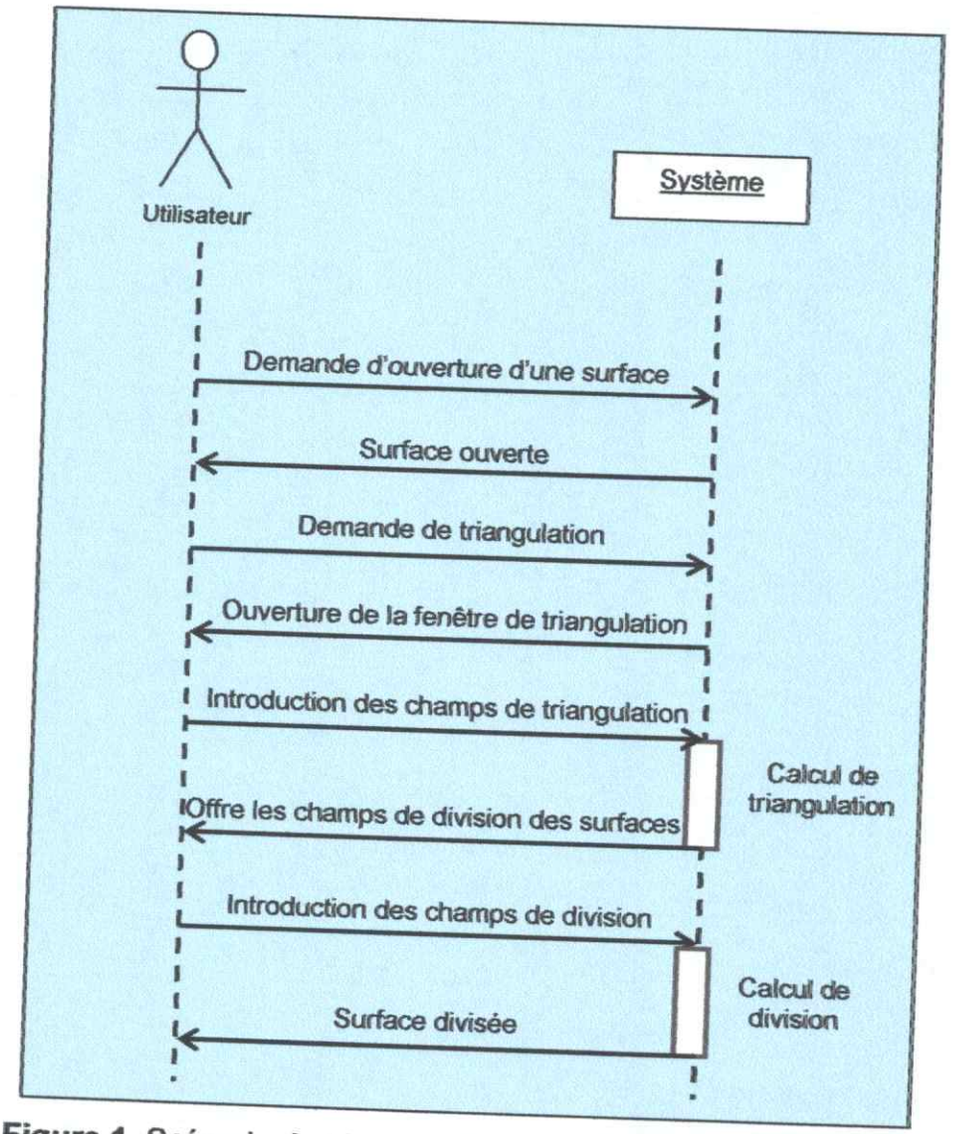


Figure 1. Scénario de triangulation et de division de la surface.

Nous allons commencer par l'ouverture d'une surface (voir figure2). Nous avons préféré la surface « demi vase cavité » car elle comporte des parties convexes et des parties concaves, ce qui permet de voir les collisions et les interférences pour les différentes parties de l'outil.

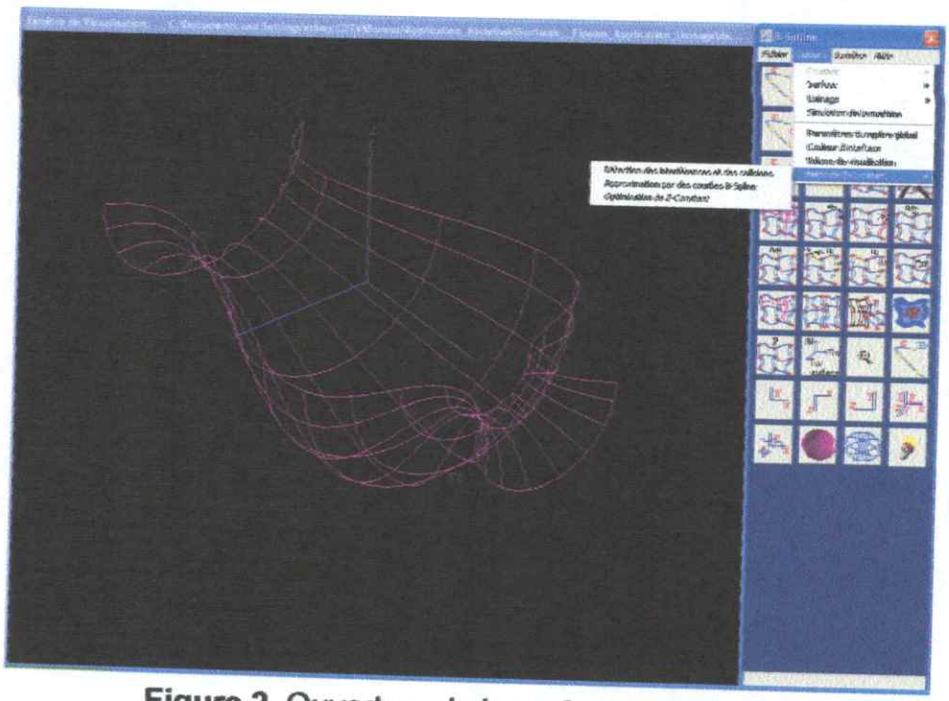


Figure 2. Ouverture de la surface considérée.

Après l'ouverture de la surface, nous allons trianguler avec une triangulation uniforme et diviser la surface en régions avec les paramètres suivants (voir figure 3) :

- Le nombre de points suivant la direction U est égal à 100.
- Le nombre de points suivant la direction U est égal à 100.
- Le nombre de régions suivant X est égal à 5.
- Le nombre de régions suivant Y est égal à 5.

La figure 3 représente les points de la surface.

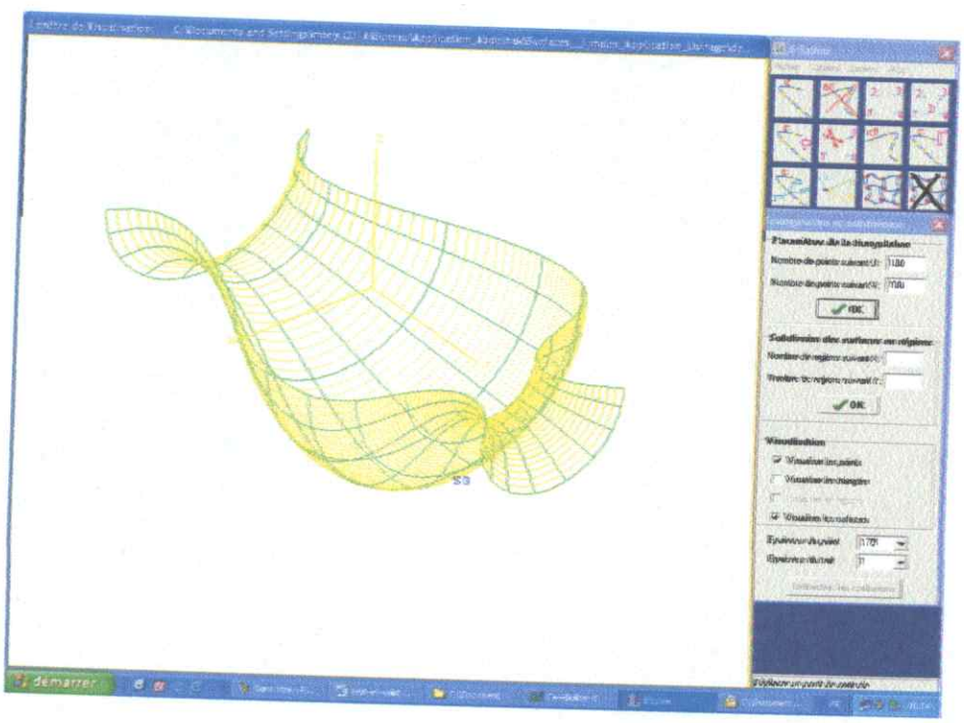


Figure 3. Points de la surface.

Les triangles de la surface sont représentés par la figure 4.

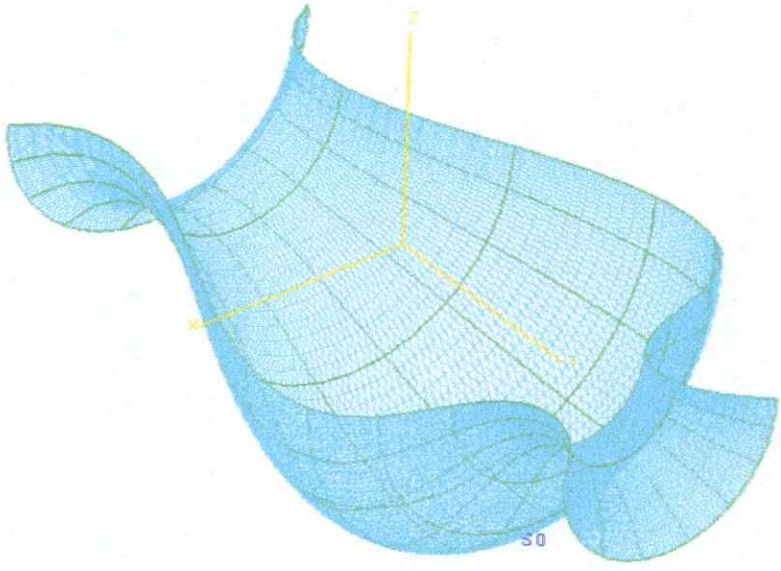


Figure 4. Triangles de la surface.

Les différentes régions obtenues sont présentées par la figure 5.

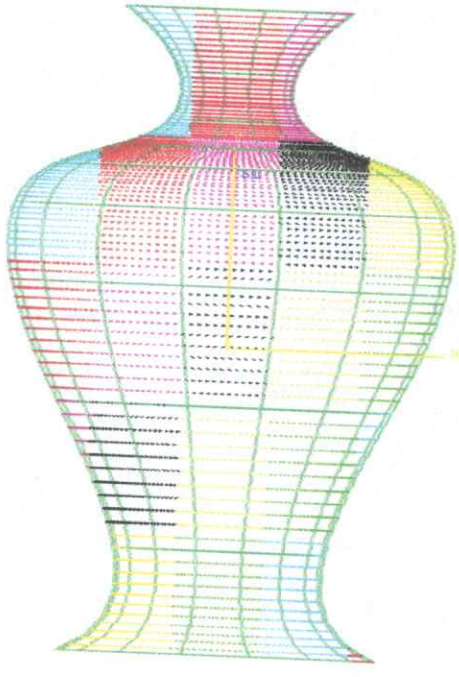


Figure 5. Régions de la surface

Après la visualisation des régions, nous allons passer maintenant à la détection effective des interférences et des collisions entre les différentes parties de l'outil et de la surface à usiner pour chaque position de l'outil sur le trajet d'usinage (voir figure 6).

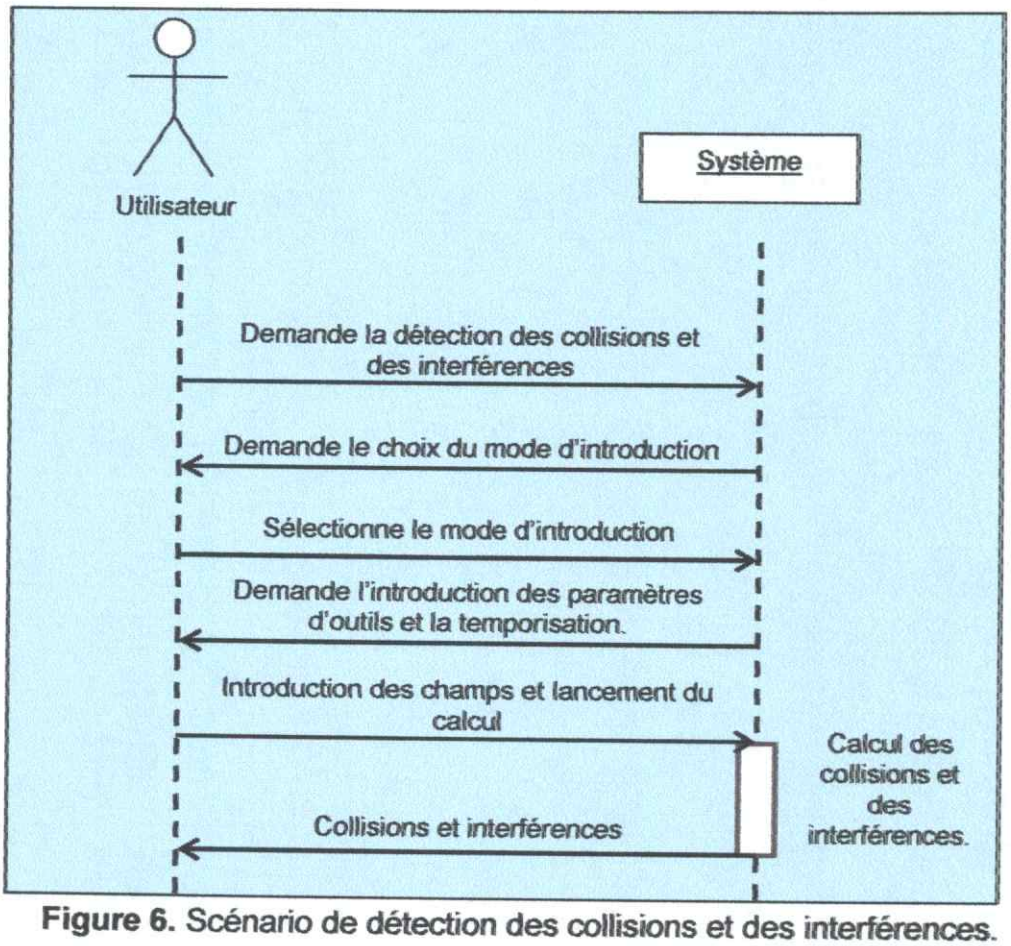


Figure 6. Scénario de détection des collisions et des interférences.

Lorsque l'utilisateur cherche les collisions et les interférences, la fenêtre des collisions apparaît (voir figure7). Dans ce cas, l'utilisateur sélectionne le mode d'introduction et introduit les paramètres de l'outil. Nous allons présenter plusieurs types de visualisations.

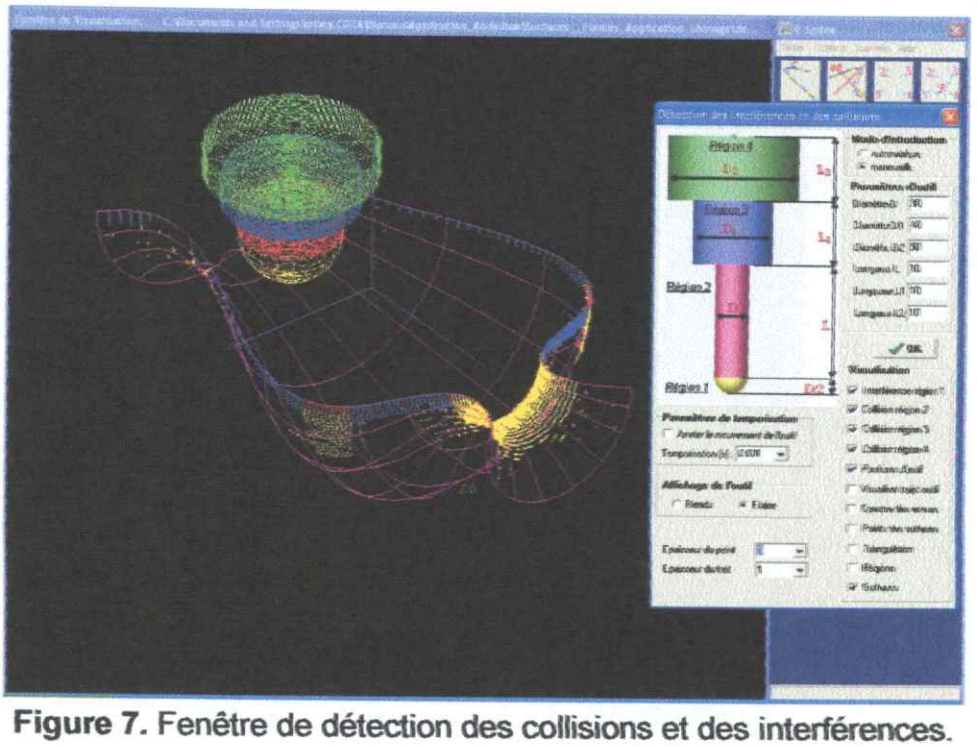
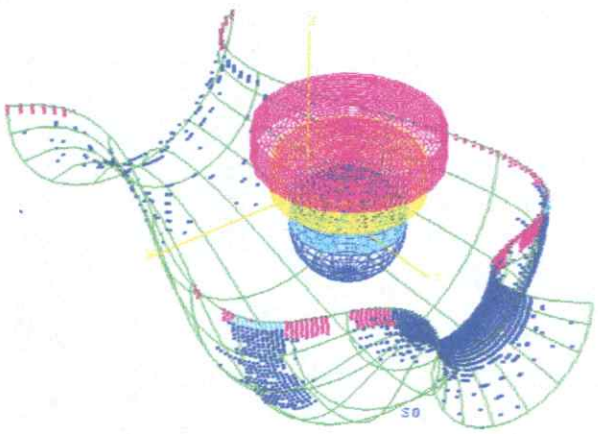
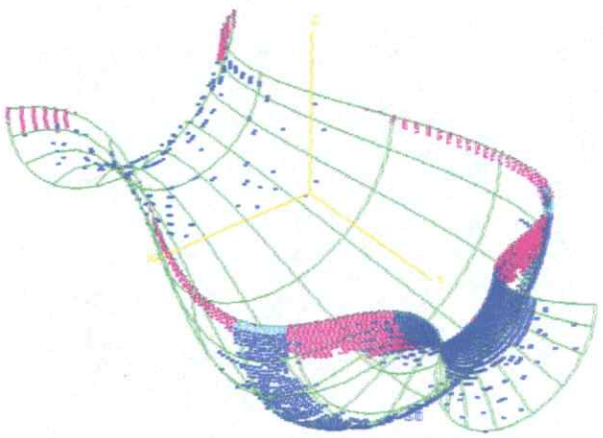


Figure 7. Fenêtre de détection des collisions et des interférences.

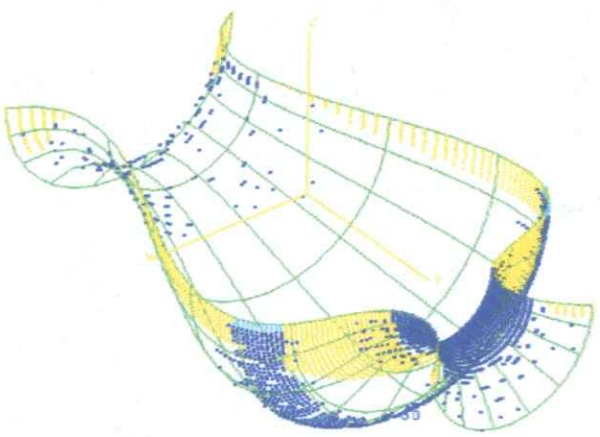
La figure 8.A montre la détection en temps réel des collisions avec visualisation de l'outil et des différentes erreurs localisées. La figure 8.B montre les collisions pour les régions 1, 2 et 4. La figure 8.C montre les collisions pour les régions 1, 2 et 3. Le spectre des erreurs après la fin de la détection est représenté par la figure 8.D.



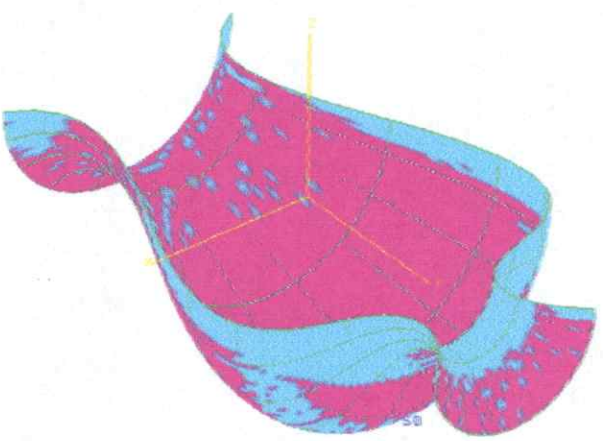
A. Détection en temps réel.



B. Collision régions 1, 2 et 4.



C. Collision régions 1, 2 et 3.



D. Spectre des erreurs.

Figure 8. Points d'interférences et de collisions.

Après la détection des collisions et des interférences, il est nécessaire de choisir un autre outil adéquat pour éviter ces problèmes.

II.2. Approximation des contours par des courbes B-Spline :

Nous allons passer maintenant au cas d'utilisation approximation des contours par des courbes B-Spline. Cette partie consiste à trouver le nombre de points de contrôle minimum nécessaire pour un degré de courbe et une précision fixés. Le scénario est le suivant :

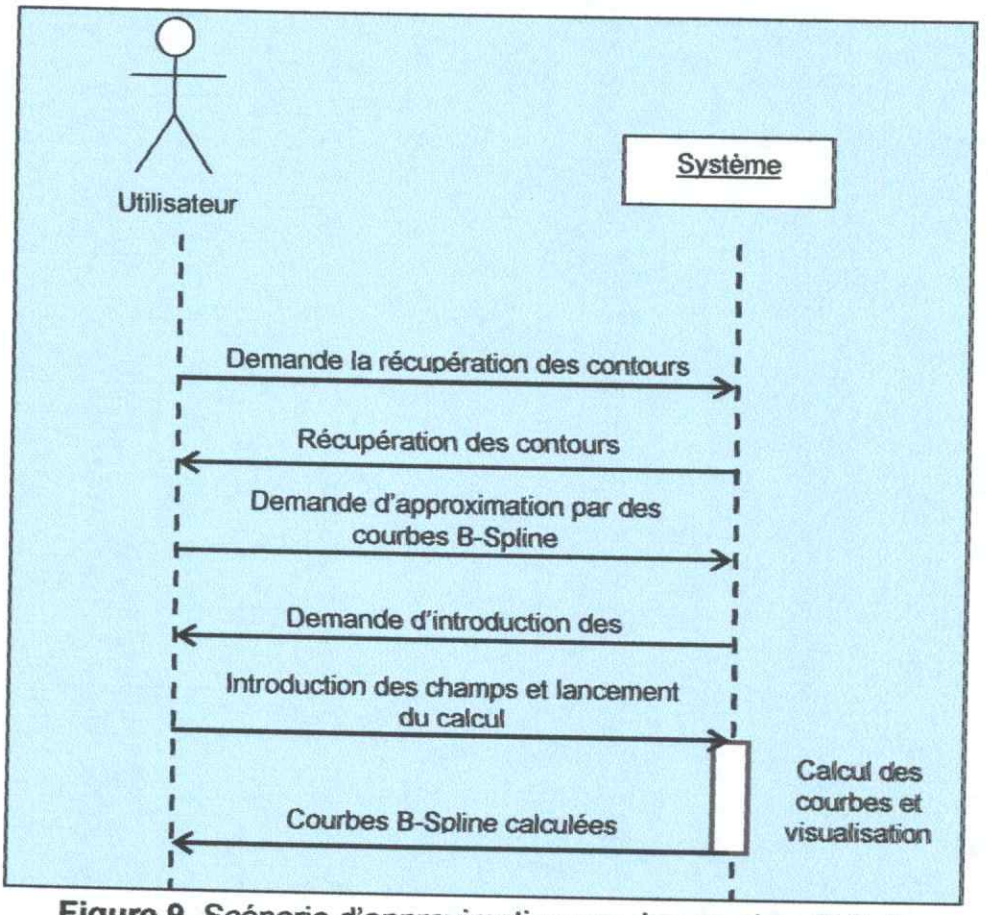


Figure 9. Scénario d'approximation par des courbes B-Spline.

Lorsque l'utilisateur demande l'approximation des contours pour une surface donnée (voir figure 10), les détails des contours (nombre de trajets, rayon d'outil, longueur d'outil, plan d'intersection, nombre de points de contrôle et les limites des contours) sont récupérés automatiquement (voir figure 11). Cette fenêtre permet de visualiser les contours suivant deux méthodes différentes.

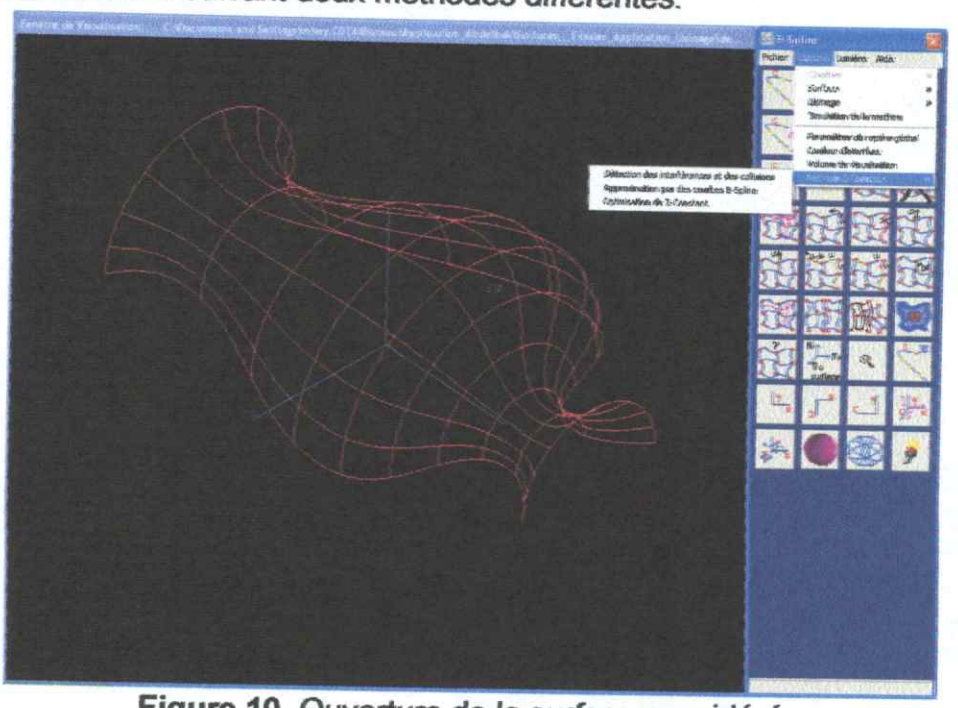


Figure 10. Ouverture de la surface considérée.

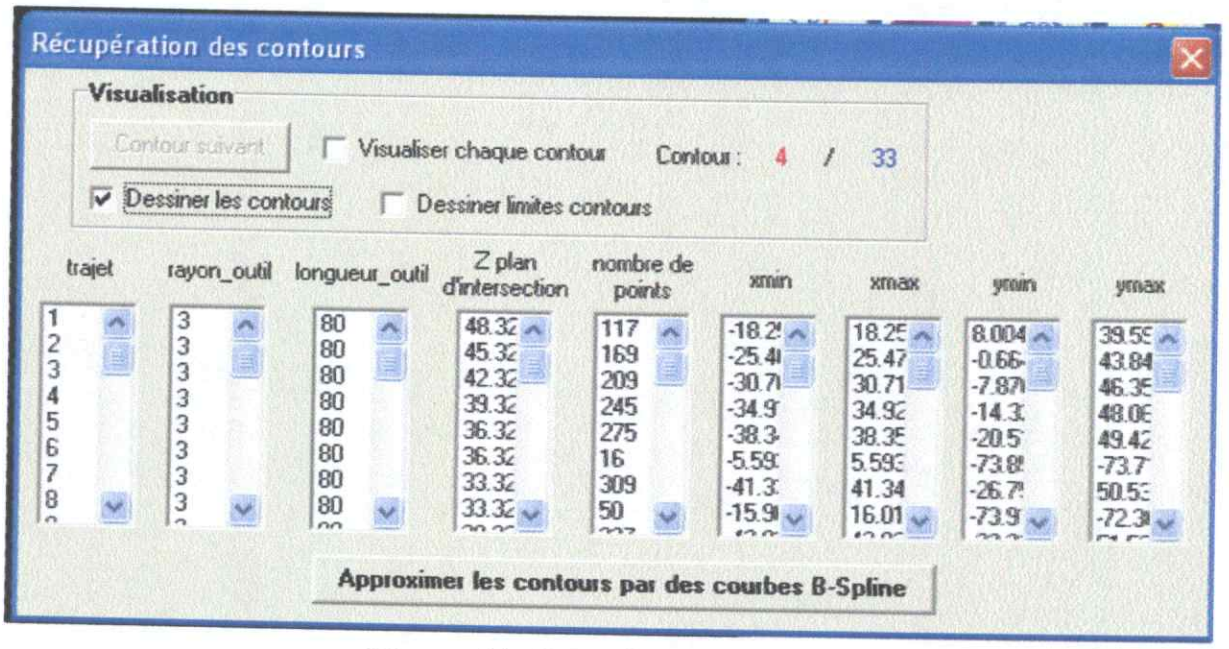


Figure 11. Paramètres des contours

La figure 12 représente différentes positions d'outil constituant le trajet d'usinage sous forme de contours pour la surface considérée.

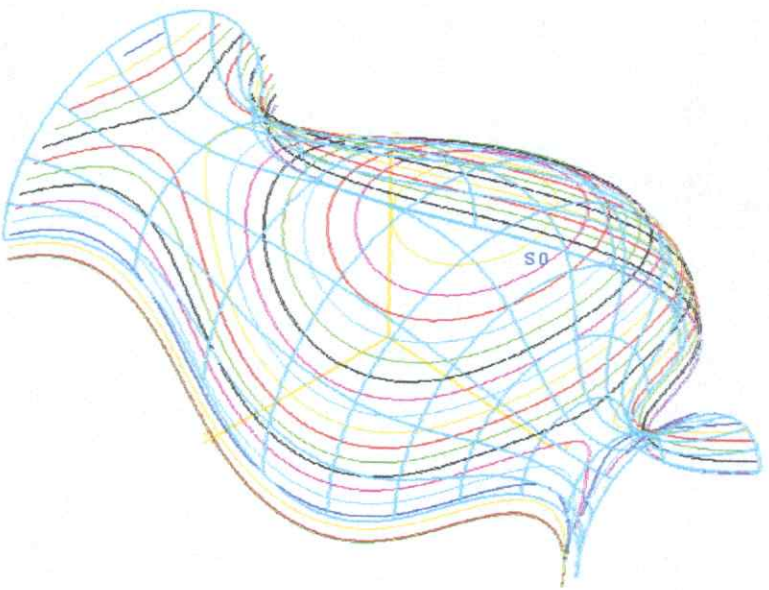


Figure 12. Visualisation des contours.

Après la récupération correcte des contours, nous passons maintenant à l'approximation de ces contours par des courbes B-Spline. Pour cela, après que l'utilisateur demande l'approximation des contours, la fenêtre d'approximation apparaît et l'utilisateur est invité à introduire le degré de la courbe et la précision d'approximation voulue (voir figure13). Pendant les calculs, les résultats intermédiaires et les résultats finaux sont affichés (nombre de points de contrôle, pourcentage de réduction, l'erreur maximale et l'erreur moyenne ainsi que les méthodes de calcul des valeurs nodales et les vecteurs nodaux).

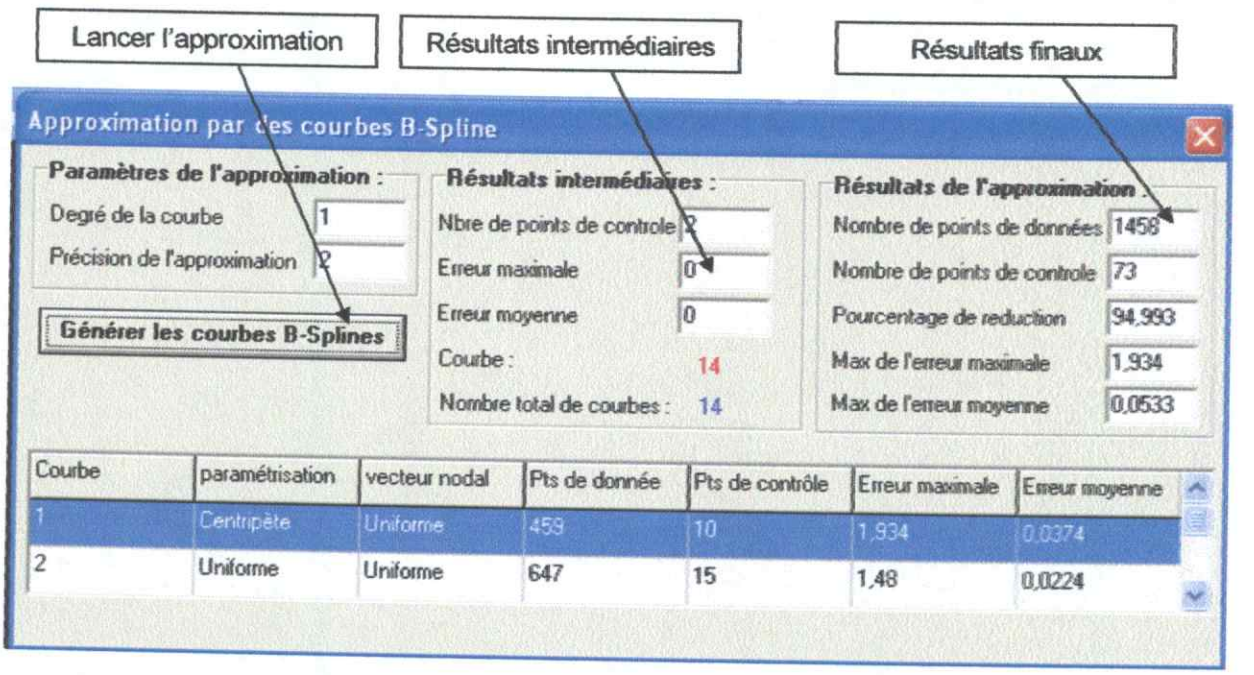
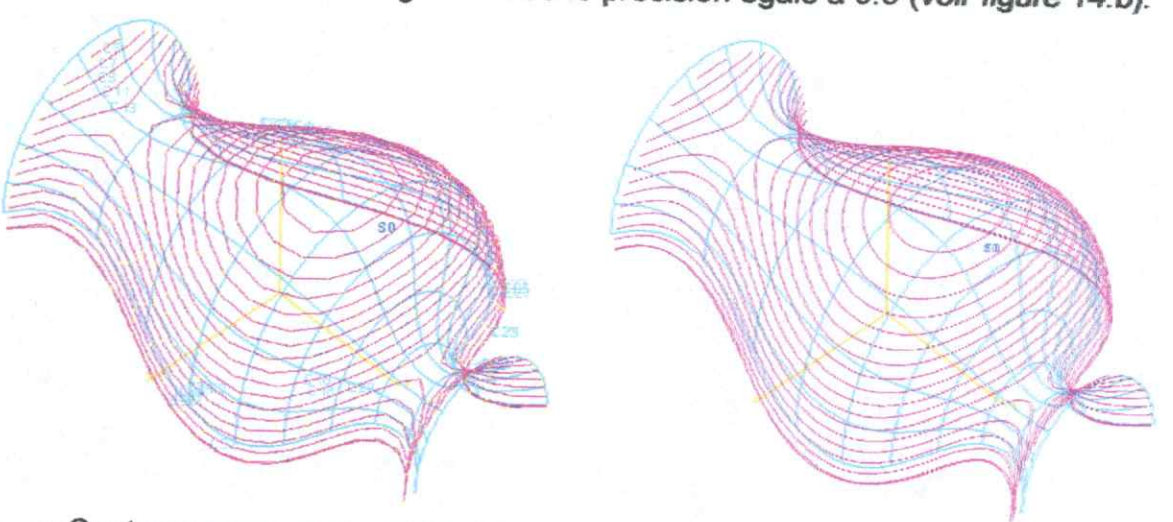


Figure 13. Affichage des résultats de l'approximation.

Les deux figures suivantes montrent deux approximations des mêmes contours avec les paramètres suivantes :

- Degré de la courbe égal à 1 et une précision égale à 1 (voir figure 14.a).
- Degré de la courbe égal à 3 et une précision égale à 0.5 (voir figure 14.b).



a. Contours avec un degré égal 1. b. Contours avec un degré égal 3.

Figure 14. Génération des contours par approximation.

D'après les deux figures ci-dessus, nous remarquons que la figure 14.b nous donne des courbes plus lisses que la figure 14.a.

II.3. Optimisation du trajet d'usinage pour la stratégie d'usinage Z-Constant :

Nous allons maintenant tester le dernier cas d'utilisation qui concerne l'optimisation du trajet d'usinage pour la stratégie d'usinage Z-Constant. Le scénario est représenté par la figure 15.

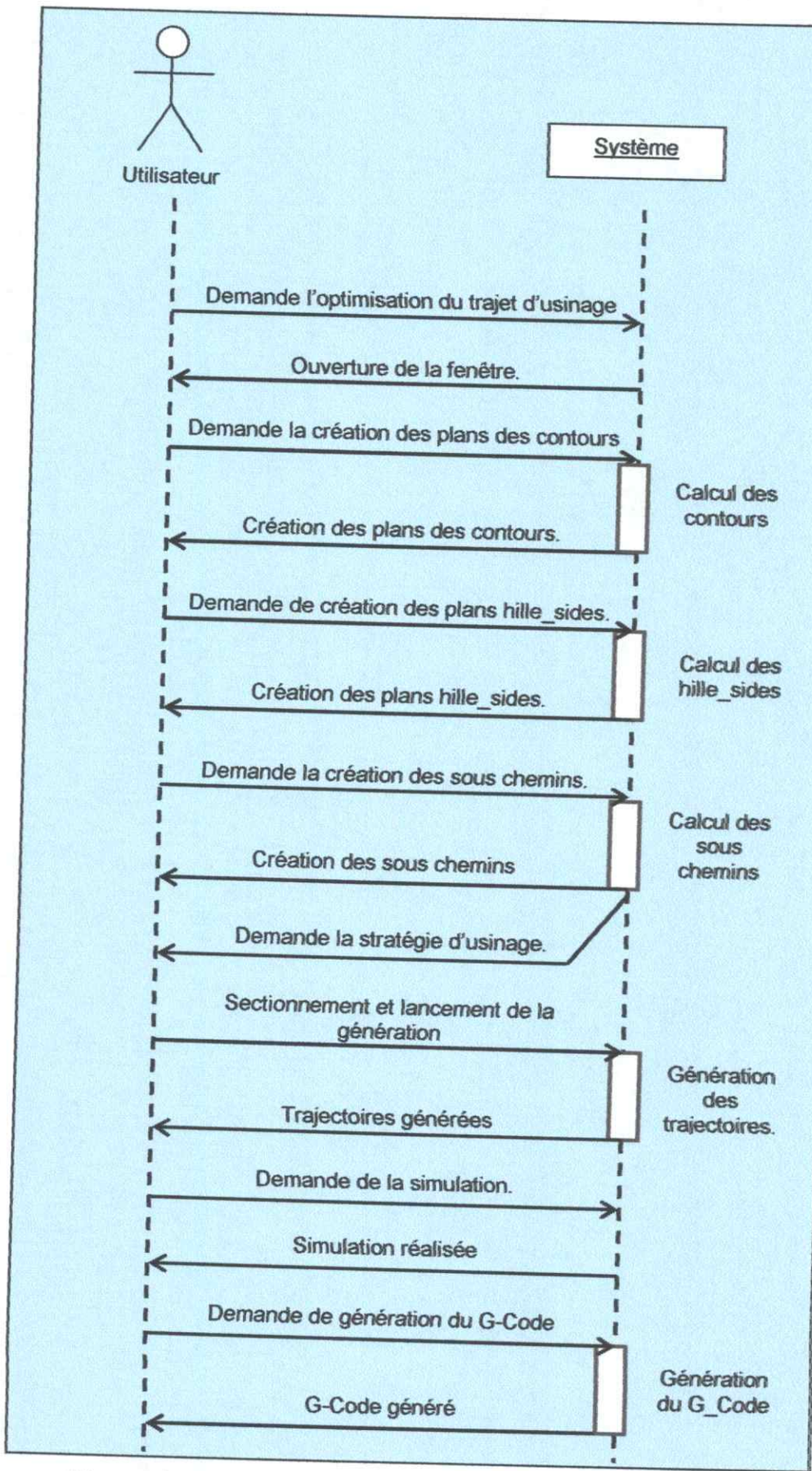


Figure 15. Scénario d'optimisation du trajet d'usinage.

Dans un premier pas l'utilisateur commence par l'ouverture de la surface (voir figure 16) et en même temps les contours des plans sont récupérés automatiquement (voir figure 17.a). Ces plans peuvent être visualisés de deux façons, tous les plans à la fois (voir figure 17.b) ou plan par plan (voir figure 17.c).

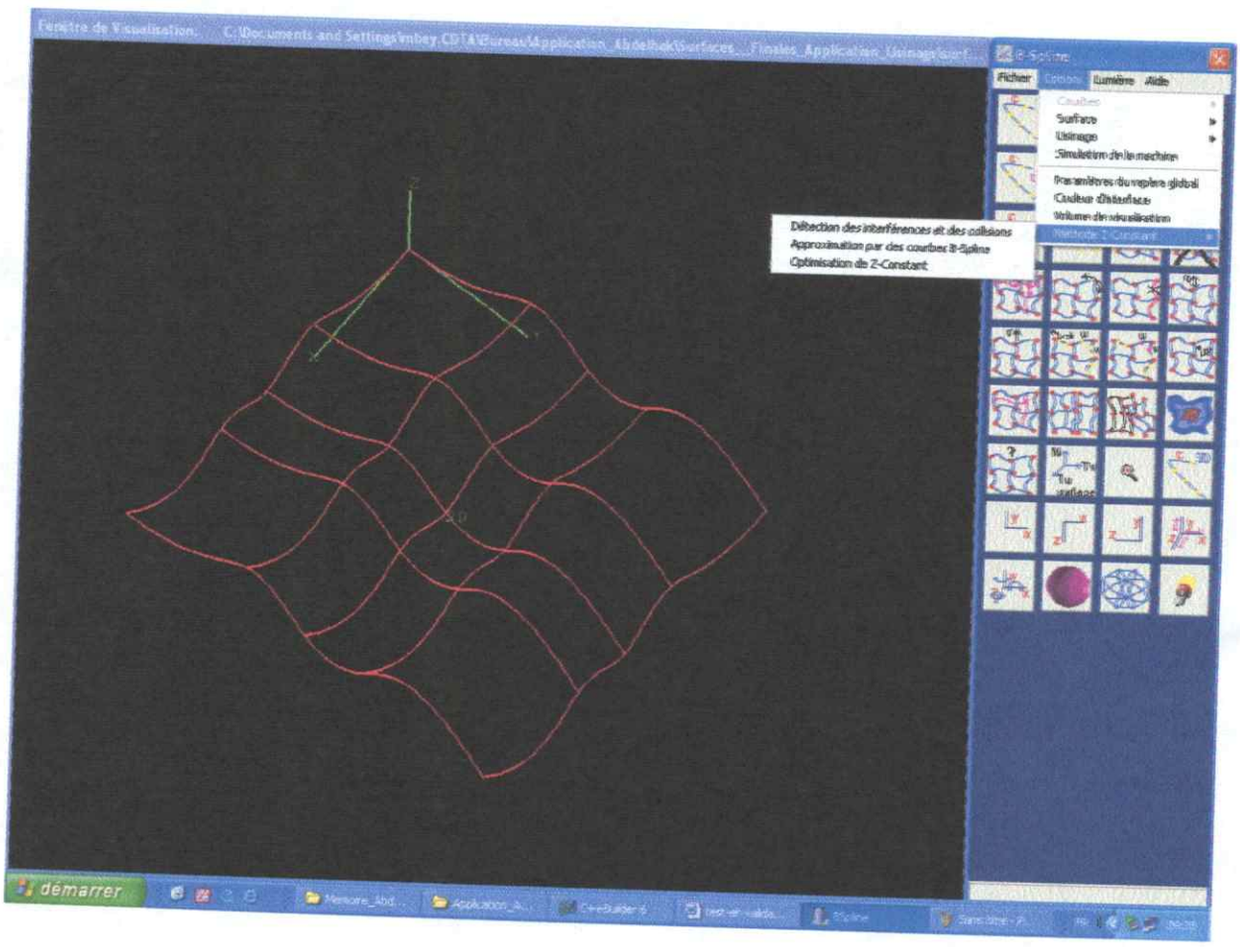
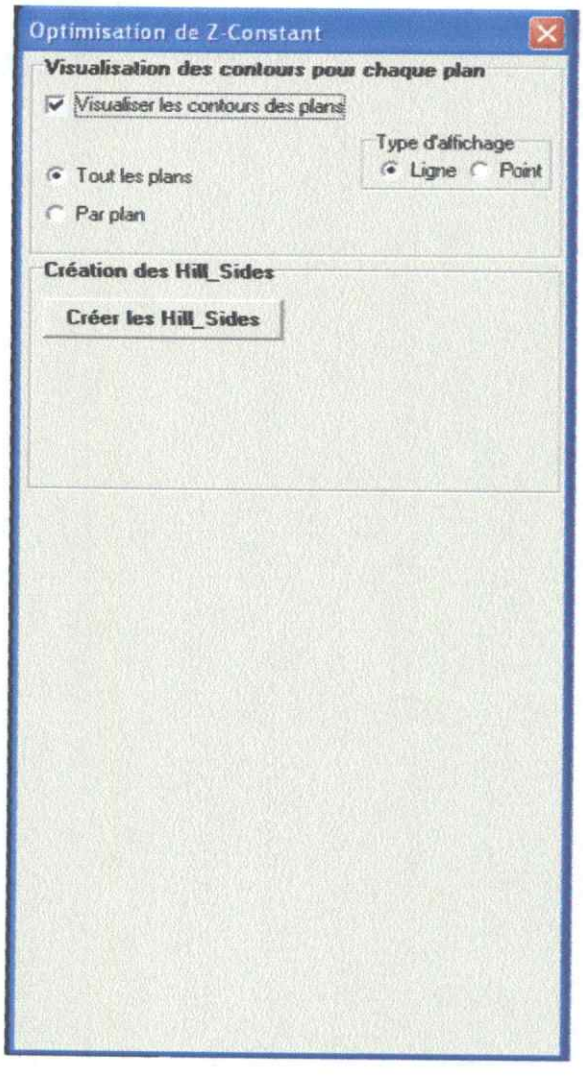
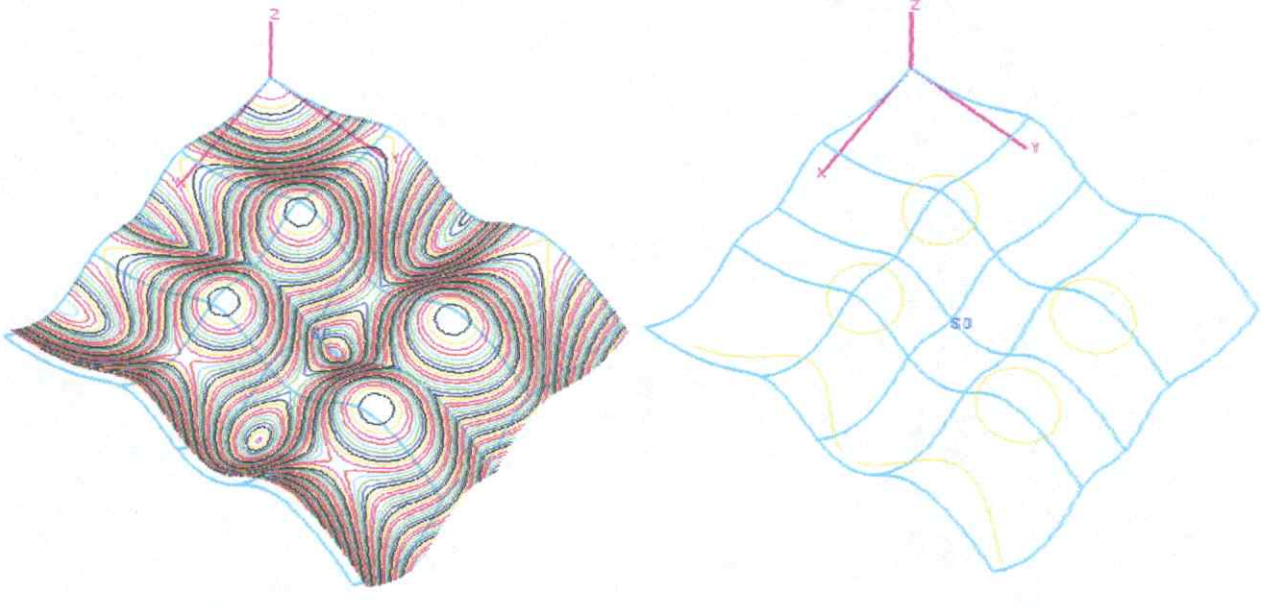


Figure 16. Ouverture de la surface considérée.



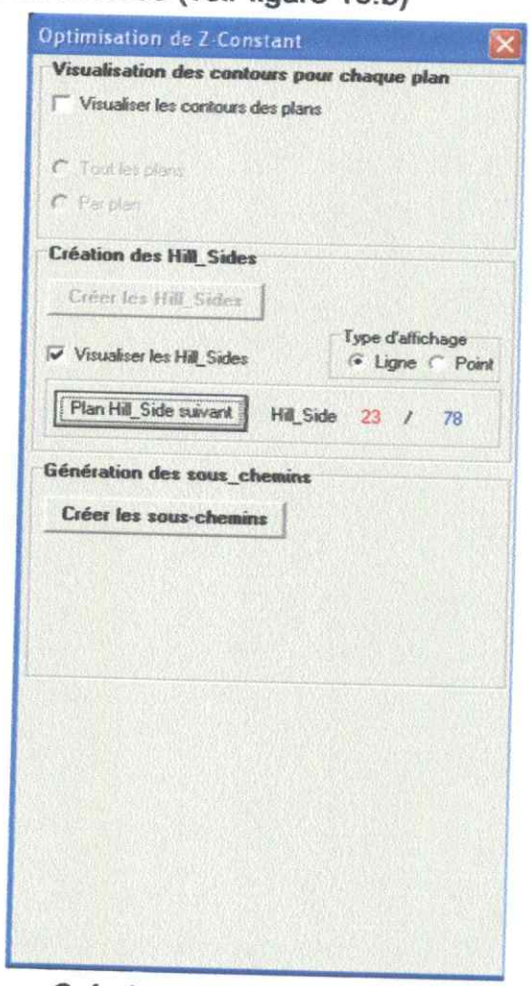
a. Création des contours des plans.



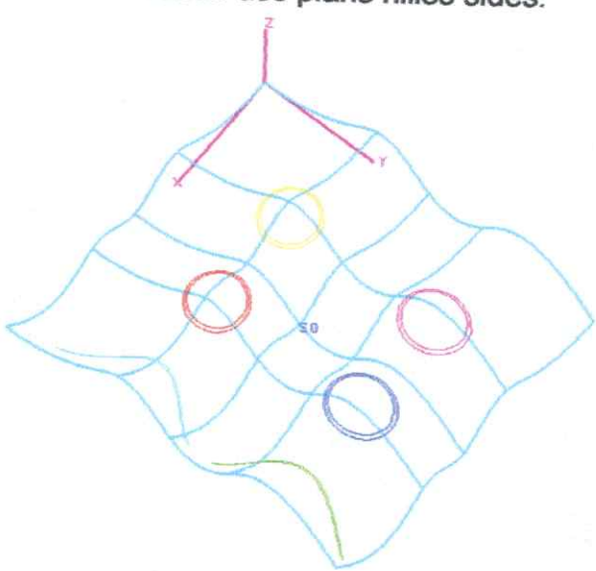
b. Tous les plans. c. Un seul plan.

Figure 17. Contours des plans de la surface.

Après récupération des contours des plans de la surface, nous allons maintenant tester la construction des hilles sides. L'utilisateur lance le calcul des plans des hilles sides (voir figure 18.a). Une fois le calcul est terminé, nous pouvons trier les hilles sides par plan pour faciliter le calcul des sous chemins. L'utilisateur a la possibilité de visualiser les plans des hilles sides (voir figure 18.b)



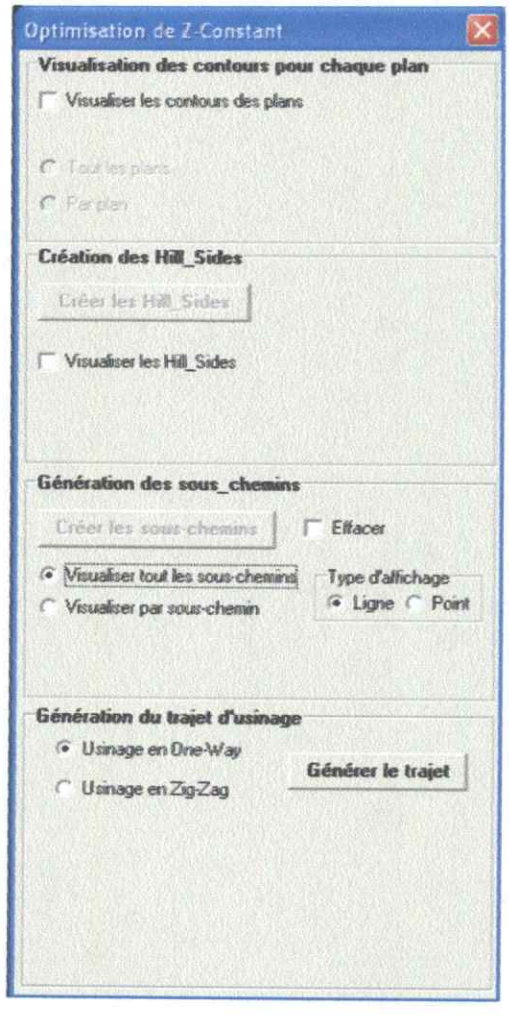
a. Création des plans hilles sides.



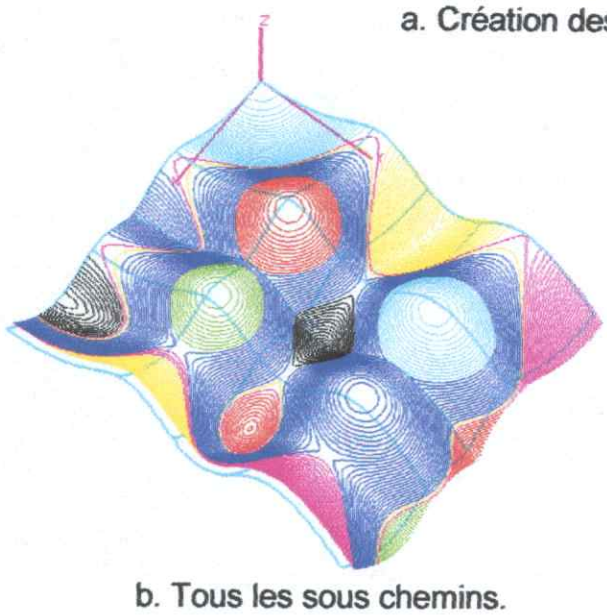
b. Plans hilles sides.

Figure 18. Hilles sides de la surface.

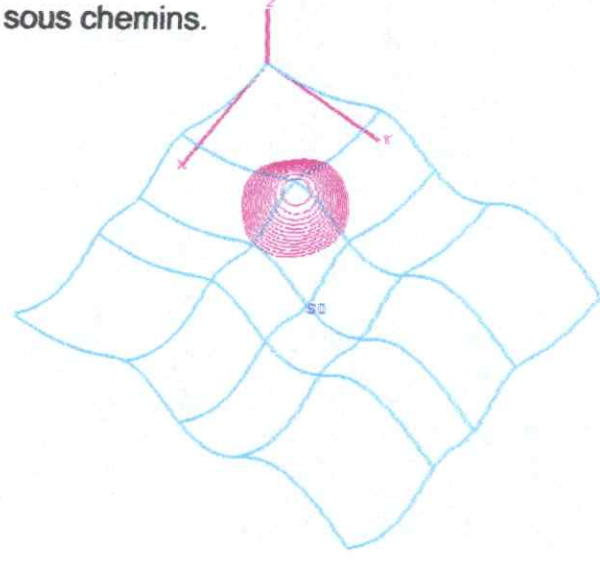
Après la construction des plans hille sides, nous allons passer au calcul des sous chemin (voir figure 19.a). Le calcul est fait en deux étapes, l'étape de calcul des sous chemins et l'étape d'optimisation des sous chemins calculés. Les sous chemins peuvent être visualisés de deux méthodes, soit tous les sous chemins (voir figure 19.b) ou sous chemin par sous chemin (voir figure 19.c)



a. Création des sous chemins.



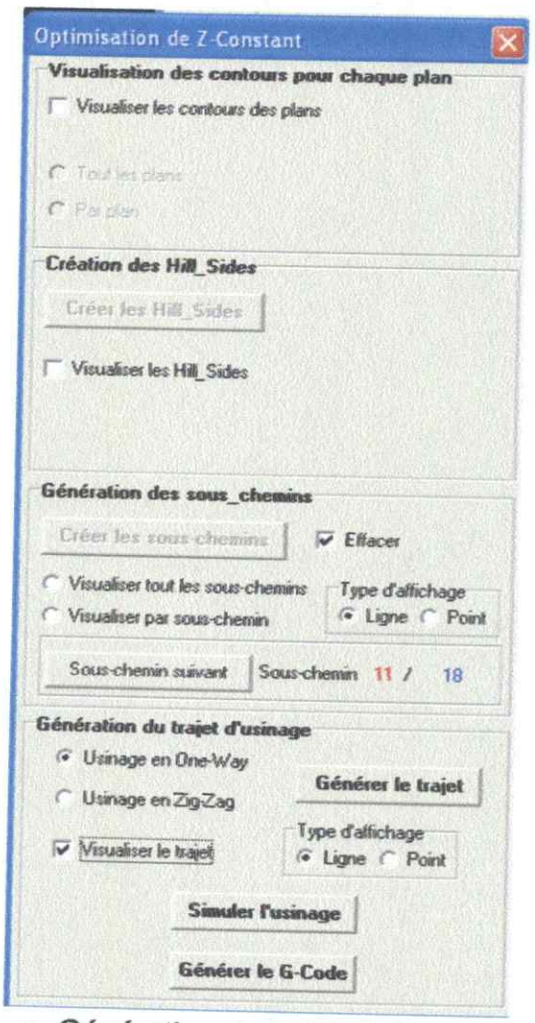
b. Tous les sous chemins.



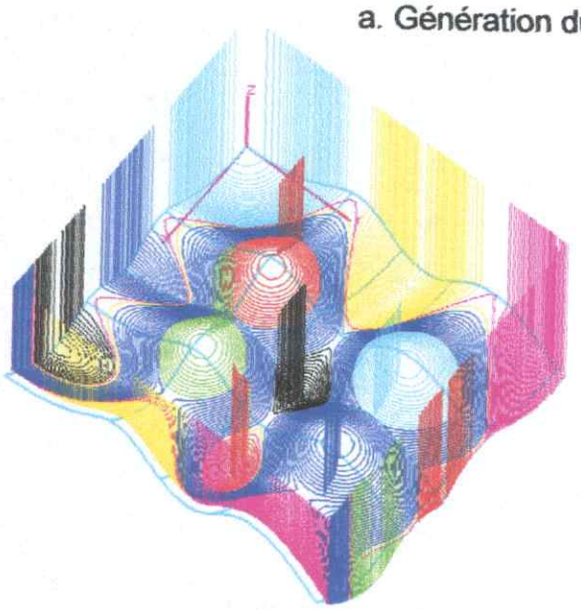
c. Un seul sous chemin.

Figure 19. Sous chemins de la surface.

Après le calcul des sous chemins, nous pouvons passer à la génération du trajet d'usinage (voir figure 20.a). Ce trajet peut être généré soit par la méthode One-Way (voir figure 20.b) soit par la méthode Zig-Zag (voir figure 20.c).



a. Génération du trajet d'usinage.



b. Trajet en One-Way.

c. Trajet en Zig-Zag.

Figure 20. Génération du trajet d'usinage.

Après la génération du trajet d'usinage, nous allons passer à l'étape de simulation du mouvement de l'outil. Après le lancement de la simulation, la fenêtre de simulation apparaît et l'utilisateur choisit le mode de la simulation (automatique ou manuel) et lance la simulation (voir figure 21). Il est possible de visualiser le trajet d'usinage par ligne, par séquence ou par plan. La figure 22 représente la simulation en temps réel.

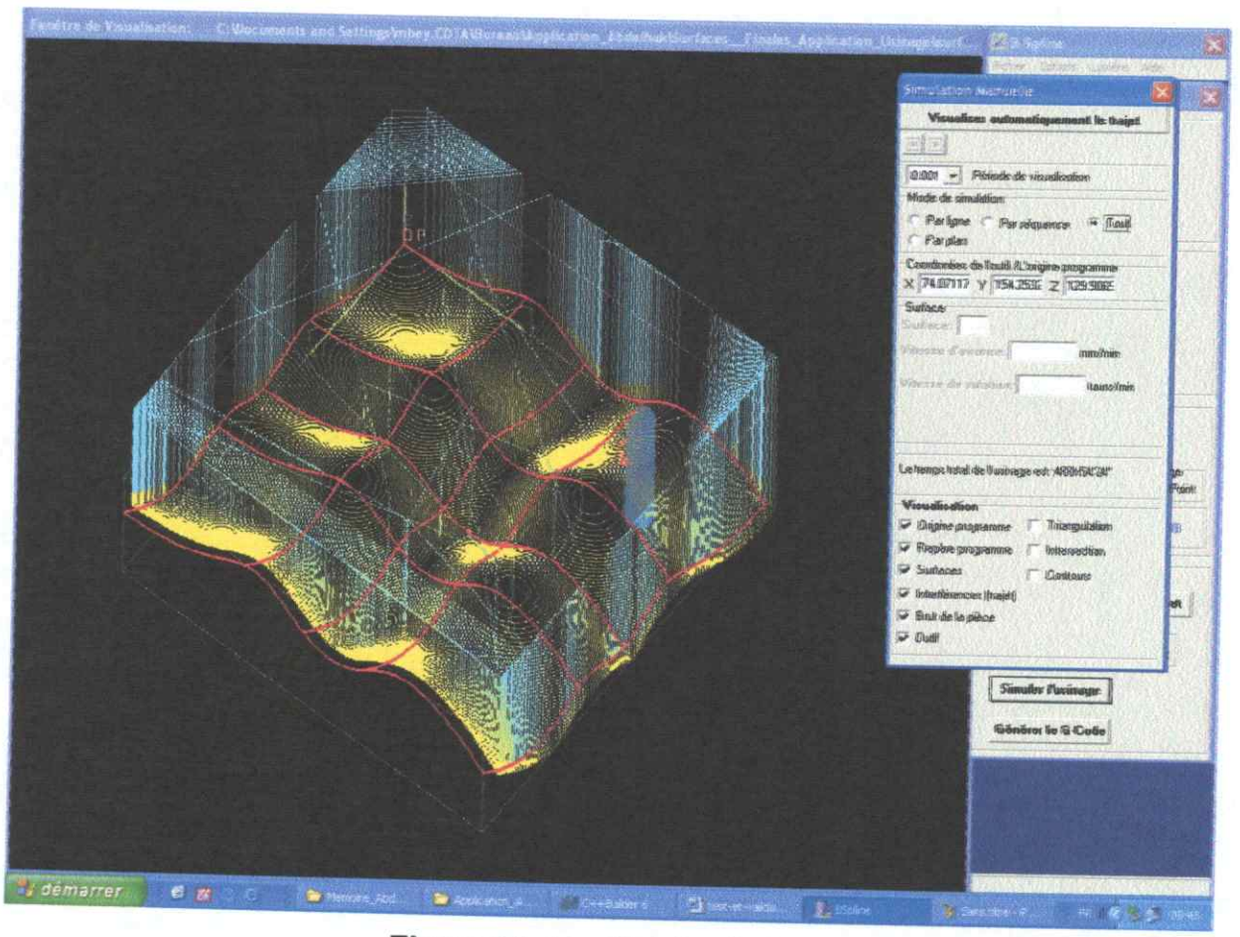


Figure 21. Fenêtre de simulation.

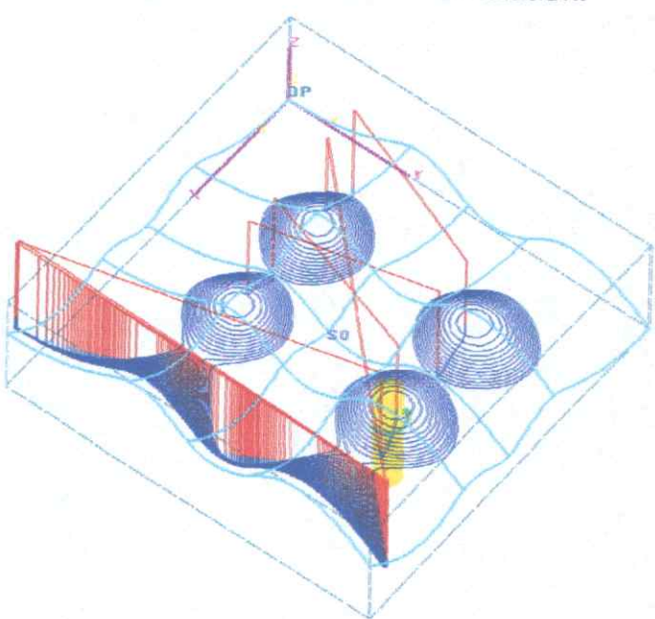


Figure 22. Simulation en temps réel.

Après la génération du trajet d'usinage, nous pouvons maintenant passer à la génération du programme d'usinage « G-Code » (voir figure 23) et qui peut être enregistré et transmis à la machine pour faire l'usinage effectif de la surface.

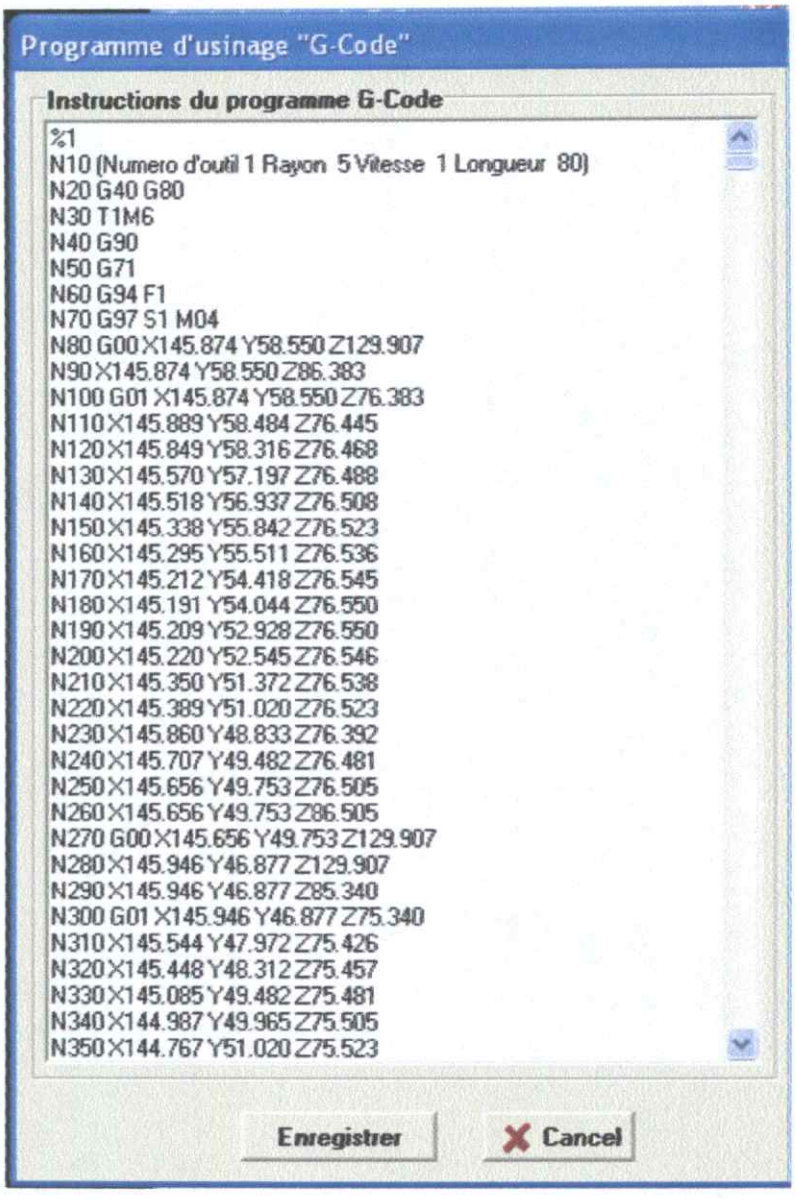


Figure 23. Génération du programme « G-Code ».

III. CONCLUSION :

Dans ce dernier chapitre, nous avons présenté les différentes fenêtres de notre application et nous avons testé toutes les fonctionnalités de notre système sur des surfaces différentes afin de valider les résultats obtenus.

CONCLUSION GENERALE

Le travail que nous avons présenté dans ce mémoire consiste en une conception et développement d'une application d'optimisation du trajet d'usinage en finition des surfaces gauches avec la stratégie Z-Constant sur des fraiseuses à commande numérique à trois axes.

Lors de la réalisation de ce projet, nous avons mené une étude bibliographique sur l'approximation et l'interpolation des contours par des courbes B-Spline. Ensuite, nous avons présenté une étude des surfaces B-Spline et NURBS ainsi que l'architecture et la programmation des machines à commande numériques et en fin l'usinage des surfaces gauches. Après cette étude théorique, nous avons présenté l'architecture de notre système et l'implémentation de notre application logicielle. A la fin, nous avons montré les différentes fonctions offertes par l'application logicielle développée suivi des tests de validation.

Le résultat de notre application est l'enrichissement de l'application logicielle graphique avec des fonctions qui permettent de :

- Triangulation et division des surfaces en régions.
- Détecter les collisions et les interférences pour les différentes parties d'outil.
- Approximer les contours du trajet d'usinage par des courbes B-Spline.
- Optimiser le nombre de points de contrôle.
- Optimiser le chemin d'outil.
- Générer le trajet d'usinage par la stratégie One-Way.
- Générer le trajet d'usinage par la stratégie Zig-Zag.
- Générer le programme d'usinage « G-Code ».
- Simuler l'usinage des pièces.

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

- Correction automatique des collisions et des interférences.
- Usinage des surfaces gauches sur des fraiseuses à cinq axes.
- Reconstruction et usinage des surfaces à partir de la mesure des pièces réelles.

ANNEXE A

OMT « OBJECT MODELING TECHNIQUE »

I. Méthode de conception OMT

La méthode OMT consiste à construire un modèle du domaine d'application, et ensuite ajouter les détails d'implémentation de cette application dans la phase de conception. La méthode (OMT) se déroule selon trois phases :

II.1. Phase d'analyse :

Cette phase commence par la spécification du problème, le modèle d'analyse est une abstraction précise qui détermine le but de l'application, les objets du modèle doivent être des concepts du domaine d'application et non pas des objets informatiques tels que les structure de données, l'analyste doit travailler en collaboration avec le client pour bien comprendre le problème.

II.2. Phase conception du système :

Le concepteur du système prend des décisions sur l'architecture globale du système, le système est découpé en sous systèmes basés sur la structure d'analyse et l'architecture proposée, les différentes décisions abouties sont :la stratégie d'attaque du problème, l'allocation des ressources et le choix du protocole de communication.

II.3. Phase conception des objets :

Le concepteur des objets construit un modèle de conception basé sur le modèle d'analyse, et il contient aussi des détails d'implémentation. L'enjeu principal de cette phase est la définition des structures des données et des algorithmes nécessaire pour implémenter chaque classe.

II.4. Phase implémentation :

Les classes d'objets et leurs relations développées durant la phase conception des objets sont traduites dans cette phase en une implémentation dans un langage de programmation.

III. Notions principales de la méthode OMT :

La notion d'objet est très importante dans la modélisation OMT, l'objet est utilisé la première fois dans le langage SIMULA et SMALTALK, il a été utilisé comme moyen de représentation des connaissances dans l'intelligence artificielle

III.1. Définition d'objet :

Un objet est une entité concrète ou abstraite présente dans l'univers du discoure, tout objet est définie par un type abstrait qui est définie par :

- Une structure de donnée composée d'attribut (partie statique).

- Un ensemble de comportement (méthodes, opérations : partie dynamique).

III.2. Types d'objets :

Un objet est atomique (primitif) si ses valeurs sont définies a partir des types atomiques (char, int.,...).

Un objet est complexe si ses valeurs sont définies à partir des types structurés.

III.3. Eléments d'analyse :

Les éléments nécessaires et suffisants pour la mise en œuvre d'un processus d'analyse sont les suivants :

III.3.1. Modèle :

C'est l'ensemble des concepts et des règles qui sont pris en considération pour modéliser la conception, il doit être puissant et, il doit offrir un ensemble complet de concepts et il doit être capable de représenter les aspects dynamiques du système.

Il existe deux types de modèles :

- **Modèle statique :** c'est la vision instantanée du système (entité, association, objets..).
- **Modèle dynamique :** c'est les changements et les échanges entre éléments du système au cours du temps.

III.3.2. Langage :

Un modèle ne peut pas être dissocié d'un langage,le langage est le moyen de représenter les concepts ,soit par une représentation graphique ou par une représentation textuelle.

III.3.3. Démarches de développement :

1. un guide qui propose un ordre de prise en compte le problème.
2. l'ensemble des étapes à suivre pour modéliser le système.
3. définition globale des étapes.

Nous prenons un exemple le cycle de vie en cacade (voir figure1) qu'il suit les étapes suivantes :

1. analyse.
2. conception.
3. implémentation.
4. tests.
5. Maintenance.

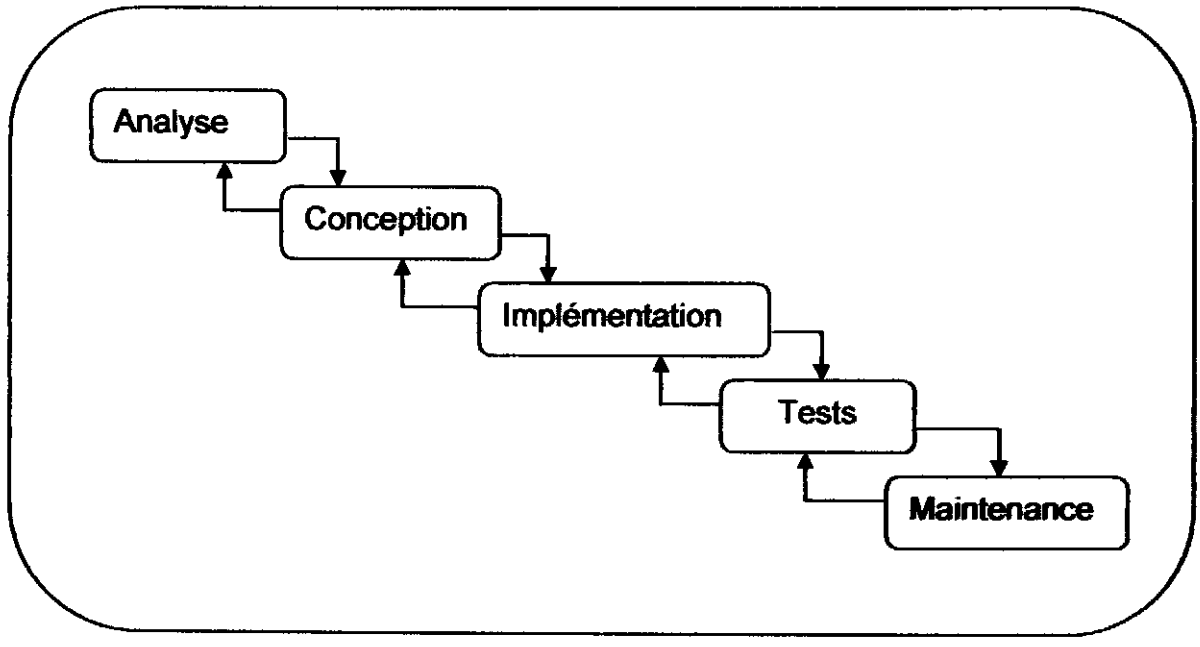


Figure A.1. Cycle de vie en cascade.

ANNEXE B

UML « UNIFIED MODELING LANGUAGE »

I. Définition :

UML (langage unifié pour la modélisation) est un langage pour la modélisation Objet, il fournit les fondements pour spécifier, construire, visualiser, et décrire tous les aspects d'un système logiciel, C'est un produit de la fusion des trois méthodes OMt, OOSE, BOOCH.

UML se concentre sur la formalisation et pas le processus de développement (aucune démarche / cycle de vie n'est pas définie par UML), il est composé d'un ensemble de notations de diagramme qu'on peut utiliser selon nos besoins.

II. Historique [25]:

A partir de 1994, Rumbaugh et Booch (rejoints en 1995 par Jacobson) ont unis leurs efforts pour mettre au point la méthode unifiée (unified method 0.8), incorporant les avantages de chacune des méthodes précédentes.

La méthode unifiée à partir de la version 1.0 devient UML (Unified Modeling Language), une notation universelle pour la modélisation objet.

UML 1.0 est soumise à l'OMG (Object Management Group) en janvier 1997, mais elle ne sera acceptée qu'en novembre 1997 dans sa version 1.1, date à partir de laquelle UML devient un standard international.

Voici le récapitulatif des évolutions de ce langage de modélisation :

- En 1995: Méthode unifiée 0.8 (intégrant les méthodes Booch'93 et OMT)
- En 1995: UML 0.9 (intégrant la méthode OOSE)
- En 1996: UML 1.0 (proposée à l'OMG)
- En 1997: UML 1.1 (standardisée par l'OMG)
- En 1998: UML 1.2
- En 1999: UML 1.3
- En 2000: UML 1.4
- En 2003: UML 1.5

. Les diagrammes d'UML sont regroupés en deux types de représentation :

II.1. Diagrammes Structurels :

C'est la représentation de la vue statique, les diagrammes d'UML qui représentent la vue statique sont les suivants :

1. Diagramme de classe.
2. Diagramme d'objet.

3. Diagramme de déploiement.
4. Diagramme des composants.

II.2. Diagrammes comportementaux :

C'est la représentation de la vue dynamique, Les diagrammes qui représentent la vue dynamique sont :

5. Diagramme d'activité.
6. diagramme de séquence.
7. diagramme d'état/ transition.
8. diagramme de cas d'utilisation.
9. diagramme de collaboration.

Le diagramme de séquence et le diagramme de collaboration sont des diagrammes d'interactions.

III. Diagrammes d'UML [2]:

III.1. Diagramme des classes :

Il exprime la vue statique en terme de classes et de relations entre ces classes, pour un système donné, il est possible d'avoir plusieurs diagrammes de classe, chacun d'eux traite un aspect du système et cela par soucis de clarté. La figureB.1 Montre la structure de la classe.

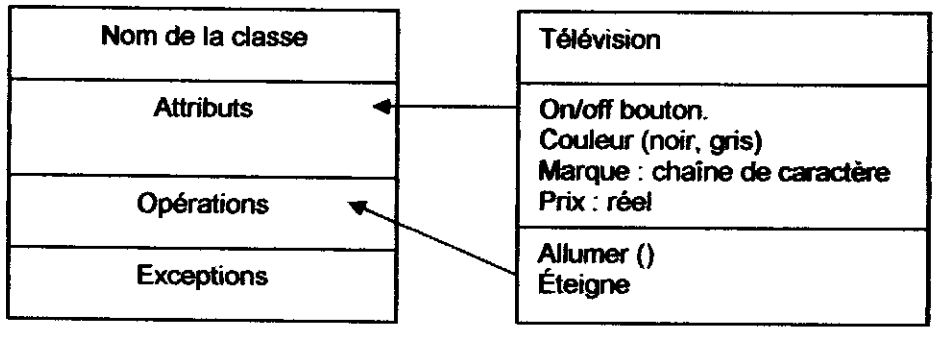


Figure B.1. Structure de la classe.

Classe :

Est une description abstraite d'un ensemble d'objets ayant des propriétés, des comportements et des relations similaires

- **Attribut :**

Les attributs peuvent être :
Des classes.
Des types prédéfinies.
Simples.
Structurés.
Le changement des valeurs des attributs peut être exprimé en fonction des valeurs prédéfinies :

Gelé : Non modifiable.
Variable : modifiable.
Ajout uniquement.

- **Attribut dérivé :**

Est attribut qui peut être déduit (calculé) à partir d'un autre attribut, il est précédé par */*. (Voir figure B.2).

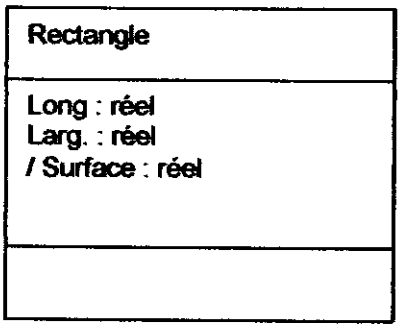


Figure B.2. Attribut calculé.

- **Opération :**

L'opération est un service q'une instance, une opération définit une fonction appliquée à des objets d'une classe.

Une méthode est l'implémentation d'une opération, la méthode accepte des arguments, des autres opérations et le type retourné.

Les arguments d'une opération sont par défaut en entrée (IN) sinon il faut préciser s'ils sont en sortie (out) ou bien en entrée / sortie (IN/OUT).

- **Association :**

Regroupe une relation structurelle qui relie deux classes d'objet et plus, la plupart des associations sont binaire chacune est représenté par un trait qui relie les deux classes (voir figure B.3).

- **Multiplicité d'association :**

  - **1** : un et un seul.
  - **0..1** : zéro ou un.
  - **N** : entier naturel.
  - **N..M** : de N à M entier.
  - ***** : zéro à plusieurs.
  - **0..*** : zéro à plusieurs.
  - **1..*** : un à plusieurs.

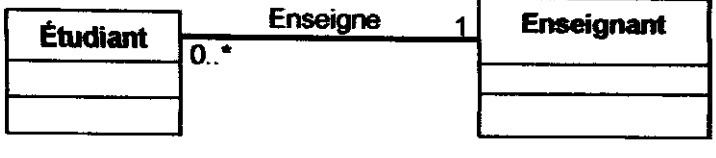


Figure B.3 Associations entre classes.

- **Agrégats :**

Un agrégat représente une association non symétrique dans laquelle une extrémité joue un rôle prédominant par rapport à l'autre extrémité (voir figure B.4).

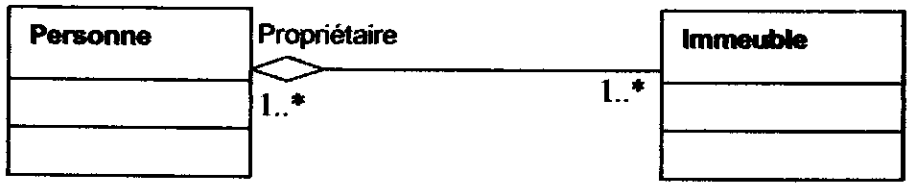


Figure B.4 Agrégation.

- **Composition :**

C'est un cas particulier d'agrégat, les attributs sont sémantiquement équivalents, et leurs représentations sont interchangeables.

- **Généralisation :**

Désigne un point de vue porté sur un arbre de classification, la généralisation est de deux types :

1. **généralisation simple** : les sous classe héritent seulement d'une superclasse
2. **généralisation multiple :** les sous classes héritent de plusieurs superclasses.

III.2. Diagrammes de cas d'utilisation :

Le diagramme de cas d'utilisation décrit la succession des opérations réalisées par un acteur (personne qui assure l'exécution d'une activité). C'est le diagramme principal du modèle UML, celui où s'assure la relation entre l'utilisateur et les objets que le système met en œuvre (voir figure B.5)
Le diagramme cas d'utilisation est caractérisé par :

1. **Cas d'utilisation :** Un cas d'utilisation correspond à une manière spécifique d'utiliser le système, c'est la représentation d'une fonctionnalité, déclenchée en réponse à une stimulation du système

2. **Acteur :** Est un entité externe qui peut agit sur le système, peut être un utilisateur ou un autre système.

Les relations entre éléments du diagramme peuvent être de trois types :

- La relation d'utilisation : Une relation d'utilisation permet de décomposer un cas d'utilisation en sous cas d'utilisation.
- La relation d'inclusion : le cas d'utilisation source comprend également le comportement de son cas d'utilisation destination.
- La relation d'extension : indique que le cas d'utilisation source étend (précise) les objectifs (le comportement) de cas d'utilisation destination.

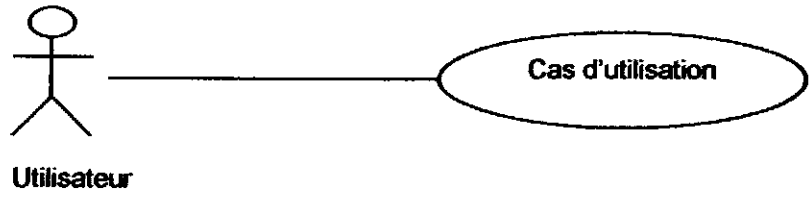


Figure B.5. Cas d'utilisation.

III.3. Diagramme de séquence :

Le diagramme de séquence représente la succession chronologique des opérations réalisées par un acteur, il est sous forme d'un scénario saisir une donnée, consulter une donnée, lancer un traitement ; il indique les objets que l'acteur va manipuler et les opérations appliquées sur ces objets (voir figureB.6)

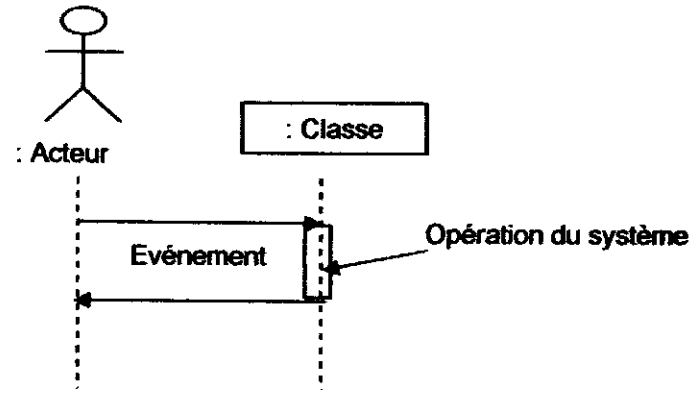
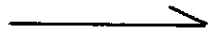
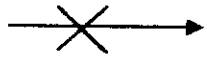
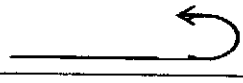
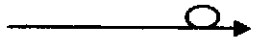


Figure B.6. Diagramme de séquence.

Les types de message inter changé entre acteur sont récapitulés dans le tableau suivant :

Type de message	Signification
	Message simple.
	Message asynchrone, pas de réponse attendue par l'émetteur.
	Message synchrone, réponse nécessaire du destinataire.
	N'interrompt pas l'exécution de l'expéditeur.
 Délai	Bloque l'expéditeur pendant un temps donné, en attendant la prise en compte du message par le récepteur.

III.4. Diagramme d'activité :

UML permet de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation, à l'aide de diagrammes d'activités (une variante des diagrammes d'états transitions).

Une activité représente un déroulement séquentiel des étapes. Le passage d'une activité vers une autre est matérialisé par une transition (voir figure B.7).

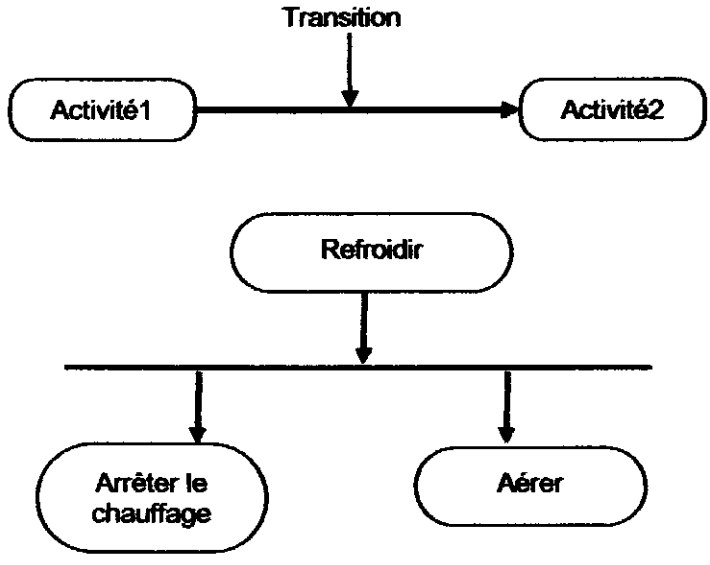


Figure B.7. Diagramme d'activité.

III.5. Diagramme de collaboration :

Est une extension des diagrammes d'objets, il montre le comportement collectif d'un ensemble d'objets en vue de réaliser une opération par la description des interactions modélisées par des envois de messages (voir figure B.8)

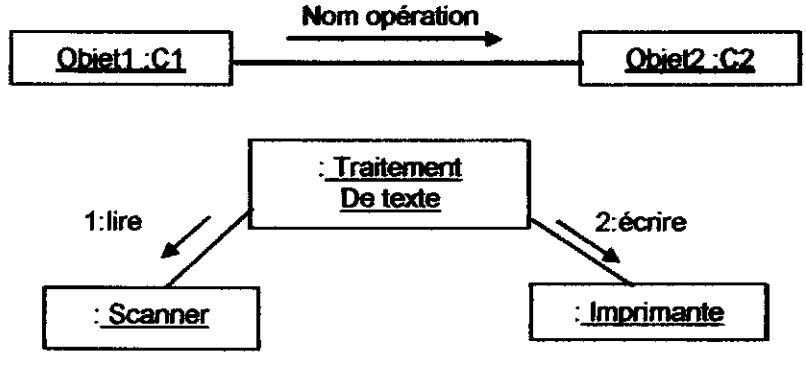
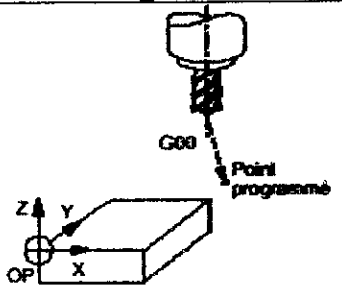
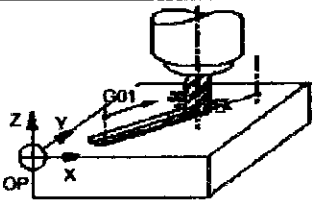
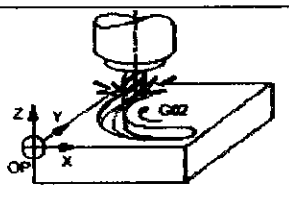
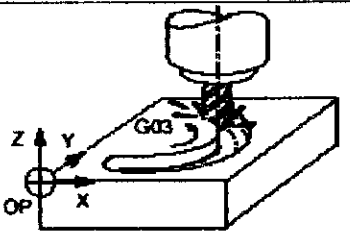
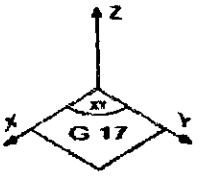
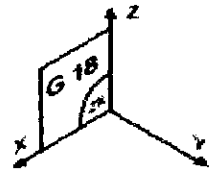
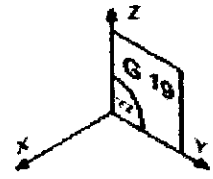
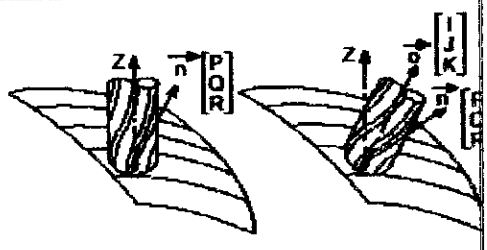
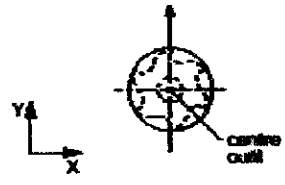
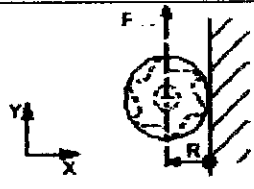
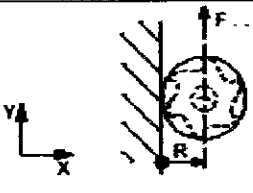
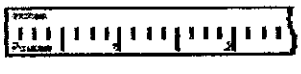


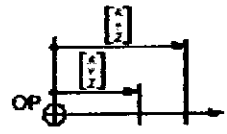
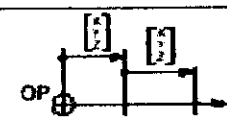
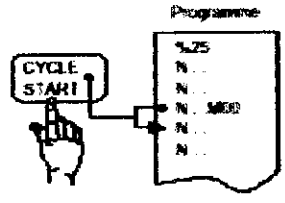
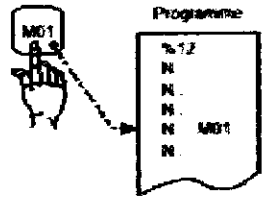
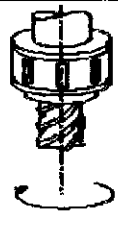
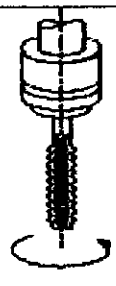
Figure B.8. Diagramme de collaboration.

ANNEXE C

PRINCIPALES FONCTIONS DU G-CODE

Fonctions	Figures
G00 : Interpolation linéaire à vitesse rapide. **Fonction modale** **Syntaxe :** N.. [G90/G91] **G00** [□±] X.. Y.. Z..	 *Figure C.1 : La fonction G00.*
G01 : Interpolation linéaire à vitesse d'avance. Programmée **Fonction modale** **Syntaxe :** N.. [G90/G91] **G01** [□±] X.. Y.. Z.. [F..]	 *Figure C.2 : La fonctionG01.*
G02 : Interpolation circulaire sens antitrigonométrique à vitesse d'avance programmée. **Fonction modale** **Syntaxe** (plan XY) : N.. [G17] [G90/G91] **G02** X.. Y..I.. J.. ou R.. [F..]	 *Figure C.3 : La fonction G02.*
G03 : Interpolation circulaire sens trigonométrique à vitesse d'avance programmée. **Fonction modale** **Syntaxe** (plan XY) : N.. [G17] [G90/G91] **G03** X.. Y.. I..J.. ou R.. [F..]	 *Figure C.4 : La fonction G03.*
G04 : Temporisation programmable. **Fonction modale** **Syntaxe :** N.. **G04** F..	 *Figure C.5 : La fonction G04.*
G17* : Choix du plan XY **Fonction modale** **Syntaxe :** N.. **G17**	 *Figure C.6. La fonction G17.*

G18 : Choix du plan ZX **Fonction modale** **Syntaxe :** **N.. G18**	 *Figure C.7 : La fonction G18.*
G19 : Choix du plan YZ **Fonction modale** **Syntaxe :** **N.. G19**	 *Figure C.8 : La fonction G19.*
G29 : Correction d'outil dans l'espace 3 ou 5 axes. **Fonction modale** **Syntaxe :** **N.. [D..] [G01] G29 X.. Y.. Z.. P.. Q.. R.. [I.. J.. K..] [A.. / B.. / C..]**	 *Figure C.9 : La fonction G29.*
G40 : Annulation de correction de rayon **Fonction modale** **Syntaxe :** **N.. [G00/G01] G40 X.. Y.. Z..**	 *Figure C.10 : La fonction G40.*
G41 : Correction de rayon à gauche du profil à usiner **Fonction modale** **Syntaxe (plan XY) :** **N.. [G17] [D..] [G00/G01/G02/G03] G41 X.. Y..**	 *Figure C.11 : La fonction G41.*
G42 : Correction de rayon à droite du profil à usiner **Fonction modale** **Syntaxe (plan XY) :** **N.. [G17] [D..] [G00/G01/G02/G03] G42 X.. Y..**	 *Figure C.12 : La fonction G42.*
G70 : Programmation en pouce **Fonction modale** **Syntaxe :** **N.. G70**	 *Figure C.13 : La fonction G70.*
G71 : Programmation en métrique **Syntaxe :** **N.. G71**	 *Figure C.14 : La fonction G71.*

G90 : Programmation absolue par rapport à l'origine programme **Fonction modale** **Syntaxe :** N.. **G90** X.. Y.. Z.. A.. B.. C..	 *Figure C.15 : La fonction G90.*
G91 : Programmation relative par rapport au point de départ du bloc **Fonction modale** **Syntaxe :** N.. **G91** X.. Y.. Z.. A.. B.. C..	 *Figure C.16 :La fonction G91.*
M00 : Arrêt programmé . **Fonction non modale** **Syntaxe :** N.. [G40] **M00** [$0 ...]	 *Figure C.17 : La fonction M01.*
M01 : Arrêt programmé optionnel. **Fonction modale** **Syntaxe :** N.. [G40] **M01** [$0 ...]	 *Figure C.18 : La fonction M02.*
M02 : Fin de programme. **Fonction modale** **Syntaxe :** N.. **M02**	
M03 : Rotation de broche sens antitrigonométrique **Fonction modale** **Syntaxe :** N.. **M03**	 *Figure C.19 : La fonction M03.*
M04 : Rotation de broche sens trigonométrique **Fonction modale** **Syntaxe :** N.. **M04**	 *Figure C.20 : La fonction M04.*
M05 : Arrêt de broche **Fonction modale** **Syntaxe :** N.. **M05**	

M08 : Arrosage numéro 1. **Fonction modale avant** **Syntaxe** : N.. **M08**	 *Figure C.21 : La fonctionM08.*
M09 : Arrêt des arrosages 1 et 2 **Fonction non modale après** **Syntaxe** : N.. **M09**	 *Figure C.22 :.La fonction M09.*
F : Avance, temporisation, nombre de filets **Syntaxe** : N.. G93 **F..** (Avance en V/L). N.. G94 **F..** (Avance en mm/min, degrés/min, pouce/min) N.. G95 **F..** (Avance en mm/t, pouce/tour) N.. G04 **F..** (Temporisation en secondes) N.. G31 **F..** (Nombre de filets)	
S : Nombre de tours/minute, nombre de répétitions de sous programme **Syntaxe**: N.. G97 **S..** (Vitesse de broche en tours/min. N.. G77 [H..] [N.. N..] **S..** (Appel et répétitions de sous programme.	
T : Numéro d'outil **Syntaxe** : N.. **T..** M06 (Appel de l'outil)	

LISTE DES FIGURES

CHAPITRE 1: INTERPOLATION ET APPROXIMATION PAR LES COURBES B-SPLINE

CHAPITRE 2: SURFACES B-SPLINE ET NURBS

CHAPITRE 3: ARCHITECTURE ET PROGRAMMATION DES FRAISEUSES A COMMANDE NUMERIQUES

CHAPITRE 4: USINAGE DES SURFACES GAUCHES

CHAPITRE 5: MODELISATION ET CONCEPTION

CHAPITRE 6: IMPLEMENTATION

CHAPITRE 7: TESTS ET VALIDATIONS

ANNEXES

Annexe A :

Annexe B :

Annexe C :

REFERENCES BIBLIOGRAPHIQUES

[1]. Brian. A. Barsky ET Tony. D. DeRose, *"Geometric Continuity of Parametric Curves: Three Equivalent Characterizations"*, IEEE Computer Graphics & Applications, Vol. 9 (1989), No. 6 (November), pp. 60-68.

[2]. M. Bey, *"Modélisation des courbes et des surfaces"* Rapport De Recherche CDTA Avril 2000.

[3]. Jean Claude Léon, *"Modélisation et construction de surfaces pour la CFAO"*, Edition Hermès, 1991.

[4]. Gerald Farin, *"Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide"*, fourth edition, Academic Press, 1990.

[5]. Gerald Farin, *"Introduction to computing with geometric notes"*, fourth edition, Academic Press, 1997.

[6]. C. K. shene, "Curves *and Surfaces for CAGD: a Practical Guide"*, Department of Computer Science, Michigan Technological University, 1997-2003.

[7]. Les Piegel et Wayne Tiller, *"The NURBS Book"*, second edition, Springer-Verlag, 1997.

[8]. Les Piegel, *"On NURBS A survey"*, University of Florida.

[9]. Bernard Méry, *"Machine à commande numérique"*, Edition Hermès, 1997.

[10]. Varady (T), Martin (R.C), Cox (J) *"Reverse enginerring of geometric models – an introduction Computer Aided Design -"*;.1997, vol. 29, no. 4, p. 255-268

[11]. Rolland Maranzana, *"Fabrication assisté par ordinateur"*, G.PA. Génie de la production automatisée.

[12]. BAGARD (P) -" *Les atouts du fraisage à grande vitesse* "- CETIM Information – Décembre 1993, no. 136, p. 47-50

[13]. E. Duc, Lefur, *"Machine outil à commande numérique Structure, Modélisation, et réglage"*, Mémoire de DEA à l'école normale supérieure de Cachan, France, septembre 1997.

[14]. *"NUM 1020/1040/1060 M; Manuel de programmation -volume1 "*. Société NUM 1996.

[15]. E. Duc, " *Usinage de formes gauches; contribution à l'amélioration de la qualité des trajectoires d'usinage*", Thèse de doctorat à l'école normale supérieure de Cachan, France 1998.

[16]. E. Duc, "*Introduction au fraisage*". ; L'école normale supérieure de Cachan, France 1998.

[17]. D.C. Anderson C.G Jensen, "A *review of numerically controlled methods for finish sculptured surface machining*".; School of Mechanical Engineering, Brigham Young University.

[18]. M. Bey, "*automatisation de la sélection de la stratégie et de la direction d'usinage pour la finition des surfaces gauches*", Rapport de Recherche, CDTA, novembre 2004.

[19]. Rezzak Samia, Taibi Hayette, "*Méthode Z-Constant pour l'Usinage des Surfaces Gauches sur des Fraiseuses à Commande Numérique à 3 axes*", Mémoire de Fin d'Etudes, Université de Blida, Algérie, 2003.

[20].Y.S.Lee, B.K.Choi, T.C. Chang, "*Cut distribution and cutter selection for sculptured surface cavity machining*", Cubic Technology Center, Ace Techno-Tower 1101, Seoul, South Korea .2001.

[21]. Frédiric Julliard, "*UML Unified Method Language*", Universite de Bretagne Sud, UFR SSIIUP, Vannes 200-2001.

[22]. James Rumbaugh ; "*Modélisation et conception orientées objet.* "; Edition Prentice Hall 1997.

[23]. V. Sundararajan ET Paul K. Wright, "*Zig-Zag Tool Path Generation for Sculptured Surface Finishing", Department* of Mechanical Engineering, University of California, Berkeley, CA, USA.

[24]. S. C. Park, "*Tool path generation for Z-constant contour machining*", Computer Aided Design, Volume 35, pages 27-36, 2003.

[25]. Joseph. Schmuller; "*Sams Teach Yourself UML in 24 Hours*"; Third Edition; Sams Publishing; Pub Date: March 15, 2004