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

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.
Option : Système d'information

Sujet :

Méthode de Z Constant pour l'Usinage des Surfaces Gauches sur des Fraiseuses à Commande Numérique à 3 axes

Présenté par : Rezzak Samia
Taibi Hayette

Promoteur : Mr Bey Mohamed
Attaché de Recherche

Organisme d'accueil : Centre de Développement des Technologies Avancées **C.D.T.A**

Soutenue le: date soutenance, devant le jury composé de :

Président

Examinateur

Examinateur

- Septembre 2005-

REMERCIEMENTS

Nous tenons à adresser au Directeur Général du Centre de Développement de Technologies avancées (CDTA) nos sincères remerciements pour nous avoir accueilli dans son laboratoire et permis de réaliser ce modeste travail.

Nous avons eu un réel plaisir à travailler avec notre promoteur Mr. M. BEY qui a su rester à notre écoute dans les moments où nous avions besoin de lui.

Nous tenons à remercier tous les professeurs de l'USTB qui nous ont accompagnés dans nos études supérieurs, ainsi qu'à tous nos enseignants et enseignantes d'étude primaire, moyenne et lycéenne, et à tout ceux qui nous aiment.

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 :

La mémoire de mon père

Ma très chère mère

Mon oncle Hacen pour son aide et soutien et à toute sa famille

Mes frères Ahmed, Mohamed, et ma sœur Houria

A tout ma famille.

Ma sœur, amie et binôme Hayette et sa famille, son fiancé Youcef,

Mes meilleures amies Sarah, Fella, Rafik

Toutes mes amies et collèges sans restriction

Samia

Je dédie ce modeste mémoire à :

Mon cher père.

Ma très chère mère.

A mes grands parents, mes frères Abd allah, Adb elhafid et Abd enour et à toute ma grande famille.

A ma deuxième famille la famille Sebaa.

A tout les gens qui m'ont aimé ou respecté surtout Sabrina et ses deux filles, Fella, Sarah, Rafik, les sœurs Fedrouche, mes cousines, mes amis d'enfance et d'étude.

A mon amie et Binôme Samia.

Hayette

SOMMAIRE

Chapitre 6: *Conception et Implémentation*

Chapitre 7: *Tests et Validation*

Liste de figures

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 nous nous intéressons à l'usinage des surfaces de formes libres sur des fraiseuses à commande numérique à 3 axes. Le but de ce travail est le développement d'une application logicielle graphique et interactive sous Windows qui permet l'usinage des surfaces gauches en utilisant la méthode Z Constant, l'optimisation de la trajectoire d'outils, l'automatisation de choix de la combinaison optimale des outils qui permettent d'usiner ces surfaces en un minimum de temps ainsi que l'automatisation de la génération des programmes d'usinage «G-Code» à partir des modèles CAO des surfaces à usiner et des différents paramètres.

Abstracts

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 Center of Development Technology Advanced (CDTA).

In this project we are interested in the machining of the free shape surfaces on tools-machine numeric order to 3 axes. The goal of this work is the development of a graphic and interactive software application under Windows that permits the left surface machining while using the method Z Constant, the optimization of the tool trajectory, the automation of choice of the optimal combination of tools that permits to mill these surfaces in a minimum of time as well as the automation of the generation of G - Code " machining " programs from the CAO models of surfaces to mill and of the different parameters

ملخص

الهدف من هذا العمل يتمثل في تطوير أدوات التصميم و الإنتاج للمساحات اليسارية , والتي بدأت بفريق التصميم و الإنتاج بجهاز الأعلام الآلي(CFAO), وذلك على مستوى الجهوية الآلية و الإنتاجية لمركز تطوير التكنولوجيات المتقدمة(CDTA).

في هذا المشروع نحن نهتم بصناعة المساحات ذات الشكل الحر بآلات ذات لوحات تحكم رقمية بثلاثة محاور. الهدف من هذا العمل هو تطوير برنامج قابل للاستعمال تحت نظام (windows) الذي يسمح بتصنيع المساحات اليسارية باستعمال طريقة Z ثابت على آلات التقطيع ذات لوحات التحكم رقمية بثلاثة محاور لإنشاء الآلي لملفات الصناعة G-Code

INTRODUCTION GENERALE

Devant les progrès accomplis, aussi bien en terme de matériels que de logiciels informatiques, mais aussi sous la pression du marché qui demande des produits de plus en plus complexes avec des temps de développement raccourcis, la tentation est forte d'insérer toujours plus de numérique dans la conception. Cette tendance, observée un peu partout aujourd'hui.

La CFAO ou Conception et Fabrication Assistées par Ordinateur est la synthèse de la CAO et de la FAO apparue dans les années 1970 avec l'introduction des machines-outils à commande numérique. L'idée est qu'un système de CAO dispose précisément de toutes les informations nécessaires pour créer le programme d'une machine-outil à commande numérique et que dans ces conditions traiter les deux questions séparément représenterait une perte de temps et d'argent, sans compter les risques d'erreur de transcription. De nos jours, la CFAO est devenue un outil incontournable dans la conception et la fabrication des produits. Elle est utilisée dans des domaines aussi variés que l'aéronautique, l'automobile, les produits liés au sport, ...etc.

Les pièces de formes gauches (moules, formes aérodynamiques, carrosseries de voitures, formes esthétiques,...etc.) sont devenues, par l'évolution du style et des techniques d'usinage, des pièces courantes de notre vie quotidienne. Comme toute pièce utilisée en mécanique, les pièces de formes gauches sont conçues dans le but d'assurer des fonctions inscrites dans le cahier des charges, par conséquent, ces pièces doivent répondre à des exigences fonctionnelles et/ou de style. Avec l'évolution des machines outils à commande numérique (MOCN), le niveau de qualité de ces pièces a augmenté d'une façon considérable, ce qui impose une attention particulière dans leur mise en production.

Le travail que nous présentons dans ce mémoire s'inscrit dans le cadre de développement d'outils de conception et de fabrication des surfaces irrégulières (surfaces gauches) initié par l'équipe Conception et Fabrication Assistées par Ordinateur (CFAO) au niveau 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-Splines et NURBS;
- La reconstruction des courbes et des surfaces B-Splines 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.
- Simulation de l'opération d'usinage (enlèvement de matière) des surfaces gauches et vérification des programmes d'usinage.

Vu la complexité de la géométrie des surfaces gauches, il est impossible de générer manuellement le chemin d'outil et le programme d'usinage correspondant. D'où vient la nécessité d'utiliser l'outil informatique pour le développement des applications permettant d'automatiser ces opérations avec la possibilité de faire un choix automatique des outils d'usinage pour les surfaces à usiner.

Dans ce mémoire, on s'intéressera à l'usinage en finition des surfaces de formes libres sur des fraiseuses à commande numérique à 3 axes en considérant la méthode d'usinage Z-Constant « plans parallèles horizontaux » ainsi que le choix automatique de la combinaison optimale des outils d'usinage, à partir d'un ensemble d'outils stockés dans une base de donnée, permettant de réduire les temps d'usinage et par conséquent les coûts de fabrication.

L'objectif de notre travail est le développement d'une application logicielle graphique et interactive sous Windows, qui permet de générer la succession des points de positionnement de l'outil par rapport aux surfaces à usiner (chemin d'outil), et à partir de ces points générer automatiquement les fichiers G-Code (langage propre de programmation des Machine Outils à Commande Numérique "*MOCN*"). Ainsi que le choix automatique des outils optimums. Ce travail permettra d'améliorer la qualité d'usinage et de réduire les temps d'écriture des programmes.

Le présent mémoire est organisé comme suit :

- Dans le premier chapitre nous allons décrire les méthodes de conception et de représentation des surfaces ;
- Dans le second chapitre nous présentons l'architecture et la programmation des fraiseuses à commande numérique ;
- Dans le chapitre trois nous décrivons l'usinage des surfaces gauches ;
- L'usinage des surfaces gauches par la méthode Z-Constant est présenté dans le chapitre quatre ;
- Dans le cinquième chapitre : définition des besoins et analyse ;
- Dans le sixième chapitre : la conception et l'implémentation ;
- Dans le dernier chapitre : test et validation.

Enfin, nous terminons ce mémoire par une conclusion générale et quelques recommandations pour la continuité de ce travail.

Chapitre1

Méthodes de Représentation et de Conception des Surfaces Gauches

I. INTRODUCTION :

La conception des surfaces en mécanique est devenue une spécialité très importante dans le domaine de la conception assistée par ordinateur *(CAO).* Cette spécialité est interdisciplinaire nécessitant des bases en informatique, des concepts mécaniques très poussés et de l'ingénierie. Après l'introduction des ordinateurs dans le procédé de conception (CAO) et le développement des machines à commande numérique qui exige la haute qualité des pièces pour assurer une bonne fonctionnalité de ces machines et pour éviter les problèmes techniques, il y a une nécessite d'avoir des méthodes de représentation et d'aide à la conception.

Pour modéliser et représenter les courbes et les surfaces de formes libres rentrant dans la conception des moules, des matrices, des formes aérodynamiques et des formes esthétiques, plusieurs méthodes ont été développé (Bézier, B-Spline et NURBS). Les surfaces B-Spline et NURBS n'existent pas dans le monde du dessin conventionnel, mais elles sont utilisées dans la modélisation 3D sur ordinateur. Elles sont à présent des standards dans le domaine de la conception et de la modélisation des surfaces. Dans ce chapitre nous allons présenter les différentes méthodes de conception et de représentation des surfaces de formes libres utilisées par les ingénieurs mécaniciens.

II. METHODES DE REPRESENTATION DES SURFACES [1,2] :

Une bonne représentation devrait être :

- Unique, chaque objet a une seule représentation (une surface peut être représenter avec une seule représentation).
- Une représentation devrait pouvoir représenter un ensemble utile d'objet.
- Non ambiguë (complète et sans doute).
- précise si aucune approximation n'est exigée (exacte).
- compacte et efficace (détermine les caractéristiques désirées et moins coûteuse).

En CAO, les méthodes de représentation des surfaces sont classées en deux grandes familles :

- surface paramétrique,
- surface implicite.

II.1 Surfaces paramétriques :

Ces surfaces sont définies par un ensemble de trois fonctions, une pour chaque coordonnée, comme suit :

$$f(u,v) = (x(u,v), y(u,v), z(u,v)) \quad (1.1)$$

Où *x, y, z* sont des fonctions réelles dépendant de deux paramètres *u, v* qui appartiennent en général à l'intervalle [0,1]. La figure 1.1 montre la correspondance d'un point de coordonnées (u,v) dans le plan paramétrique et le point f(*u*,v) de la surface paramétrique dans l'espace de coordonnées X, Y et Z.

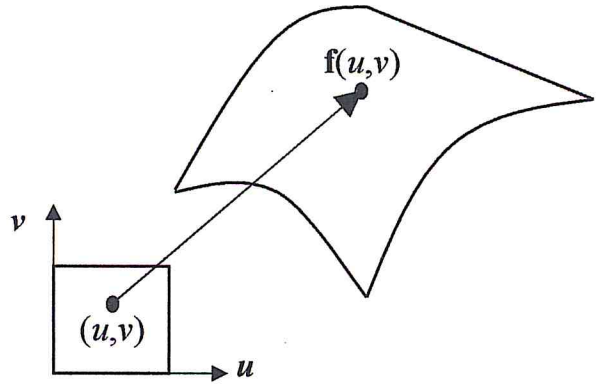


Figure1. 1 : Localisation d'un point sur une surface paramétrée.

Cette représentation est la plus utilisée en CAO grâce à sa souplesse et sa simplicité. Les surfaces paramétriques ne sont pas utilisées individuellement pour représenter un objet de forme complexe, mais plusieurs surfaces sont jointes pour former cet objet.

II.1.1. Propriétés géométriques des surfaces paramétriques :

Les propriétés géométriques des surfaces paramétriques sont les suivantes :

II.1.1.1. Vecteurs tangents et vecteur normal à la surface paramétrée en un point :

Pour calculer les vecteurs tangents et le vecteur normal en un point de la surface de coordonnées paramétriques (u, v), nous avons besoins de calculer les dérivées partielles. Les vecteurs tangents à la surface au point P (*u, v*) sont donnés par :

$$T_u = \frac{\partial f}{\partial u} = \left(\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u}\right) \quad (1.2)$$

$$T_v = \frac{\partial f}{\partial v} = \left(\frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v}\right) \quad (1.3)$$

Ces deux vecteurs définissent le plan tangent à la surface au point P (*u, v*). Le vecteur normal à la surface ***n*** (*u, v*), est le produit vectoriel des deux vecteurs tangents et est donné par :

$$n = \frac{\frac{\partial f}{\partial u} \times \frac{\partial f}{\partial v}}{\left| \frac{\partial f}{\partial u} \times \frac{\partial f}{\partial v} \right|} \qquad (1.4)$$

La figure 1.2 présente les vecteurs tangents *Tu* et *Tv*, la normale n et le plan tangent *P* en un point *f(u,v)* d'une surface paramétrique.

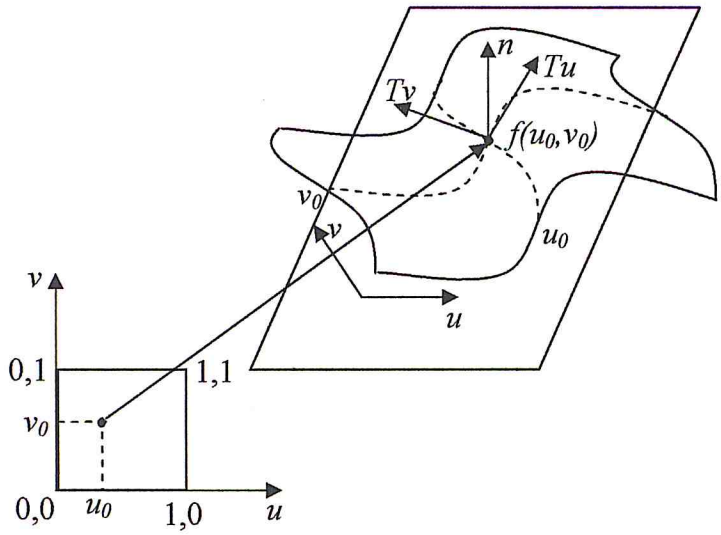


Figure 1.2 : Vecteurs tangents et vecteur normal en un point sur une surface paramétrique.

II.1.1.2. Courbes isoparamétriques :

La surface paramétrique *f* (*u*, *v*) est obtenue en fixant l'un des deux paramètres et en faisant varier l'autre paramètre ce qui génère une courbe appelée courbe isoparamétrique dans la direction de la variable variée. La figure 1.3 montre les courbes isoparamétriques suivant les directions U et V.

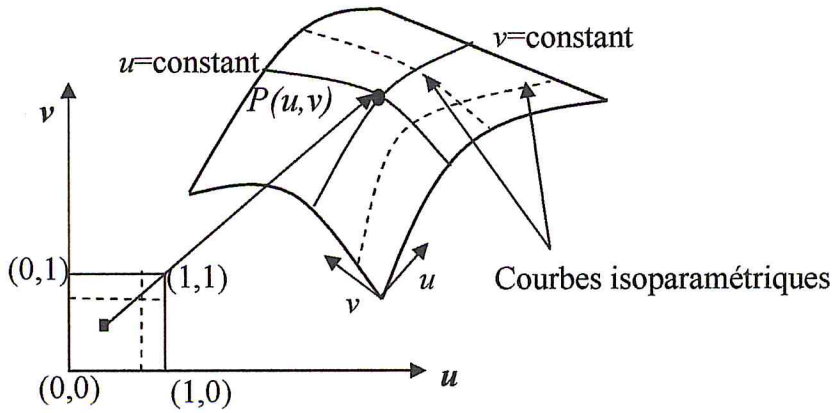


Figure 1.3 : Courbes isoparamétriques d'une surface paramétrique.

II.1.1.3. Notion de courbure [3] :

En un point d'une courbe, la courbure est l'inverse du rayon du cercle osculateur (le cercle qui approxime le mieux la courbe que tout autre cercle), ce rayon est appelé rayon de courbure (voir figure 4). Cette courbure est donnée par l'équation suivante :

$$k = \frac{1}{R} \tag{1.5}$$

R est le rayon du cercle de courbure.

K est la courbure.

t est le vecteur tangent et *n* est le vecteur normal au point considéré *x*.

La figure 1.4 montre le cercle osculateur et la courbure d'une courbe :

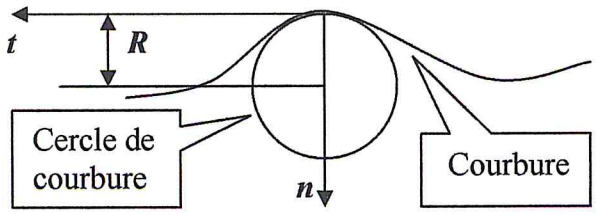


Figure1.4 : Courbure d'une courbe en un point x.

Une courbe possède une seule valeur de la courbure, mais une surface possède plusieurs types de courbures et le calcul d'un type de courbure d'une surface permet la détection des anomalies extérieures possibles de cette surface. Ainsi, c'est souvent nécessaire de calculer plusieurs genres de courbures pour obtenir une idée précise sur la forme de la surface. En n'importe quel point donné sur une surface paramétrique, il y a un nombre infini de courbes appartenant à cette surface et passant par ce point et dont chacune pourrait avoir une valeur différente pour la courbure. Parmi ces valeurs de courbures, il y a deux valeurs dont la première est la valeur minimale K_1 et la deuxième représente la valeur maximale K_2. Ces deux valeurs extrêmes correspondent aux courbures extrêmes et sont appelées les courbures principales. Les rayons de courbures principaux d'une surface sont les solutions de l'équation du second degré suivantes :

$$(LN - M^2)R^2 + (2MF - GL - EN)R + EG - F^2 = 0 \tag{1.6}$$

Où *L, E, N, G, M* et *F* sont les paramètres utilisés pour calculer les deux formes fondamentales et sont donnés par les relations suivantes :

$$E = \frac{\partial P}{\partial u} \times \frac{\partial P}{\partial u} \tag{1.7}$$

$$F = \frac{\partial P}{\partial u} \times \frac{\partial P}{\partial v} \tag{1.8}$$

$$G = \frac{\partial P}{\partial v} \times \frac{\partial P}{\partial v} \tag{1.9}$$

$$L = n \times \frac{\partial^2 P}{\partial u^2} \tag{1.10}$$

$$M = n \times \frac{\partial^2 P}{\partial u \partial v} \tag{1.11}$$

$$N = n \times \frac{\partial^2 P}{\partial v^2} \tag{1.12}$$

On appelle courbure moyenne d'une surface au point *P (u, v)*, le nombre *H* défini comme suit :

$$H = \frac{K_1 + K_2}{2} \tag{1.13}$$

La courbure gaussienne *K* d'une surface en un point est donnée par :

$$K = K_1 \times K_2 \tag{1.14}$$

Un autre type de courbure est la courbure absolue qui est calculée par la formule suivante :

$$A = |K_1| + |K_2| \tag{1.15}$$

Chaque point de la surface est caractérisé par deux rayons :

- Rayon minimum : c'est le rayon qui correspond à la courbure maximale.
- Rayon maximum : c'est le rayon qui correspond à la courbure minimale.

Suivant les valeurs de ces deux courbures (H, K) en un point *x* sur la surface, et la valeur de produit des deux rayons, celle-là aura les propriétés suivantes :

- Si *K*>0 alors *x* est un point elliptique. Si les deux courbures principales sont positives, alors c'est un point concave. Si les deux courbures principales sont négatives, alors c'est un point convexe.
- Si *K*<0 alors *x* est un point hyperbolique. C'est un point en selle de cheval.
- Si *K*=0 alors *x* est un point parabolique. C'est un point développable
- Si *K*<0 et *H*=0 alors *x* est un point plat.

II.2. Surfaces non paramétriques [2] :

II.2.1. Surfaces explicites :

Une surface explicite est donnée par une équation algébrique de la forme :

$$Z= f(x, y) \tag{1.16}$$

Ou à chaque valeur de x et y correspond une seule valeur de Z. Donc ces surfaces ne permettent pas de représenter des surfaces fermées puisque des valeurs multiples ne sont pas permises. La forme explicite n'est pas adaptée à la CAO pour les raisons suivantes :

- limitation des surfaces représentées.
- la difficulté de la modélisation et la modification interactive des surfaces.

II.2.2. Surfaces implicites :

Les surfaces implicites sont définies par une équation de la forme :

$$f(x,y,z) = 0 \tag{1.17}$$

Ces surfaces n'ont pas les limitations des surfaces explicites, mais toujours il est difficile de concevoir et de manipuler ces surfaces.

III. METHODES DE CONCEPTION DES SURFACES [1] :

La conception des surfaces a besoin d'un système de modélisation qui leur permet de concevoir et de manipuler les surfaces d'une manière interactive et en temps réel. Dans la pratique, les ingénieurs concepteurs ne s'inquiètent pas des mathématiques, mais ils se concentrent sur leurs travaux. Les méthodes qui aident les concepteurs des surfaces doivent avoir les propriétés suivantes :

- **Intuitif :** chaque étape et chaque algorithme doit avoir une interprétation intuitive et géométrique.
- **Flexible :** le système devrait fournir aux utilisateurs plus de commande pour concevoir et éditer la forme d'une surface. La manière de créer et d'éditer une surface devrait être facile et géométrique.
- **Approche Unifiée :** les différentes types de surfaces peuvent être conçues, crées et éditées de la même manière.
- **Invariable :** la surface représentée ne changera pas sa géométrie sous des transformations géométriques telles que la translation, la rotation et le changement d'échelle.

- **Efficacité et stabilité numérique :** la conception doit être rapide et exacte, de plus la grande quantité de calcul ne déformera pas la surface.

Les méthodes de conception de surfaces sont classées en deux grandes classes :

- Méthodes basées sur les points : pour ces méthodes, l'information de base de conception est le point.
- Méthodes basées sur les courbes : pour ces méthodes, l'information de base de conception est la courbe.

La figure 1.5 représente les différentes méthodes de conception des surfaces.

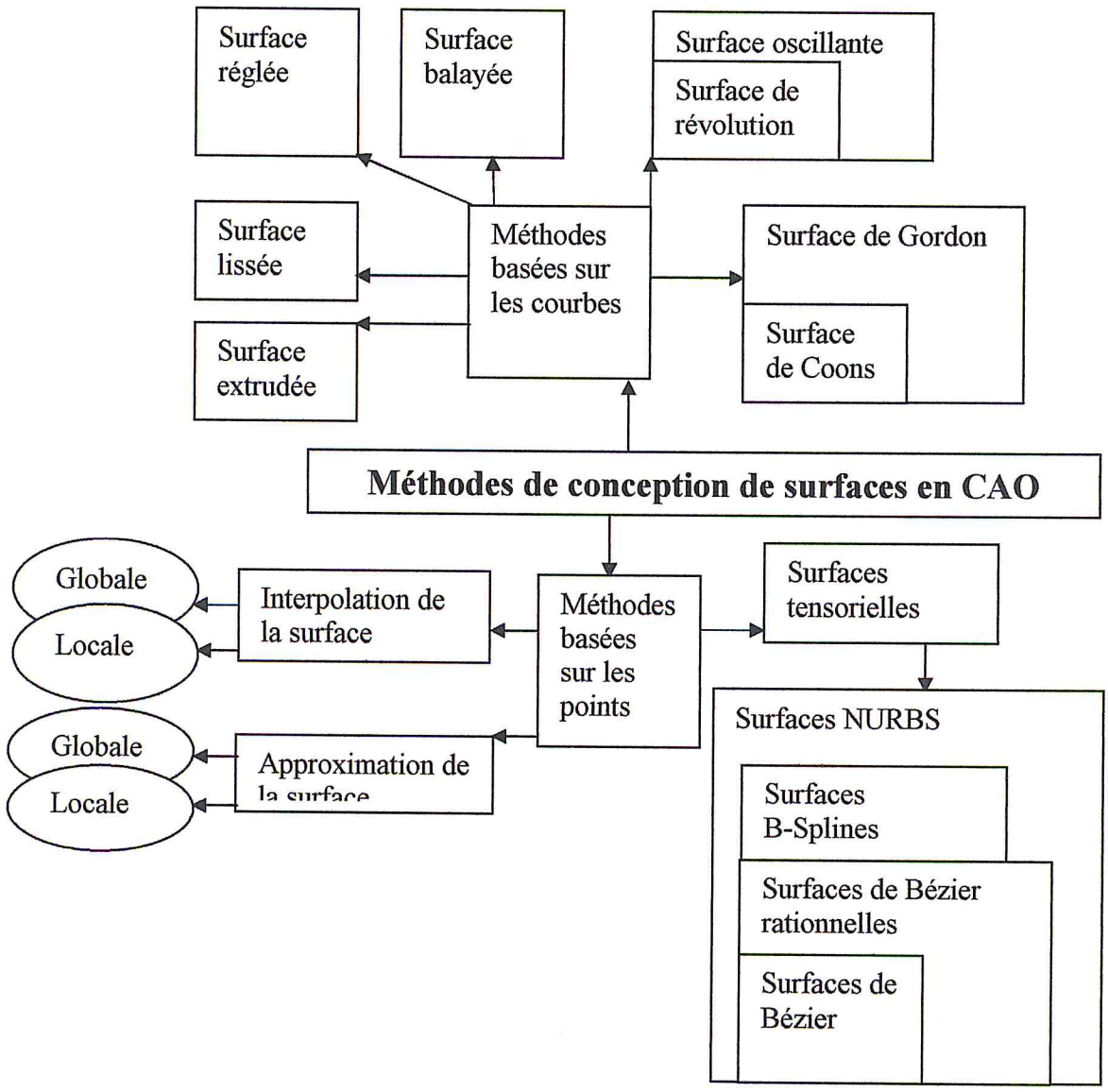


Figure1.5 : Les différentes méthodes de conception d'une surface en CAO.

IV. SURFACE B-SPLINE [2, 4, 6]:

Pour concevoir une surface B- Spline, nous avons besoin des informations suivantes :

- Un réseau de (m+1) lignes et (n+1) colonnes de points de contrôle $P_{i,j}$ ($0 \le i \le m$, $0 \le j \le n$);
- Un vecteur nodal de (h+1) noeuds dans la direction u, $U = \{u_0,..,u_h\}$;
- Un vecteur nodal de (k+1) noeuds dans la direction v, $V = \{v_0,...,v_k\}$;
- Le degré p dans la direction u ;
- Le degré q dans la direction v ;

L'équation d'une surface de B-Spline définie par les paramètres précédents est donnée par :

$$p(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)p_{i,j} \qquad (1.18)$$

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

La figure 1.6 montre les paramètres de définition d'une surface B-Spline.

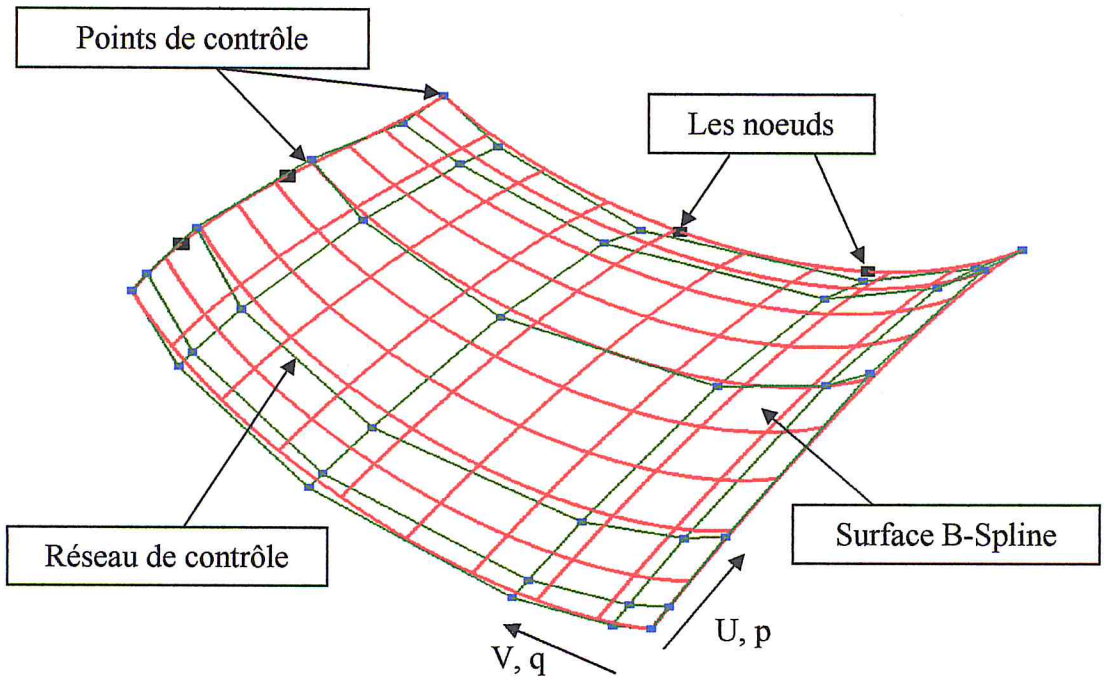


Figure 1.6 : Paramètres d'une surface B-Spline.

IV.1. Vecteur nodal :

Pour les courbes l'ensemble de (m+1) points $u_0, u_1, ..., u_m$ tel que chaque point appartient à la courbe, subdivise la courbe en petits segments de courbes, est appelé vecteur nodal. Ce vecteur noté U est un ensemble de valeurs réelles croissantes $u_0 \leq u_1 \leq ..... \leq u_m$ et dont les valeurs appartiennent à l'intervalle [0, 1]. Pour les surfaces, les valeurs nodales subdivisent la surfaces en sous surfaces appelées « patch ».

Si un nœud u_i apparaît k fois (k>1), alors ce nœud est dit un nœud multiple de multiplicité k, sinon (k = 1) il est dit simple.

Le concept de vecteur nodal apporte une grande flexibilité aux surfaces B-Splines puisqu'il il permet de gérer indépendamment le degré de la surface B-Spline du nombre de points de contrôle.

Les surfaces B-Spline nécessitent deux vecteurs nodaux, un pour chaque direction. Si les nœuds sont à distance égale, alors le vecteur nodal est uniforme, sinon il est dit non uniforme.

IV.2. Fonction de base de B-Spline :

Le coefficient de point de contrôle $p_{i,j}$ est le produit de deux fonctions unidimensionnelles de base B- Spline :

- dans le u- direction, $N_{i,p}(u)$.
- dans le *v*- direction, $N_{j,q}(v)$.

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

Les fonctions base, de degré 0 suivant *u* et suivant *v* sont définies par :

$$N_{i,0}(u) = \begin{cases} 1 & \text{si } u_i \leq u \prec u_{i+1} \\ 0 & \text{sinon} \end{cases} \quad (1.19)$$

$$N_{i,0}(u) = \begin{cases} 1 & \text{si } v_i \leq v \prec v_{i+1} \\ 0 & \text{sinon} \end{cases} \quad (1.20)$$

Les fonctions base, de degré *p* suivant *u* et *q* suivant *v* sont définies par :

$$N_{i,p}(u) = \frac{u-u_i}{u_{i+p}-u_i} N_{i,p-1}(u) + \frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}} N_{i+1,p-1}(u) \quad (1.21)$$

$$N_{j,q}(v)=\frac{v-v_j}{v_{j+p}-v_j}N_{j,q-1}(v)+\frac{v_{j+q+1}-v}{v_{j+q+1}-v_{j+1}}N_{j+1,q-1}(v) \qquad (1.22)$$

Une fonction base B-Spline $N_{i,p}(u)$ de degré p est une combinaison linéaire de deux fonctions base B-Spline $N_{i,p-1}(u)$ et $N_{i+1,p-1}(u)$ de degré $p-1$ (cette remarque est valable pour l'autre direction v).

IV.3. Les formes d'une surface B-Spline : [5]

Une surface B-Spline peut avoir trois formes possibles dans chaque direction (pincée, ouverte ou fermée).

- Si le vecteur nodal dans une direction n'a aucune structure particulière, la surface produite est ouverte dans cette direction et ne passe pas par les points de contrôle $P_{0,0}$, $P_{m,0}$, $P_{0,n}$ et $P_{m,n}$;
- La surface B-Spline est pincée dans la direction U (respectivement dans la direction V) si le premier et le dernier nœud est répété $p+1$ (respectivement $q+1$) fois. Si la surface est pincée dans les deux direction, alors la surface passe par les points de contrôle $P_{0,0}$, $P_{m,0}$, $P_{0,n}$ et $P_{m,n}$ et de plus elle est tangente aux huit segments du réseau de contrôle en ces points de contrôle ;
- En répétant quelques nœuds et points de contrôle, la surface produite peut être fermée, et dans ce cas les premiers et derniers points de contrôle (dans la direction où la surface est fermée) sont égaux. Si la surface est fermée dans une direction, alors toutes les courbes isoparamétriques sont fermées dans cette direction.

IV.4. Propriétés des surfaces B-Spline :

Les propriétés d'une surface B-Spline sont les suivantes :

1. Non négativité : $N_{i,p}(u)\ N_{j,q}(v)$ est non négative pour tous les p, q, l, j , u et v.
2. Partition d'unité : la somme de toutes les fonctions $N_{i,p}(u)\ N_{j,q}(v)$ est égale à 1 dans l'intervalle [0,1].

$$\sum_{i=0}^{m}\sum_{j=0}^{n}N_{i,p}(u)N_{j,q}(v)=1 \qquad (1.23)$$

3. Propriété de l'enveloppe convexe : si (u,v) est dans l'intervalle $[u_i, u_{i+1}[\times[v_j, v_{j+1}[$, alors $P(u,v)$ est contenue dans l'enveloppe convexe définie par les points de contrôle $P_{s,t}$ tel que $i-p \le s \le i$ et $j-q \le t \le j$.
4. Si $m = p$, $n = q$, et $U = \{0, 0\ldots, 0, 1, 1\ldots, 1\}$, alors une surface de B- Spline devient une surface de Bézier.

5. $P(u,v)$ a une continuité C_{p-s} (respectivement C_{q-t}) dans la direction U (respectivement la direction V) si u (respectivement v) est un noeud de multiplicité s (respectivement t).
6. Schéma de modification locale : $N_{i,p}(u)N_{j,q}(v)$ est non nulle si (u,v) est dans l'intervalle $[u_i,u_{i+p+1}[\times [v_j,v_{j+q+1}[$ et nulle ailleurs.
7. N'importe quelle courbe isoparamétrique sur la surface B-Spline est une courbe B-Spline définie par un ensemble de points de contrôle.
8. Les deux entités fondamentales doivent être satisfaites :

$$\begin{cases} n = m + p + 1 \\ et \\ k = n + q + 1 \end{cases} \qquad (1.24)$$

9. Les courbes frontières pour u=0, u=1, v=0 et v=1 sont définies par les points de contrôle suivants :

   $P_{0,0}$, $P_{1,0}$,..., $P_{m,0}$ pour la courbe $P(0,v)$,
   $P_{0,n}$, $P_{1,n}$,..., $P_{m,n}$ pour la courbe $P(1,v)$,
   $P_{0,0}$, $P_{0,1}$,..., $P_{0,n}$ pour la courbe $P(u, 0)$,
   $P_{m,0}$, $P_{m,1}$, ..., $P_{m,n}$ pour la courbe $P(u, 1)$.

10. Pour une surface B-Spline pincée, la surface passe par les quatre points : $p_{0,0}$, $p_{m,0}$, $p_{0,n}$ et $p_{m,n}$ et elle est tangente aux huit segments du réseau de contrôle.
11. Puisque $N_{i,p}(u)$ et $N_{j,q}(v)$ sont des fonctions polynomiales de degré p et degré q, alors la surface B-Spline est une surface polynomiales de degré (p,q).
12. La surface est indépendante du système de coordonnées.
13. Invariance affine : pour appliquer une transformation affine à une surface B-Spline, il suffit d'appliquer cette transformation à tous les points de contrôle.

V. SURFACES NURBS [1] :

Le terme NURBS signifie ***Non-Uniform Rational B-Spline*** (B-Spline rationnelle non uniforme). Plus précisément :

- *Non uniforme* signifie que les nœuds peuvent ne pas être équidistants. Cela peut être utile pour la modélisation des surfaces irrégulières.
- *Rationnelle* signifie que l'équation représentant la courbe ou la surface est exprimée sous la forme d'un rapport entre deux polynômes, et non sous la forme polynomiale comme pour le cas des B-Spline. L'équation rationnelle permet d'obtenir un meilleur modèle pour certaines courbes et surfaces importantes, en

particulier pour les sections coniques, (les cônes, les sphères,... etc.).

- Une *B-Spline* correspond à *Spline de base.*

VI. Pourquoi NURBS ?

1 Peuvent représenter n'importe quelle forme désirée, des points, des lignes droites, des courbes de forme libre, des sections coniques (cercles, ellipses, paraboles, et hyperboles);

2 Donnent un grand contrôle de la forme d'une courbe ou d'une surface ;

3 Peuvent représenter des formes très complexes avec peu de données ;

4 Les surfaces NURBS sont employés pour des raisons informatiques : faciles à traité par ordinateur, stables à des erreurs et ayant peu de conditions de mémoire.

Une surface NURBS est définie à partir des informations suivantes :

- Un réseau de (m+1) lignes et (n+1) colonnes de points de contrôle $P_{i,j}$ ($0 \le i \le m$, $0 \le j \le n$);
- Un vecteur nodal de (h+1) noeuds dans la direction u, $U = \{u_0,..,u_h\}$;
- Un vecteur nodal de (k+1) noeuds dans la direction v, $V = \{v_0,...,v_k\}$;
- Le degré p dans la direction u ;
- Le degré q dans la direction v ;
- Un ensemble de poids Wi,j pour chaque point de contrôle Pi,j ($0 \le i \le m$, $0 \le j \le n$) avec $Wi,j \ge 0$

Une surface NURBS définie par les paramètres précédents est donnée par :

$$p(u,v) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)w_{i,j}p_{i,j}}{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \qquad (1.25)$$

La figure 1.7 montres une surface NURBS et ses paramètres de définition.

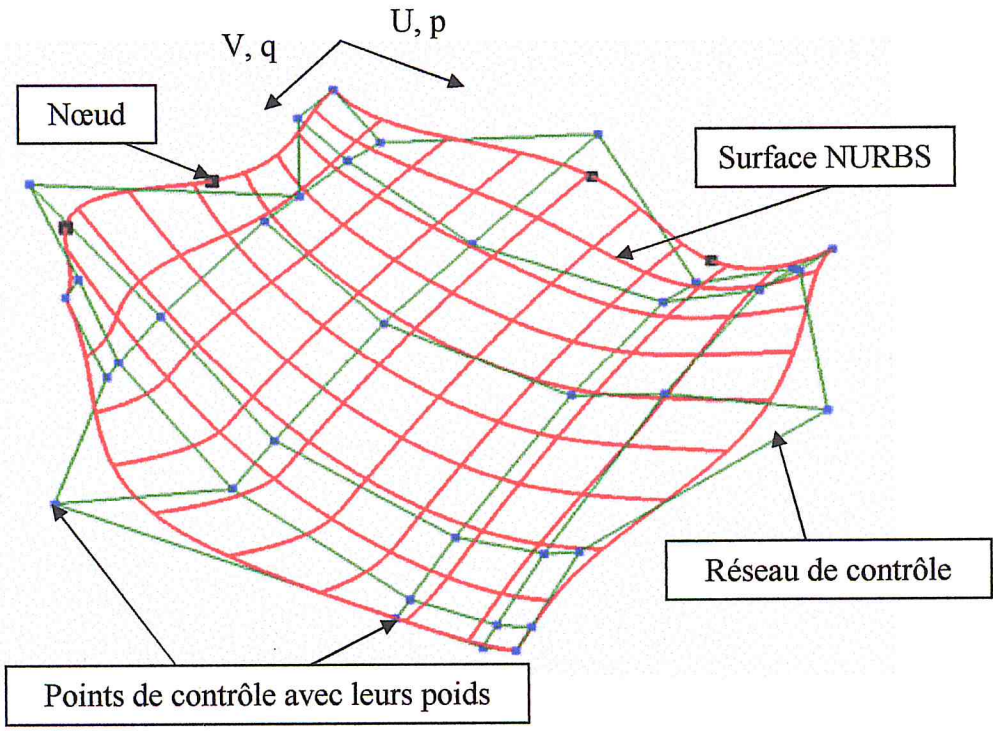


Figure 1.7 : Paramètres d'une surface NURBS.

Puisque $N_{i,p}(u)$ et $N_{j,q}(v)$ sont des fonctions de degré p et degré q, alors on dit que la surface NURBS est de degré (p,q).

Les surfaces NURBS sont des surfaces à produit tensoriel. Comme les surfaces B-Spline, une surface NURBS peut être ouverte, fermée ou pincée tout dépend des vecteurs U et V et du réseau de contrôle.

VI.1. Propriétés d'une surface NURBS :

En plus des propriétés des B-Splines, nous avons les propriétés suivantes :

1. Si on augmente (resp. diminue) la valeur du poids $W_{i,j}$ du point de contrôle $P_{i,j}$ une portion de la surface est approchée (resp. éloignée) de ce point.
2. Les surfaces NURBS sont une généralisation des surfaces Bézier & des B-Spline. Si tous les poids sont égaux, la surface NURBS devient une surface B-Spline. Si de plus, $m = p$, $n = q$, $U = \{0, 0, ..., 0, 1, 1, ..., 1\}$ et $V = \{0, 0, ..., 0, 1, 1, ..., 1\}$, alors une surface NURBS devient une surface de Bézier.
3. Toute courbe isoparamétrique sur la surface NURBS est une courbe NURBS définie par un ensemble de points de contrôles.
4. Une surface NURBS est une surface rationnelle ce qui permet de représenter toutes les coniques. Les surfaces NURBS permettent de représenter les coniques (sphère, cylindre, ellipsoïde,...etc.).

5. Invariance projective : si une transformation projective est appliquée à une surface NURBS, alors la surface projetée est construite en projetant les points de contrôle.

VII. MODIFICATIONS DE LA FORME D'UNE SURFACE B-SPLINE OU NURBS :

La modification d'une surface B-Spline ou NURBS peut être faite en agissant sur un ou plusieurs paramètres et qui sont les suivants :

1 La position des points de contrôle ;

2 Insertion et suppression des points de contrôle ;

3 Les valeurs des nœuds dans les deux directions ;

4 Insertion des nœuds dans les deux directions ;

5 Le degré de la surface dans les deux directions ;

6 Les poids des points de contrôle pour les surfaces NURBS.

VIII. CONCLUSION :

Dans ce chapitre nous avons étudié les différentes méthodes de représentation des surfaces paramétrique et non paramétrique. Ensuite nous sommes passé à la présentation des méthodes de conception des surfaces et nous avons terminé par la définition des surfaces les plus utilisées dans la conception et la modélisation des surfaces en conception assistée par ordinateur (CAO) et qui sont les surfaces B-Spline et NURBS. Dans la suite de notre travail, les surfaces à usiner sont soit des surfaces B-Spline ou des surfaces NURBS.

Chapitre2

Architecture et Programmation des Fraiseuses à Commande Numérique

I. INTRODUCTION :

Le fraiseur mécanicien dans son travail donne la forme voulue à des pièces mécaniques en faisant des rainures, des engrenages, des perçages...etc. Pour réaliser ces usinages, il utilise des machines dédiées appelées « fraiseuses ». Avec l'introduction des nouvelles technologies, les fraiseuses manuelles sont de plus en plus remplacées par des machines automatisées à commande numérique.

Suite à la complexité des surfaces gauches (surface de forme libre), ces surfaces ne peuvent être usinées sur des fraiseuses conventionnelles et nécessitent l'utilisation des machines automatiques. Ces machines sont les fraiseuses à commande numérique que nous allons présenter dans ce chapitre.

II. DEFINITION D'UNE MACHINE OUTIL A COMMANDE NUMERIQUE [7] :

Une Machine Outil à Commande Numérique (M.O.C.N.) est une machine-outil d'usinage programmable équipée d'une commande numérique par calculateur (CNC) et elle est composée principalement de deux parties, une partie opérationnelle et une partie commande,(La figure 2.1 montre deux types de fraiseuses).

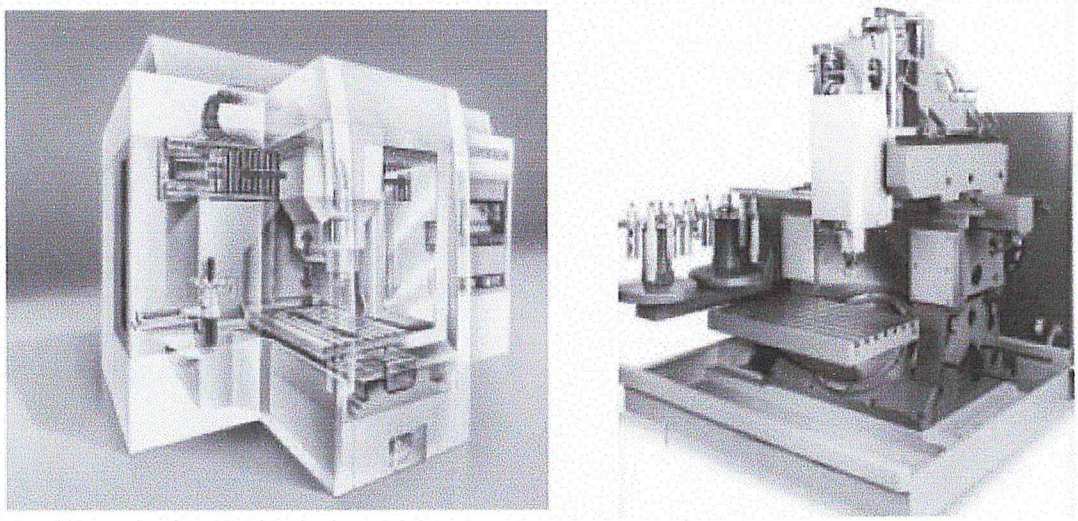


Figure 2. 1 : Vue générale des machines à fraiser.

III. LES FONCTIONS D'UNE MACHINE OUTIL A COMMANDE NUMERIQUE :

Une machine outil est utilisée dans le but :

- de réaliser physiquement les mouvements de coupe nécessaires à l'obtention d'une surface par enlèvement de matière.
- de réaliser le mouvement de coupe et le mouvement d'avance de l'outil par rapport à la pièce.
- d'obtenir des pièces respectant les spécifications fonctionnelles.

Une machine outil à commande numérique permet d'assurer la réalisation automatisée des pièces de différentes formes en décrivant les mouvements nécessaires à l'usinage de la pièce dans un programme d'usinage.

IV. ARCHITECTURE MECANIQUE D'UNE FRAISEUSE A COMMANDE NUMERIQUE :

Une machine à commande numérique est destinée aux travaux de fraisage ou le mouvement de coupe est donné à l'outil fixé à la broche (rotation de la fraise), et qui peut translater suivant les 3 axes de la machine.

Une M.O.C.N est composée de deux parties complémentaires :

IV.1. Partie opérationnelle:

Cette partie englobe les éléments mécaniques de la machine :

- Axes de la machine : des systèmes, autant que nécessaire, assurant la mise en position de l'outil par rapport à la pièce et les mouvements d'avance.
- La broche : un système qui réalise le mouvement de coupe par mise en rotation des outils de la machine ;
- le bâti : l'élément mécanique qui assure le lien entre ces systèmes:
- Les porte-outils.

IV.2. Partie commande :

C'est l'ensemble des commandes associées à la machine qui gèrent les mouvements à partir d'un programme d'usinage. Ce programme est introduit à la machine soit avec une ligne de communication pour charger des programmes d'un équipement informatique externe (serveur ou ordinateur), soit par le pupitre de commande.

Parmi les composants électroniques de cette partie :

- Un système de contrôle - commande, qui permet le suivi automatique du programme de commande de la machine;
- Le pupitre de commande;
- L'armoire électronique.

IV.3. La broche :

La broche crée le mouvement de coupe nécessaire à l'usinage. Elle assure donc la mise en rotation de l'outil. La broche doit être très rigide, et stable thermiquement de façon à garantir la position relative de l'outil par rapport à la pièce durant l'usinage.

IV.4. L'armoire électronique [8] :

L'armoire électronique sert de relais entre la machine et le directeur de commande numérique (DCN). Elle renferme des câbles, des amplificateurs et des fusibles.

IV.5. Le pupitre de commande [9,10] :

Le pupitre de commande sert à dialoguer avec le (DCN) et envoi des ordres de commande codés. Il possède des touches sensitives, ainsi qu'un écran graphique qui sert à visualiser le programme ou le profil fini de la pièce et la trajectoire des outils.

IV.6. Le bâti :

Le bâti est un support sur lequel sont fixées les pièces d'une machine. Il assure le guidage des axes de mouvements et l'agencement des autres organes de la machine. Pour assurer une géométrie correcte et encaisser les actions mécaniques dues aux accélérations élevées des mobiles, le bâti doit être rigide.

IV.7. Le directeur de commande numérique :

Le directeur de commande numérique (D.C.N.) est un automatisme composé d'éléments électroniques, qui à partir d'un programme d'usinage établi par un opérateur, fournit des ordres aux servocommandes des axes de la machine. La commande numérique assure l'asservissement en position et en vitesse des déplacements des mobiles. C'est purement de la commande d'axes avec un traitement numérique pour élaborer les consignes de commande en temps réel en fonction des paramètres de la trajectoire et de l'état de la chaîne d'action.

Les différentes fonctions assurées par le D.C.N sont les suivantes :

- Interprétation du programme d'usinage,
- Détermination des phases de travail (blocs exécutables),
- Calcul des consignes successives sur la trajectoire,
- Elaboration de l'écart de poursuite et des corrections nécessaires,
- Gestion des données et des mesures,
- Surveillance des erreurs.

En plus à ces fonctions, il gère l'ensemble des fonctions séquentielles associées à la machine :

- Commande des actionneurs auxiliaires,
- Modes de marche et d'arrêt,
- Commande de distribution d'énergie,
- Traitement des informations de sécurité,
- Dialogue avec l'opérateur.

IV.8. Les porte-outils :

Les porte-outils ont pour fonction d'assurer la liaison entre l'outil et la machine suivant le mode d'usinage. Dans le cadre du fraisage, les porte-outils assurent la liaison au moyen d'un cône normalisé.

IV.9. Les axes de déplacement [8,9] :

Les déplacements de l'outil ou du porte-pièce s'effectuent par combinaisons de translations et/ou de rotations. Donc un système de coordonnées qui permet de repérer les positions et les déplacements d'un objet par rapport à un point origine est nécessaire. Chaque mouvement élémentaire (axe) est repéré par une lettre affectée du signe + ou - indiquant le sens du déplacement. Un système de coordonnées cartésiennes rectangulaire est un trièdre de sens direct constitué de trois axes linéaires *X*, *Y* et *Z* auxquels sont associés trois axes rotatifs *A*, *B* et *C* (voir figure 2.2).

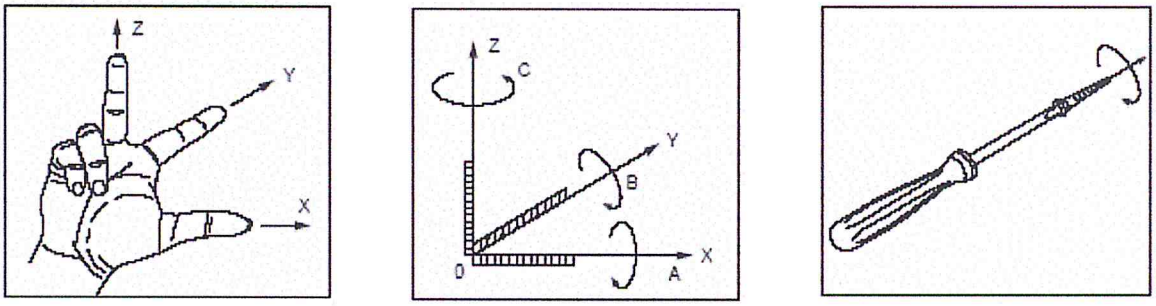


Figure 2.2: Orientation des axes de déplacement.

IV.10. Orientation des axes d'une fraiseuse:

La règle des trois doigts permet de retrouver facilement l'orientation des axes X, Y et Z. L'orientation positive d'un axe rotatif correspond à la rotation d'une vis de pas à droite avançant dans le sens positif de l'axe associé (sens du vissage) (voir figure 2.2).

L'orientation des axes d'une machine dépend du type de machine et de la disposition des éléments qui la constituent. Pour une fraiseuse :

- L'axe Z est parallèle à la broche principale.
- L'axe X est celui définissant le plus grand déplacement horizontal, et il est perpendiculaire à l'axe Z.
- L'axe *Y* forme le trièdre direct avec les axes X et Z.

Les angles A, B, C définissent les mouvements de rotation respectivement au tour des axes X, Y et Z.

Le système de coordonnées mesure les déplacements des outils par rapport à la pièce à usiner supposée fixe.

La commande d'axe permet d'asservir en position et/on en vitesse le déplacement des mobiles. Les déplacements sont contrôlés avec des capteurs de mesure.

Les axes de la machine sont de deux types :

- Axe numérique : axe de déplacement pour lequel une infinité de positions peut être atteinte à la résolution de positionnement près - ou - axe de déplacement asservi en position et en vitesse.
- Demi-axe numérique : axe de déplacement pour lequel un ensemble fini de positions peut être atteint - ou - axe de déplacement asservi en position ou en vitesse.

Les machines-outils sont classées par le nombre de mouvements élémentaires qu'elles peuvent mettre en oeuvre lors du déplacement de l'outil par rapport à la pièce. Seuls les axes sont décomptés, par exemple, les fraiseuses à 2.5 axes, 3 axes ,4 axes et à 5 axes. Une fraiseuse à 5 axes offre deux axes supplémentaires par rapport à une fraiseuse à 3 axes. Ces deux axes permettent d'orienter l'axe de l'outil.

V. CLASSIFICATION DES FRAISEUSES A COMMANDE NUMERIQUE [5] :

Le classement des machines est nécessaire car il aide au choix de machines lors d'étude de gammes de fabrication. Les machines sont classés en fonction du :

- mode de fonctionnement de la fraiseuse;
- nombre d'axes de la fraiseuse;
- mode de fonctionnement du système de mesure;
- mode d'entrée des informations.

V.1. Les types de fraiseuses à commande numérique :

Les fraiseuses à commande numérique sont de deux types :

V.1.1. Fraiseuses verticales :

Dans ce type de fraiseuses, l'axe Z du mouvement est vertical et il est parallèle à l'axe de la broche principale (voir figure 2.3). Le sens X positif est dirigé vers la droite, lorsqu'on regarde de la broche principale vers le montant de la machine. L'axe Y du mouvement forme avec les axes X et Z un trièdre de sens direct. Généralement, ce type de fraiseuses est le plus utilisé dans l'industrie mécanique et particulièrement dans l'usinage des surfaces gauches.

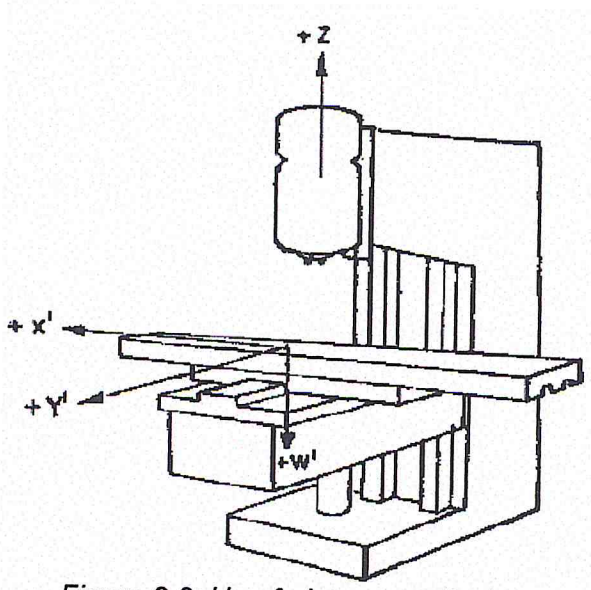


Figure 2.3: Une fraiseuse verticale.

V.1.2. Fraiseuses horizontales :

Malgré que l'axe Z est toujours celui de la broche, le système d'axes est différent Z est horizontale, le sens X positif est dirigé vers la droite, lorsqu'on regarde de la broche principale vers le montant de la machine et l'axe Y forme avec les axes X et Z un trièdre direct, (figure 2.4).

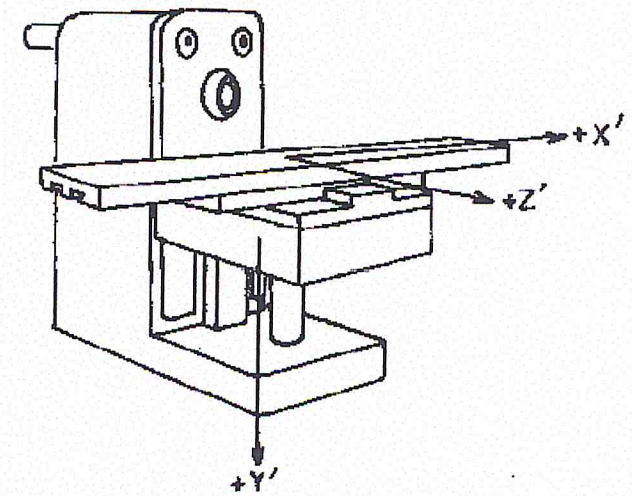


Figure 2.4: Une fraiseuse horizontale.

VI. FONCTIONNEMENT D'UNE FRAISEUSE A COMMANDE NUMERIQUE [11] :

A partir d'un programme d'usinage établi par un opérateur, le directeur de commande numérique (D.C.N.) fournit des ordres aux servocommandes des axes de la machine. Le système comprend un ou plusieurs microprocesseurs préprogrammés pour l'exécution des fonctions de la C.N. Le parcours de la trajectoire programmée s'effectue en déplaçant l'outil par rapport à la pièce.

Les machines effectuent des usinages soit par :

- Usinage point à point : qui s'effectue en programmant la position finale et le trajet parcouru pour atteindre cette position n'est pas contrôlée par le directeur de commande numérique ;
- Usinage en par axial : les trajectoires sont parallèles aux axes de déplacement ;
- Usinage en contourne : dans ce type d'usinage, plusieurs axes sont déplacés simultanément.

VII. OUTILS D'USINAGE [5,11] :

L'outil d'usinage utilisé pour le fraisage est la fraise qui caractérisée par :

- Son sens de coupe;
- Son type de denture;
- Son nombre de dents;
- Sa géométrie de coupe;
- Sa matière.

VII.1. Définition des dimensions de l'outil (Jauge) [1] :

La jauge d'outil est la distance entre l'arête coupante de l'outil par rapport au point de référence broche. Pour une fraise, les jauges sont : la longueur L, le rayon de la fraise R et le rayon @ de la parie torique pour une fraise torique (voir figure 2.5).

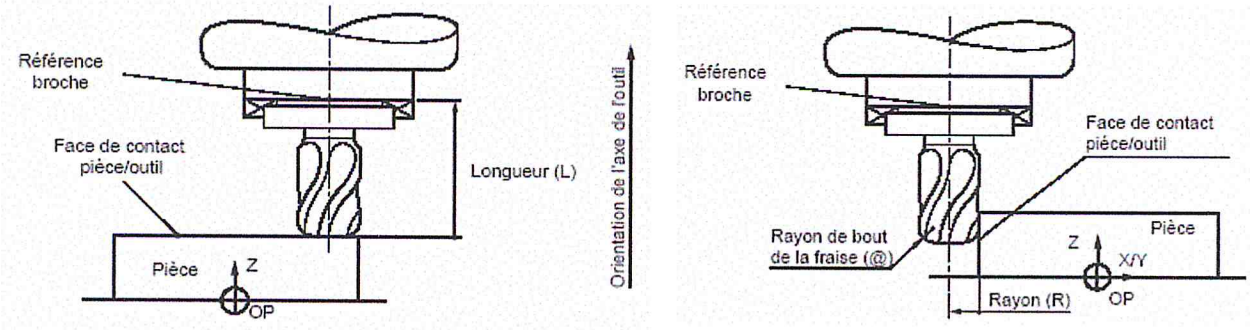


Figure 2.5 : Les jauges d'outil.

VIII. AVANTAGES DES FRAISEUSES A COMMANDE NUMERIQUE [5]:

Les fraiseuses à commande numérique ont les avantages suivants :

- la rapidité de lecture et de traitement des informations ;
- la rapidité de réponse des divers organes mobiles ;

- à sa conception et ses vitesses de déplacement ;
- à la précision que confère son architecture ;
- possibilités de réalisation d'un profil de pièce ;
- la souplesse de changement de type de pièce et à la polyvalence des Travaux permis ;
- aux cycles programmés ;
- modifications des paramètres d'usinage en cours d'usinage.

La programmation interactive de fabrication de pièces prismatiques fournit des fonctions de programmation CN d'entrée dédiées aux opérations classiques d'usinage. En se basant sur la géométrie, l'utilisateur crée des opérations d'usinage qui modélise l'environnement de fabrication spécifique à l'entreprise. Les paramètres d'usinage corrects sont alors définis (machine et outils, vitesse et rotation de l'outil et type d'usinage (perçage, fraisage plan ou point à point).

X. DEFINITION D'UN PROGRAMME [9]:

Un programme est une suite d'instructions écrites dans un langage codé propre à la commande numérique (le plus utilisé est le code ISO : International Organization for Standardization). La commande numérique interprète le programme pour commander un usinage sur la machine outil.

XI. ELABORATION D'UN PROGRAMME [9]:

Le programme pièce peut être créé par programmation traditionnelle ou par l'intermédiaire d'un système CFAO (voir figure2.6).

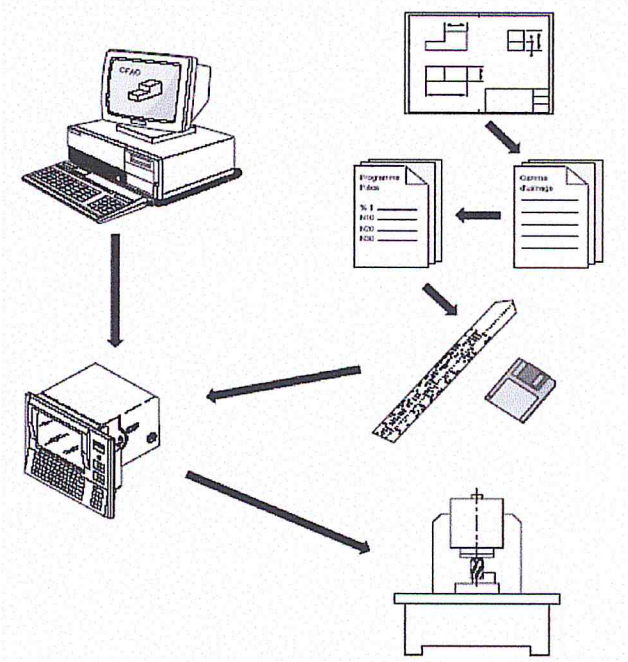


Figure 2.6 : Modes d'élaboration d'un programme d'usinage

Avant l'usinage d'une pièce sur une fraiseuse numérique il faut :

- Etudier les dessins de définition;
- Analyser le travail à effectuer ;
- Choisir les outillages de coupe (fraises) et de contrôle;
- Définir le montage de la pièce ;
- Déterminer l'origine programme (OP);
- Numéroter et coter les points du profil;
- Imaginer la trajectoire de la fraise;
- Ecrire le programme;
- Mettre la machine sous tension;
- Charger le programme dans le (DCN);
- Actionner la puissance;
- Introduire les dimensions des outils (jauges);
- Tester le programme mode rapide sans outil;
- Activer le bouton départ du cycle;
- Usiner la pièce;
- Contrôler la pièce usinée.

XI.1. Structure d'un programme :

Un programme CN comporte des caractères obligatoires de début et fins. Un programme est exécuté dans l'ordre d'écriture des blocs situés entre les caractères de début et de fin de programme.

XI.1.1. Format des blocs :

Un bloc (ou séquence) définit une ligne d'instructions composée de mots codés à transmettre au système de commande. Le format de bloc définit la syntaxe des mots de fonction et de dimension composant chaque bloc de programmation (figure 2.7).

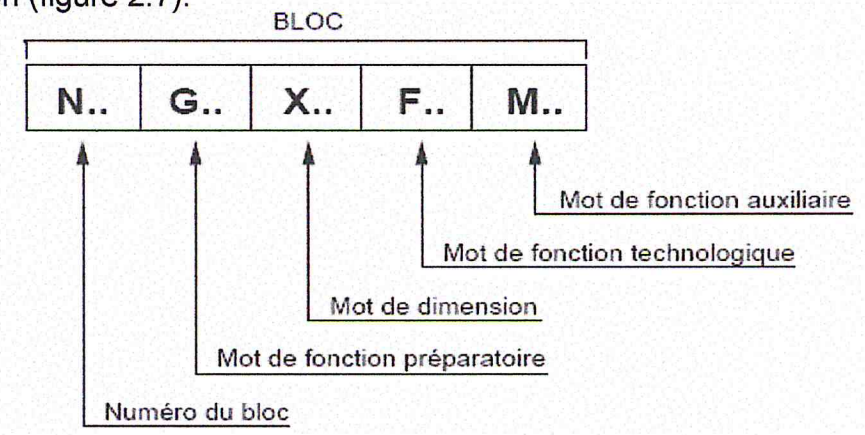


Figure 2.7 : Format de bloc.

XI.1.2. Format de mot :

Le mot définit une instruction ou donnée à transmettre au système de commande. Le format général des mots est la suivante :

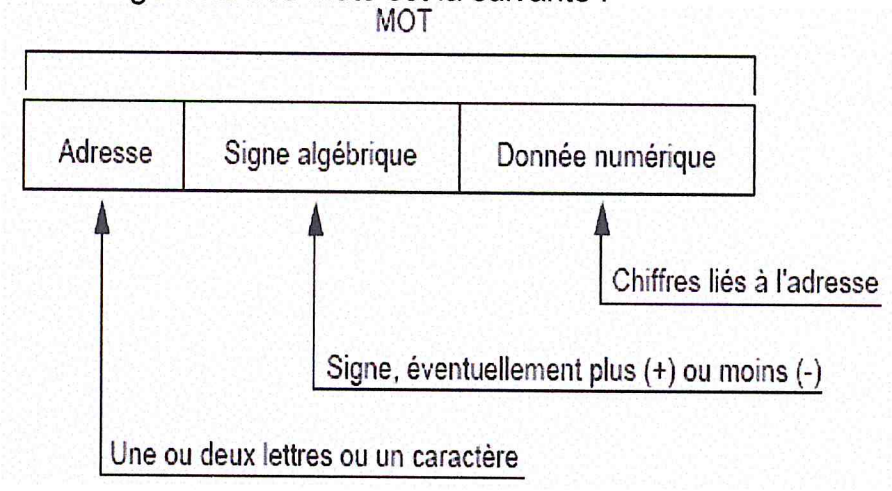


Figure 2.8 : Format de mot.

Il y a deux types de mots :

Mots définissant des fonctions :

- **G---** : fonctions préparatoires
- **F---**: fonction vitesse d'avance ("**F**eedrate" = avance)
- **S---**: fonction vitesse de broche ("**S**peed" = vitesse)
- **T---**: fonction outils ("**T**ools" = outils)
- **M---** : fonctions auxiliaires ("**M**iscellaneous" = divers)

Mots définissant des dimensions :

- **X---**: mouvement suivant l'axe X
- **Y---**: mouvement suivant l'axe Y
- **Z---**: mouvement suivant l'axe Z.

XI.2. Structure d'un programme ISO :

La figure 2.9 montre les différents paramètres à mettre dans un programme d'usinage.

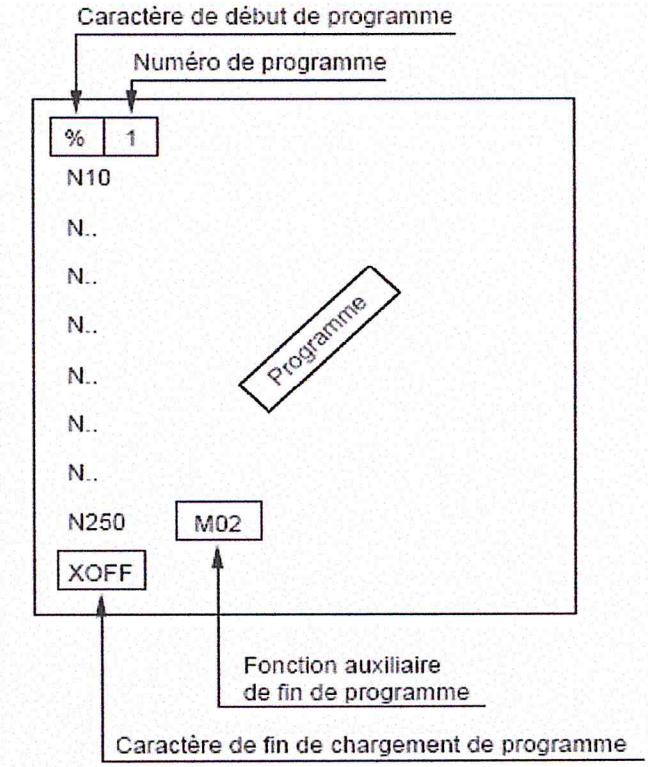


Figure 2.9 : Structure d'un programme.

Ces paramètres sont les suivants :

- Début de programme : caractère % suivi du numéro du programme.
- Fin de programme : code M02.
- Fin de chargement de programme: caractère XOFF.
- Commentaires : entre parenthèses.

XI.3. Classification des fonctions préparatoires G et auxiliaires M :

En programmation, nous avons deux types de fonctions :

- Fonctions auxiliaires M ;
- Fonctions préparatoires G.

XI.3.1. Classification des fonctions préparatoires G :[1]

Les fonctions préparatoires sont de deux types :

XI.3.1.1. Fonctions G modales :

Fonctions appartenant à une famille de fonctions G se révoquant mutuellement. La validité de ces fonctions est maintenue jusqu'à ce qu'une fonction de même famille révoque leur validité.

[1] Voir Annexe A.

N.. G00 X.. Y.. Interpolation linéaire à vitesse rapide
N.. G01 Z.. Interpolation linéaire à vitesse d'usinage révoque G00

XI.3.1.2. Fonctions G non modales :

Fonctions uniquement valide dans le bloc où elles sont programmées (révoquée en fin de bloc).

N.. G09 X.. Fonction d'arrêt précis en fin de bloc révoquée en fin de bloc

XI.3.2. Classification des fonctions auxiliaires M :

Les fonctions auxiliaires M sont de deux types :

- fonctions M modales,
- fonctions M non modales.

Elles peuvent être aussi:

- des fonctions «avant» ou «après»,
- des fonctions codées ou décodées.

XI.3.2.1. Fonctions M modales :

Fonctions appartenant à une famille de fonctions M se révoquant mutuellement. La validité de ces fonctions est maintenue jusqu'à ce qu'une fonction de même famille révoque leur validité.

N.. S500 M03 Mise en rotation de la broche
N.. M05 Arrêt de la broche, révoque M03

XI.3.2.2. Fonctions M non modales :

Fonctions uniquement valides dans le bloc ou elles sont programmées.

N.. M00 Fonction d'arrêt programmé

XI.3.2.3. Fonctions M «avant» :

Fonctions exécutées avant déplacements sur les axes programmés dans le bloc.

N.. X100 Y50 M08 : la fonction d'arrosage M08 est exécutée avant le déplacements sur X et Y.

XI.3.2.4. Fonctions M «après» :

Fonctions exécutées après déplacements sur les axes programmés dans le bloc.

N.. X50 Y100 M09 : la fonction d'arrêt arrosage (M09) est exécutée après déplacements sur X et Y.

XII. CONCLUSION :

Dans ce chapitre nous avons présenté la structure mécanique des machines à commande numérique et en particulier les fraiseuses à commande numérique, la structure d'un programme d'usinage en détaillant les différentes fonctions préparatoires et auxiliaires de ce programme. Ces données seront utilisées lors de la génération automatique des programmes d'usinage. Ce chapitre nous permettra de passer à l'usinage des surfaces gauches sur des fraiseuses à commande numérique à 3 axes.

Chapitre3

Usinage des Surfaces Gauches

I. INTRODUCTION [12] :

Pour usiner des surfaces gauches conçues dans un environnement de CAO, il est nécessaire de générer les trajets d'usinage permettant la réalisation physique de ces surfaces. Pour réaliser ces surfaces, on a recours à des logiciels de FAO (Fabrication Assistée par Ordinateur) dont le but est de générer des trajets outil capables d'usiner des formes complexes et éventuellement de transformer ces trajets en un programme compréhensible par le directeur de commande numérique.

Dans ce chapitre nous allons décrire les différents paramètres à considérer dans le processus de génération des trajets d'usinage des surfaces gauches.

II. PROCESSUS DE REALISATION D'UNE SURFACE GAUCHE [12] :

Le processus de réalisation d'une forme gauche passe par deux étapes :

- Conception : qui a pour but d'exprimer l'ensemble des contraintes fonctionnelles écrites dans le cahier des charges sous forme de surfaces ou d'éléments géométriques;
- Fabrication : qui pour but de transformer la conception obtenue en une forme réelle qui permet à l'objet de répondre à sa fonction.

La figure (3.1) montre la succession des différentes étapes permettant de passer de la conception jusqu'à l'obtention de la pièce réelle.

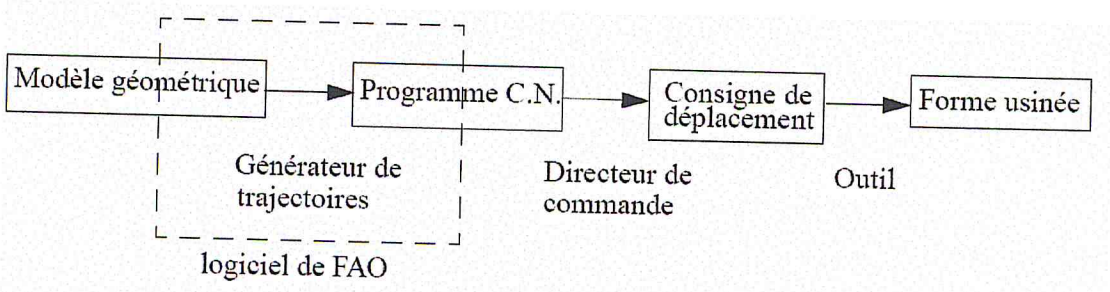


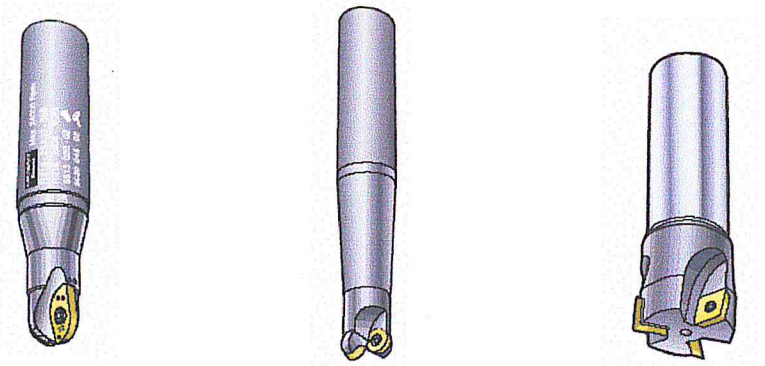
Figure 3.1 : Processus de réalisation d'un usinage.

III. LES DIFFERENTS PARAMETRES D'USINAGE :

III.1. Choix de l'outil [5] :

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 principaux types d'outils utilisés dans l'usinage des surfaces gauches sont les suivants :

- Fraise cylindrique ;
- Fraise sphérique ;
- Fraise torique.



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

Figure 3.2 : Différents types de fraises.

Pour usiner une surface gauche il faut avoir un enchaînement de déplacements et une forme d'outil qui soit toujours tangente à la surface à usiner en chaque point de contact. La forme la plus simple est la sphère. Mais pour garantir l'existence de la tangence et donc d'un unique point de contact sur des zones concaves et pour éviter les problèmes d'interférences, il faut que la valeur de son rayon soit plus petite que le plus petit des rayons de courbures principaux de la forme. L'outil hémisphérique est donc la solution pour tous les usinages du type 3 axes. Mais, il est nécessaire de conserver des petits rayons pour permettre d'usiner toute la pièce, ce qui diminue la rigidité de l'outil et augmente le nombre de passes nécessaires au respect d'une hauteur entre crêtes donnée. De plus l'outil hémisphérique possède une vitesse de coupe très faible sur les zones de l'arête de coupe proches de l'axe de rotation. Cette vitesse trop faible entraîne un mauvais comportement de la coupe et une usure importante.

L'usinage d'une surface en finition, générée par un logiciel de FAO, est une succession de trajets outil. Les trajectoires suivies par l'outil sont établies en fonction du mode d'usinage, des paramètres de guidage et des paramètres de passe.

III.2. Les modes d'usinage :

Les trajets outils sont déterminés par les intersections entre les surfaces à usiner et les surfaces de guidage de l'outil. Ce sont dans la plupart des cas des plans parallèles verticaux dont le choix de l'orientation est laissé à l'utilisateur. Mais il existe d'autres modes d'usinage, comme l'usinage suivant les isoparamétriques des surfaces paramétrées ou suivant des plans horizontaux (méthode de Z constant).

III.3. Les paramètres de guidage :

Le guidage de l'outil le long de la courbe peut se faire de deux manières différentes, soit par guidage du point extrémité de l'outil, soit par guidage du point de contact outil matière. En effet, pour l'usinage d'une surface quelconque, le point extrémité outil et le point de contact n'ont pas la même trajectoire et les

algorithmes de calcul de trajets ne sont pas les mêmes si l'on guide l'extrémité de l'outil ou le point de contact.

Le mode de balayage de la surface permet d'usiner dans le sens «aller» ou dans les deux sens «aller et retour», couramment appelés «one way» et «zigzag» dans les logiciels. Pour les opérations de finition, on utilise le mode «one way» pour usiner toujours en «avalant» afin d'améliorer l'état de surface. Pour usiner en avalant, il faut fixer le sens de parcours en fonction du sens de rotation de la broche.

III.4. Les paramètres de passe :

Les courbes de l'espace que suit l'outil sont approchées par interpolation linéaire et l'outil usine une ligne brisée, car les directeurs de commande numérique fonctionnent par interpolation linéaire.

III.4.1. Trajet d'outil :

Un trajet est représenté par une succession de couples formés par la position de l'extrémité de l'outil et la direction de l'axe outil. Générer des trajets outil, c'est trouver les positions de l'outil successives qui permettent d'usiner une surface selon des trajectoires définies par le programmeur machine outil. Ces trajectoires sont en général des courbes 3D.

Les déplacements de l'outil étant rectilignes, les courbes que doit suivre l'outil sont approchées par interpolation linéaire. Un trajet outil est donc une ligne brisée qui suit au mieux les surfaces selon un pas longitudinal (voir figure 3.3).

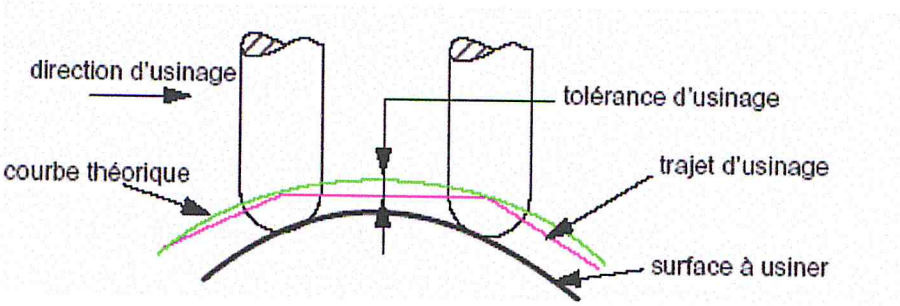


Figure 3.3 : Trajet réel de l'outil pendant l'usinage.

Afin de respecter la forme conçue, deux erreurs sont à satisfaire :

III.4.1.1.Erreur de flèche:

Les courbes de l'espace que suit l'outil sont approchées par interpolation linéaire et l'outil usine une ligne brisée, Le pas longitudinal d'usinage est calculé en général par le respect de la tolérance d'usinage qui est la valeur maximale de l'erreur de flèche entre la courbe et chacun des segments de la ligne brisée (voir figure 3.4).

Répéter
Usiner la courbe isoparamétrique (*u* ou *v)* jusqu'à sa fin,
Faire déplacer l'outil en quittant la surface,
Passer à la courbe suivante,
Jusqu'à fin surface

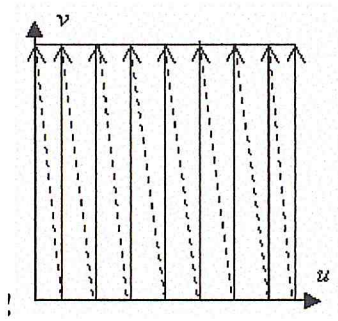


Figure 3.10 : Le trajet aller simple.

V.1.2. Aller retours (ZigZag):

Cette méthode consiste à usiner toute la surface selon l'isoparamétrique u ou v sans que l'outil quitte la surface (voir figure 3.11) selon l'algorithme suivant :

Répéter
Usiner la courbe isoparamétrique (*u* ou *v)* jusqu'à sa fin,
Faire déplacer l'outil sans quitter la surface,
Passer à la courbe suivante dans l'autre sens,
Jusqu'à fin surface

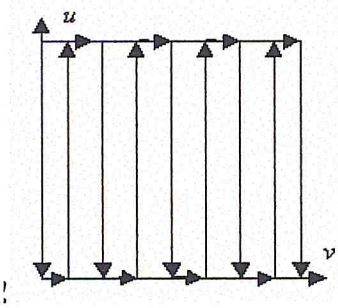


Figure 3.11 : Le trajet ZigZag.

V.1.3. Concentrique :

Cette méthode consiste à usiner la surface selon des courbes parallèles aux frontières du domaine de définition de la surface jusqu'à la fin de la surface (voir figure 3.12).

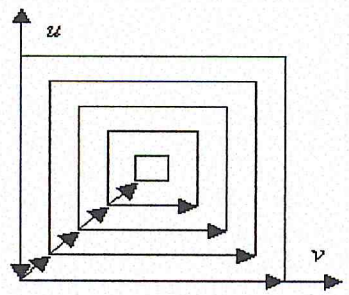


Figure3.12 ; trajet concentrique.

V.1.4. Spiral In :

Dans cette méthode l'outil d'usinage suit une trajectoire en forme de spirale jusqu'à usinage de toute la surface en partant de l'extérieure de la surface vers l'intérieur (voir figure 3.13).

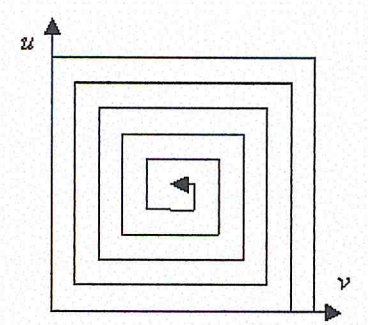


Figure3.13 : Le trajet spiral in.

V.1.5. Spiral Out:

Dans cette méthode l'outil d'usinage suit une trajectoire en forme de spirale jusqu'à usinage de toute la surface en partant de l'intérieure de la surface vers l'extérieur (voir figure 3.14).

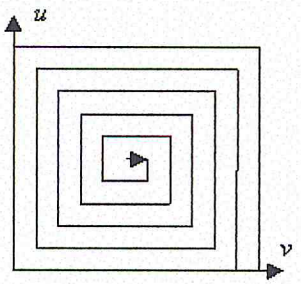


Figure 3.14 : Le trajet spiral out.

V.1.6. Radiale :

Cette méthode consiste à usiner la surface en commençant du centre de la surface jusqu'à la frontière de la surface et refaire la même étape jusqu'à la fin de la surface en déplaçant chaque fois le point de l'extrémité (voir figure 3.15).

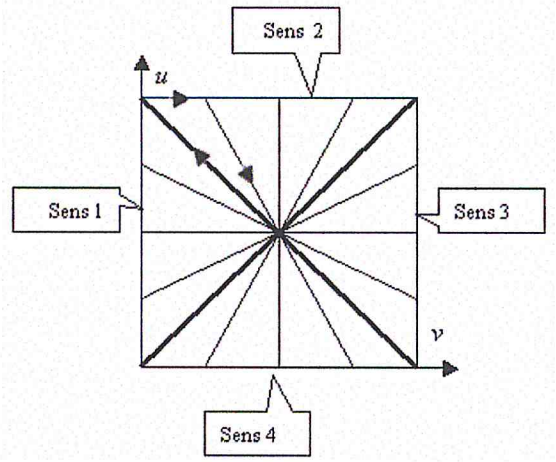


Figure 3.15: Le trajet radial.

En plus de ces dernières methodes, nous pouvons utiliser d'autres méthodes d'usinage.

V.2. Par plan parallèle au plan (x, z) :

Ce type de trajectoire est obtenu par la construction de courbes obtenues par intersection de la surface à usiner avec des plans parallèles au plan (XZ). De même, cette intersection peut être faite avec n'importe quel plan vertical avec la surface à usiner (voir figure 3.16).

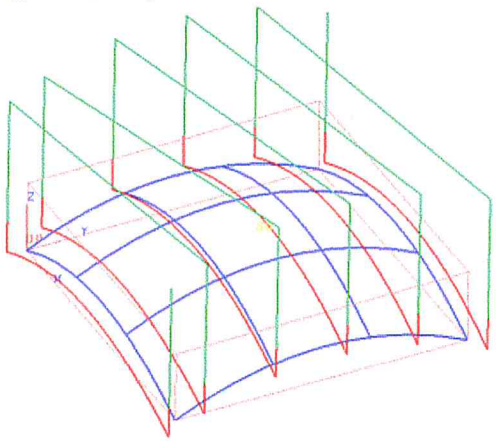


Figure3.16 : Découpage par plan parallèle au plan (x, z).

V.3. Par courbe de niveau (Z constant) :

Ce type de trajectoire est obtenu par intersection de la surface à usiner avec des plans horizontaux avec différentes valeurs de z. Les courbes obtenues sont appelés courbes de niveau (voir figure 3.17).

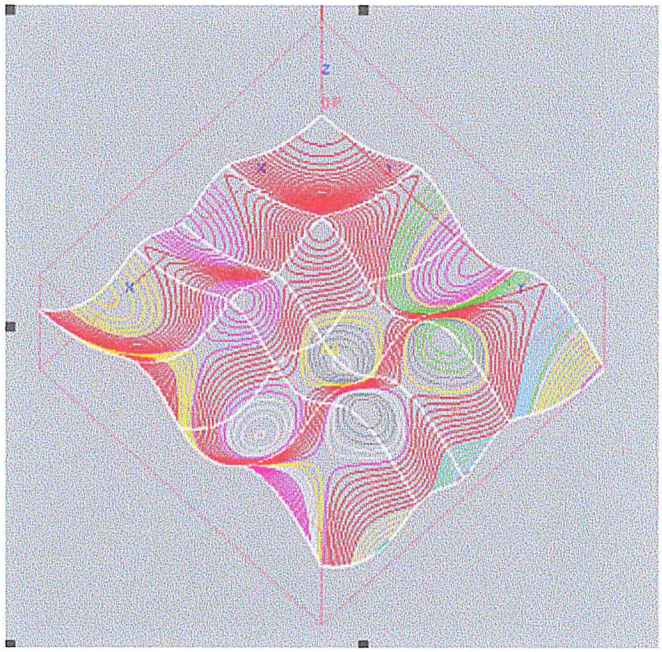


Figure3.17 : Une surface à usiner avec des plans horizontaux

VI. CONCLUSION :

Dans ce chapitre on a présenté les différentes méthodes d'usinage en finition des surfaces gauches, les paramètres d'usinage et les différentes méthodes de positionnement de l'outil. Dans le chapitre suivant on va détailler la méthode d'usinage Z-Constant.

On appelle :

- 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 $\vec{n}$, la normale à la surface au point de contact,
- le vecteur $\vec{u}$, l'axe de l'outil,
- r le rayon de l'outil hémisphérique et le petit rayon de l'outil torique,
- R le grand rayon de l'outil torique.

Les positions des points de l'outil sont données par les relations suivantes :

Pour un outil hémisphérique :

$$\vec{OC_E} = \vec{OC_C} + r\vec{n} \tag{3.3}$$

$$\vec{OC_L} = \vec{OC_E} - r\vec{u} = \vec{OC_E} = \vec{OC_C} + r\vec{n} - r\vec{u} \tag{3.4}$$

Pour un outil torique :

$$\vec{k} = \frac{\vec{u} \wedge \vec{n}}{\left\| \vec{u} \wedge \vec{n} \right\|} \tag{3.5}$$

$$\vec{OC_E} = \vec{OC_C} + r\vec{n} + R\frac{\vec{k} \wedge \vec{u}}{\left\| \vec{k} \wedge \vec{u} \right\|} \tag{3.6}$$

$$\vec{OC_L} = \vec{OC_E} - r\vec{u} = \vec{OC_E} = \vec{OC_C} + r\vec{n} + R\frac{\vec{k} \wedge \vec{u}}{\left\| \vec{k} \wedge \vec{u} \right\|} - r\vec{u} \tag{3.7}$$

Les équations précédentes permettent de calculer soit les coordonnées du centre de l'outil à partir du point de contact, soit le point de contact à partir des coordonnées du centre. Donc, nous avons deux méthodes de posage de l'outil :

- Posage par le point de contact de l'outil: c'est l'opération de calcul de la position du centre de l'outil à partir d'une position de contact donnée;
- Posage par le centre de l'outil: c'est l'opération de calcul direct d'une position du centre de l'outil sans la donnée initiale de la position du point de contact.

IV.1. Méthodes de calcul des positions d'outils :

Plusieurs méthodes sont utilisées pour le calcul des positions d'outils :

IV.1.1. Calcul par offset de la forme :

Cette méthode est basée sur le calcul a priori de la surface offset à la surface à usiner d'une valeur égale à celle du rayon de l'outil. Dans le cas de l'usinage à trois axes avec un outil hémisphérique, la position de l'outil tangente à la surface à usiner appartient à la surface parallèle (ou offset) de la valeur du rayon de l'outil. Le calcul de la position est alors basé sur l'adaptation de la stratégie d'usinage à la surface offset. L'avantage de cette méthode est qu'elle permet d'éviter les interférences locales, par contre son temps de calcul est plus long et les problèmes se posent aux raccordements entre les surfaces offset obtenues. L'offset d'une surface donnée avec une valeur de r est donnée par :

$$\vec{Q}^{off}(u,v) = \vec{Q}(u,v) + r\vec{n} \qquad (3.8)$$

IV.1.2. Calcul par la méthode de plongée (le copiage informatique) :

Le copiage informatique est une méthode non référencée, directement adaptée des méthodes de copiage traditionnelles. C'est une méthode de posage par le centre, implicite, directe ou indirecte, qui élimine les risques d'interférence locale. Lors d'une opération de copiage classique, l'outil suit la surface. Physiquement, de par la valeur de son rayon, il est toujours tangent à la surface. En fait, l'outil tombe sur la surface jusqu'à ce qu'il la touche. Le copiage informatique suit cette stratégie, l'outil glisse le long d'une droite jusqu'à ce qu'il touche la surface (voir figure 3.9).

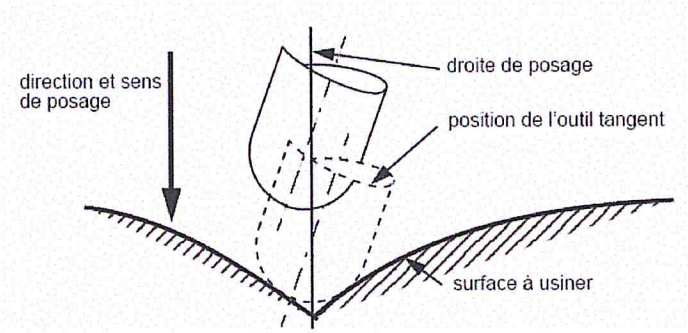


Figure 3.9 : Copiage informatique, guidage de l'outil tangent à la surface.

Les droites de posage sont parallèles à la direction de posage, qui n'est pas forcément l'axe de l'outil. Le calcul du trajet est partagé en deux activités, la première fait suivre à l'outil une courbe plane caractérisée par la stratégie requise, et la seconde positionne verticalement l'outil de façon à le rendre tangent à la surface.

V. METHODES D'USINAGE DES SURFACES GAUCHES EN FINITION [5] :

Différents types de méthodes sont utilisés dans l'usinage des surfaces gauches :

V.1. Méthodes isoparamétriques :

V.1.1. Aller simple (One Way):

Cette méthode consiste à usiner la surface selon l'isoparamétrique u ou v (voir figure 3.10) donnée par l'algorithme suivant :

Chapitre4

Usinage des Surfaces Gauches par la Méthode Z-Constant

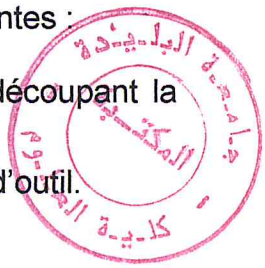
I. INTRODUCTION :

Pour l'usinage des surfaces gauches nous avons le choix entre plusieurs méthodes d'usinage. Mais, chaque méthode a son domaine d'application et le choix d'une méthode dépend de plusieurs facteurs. Dans notre cas, la méthode Z-Constant est choisie dans le cas d'usinage des cavités profondes et pour éviter les longues descentes et montées d'outils et pour les surfaces présentant des régions verticales ou fortement inclinées, tandis que la méthodes des plans parallèles est utilisée dans l'usinage des surfaces ne présentant pas des surfaces avec une forte inclinaison. Dans ce chapitre, nous allons présenter toutes étapes permettant de générer le trajet outil

II. PROCEDE DE GENERATION DU CHEMIN D'OUTIL AVEC LA METHODE Z-CONSTANT :

Le procédé de génération du chemin d'outil lors de l'usinage des surfaces gauches par la méthode Z-Constant passe par les deux étapes suivantes :

- Calcul des contours (éléments de chemin d'outil) en découpant la surface avec les plans horizontaux ;
- Liaison des différents contours pour générer le chemin d'outil.



II.1. Intersection d'un plan et d'une surface libre [14] :

Le problème d'intersection surface-plan peut être résolu de la même manière que l'intersection surface-surface. Généralement, pour calculer les intersections surface-surface deux méthodes sont utilisées :

- Subdivision récursive (triangulation) ;
- Méthodes analytiques.

Lors du calcul de l'intersection entre un plan infini et une surface quelconque, le résultat peut être :

1. Un ensemble vide;
2. Une collection de points;
3. Une collection de courbes lisses;
4. Toutes les combinaisons de (2) et (3).

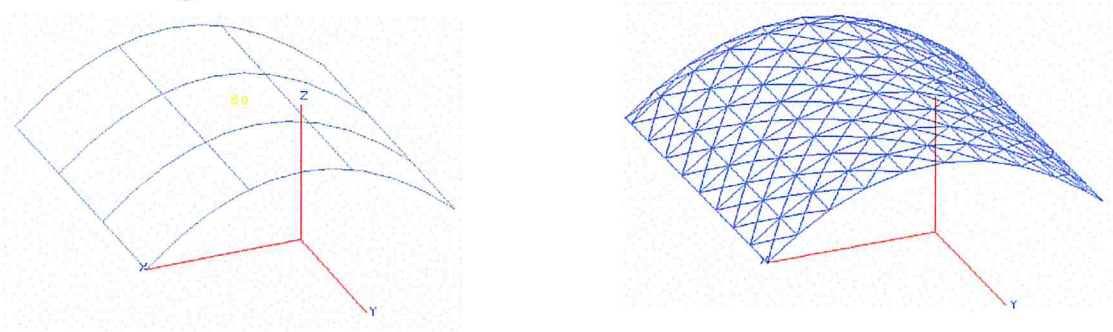
II.1.1. Triangulation :

Pour simplifier les calcul des intersection entre une surface quelconque et un plan infini et pour avoir un système d'équations linéaires, nous devons approximer cette surface avec des éléments géométriques simples soit des quadrilatères soit des triangles. Généralement, la surface est approximée par un ensemble de triangles puisque les calculs avec les triangles sont plus simples et

le triangle permet de représenter un plan. Cette étape d'approximation avec des triangles est appelée « triangulation ». Cette triangulation peut être faite de deux manières différentes :

- Triangulation adaptative : dans ce mode de triangulation, le nombre de triangles est crée d'une manière dynamique et adaptative à partir des critères de précision fixés par l'utilisateur.
- Triangulation uniforme : pour ce mode de triangulation, pour chaque surface sont fixées les nombres de lignes et de colonnes une fois pour toute.

La figure (4.1) montre la surface nominale théorique et son approximation avec des triangles.



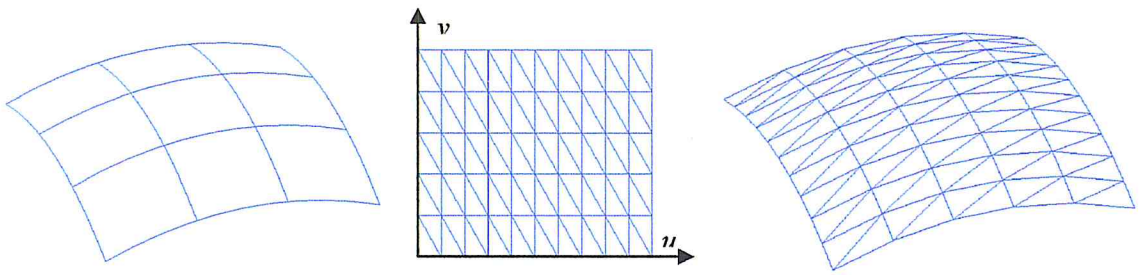
a. Surface nominale. *b. Surface triangulée.*

Figure 4.1 : Triangulation d'une surface nominale.

II.1.1.1. Triangulation uniforme :

Dans cette triangulation le nombre de triangles, pour chaque surface, sont fixés suivant la direction u et la direction v dans le plan paramétrique (u,v), ensuite cette information est utilisée pour faire la triangulation dans l'espace 3D. Avec cette méthode de subdivision, la triangulation est rapide, mais pour avoir une bonne approximation il faut choisir un nombre important de triangles dans chaque direction, ce qui augmente l'espace mémoire nécessaire au stockage des différentes informations de chaque triangle et augmente le temps de calcul des intersections.

La figure (4.2) montre une surface nominale et sa triangulation avec un nombre de triangles égal à 10 et à 5 suivant les directions u et v respectivement.



a. Surface nominale. *b. Paramètres de subdivision* *c. Surface triangulée.*

Figure 4.2 : Triangulation uniforme d'une surface nominale.

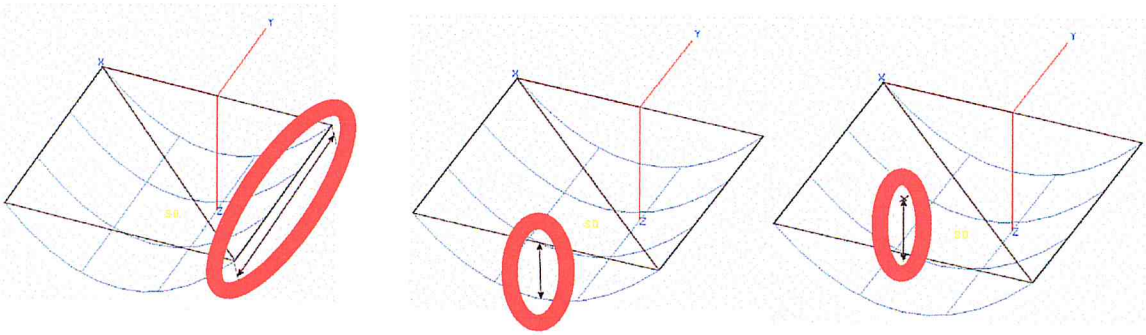
II.1.1.2. Triangulation adaptative [15] :

Généralement, pour approximer une surface de forme libre, c'est la triangulation adaptative qui est utilisée puisqu'elle permet de réduire le nombre de triangles générés et permet de réduire le temps de calcul des intersections. De plus, elle s'adapte mieux aux variations géométriques de la forme de la surface.

La triangulation adaptative nécessite la spécification des critères d'approximation de la surface afin que l'ensemble des triangles donne une bonne représentation de la surface théorique. Les conditions à respecter pour chaque triangle sont les suivantes :

- La longueur maximale de chaque coté du triangle doit être inférieure à une tolérance « d » spécifiée par l'utilisateur. Donc, trois vérifications sont nécessaires (voir figure 4.3.a).
- La distance entre le milieu de chaque coté du triangle et la surface nominale doit être inférieure à une tolérance « e » spécifiée par l'utilisateur. Donc, trois vérifications sont nécessaires (voir figure 4.3.b).
- La distance entre le centre de gravité du triangle et la surface nominale doit être inférieure à une tolérance « e » spécifiée par l'utilisateur. Donc, une vérification est nécessaire (voir figure 4.3.c).

Donc, pour trianguler une surface donnée, nous avons sept conditions à vérifier pour chaque triangle généré. Si une des conditions n'est pas vérifiée, le triangle est subdivisé plusieurs fois jusqu'à vérification de toutes les conditions.



a. 1ère condition. *b. 2ème condition.* *c. 3ème condition.*

Figure 4.3 : Conditions de subdivision d'un triangle.

Le triangle est subdivisé dans les cas suivants (voir figure 4.4):

- Si un coté ne vérifie pas une condition (« d » ou « e »), le triangle est subdivisé en deux triangles (voir figure 4.4.a).
- Si deux cotés ne vérifient pas une condition (« d » ou « e »), le triangle est subdivisé en trois triangles (voir figure 4.4.b).

- Si trois cotés ne vérifient pas une condition (« d » ou « e »), le triangle est subdivisé en quatre triangles (voir figure 4.4.c).
- Si la distance entre le centre de gravité du triangle et la surface est plus grande que « e », le triangle est subdivisé en quatre triangles (voir figure 4.4.d).

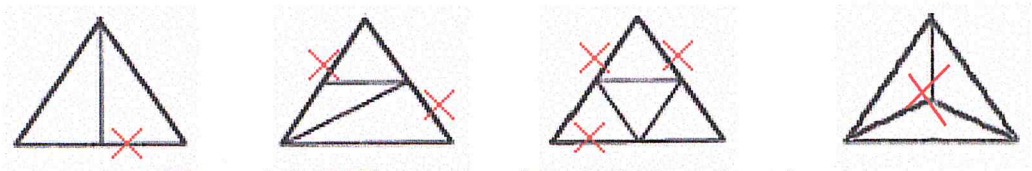
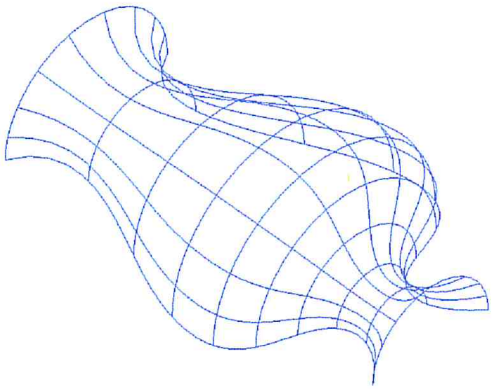


Figure 4.4 : Différents cas de subdivision d'un triangle.

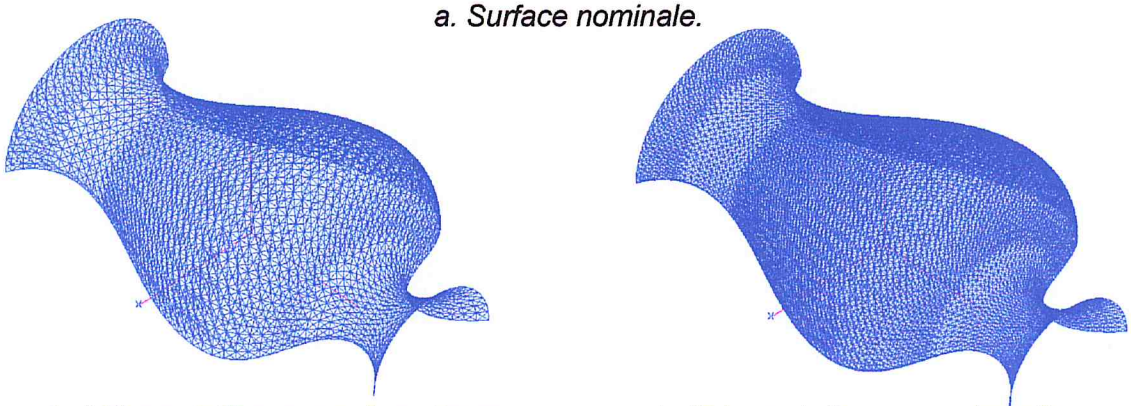
A la fin de la triangulation, nous obtenons un ensemble qui donne une bonne approximation de la surface où tous les sommets des triangles appartiennent à la surface nominale. Les triangles générés seront utilisés dans l'étape de calcul de l'intersection.

La figure (4.5) montre une surface nominale et sa triangulation adaptative pour deux cas :

- d = e = 5mm,
- d = e = 3mm.



a. Surface nominale.



b. Triangulation avec d=e=5mm. *b. Triangulation avec d=e=3mm.*

Figure 4.5 : Triangulation adaptative d'une surface nominale.

II.2. Intersection d'un triangle et un plan dans l'espace tridimensionnelle [16] :

Pour qu'un plan (π) intersecte un triangle, il faut et il suffit qu'il intersecte au moins un segment du triangle. Pour que l'intersection entre un plan et un triangle soit un ensemble vide, il faut qu'il existe deux segments de triangle tels que l'intersection entre chacun d'eux et le plan soit un ensemble vide. La figure (4.6) illustre les différents résultats de l'intersection entre un plan et un triangle.

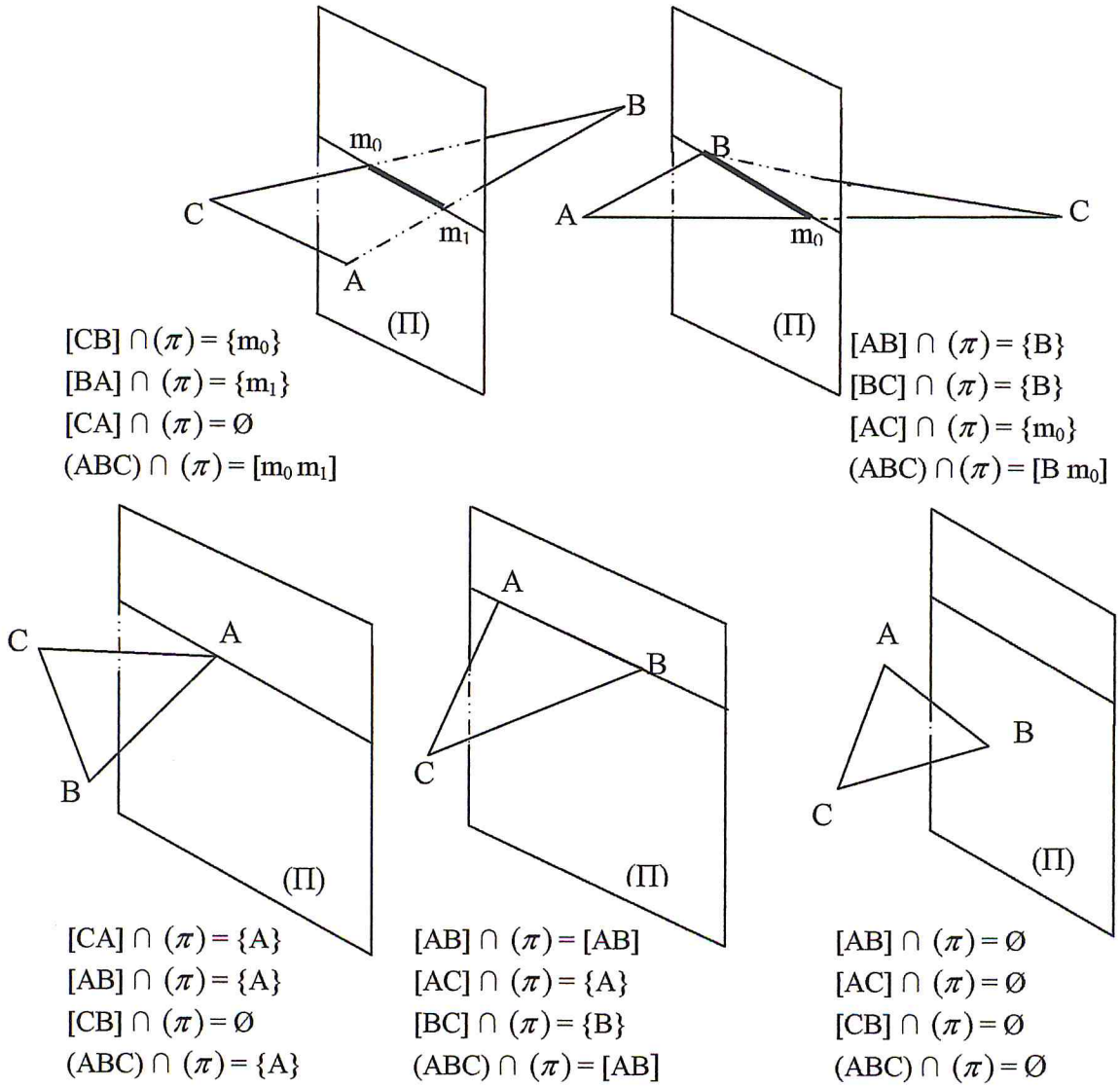


Figure 4.6 : Différents cas d'intersection entre un triangle et un plan.

Soit $P_0P_1P_2$ un triangle telle que $P_0(x_0, y_0, z_0)$, $P1(x_1, y_1, z_1)$ et $P_2(x_2, y_2, z_2)$, et (π) un plan dont l'équation est :

$$Ax+By+Cz+d=0 \qquad (4.1)$$

Le calcul de l'intersection entre le triangle $P_0P_1P_2$ et le plan (π), revient à trouver l'intersection entre le coté $[P_0P_1]$ et (π), le coté $[P_0P_2]$ et (π), le coté $[P_1P_2]$ et (π).

Pour calculer l'intersection entre le coté $[P_0P_1]$ et le plan (π), il faut trouver le système d'équations qui définie la droite (P_0P_1). Supposons que le système d'équations est :

$$\begin{cases} A'x + B'y + C'z + d' = 0 \\ A''x + B''y + C''z + d'' = 0 \end{cases} \quad (4.2)$$

La détermination des coordonnées du point d'intersection entre la droite (P_0P) avec P est un point sur la droite (P_0P_1) et le plan (π) passe par la résolution du système d'équations linéaires suivant :

$$\begin{cases} A\,x + B\,y + C\,z + d = 0 \\ A'\,x + B'\,y + C'\,z + d' = 0 \\ A''\,x + B''\,y + C''\,z + d'' = 0 \end{cases} \quad (4.3)$$

Si le système n'admet pas de solution, donc $(P_0P) \cap (\pi) = \emptyset$, d'où $[P_0P_1] \cap (\pi) = \emptyset$,

- Si le système admet une infinité de solution, donc $(P_0P) \cap (\pi) = (P_0P)$, d'où $[P_0P_1] \cap (\pi) = [P_0P_1]$,
- Si le système admet une solution unique, alors $(P_0P) \cap (\pi) = \{m\}$, dont on vérifie si $\{m\} \in [P_0P_1]$. Si $m \in [P_0P_1]$ alors $[P_0P_1] \cap (\pi) = \{m\}$, sinon $[P_0P_1] \cap (\pi) = \emptyset$.

II.3. La droite dans l'espace [16]:

Par deux points P_0 et P1 dans l'espace $\Re^3$, on ne peut faire passer qu'une seule droite. La droite qui passe par P_0 et P_1 est définie comme l'ensemble des points M qui vérifient la relation suivante (voir figure 4.7) :

$$\overrightarrow{P_0M} \,//\, \overrightarrow{P_0P_1}$$

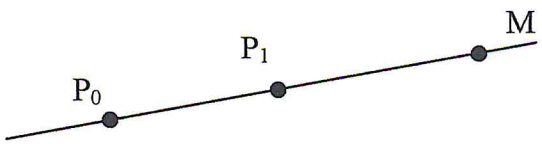


Figure 4.7. Droite dans l'espace.

Soient $P_0(x_0, y_0, z_0)$, $P1(x_1, y_1, z_1)$ deux points de l'espace, soit M(x, y, z) un point de la droite (Δ_M) qui passe par les deux point P_0 et P_1.

$$M \in (\Delta_M) \Leftrightarrow \overrightarrow{P_0M} \,//\, \overrightarrow{P_0P_1} \Leftrightarrow \exists\ \alpha \in \Re \,/\, \overrightarrow{P_0M} = \alpha \,.\, \overrightarrow{P_0P_1}\,.$$

$$\Leftrightarrow \exists\ \alpha \in \Re \,/ \begin{vmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{vmatrix} = \alpha \,.\, \begin{vmatrix} x_1 - x_0 \\ y_1 - y_0 \\ z_1 - z_0 \end{vmatrix} \quad (4.4)$$

Donc :

$$\begin{cases} x - x_0 = \alpha(x_1 - x_0) \\ y - y_0 = \alpha(y_1 - y_0) \\ z - z_0 = \alpha(z_1 - z_0) \end{cases} \Leftrightarrow \begin{cases} \alpha = \dfrac{x - x_0}{x_1 - x_0} & si \quad x_1 \neq x_0 \\ \alpha = \dfrac{y - y_0}{y_1 - y_0} & si \quad y_1 \neq y_0 \\ \alpha = \dfrac{z - z_0}{z_1 - z_0} & si \quad z_1 \neq z_0 \end{cases} \tag{4.5}$$

D'où on peut écrire :

$$\frac{x - x_0}{x_1 - x_0} = \frac{y - y_0}{y_1 - y_0} = \frac{z - z_0}{z_1 - z_0} \tag{4.6}$$

De la relation (4.6), on peut définir un système de trois équations linéaires :

$$(I)\begin{cases} (x - x_0)(y_1 - y_0) = (x_1 - x_0)(y - y_0)...(1') \\ (y - y_0)(z_1 - z_0) = (y_1 - y_0)(z - z_0)...(2') \\ (x - x_0)(z_1 - z_0) = (x_1 - x_0)(z - z_0)....(3') \end{cases} \tag{4.7}$$

Dans ce système d'équations, on peut remarquer que chaque équation peut être obtenue par les deux autres équations. Donc on peut éliminer une des équations, cette élimination ce fait selon les cas suivants :

1ercas: si $x_1 \neq x_0$, $y_1 \neq y_0$, $z_1 \neq z_0$:

Dans ce cas, on peut éliminer n'importe quelle équation. En éliminant l'équation (2') on obtient le système :

$$\begin{cases} (y_1 - y_0)x - (x_1 - x_0)y - (y_1 - y_0)x_0 + (x_1 - x_0)y_0 = 0 \\ (z_1 - z_0)x - (x_1 - x_0)z - (z_1 - z_0)x_0 + (x_1 - x_0)z_0 = 0 \end{cases} \tag{4.8}$$

2emecas: si $x_1 \neq x_0$, $y_1 \neq y_0$, $z_1 = z_0$:

Remplaçons dans le système (I) z_1-z_0 par 0 on obtient :

$$\begin{cases} (x - x_0).(y_1 - y_0) = (x_1 - x_0).(y - y_0) \\ (y - y_0).0 = (y_1 - y_0).(z - z_0) \\ (x - x_0).0 = (x_1 - x_0).(z - z_0) \end{cases} \Leftrightarrow \begin{cases} (x - x_0).(y_1 - y_0) = (x_1 - x_0).(y - y_0) \\ (y_1 - y_0).(z - z_0) = 0 \\ (x_1 - x_0).(z - z_0) = 0 \end{cases} \tag{4.9}$$

On sait que x_1- $x_0 \neq 0$ et y_1-$y_0 \neq 0$, d'où le système devient :

$$\begin{cases} (x - x_0)(y1 - y_0) = (x_1 - x_0)(y - y_0) \\ z - z_0 = 0 \\ z - z_0 = 0 \end{cases} \tag{4.10}$$

On remarque que la troisième équation n'est rien d'autre que la deuxième équation, donc on élimine la troisième équation et le système équivalent est donné par :

$$\begin{cases} x(y_1 - y_0) - y(x_1 - x_0) - x_0(y_1 - y_0) + y_0(x_1 - x_0) = 0 \\ z - z_0 = 0 \end{cases} \quad (4.11)$$

3eme cas: si $x_1 \neq x_0$, $z_1 \neq z_0$, $y_1 = y_0$:

Par les mêmes étapes, on peut trouver le système suivant :

$$\begin{cases} x(y_1 - y_0) - y(x_1 - x_0) - x_0(y_1 - y_0) + y_0(x_1 - x_0) = 0 \\ z - z_0 = 0 \end{cases} \quad (4.12)$$

4eme cas: si $x_1 \neq x_0$, $y_1 = y_0$, $z_1 = z_0$:

$$\begin{cases} y - y_0 = 0 \\ z - z_0 = 0 \end{cases} \quad (4.13)$$

Cette droite est parallèle à l'axe (OX).

5eme cas: si $x_1 = x_0$, $y_1 \neq y_0$, $z_1 \neq z_0$:

$$\begin{cases} y(z_1 - z_0) - z(y_1 - y_0) - y_0(z_1 - z_0) + z_0(y_1 - y_0) = 0 \\ x - x_0 = 0 \end{cases} \quad (4.14)$$

6eme cas: si $x_1 = x_0$, $y_1 \neq y_0$, $z_1 = z_0$:

$$\begin{cases} x - x_0 = 0 \\ z - z_0 = 0 \end{cases} \quad (4.15)$$

Cette droite est parallèle à l'axe (OY).

7eme cas: si $x_1 = x_0$, $y_1 = y_0$, $z_1 \neq z_0$:

$$\begin{cases} x - x_0 = 0 \\ y - y_0 = 0 \end{cases} \quad (4.16)$$

Cette droite est parallèle à l'axe (OZ).

II.4. Intersection entre un plan et une droite dans l'espace [16]:

L'intersection entre un plan et une droite dans l'espace donne les résultas suivants:

- Ensemble vide,
- La droite elle même,
- Un point.

Pour déterminer l'ensemble d'intersection entre un plan et une droite, il suffit de résoudre le système d'équations linéaires dont la forme générale est :

$$(II)\begin{cases} Ax+By+Cz+d=0 \\ A'x+B'y+C'z+d'=0 \\ A''x+B''y+C''z+d''=0 \end{cases} \qquad (4.17)$$

Avec :

- L'équation du plan est donné par : Ax+By+Cz+d=0.
- L'équation de la droite est définie par le système suivant :

$$\begin{cases} A'x+B'y+C'z+d'=0 \\ A''x+B''y+C''z+d''=0 \end{cases} \qquad (4.18)$$

On remarque le système (II) est un système linéaire a trois inconnues x, y, z et à trois équations, qui peut être résolue par la méthode Gauss ou toute autre méthode de résolution. Donc, la triangulation permet de rendre le problème d'intersection entre une surface de forme quelconque et un plan, un problème linéaire.

II.5. Appartenance d'un point à un segment de droite [16] :

Soit [AB] un segment de droite dans l'espace et M un point.

$M \in [AB] \Leftrightarrow \exists \alpha \,/\, 0 \le \alpha \le 1, \overrightarrow{AM} = \alpha.\overrightarrow{AB}$

La figure (4.8) montre les positions de trois points par rapport au segment de droite [AB].

$$\begin{cases} M_1 \in [AB] & et \quad 0 \le \alpha \le 1 \\ M_2 \notin [AB] & car \quad \alpha \ge 1 \\ M_3 \notin [AB] & car \quad \alpha < 0 \end{cases} \qquad (4.19)$$

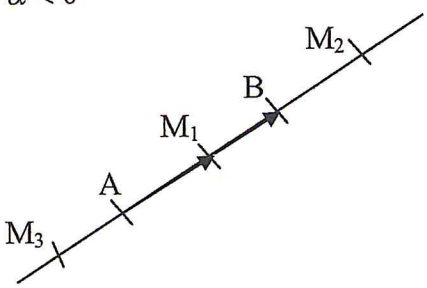


Figure 4.8. Position des différents points par rapport à AB.

De la relation précédente, on peut déduire les relations suivantes :

$$\overrightarrow{AM} = \alpha.\overrightarrow{AB} \quad 0 \leq \alpha \leq 1$$

On pose $A(x_A, y_A, z_A)$, $B(x_B, y_B, z_B)$ et $M(x, y, z)$,donc on a :

$$\begin{pmatrix} x - x_A \\ y - y_A \\ z - z_A \end{pmatrix} = \alpha. \begin{pmatrix} x_B - x_A \\ y_B - y_A \\ z_B - z_A \end{pmatrix} \quad et \quad 0 \leq \alpha \leq 1$$

Donc :

$$\frac{x - x_A}{x_B - x_A} = \frac{y - y_A}{y_B - y_A} = \frac{z - z_A}{z_B - z_A} = \alpha \tag{4.20}$$

Donc, il suffit que :

$$0 \leq \frac{x - x_A}{x_B - x_A} \leq 1 \text{ ou } 0 \leq \frac{y - y_A}{y_B - y_A} \leq 1 \text{ ou } 0 \leq \frac{z - z_A}{z_B - z_A} \leq 1 \tag{4.21}$$

III. ETAPES D'USINAGE AVEC LA METHODE Z-CONSTANT [12] :

L'usinage avec la méthode Z-Constant consiste à calculer les trajets d'usinage sur des plans horizontaux par la recherche des intersections entre la surface nominale et des plans horizontaux avec différentes valeurs de Z décroissantes.

La génération du trajet d'usinage lors de l'utilisation de la méthode Z-Constant nécessite le passage par les étapes suivantes :

- Triangulation des surfaces à usiner (uniforme ou adaptative),
- Calcul des segments d'intersection des triangles et des plans horizontaux pour différentes valeurs de Z,
- Fusion des segments de chaque plan et construction des différents contours,
- Calcul du trajet d'usinage sur chaque plan,
- Construction du trajet d'usinage total.

VI. CONDITIONS TECHNOLOGIQUES LORS DE L'USINAGE AVEC LA METHODE Z-CONSTANT [17] :

Lors de l'usinage avec la méthode Z-Constant, on doit tenir compte des contraintes technologiques suivantes :

1-Le chemin d'outil doit être généré en considérant les contraintes d'usinage telles que les efforts de coupe et les collisions. S'il y a deux trajets d'usinage sur la même paroi, alors on doit commencer l'usinage par le trajet le plus haut C_H et ensuite passer au trajet le plus bas C_L (voir figure 4.9). Commencer l'usinage par le trajet le plus bas sans usiner le trajet le plus haut peut générer des efforts de coupe très importantes ou des collisions entre l'outil et la pièce. Donc, lors de l'usinage, nous devons toujours commencer par les trajets d'usinage les plus hauts.

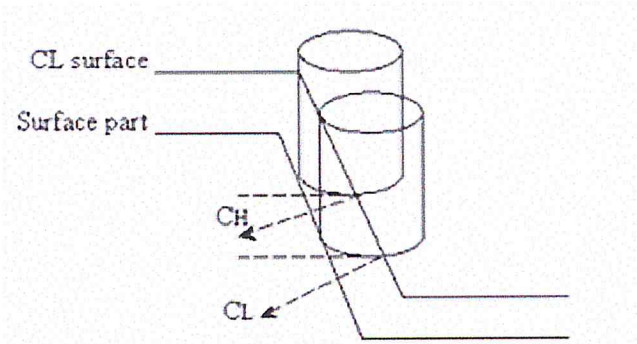


Figure 4.9 : Contraintes d'usinage entre les contours.

2-La longueur totale du trajet d'usinage doit être minimale :

Pour l'efficacité d'usinage, il est souhaitable de réduire au maximum la longueur du trajet d'usinage. Cette réduction permet de réduire considérablement les temps d'usinage. Pour éviter les déplacements de l'outil à vide, on doit lier les trajets d'usinage les plus proches.

3-Des options de fraisage en One-Way et Zig-Zag doivent être possibles :

Le choix de la méthode de balayage de l'outil en One-Way ou en Zig-Zag doit être fait en compromis entre la qualité de la surface usinée (état de surface) et la productivité (temps d'usinage). Dans la pratique industrielle, One-Way est employé pour usiner des aciers, alors que Zig-Zag est utilisée dans l'usinage de matériaux non dur.

V. CONCLUSION :

Dans ce chapitre on a présenté la méthode d'usinage Z-Constant, les différentes méthodes de triangulation des surfaces et de calcul des intersections entre un triangle et un plan.

Chapitre5

Analyse et Spécification des Besoins

I. INTRODUCTION :

Dans les chapitres précédents nous avons présenté les différents aspects mécaniques liés à l'usinage des surfaces gauches sur des fraiseuses à commande numérique à 3 axes. Dans ce chapitre, nous allons présenter notre problème et la solution proposée ainsi que la définition des objectifs de notre système

II. PROBLEMATIQUE ET SOLUTION PROPOSEE :

Les pièces de formes libres sont très utilisées dans la conception et la fabrication des moules, matrices, formes aérodynamiques, formes esthétiques, ...etc. Vu la complexité des formes géométriques, ces pièces ne peuvent être usinées que sur des machines outils à commande numérique et le plus souvent sur des fraiseuses à commande numérique à 3 axes et à 5 axes. De ce fait, il est indispensable de générer des programmes d'usinage appelés programmes « G-Code », mais pour ces surfaces, il est impossible d'écrire manuellement des programmes de milliers de lignes d'instructions avec différentes méthodes d'usinage. Généralement, le processus d'usinage de ces surfaces passe par 03 étapes : ébauche, demi-finition et finition. Pour l'opération de finition, nous avons le choix entre plusieurs méthodes d'usinage dont le choix dépend de plusieurs facteurs. Dans le cas de l'usinage des cavités profondes, la méthode la plus adaptée est la méthode de Z-Constant (courbes de niveau).

Dans ce projet on s'intéressera à l'usinage en finition des surfaces de formes libres sur des fraiseuses à commande numérique à 3 axes. Le but de ce travail est le développement d'une application logicielle graphique et interactive sous Windows qui permet en utilisant la méthode d'usinage en Z-Constant :

- De développer des algorithmes d'usinage des surfaces en prenant en compte la forme et les dimensions de l'outil, paramètres d'usinage et le modèle de la surface.
- Par la suite, automatiser le choix des outils utilisés pour l'usinage des surfaces (les rayons d'outils sont théoriques (calculés), modifiés en fonction des rayons disponibles dans la base de données des outils).
- Pour le processus d'usinage, la combinaison optimale des d'outils est celle qui minimise les temps d'usinage.
- Et à la fin l'automatisation de la génération des programmes d'usinage « G-Code » à partir des modèles CAO des surfaces à usiner et des différents paramètres d'usinage.

III. ANALYSE [18,19] :

Pour développer un logiciel, il est nécessaire de passer par plusieurs étapes. Pour cela, des méthodes de développement ont été définies, ces méthodes permettent de mieux organiser, d'avoir une meilleure compréhension, de réduire

la complexité des applications et permettent une plus grande facilité dans l'interprétation des concepts logiciels.

UML est un langage qui permet de représenter des modèles, mais il ne définit pas le processus d'élaboration des modèles. Dans le cadre de la modélisation d'une application informatique, les auteurs d'UML préconisent d'utiliser une démarche de développement. Dans notre cas on utilise un processus de développement «en cascade », ce cycle de vie de logiciel est hérité du bâtiment. Ce modèle repose sur les hypothèses suivantes :

- On ne peut pas construire la toiture avant les fondations,
- Les conséquences d'une modification en amont du cycle ont un impact majeur sur les coûts en aval (on peut imaginer la fabrication d'un moule dans l'industrie du plastique).

Les phases traditionnelles de développement sont effectuées simplement les unes après les autres (figure 5.1), avec un retour sur les précédentes, voire au tout début du cycle.

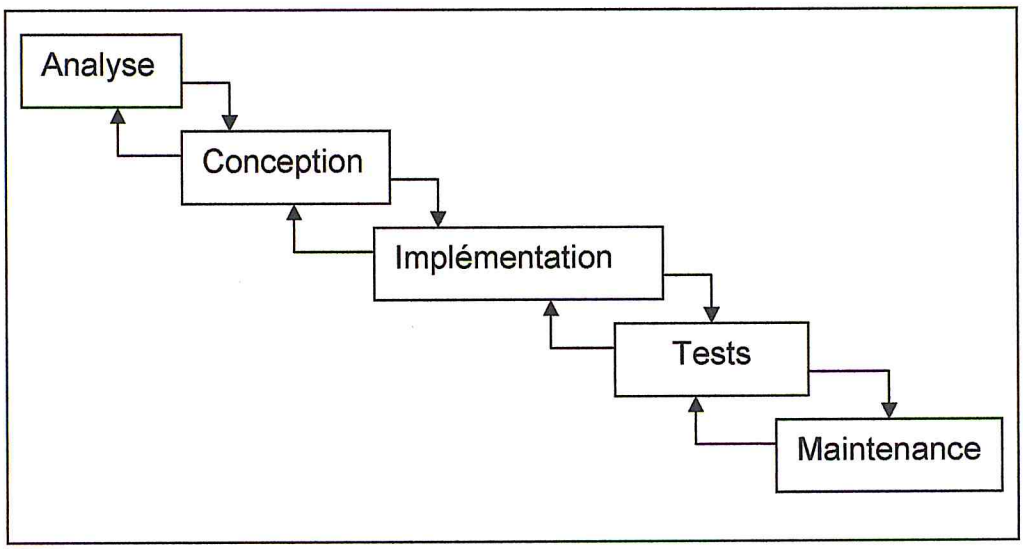


Figure 5.1:Le processus de développement en cascade.

III.1. Définition des besoins :

Les besoins d'un système (cahier des charges) sont souvent exprimés de manière non structurée, sans forte cohérence (imprécision, oublis, contradictions) pour cela on utilise le diagramme de cas d'utilisation d'UML.

III.1.1. Les diagrammes de cas d'utilisation :

Les use cases permettent de structurer les besoins des utilisateurs et les objectifs correspondants d'un système. Ils identifient les utilisateurs du système (acteurs) et leur interaction avec le système.

Les acteurs : l'acteur est une entité externe qui agit sur le système (opérateur, autre système...) et qui peut consulter ou modifier l'état du système.

Notre système dispose d'un seul type d'utilisateur (l'utilisateur de système) qui peut être un mécanicien ou non.

Use cases (cas d'utilisation) : ensemble d'actions réalisées par le système en réponse à une action d'un acteur. Les use cases permettent de structurer les besoins des utilisateurs et les objectifs correspondants d'un système.

Notre système est divisé en deux grands modules (voir figure 5.2):

- Usinage des surfaces gauches par la méthode Z-Constant.
- Le choix des outils.

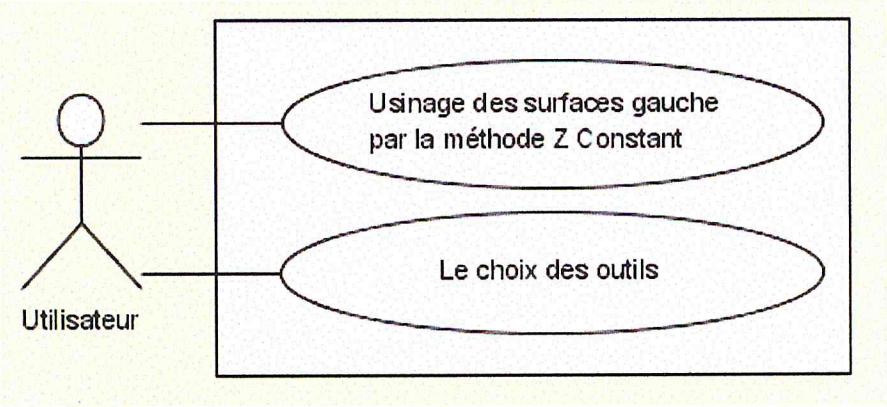


Figure 5.2:Diagramme de cas d'utilisation pour les cas d'utilisation principaux.

Dans ce qui suit, on va présenter les cas d'utilisations correspondants à chaque module :

✓ Usinage des surfaces gauches par la méthode Z-Constant :

- Choix des surfaces à usiner;
- Choix de la méthode d'usinage
- Définition des paramètres d'usinage;
- Lancement de l'usinage;
- Simulation.

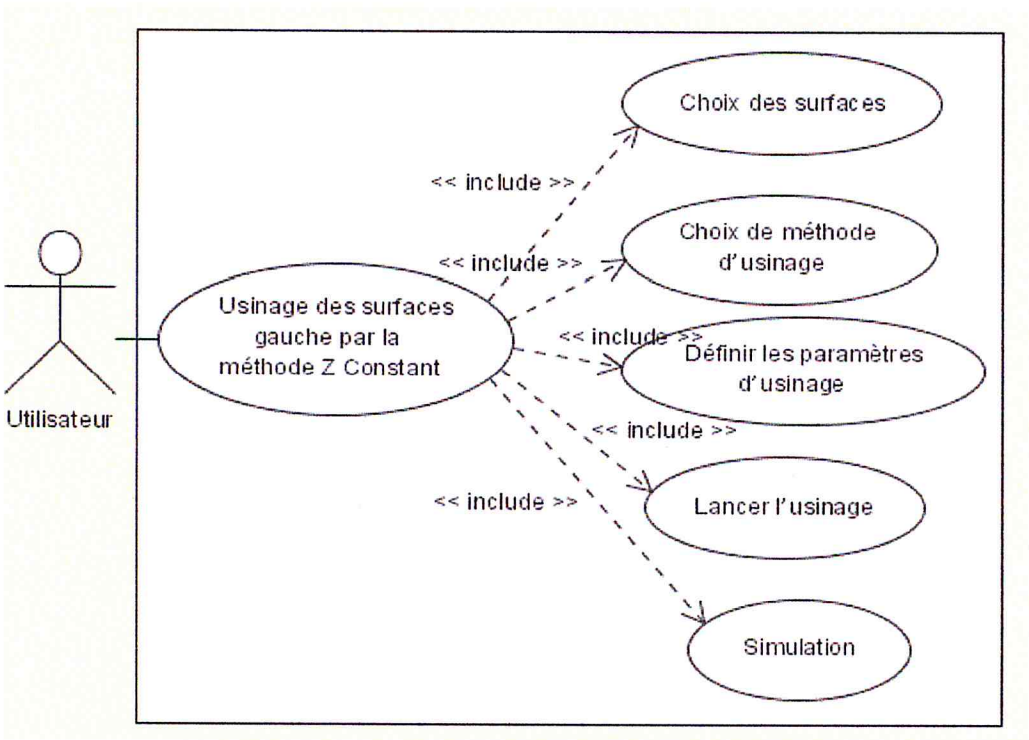


Figure 5.3: Diagramme de cas d'utilisation usinage des surfaces gauches par la méthode Z-Constant.

➢ Simulation :

- Choix de la période de visualisation;
- Choix de mode de simulation;
- Choix de mode de visualisation;
- Lancement de la simulation.

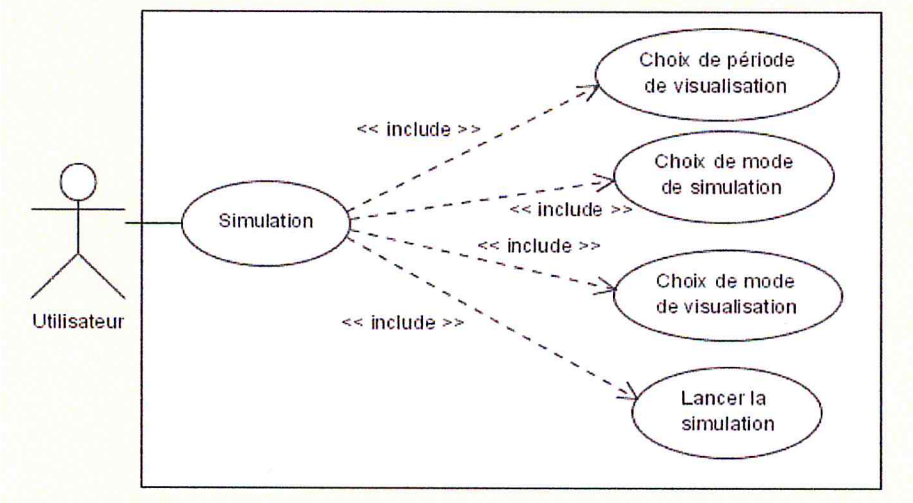


Figure 5.4: Diagramme de cas d'utilisation simulation.

➢ Lancer l'usinage :

- Triangulation;
- Classification;
- Usinage avec Z constant.

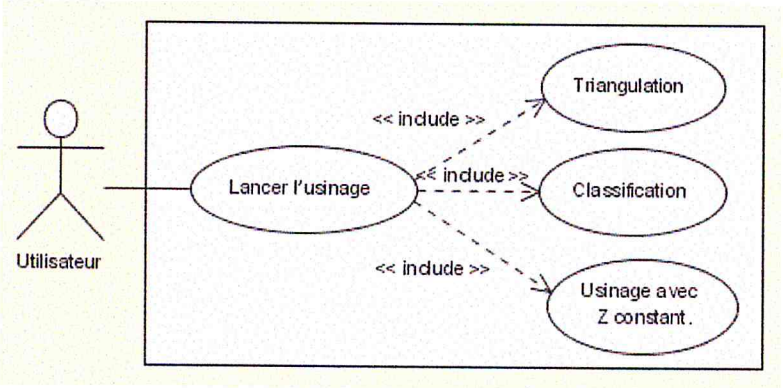


Figure 5.5: Diagramme de cas d'utilisation lancer l'usinage.

♦ Usinage avec Z-Constant :

- Calcul des intersections;
- Création des contours;
- Optimisation de chemin d'outil;
- Sauvegarde des intersections;
- Calcul des positions de contact outil-surface.
- Vérification des interférences locales et globales sur le trajet d'usinage.

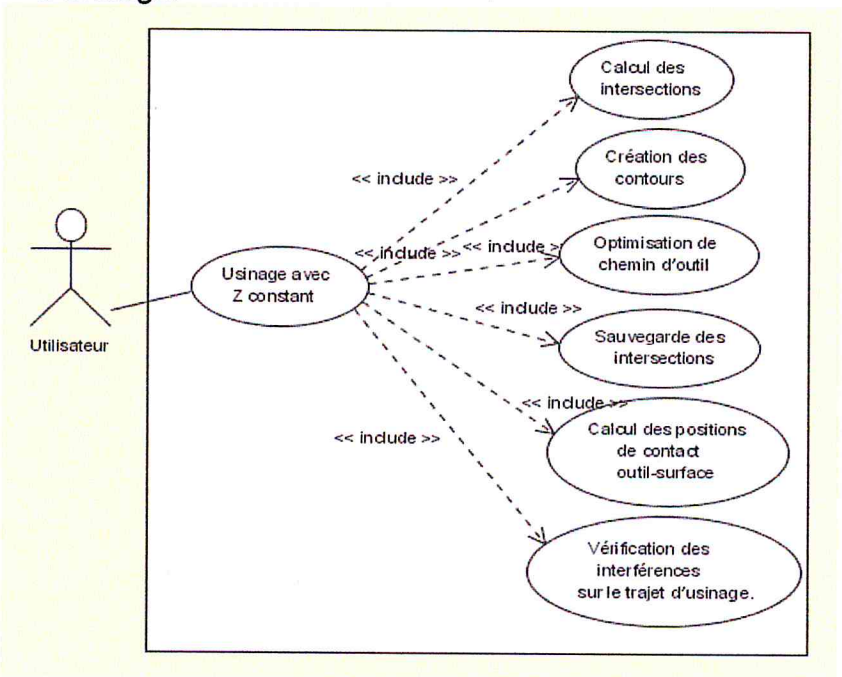


Figure 5.6: Diagramme de cas d'utilisation usinage avec Z constant.

✓ Le choix des outils :

- Calcul des rayons théoriques;
- Choix automatique d'outils à partir de la base de données des outils;
- Optimisation de choix d'outils;
- Gestion de la base de données.

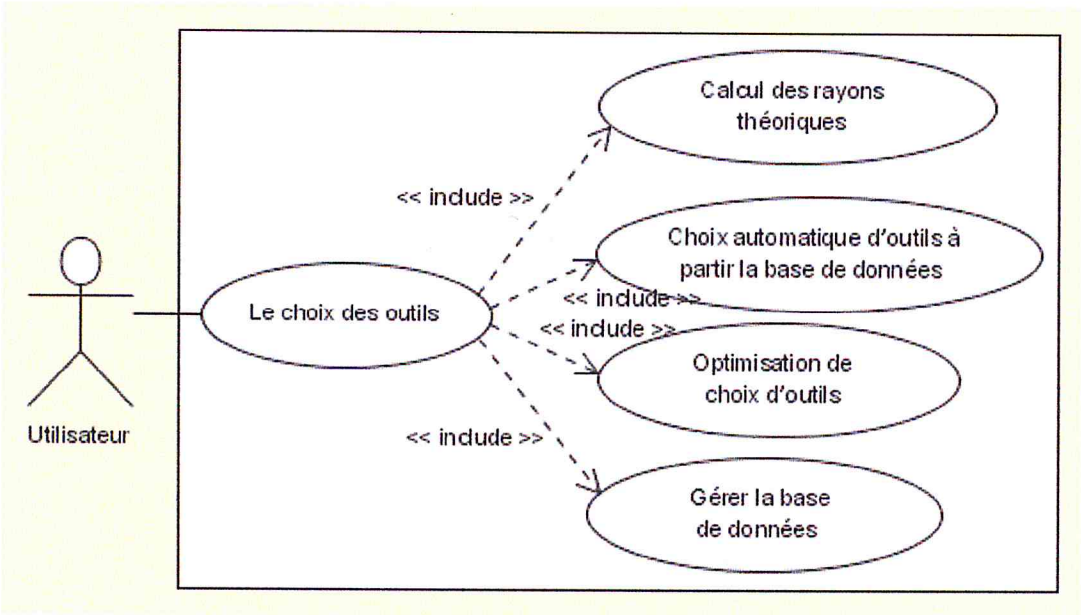


Figure 5.7: Diagramme de cas d'utilisation le choix des outils.

➢ Optimisation de choix d'outils :

- Définir les paramètres d'optimisation;
- Déterminer la combinaison optimale des outils.

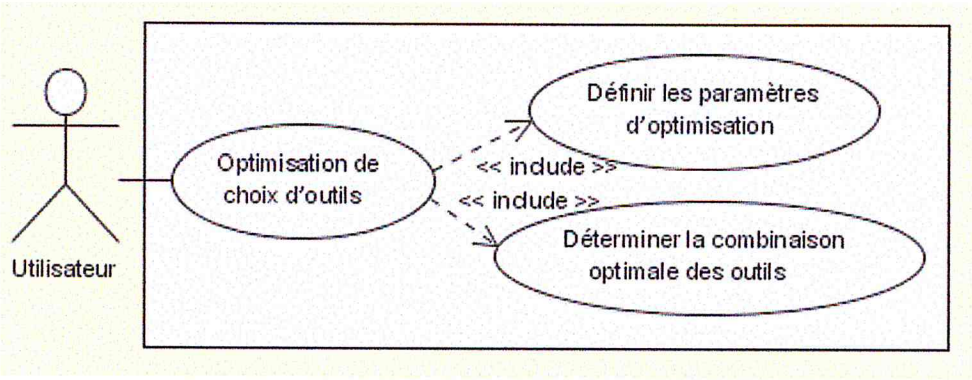


Figure 5.8: Diagramme de cas d'utilisation optimisation de choix d'outils.

➢ Gestion de la base de données :

- Rajouter un nouvel outil à la base de données;
- Supprimer un outil de la base de données;
- Modifier un outil existant dans la base de données.

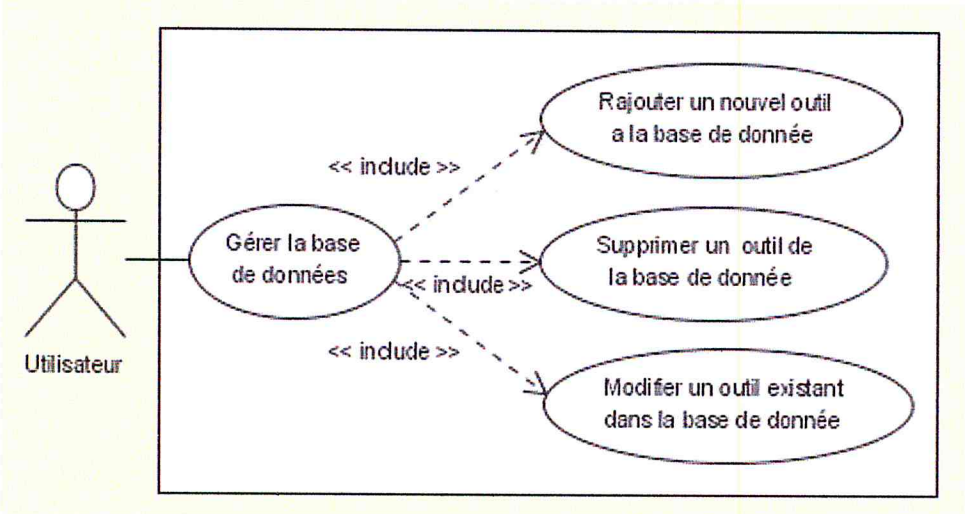


Figure 5.9: Diagramme de cas d'utilisation pour gérer la base de données.

III.1.2. Diagrammes des séquences :

Le diagramme de séquences est le plus apte à modéliser les aspects dynamiques de système en temps réel.

✓ Usinage des surfaces gauches par la méthode Z-Constant.

Pour faire l'usinage d'une ou plusieurs surfaces :

- L'utilisateur demande l'ouverture d'une surface ou plusieurs surfaces;
- Le système ouvre la surface;
- L'utilisateur choisi la méthode d'usinage;
- Le système sélectionne la méthode choisie;
- L'utilisateur demande la sélection d'une surface;
- Le système sélectionne la surface et demande les paramètres d'usinage ;
- L'utilisateur définit les paramètres d'usinage;
- Le système récupère les paramètres d'usinage;
- L'utilisateur lance l'usinage;
- Le système usine la surface ou les surfaces.

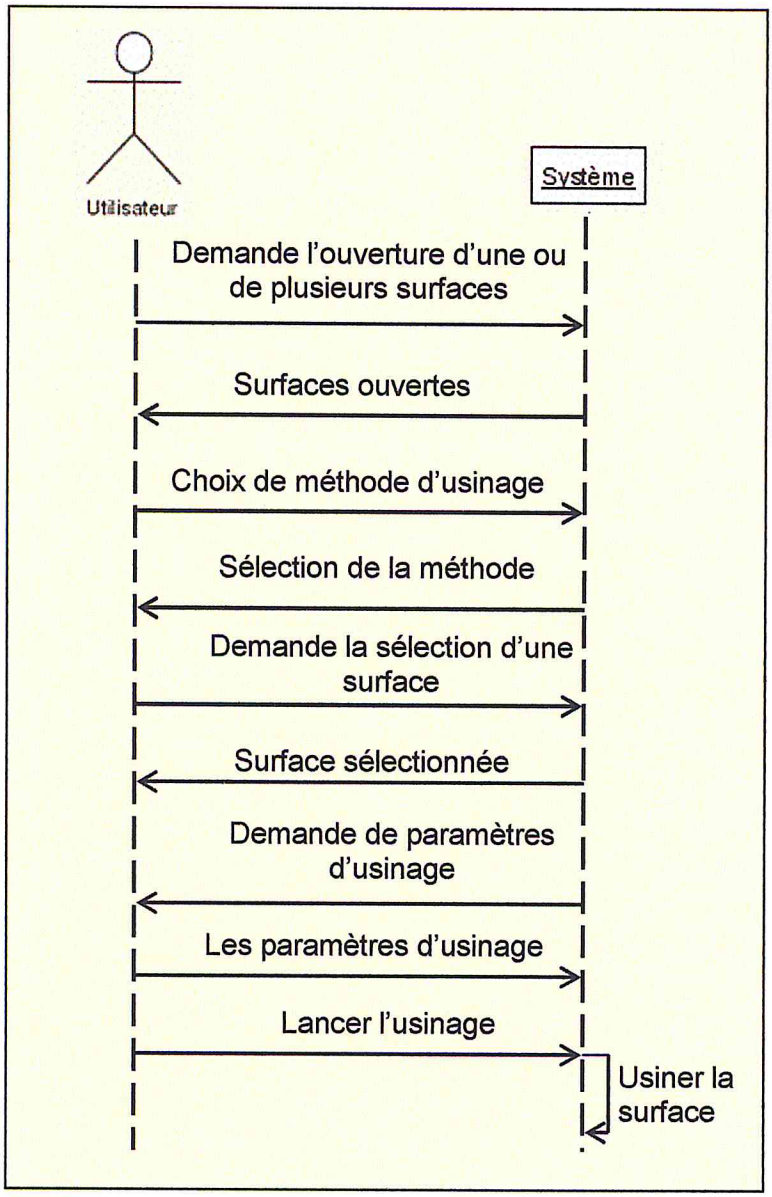


Figure 5.10: Diagramme de séquence pour l'usinage des surfaces gauches par la méthode Z-Constant.

✓ Gestion de la base de données :

- L'utilisateur demande de gérer la base de données;
- Le système demande le choix de type d'attachement d'outils;
- L'utilisateur choisit le type d'attachement;
- Le système affiche la liste des outils;
- L'utilisateur demande la mise à jour d'un outil;
- Le système fait la mise à jour de l'outil dans la base de données;

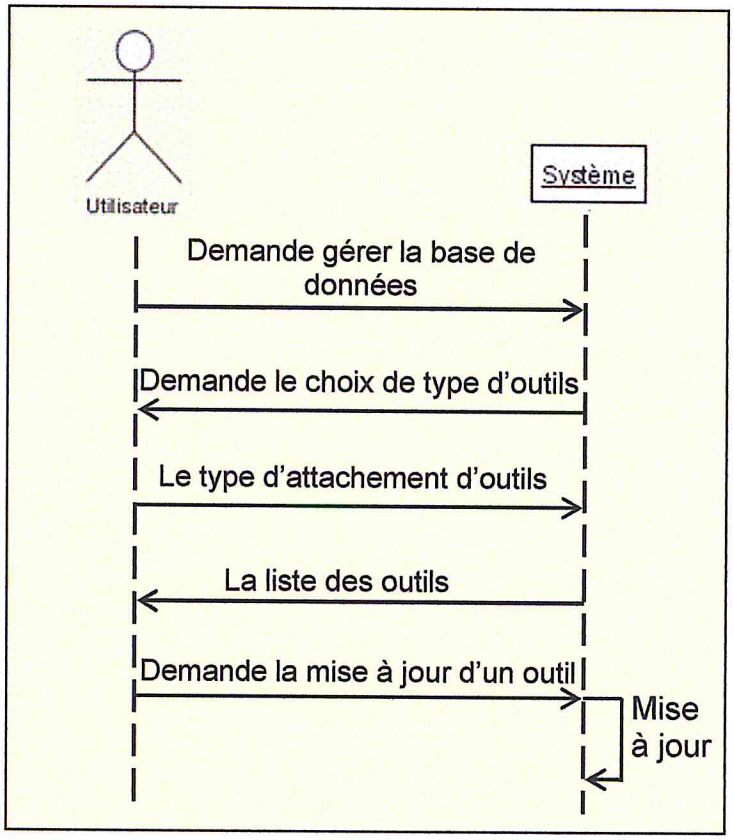


Figure 5.11: Diagramme de séquence pour gérer la base de données.

✓ Suppression d'un outil:

- L'utilisateur sélectionne un outil;
- L'utilisateur demande la suppression de l'outil;
- Le système demande la confirmation de la suppression;
- L'utilisateur donne sa confirmation;
- Le système supprime l'outil de la base de données.

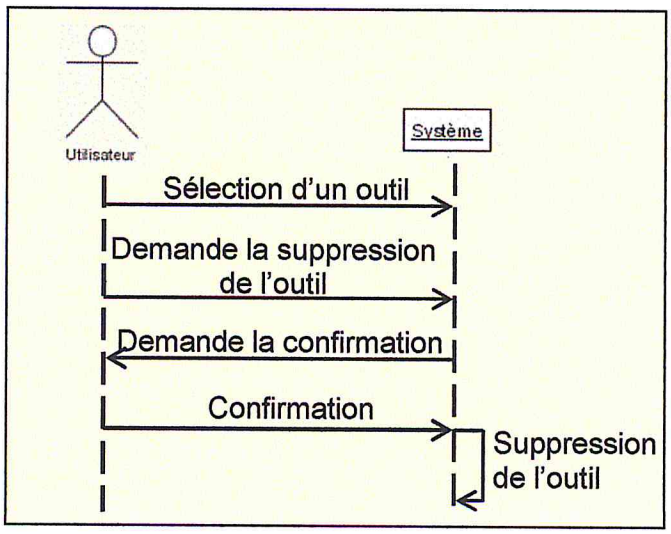


Figure 5.12: Diagramme de séquence pour la suppression d'un outil.

✓ Insertion d'un nouvel outil:

- L'utilisateur donne les informations d'un nouvel outil;
- L'utilisateur demande l'insertion de l'outil;
- Le système vérifier si tous les champs sont remplis et si l'outil existe dans la base de données;
- Le système insère l'outil dans la base de données.

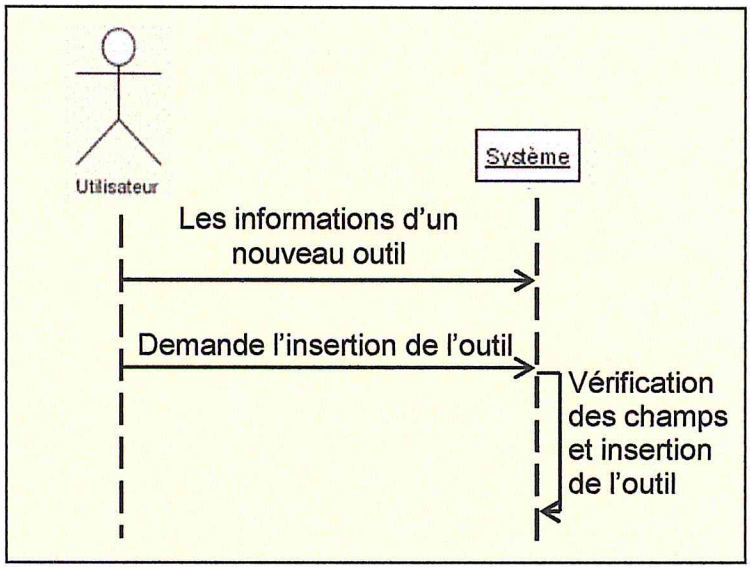


Figure 5.13: Diagramme de séquence pour l'insertion d'un nouvel outil.

✓ La modification d'un outil:

- L'utilisateur sélectionne un outil ;
- L'utilisateur modifie la disponibilité de l'outil ;
- Le système valide la modification de l'outil.

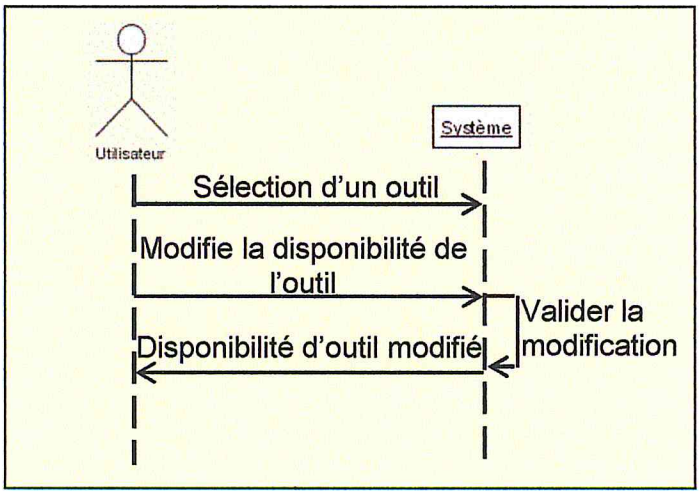


Figure 5.14 : Diagramme de séquence pour la modification d'un outil.

✓ L'optimisation de choix d'outils:
- L'utilisateur demande l'optimisation de choix d'outils;
- Le système demande les paramètres d'optimisation;
- L'utilisateur définir les paramètres d'optimisation;
- Le système fait l'optimisation;
- Le système donne la combinaison optimale des outils.

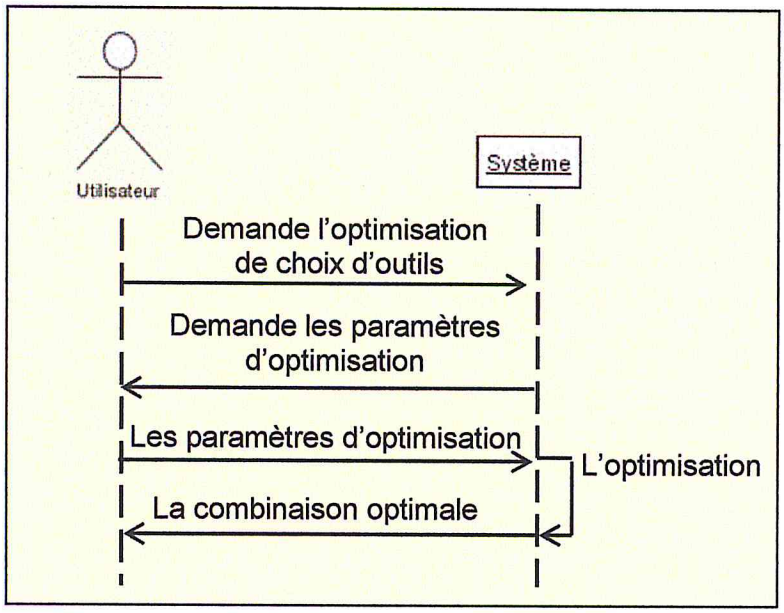


Figure 5.15 Diagramme de séquence pour l'optimisation de choix d'outils.

✓ Calculer les rayons théoriques:
- L'utilisateur demande le choix des outils;
- Le système calcule les rayons des outils;
- Le système affiche les rayons d'outils théoriques;

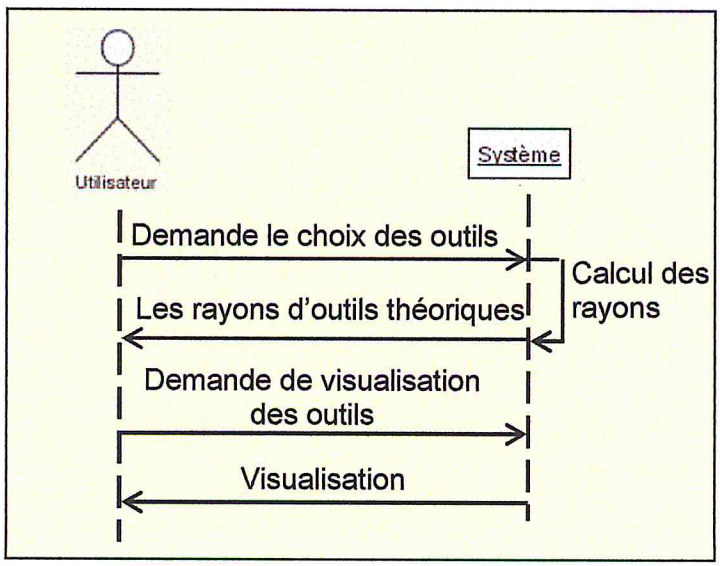


Figure 5.16: Diagramme de séquence pour calculer les rayons théoriques.

✓ Choix automatique d'outils à partir de la base de données :

- L'utilisateur demande le choix des outils à partir de la base de données;
- Le système demande le choix de type d'attachement d'outils;
- L'utilisateur choisit le type d'attachement d'outils;
- Le système modifie les rayons d'outils à partir de la base de données.

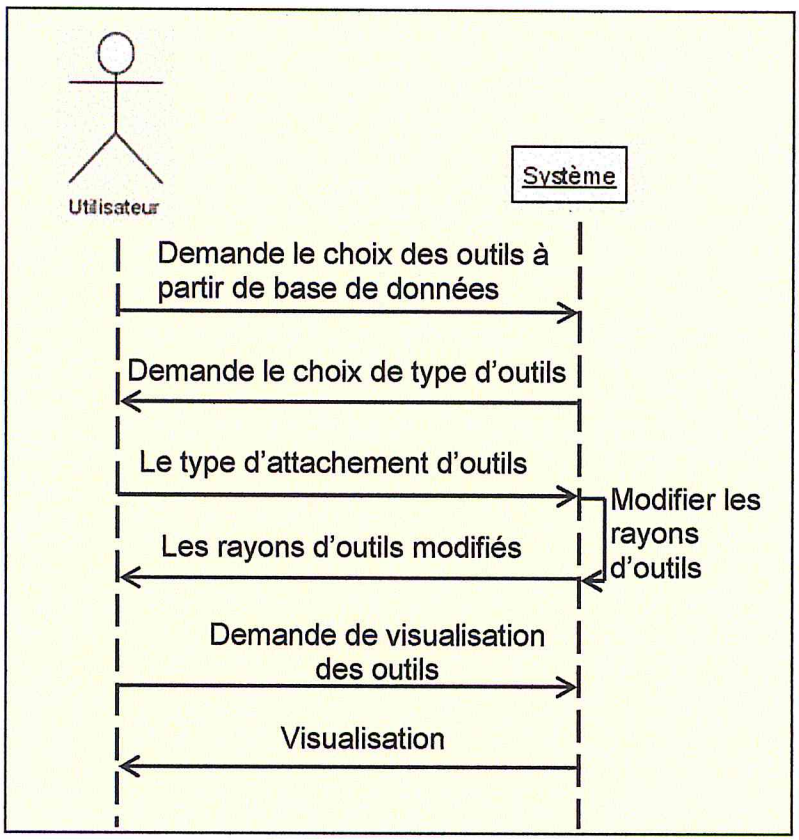


Figure 5.17: Diagramme de séquence pour le choix automatique d'outils à partir de la base de données.

✓ Simulation :

- L'utilisateur demande la simulation de l'usinage;
- Le système demande les paramètres de simulation;
- L'utilisateur définit le mode de simulation, la période de visualisation et le mode de visualisation ;
- L'utilisateur lance la simulation;
- Le système fait la simulation;
- L'utilisateur demande la visualisation de différentes informations (contours, intersection, ...etc.);
- Le système visualise les informations.

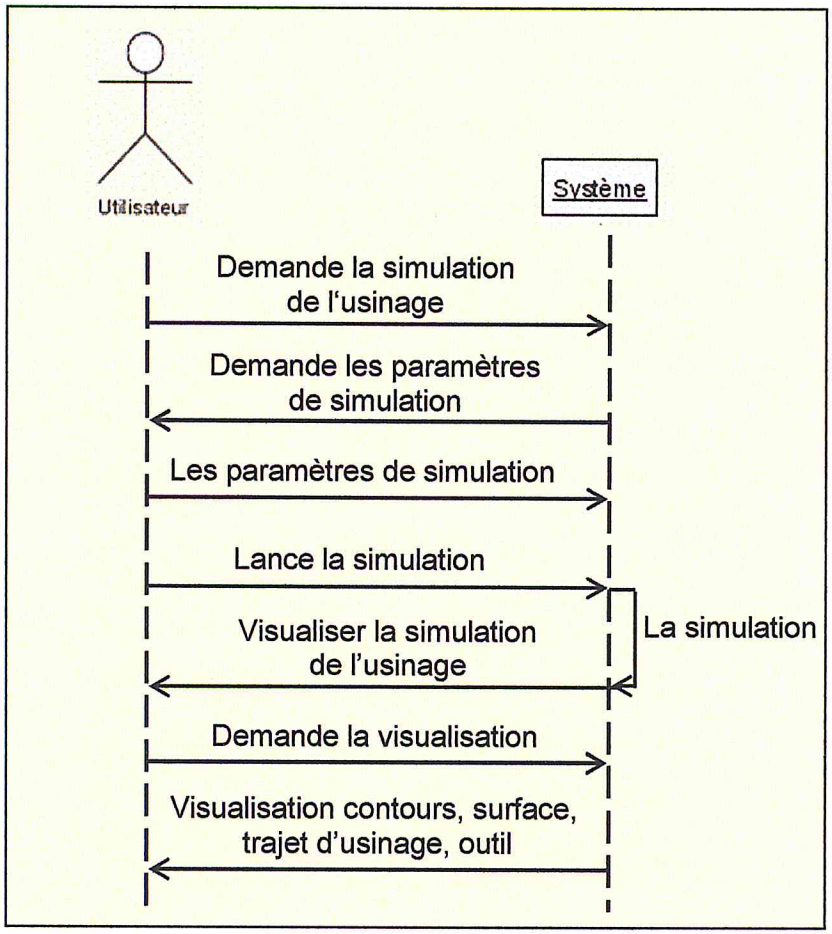


Figure 5.18: Diagramme de séquence pour la simulation.

III.1.3. 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 activité représente une exécution d'un mécanisme, un déroulement d'étapes séquentielles.

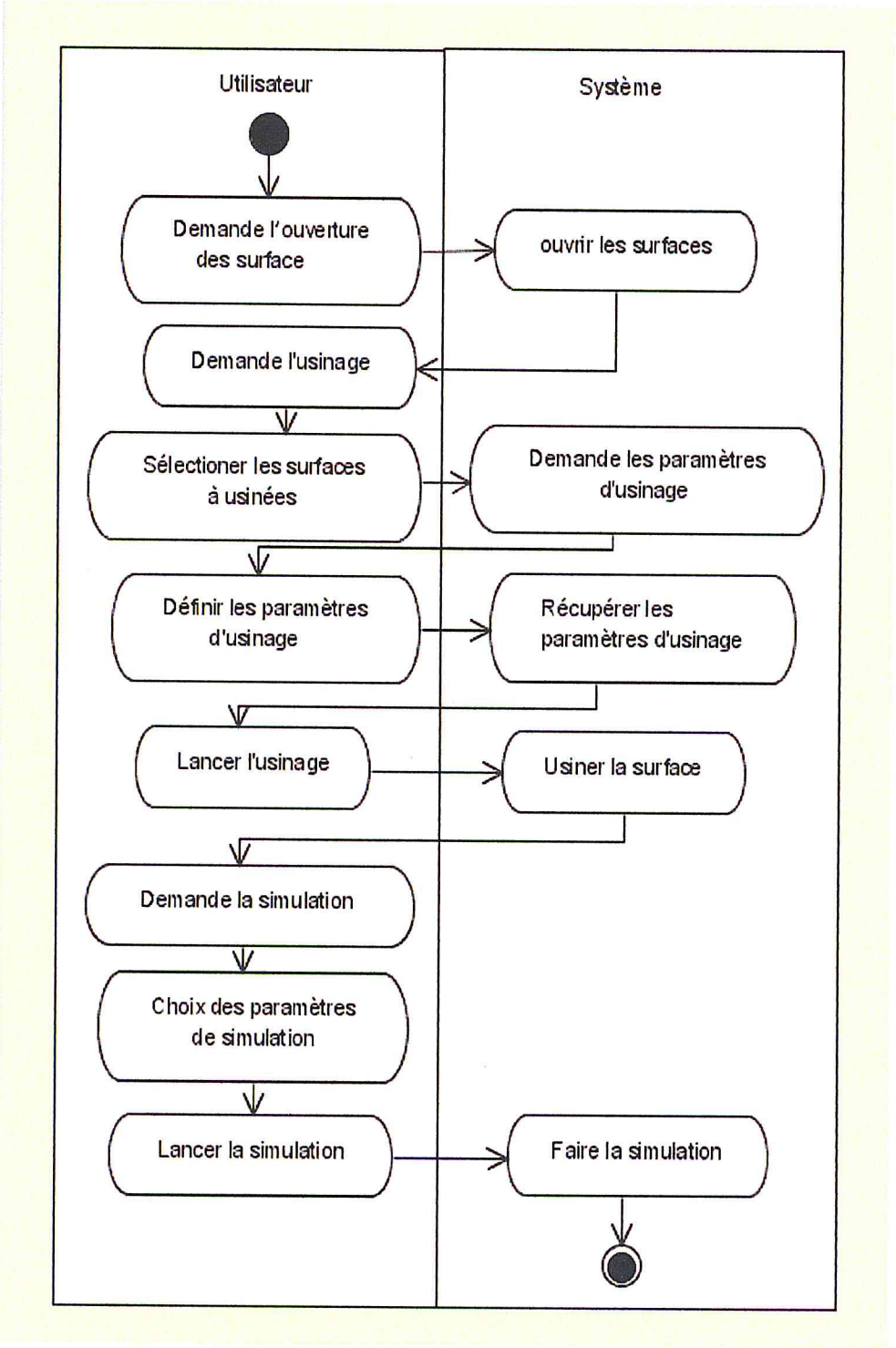


Figure 5.19 : Diagramme d'activités pour l'usinage des surfaces gauches par la méthode Z-Constant:

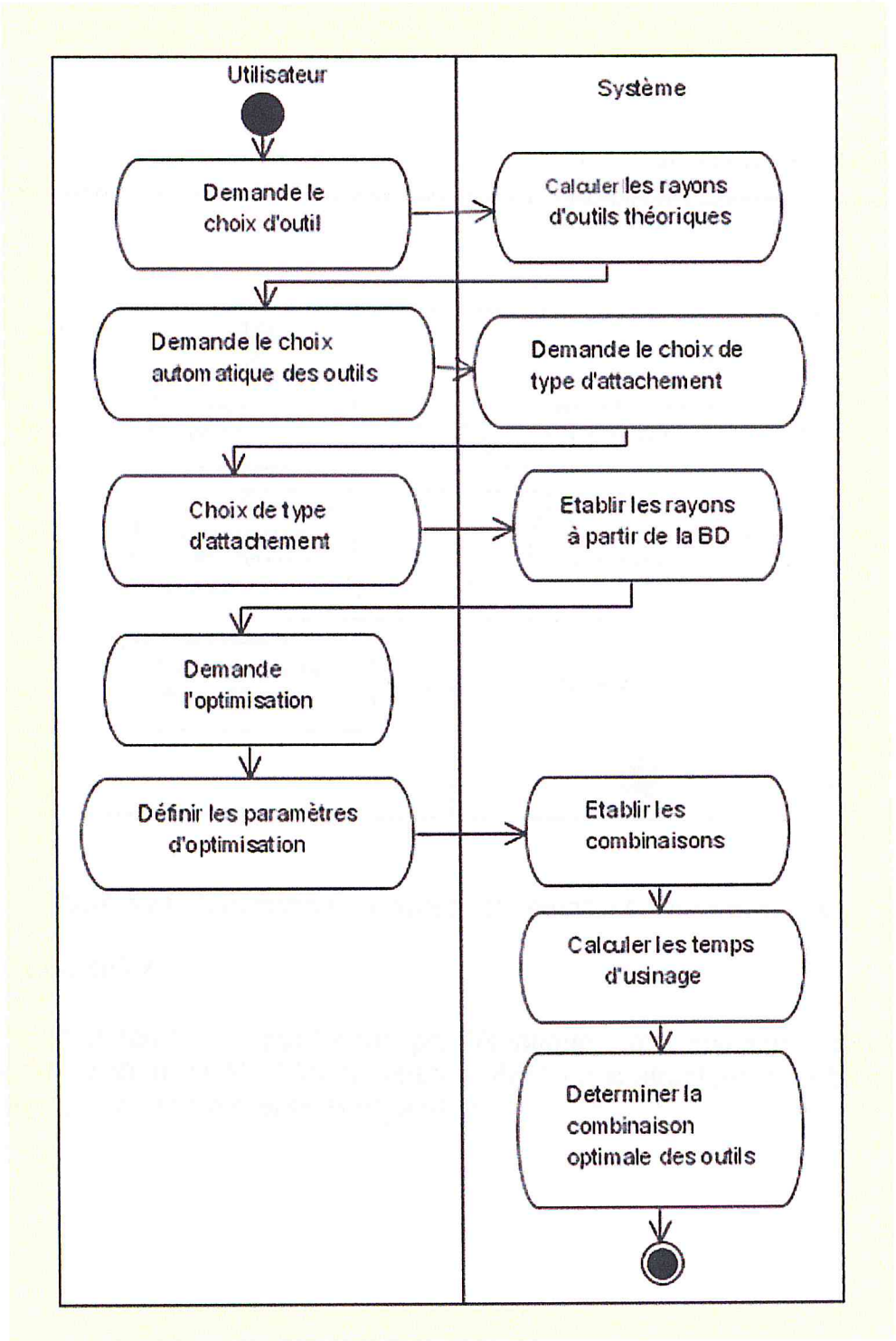


Figure 5.20 : Diagramme d'activités pour le choix d'outils.

Chapitre6

Conception et Implémentation

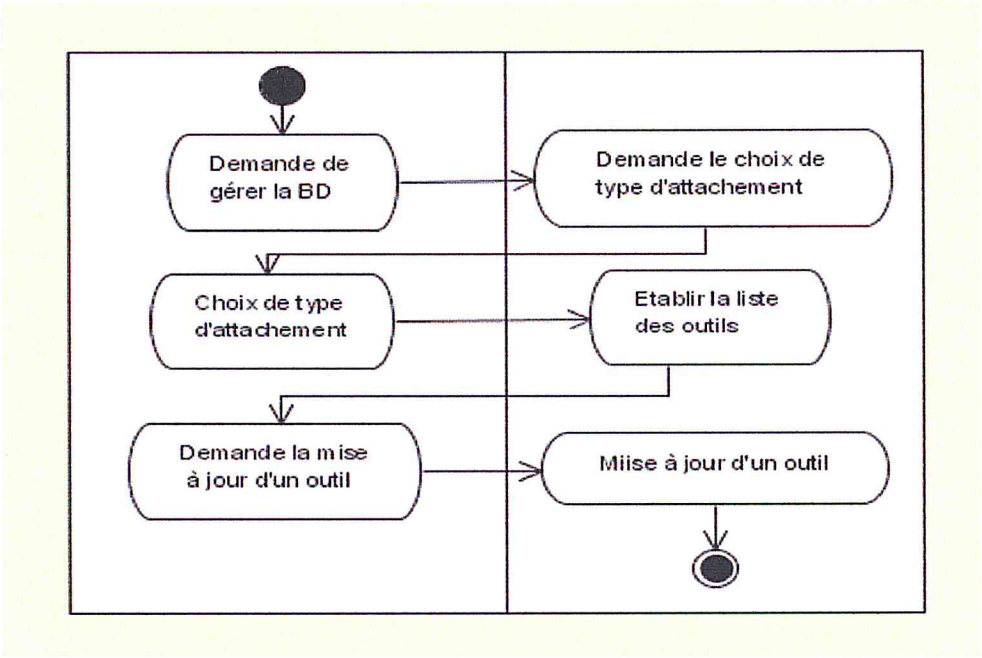


Figure 5.21 : Diagramme d'activités pour gérer la base de données.

IV. CONCLUSION :

Dans ce chapitre on a posé notre problématique, notre solution proposée, et à l'aide de notation d'UML (diagrammes d'UML) on a structuré les objectifs du système pour les rendre claires et organisés.

I. INTRODUCTION :

Après la réponse sur la question «quoi faire» et après avoir compris le système dans le chapitre précédent, l'étape logique suivante consiste à trouver les solutions et ensuite passer à la réalisation de ces solutions avec à la prise en compte des différents besoins des utilisateurs.

II. LA CONCEPTION [20,21] :

La phase d'analyse est suivie de la phase de conception, généralement décomposée en deux phases successives dont la première est appelée conception générale, conception globale, conception préliminaire ou conception architecturale, et la deuxième conception détaillée. A partir du document de spécification, la phase de conception doit fournir une réponse à la question "Comment réaliser le logiciel ?".

II.1. La conception globale :

C'est une définition de l'architecture technique générale, et qui consiste à décomposer notre logiciel en modules, ce dernier représente une unité pour la manipulation des applications.

Dans notre cas, on a proposé une architecture de deux modules :

- Module d'usinage ;
- Module de choix d'outils.

Les composants de notre système peuvent être décrits à l'aide de diagramme de composants (voir figure 6.1).

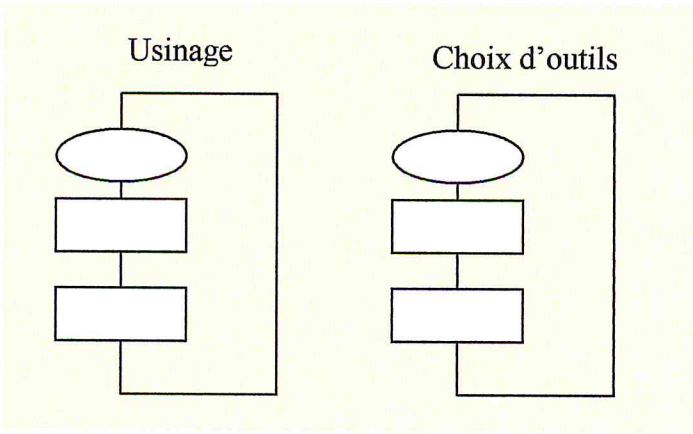


Figure 6.1: Diagramme de composants.

II.1.1. Module «Usinage» :

C'est le module responsable de l'usinage des surfaces et qui permet de générer le trajet d'usinage. Pour atteindre ce résultat, il faut d'abord choisir les surfaces à usiner et une méthode d'usinage avec la définition des paramètres d'usinage et de visualisation.

II.1.2. Module « Choix d'outils »:

Pour déterminer les outils optimums (les rayon d'outils) qui permettent d'usiner les surfaces, nous avons proposé le module de choix d'outils qui consiste premièrement à calculer les rayons d'outils théoriques nécessaires pour usiner chaque plan et chaque contour, et en deuxième étape, ces rayons sont modifiés par une mise à jour à partir de la base de données des outils disponibles. En plus de choix de l'outil pour chaque plan, nous avons la possibilité de choisir entre deux ou trois outils pour usiner la totalité de la surface. Ce choix permet de réduire les temps d'usinage.

II.2.Conception détaillé [13] :

La conception détaillée est le raffinage des modèles d'analyse et une préparation de codage en langage cible

II.2.1. Le module «Usinage» :

- ✓ Usinage avec la méthode Z constant :

Le problème qui se pose est comment usiner une surface gauche avec la méthode Z-Constant (méthode des plans horizontaux) ?, c'est-à-dire faire le traitement de chaque plan à part (la coordonnée Z est constante pour tous les points d'un plan horizontal).

Tout d'abord, il faut récupérer les paramètres d'usinage de la surface. Le diagramme de classe de l'usinage avec la méthode Z-Constant (figure6.2) représente la classe TForm202 qui s'occupe de la récupération de ces paramètres, à base de ces données se fait la classification des surfaces et la création des états (les surfaces usinées avec les mêmes paramètres d'usinage appartiennent au même état).

L'usinage avec la méthode Z-Constant (représentée par la classe TForm 201 de diagramme de classes de l'usinage avec la méthode Z-Constant) adopté consiste à calculer les intersections entre les plans horizontaux et la surface après l'étape d'approximation de la surface avec des triangles (triangulation).

Le prochain problème à résoudre est comment organiser les points d'intersection pour arriver à construire le trajet d'usinage. La structure de contour nous permet de grouper une séquence de points. Donc, plusieurs contours peuvent exister dans un plan. La séquence de ces contours forme le trajet d'usinage. Une fois tous les contours des différents plans sont déterminés, on passe au calcul des différentes positions d'outil en prenant en compte le dégagement et l'engagement d'outil.

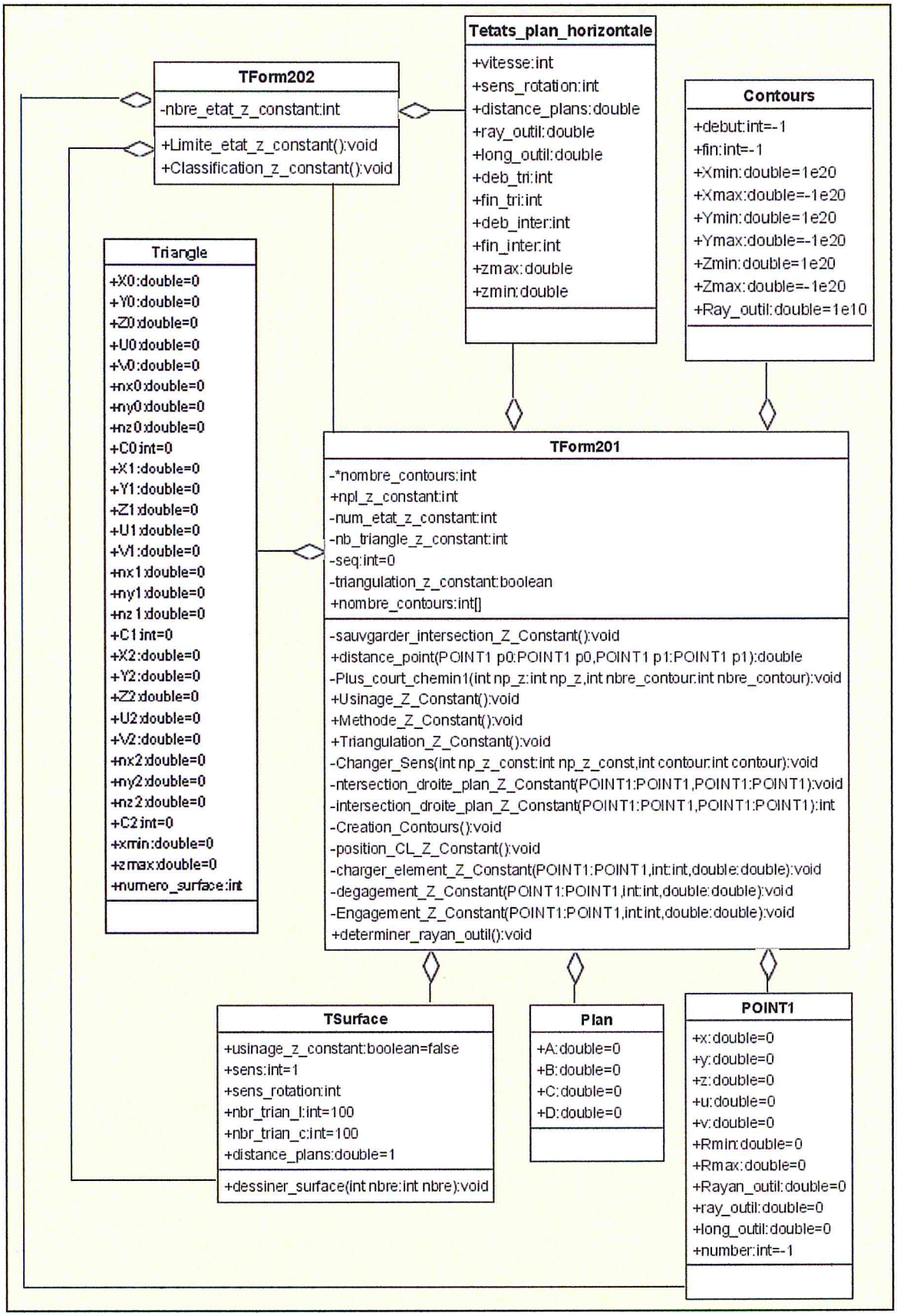


Figure6.2 : Diagramme de classes de l'usinage avec la méthode Z-Constant.

✓ La simulation :

La simulation ne peut se faire qu'après avoir généré le trajet d'usinage. Le diagramme de classes de simulation (figure6.3) représente la classe TForm210 qui s'occupe de la simulation d'usinage par plan. La simulation c'est juste la visualisation (faite par la classe TForm1) des différents résultats d'usinage (trajet d'usinage et outil) stockés dans la classe TForm204. Les temps d'usinage réels sur machine sont calculés par la classe TForm210.

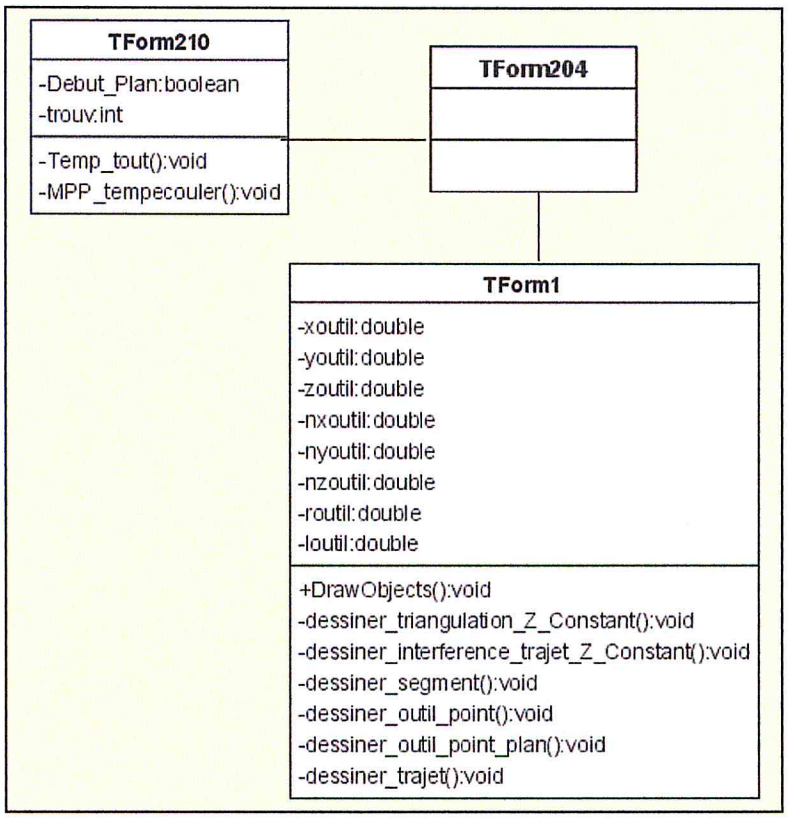


Figure6.3 : Diagramme de classes de simulation.

✓ La visualisation :

Après l'usinage, les résultats sont prêts à être visualisé. La classe TForm1 (voir figure 6.4) représente la visualisation et le dessin de toutes les informations de l'usinage (surface, triangulation, trajet, contours, points, outil, ...etc.).

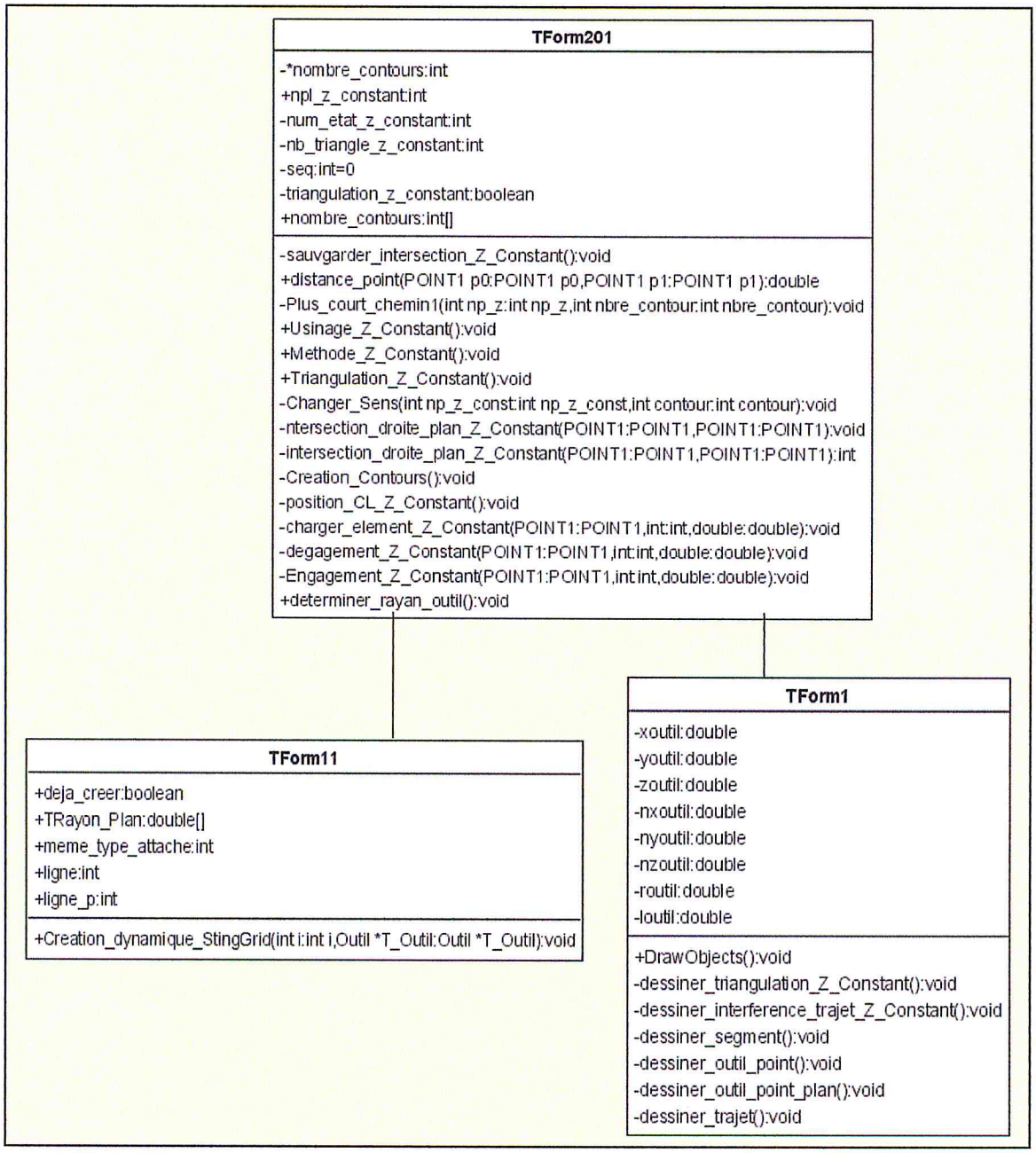


Figure6.4 : Diagramme de classes de visualisation.

II.2.2. Module «Choix d'outils» :

✓ Choix automatique des rayons d'outils :

Dans cette première étape dans le choix des outils, la classe TForm11 (figure 6.5) calcule pour chaque plan et contour un rayon d'outil théorique (des rayons non réels) nécessaire pour l'usinage afin d'éviter les interférences.

✓ Choix d'outils à partir la base de données :

Les rayons d'outils calculés dans la phase précédente sont des rayons théoriques (des rayons d'outils qui n'existent pas dans la réalité). Donc pour les rendre réels, la classe TForm155 dans le diagramme de classes de choix d'outils modifie ces rayons d'outils à partir des outils disponibles dans la base de données.

✓ Optimisation de choix d'outils :

Le but de cette phase est de minimiser le temps d'usinage (le temps de parcourir de l'outil). Cette phase est représentée par la classe TForm155 (voir figure 6.5). L'optimisation consiste à réduire le nombre des outils utilisés afin de réduire le temps d'usinage ainsi que le nombre des changements d'outils. Donc, le problème est de trouver la bonne combinaison optimale d'outils disponibles dans la base de données pour usiner une surface.

✓ Gestion de la base de données :

Cette phase consiste à faire une mise à jour des tables d'outils de la base de données.

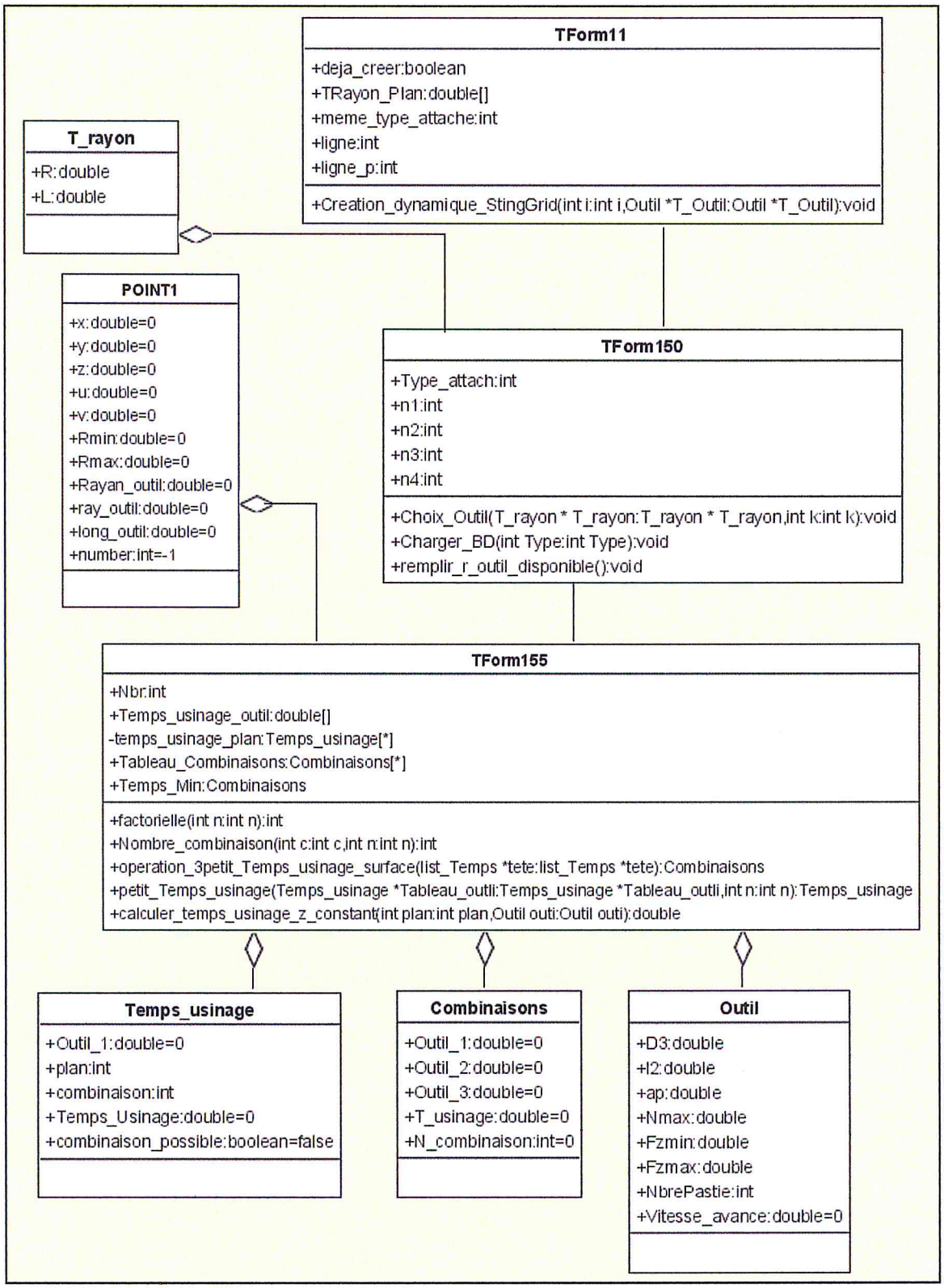


Figure6.5 : Diagramme de classes de choix d'outils.

III. IMPLEMENTATION [20,5]:

Après la conception détaillée, on peut passer à la phase d'implémentation, également appelée phase de construction, phase de réalisation ou phase de codage. Lors de cette phase, la conception détaillée est traduite dans un langage de programmation évolué. Il faut également préparer les données nécessaires à l'exploitation du logiciel.

III.1. Environnement de développement :

Nous avons implémenté notre application sous un système d'exploitation Windows (qui supporte les différentes versions Windows : Windows 95, Windows 98, Windows XP…) sur un PC (Personal computer). Pour l'implémentation, nous avons utilisé le langage de programmation Builder C++ avec son concept oriente objet sa simplicité de programmation qui permet de minimise considérablement le temps de programmation ainsi que sa bibliothèque riche de classes.

III.2. Programmation avec la bibliothèque Open GL :

Toute application graphique requiert, à son plus bas niveau, l'interfaçage avec le matériel graphique utilisé. C'est le rôle d'une bibliothèque graphique.

Toute application graphique requiert, à son plus bas niveau, l'interfaçage avec le matériel graphique utilisé. C'est le rôle d'une bibliothèque graphique. OpenGL (Open Graphics Library) est une bibliothèque graphique standard qui regroupe un ensemble de primitives utilisées lors de l'implémentation des applications graphiques afin d'éviter tout interfaçage direct avec le hardware. Elle regroupe environ 200 commandes différentes qui permettent de spécifier un objet ou de développer des applications interactives en 3D.

D'une manière générale, les fonctions élémentaires d'OpenGL se résument aux taches suivantes :

- Création de formes géométriques à partir des primitives géométriques (OpenGL considère les points, les lignes et d'autres images comme étant primitives),
- Conversion de l'information concernant la description mathématique de l'objet et sa couleur en un ensemble de pixels sur l'écran,
- Rendu réaliste des scènes par définition des textures,
- Manipulation des scènes complexes,
- Représentation et manipulation des objets 3D.

III.3.Structuration de l'application logicielle :

L'implémentation des différents modules composant notre système peut être détaillée comme suite :

III.3.1. Implémentation de module « Usinage » :

Pour la réalisation de ce module nous avons défini les classes suivantes :

- La classe Tsurface : pour représenter les surfaces à usiner.
- La classe Contours : qui permet de représenter un ensemble de points segmentés. Chaque contour a un début et une fin.
- La classe Tetats_plan_horizontale : c'est une classe qui regroupe un ensemble des surfaces usinées avec les mêmes paramètres.
- La classe Plan : cette classe représente l'objet Plan (plan horizontal dans notre cas).
- La classe POINT1 : permet d'identifier chaque point d'intersection.
- La classe Triangle : avec l'objet Triangle de cette classe nous approximons une surface avec un ensemble de triangle.

Pour manipuler et stocker les informations des objets de ces classes, nous avons utilisé deux structures de données, les tableaux et les listes chaînées.

Une liste chaînée « list_POINT1» : un maillon de cette liste est un objet de la classe « POINT1 ».

```
Struct list_POINT1 {

        POINT1 p;
        list_POINT1 *svt;
    };
```

Les tableaux : nous pouvons citer un tableau de deux dimensions « contours » pour stocker les objets de la classe « Contours », un tableau « surfaces » de la classe TSurfase, un tableau de deux dimensions « point_inters_plan_z_constant » de la classe « POINT1 ».

Nous avons implémenté les fonctions suivantes :

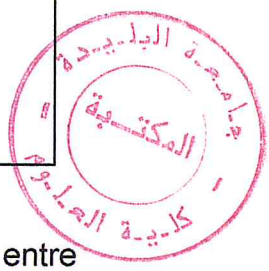
Usinage Z Constant () : après la triangulation et la classification des surfaces à usiner dans des états, cette fonction lance l'usinage des surfaces avec la méthode Z-Constant. S'il y a des points d'intersection entre les surfaces et les plans horizontaux, la fonction création des contours est appelée. Tous les points d'intersection sont sauvegardés pour générer le trajet d'usinage.

```
Usinage Z Constant ()
Début
    Pour tout les surfaces
    Faire
        Si (une surface usiné par z constant)
        Alors
            nombre surface z constant++ ;
        Fin si
    Fait
    Si (nombre surface z constant>0)
    Alors
        début intersection=0 ;
        Pour tout les états
        Faire
            plan.A=0;
            plan.B=0;
            plan.C=1;
            ouvrir un fichier Intersection_Z_Constant.dat ;
            distance =etat.distance ;
            sortir=faux ;
            dmin=état.zmin ;
            dmax=état.zmax ;
            plan.D= -dmax
            Tant que (plan.D <=-dmin et sortir faux)
            Faire
                Si (plan.D==-dmin)
                Alors
                    sortir=vrai ;
                Fin si
                Pour tout les points de ce état
                Faire
                    intersection_plan_triangle_Z_Constant(P0, P1, P2);
                Fait
                Si (intersection=vrai)
                Alors
                    Création des contours();
                    Plan++;
                Fin si

            Fait
        Fait
        sauvgarder_intersection_Z_Constant();
        Pour tout les états
        Faire
            position_CL_Z_Constant();
        Fait
    Fin si
Fin
```

sauvgarder_intersection_Z_Constant() : le rôle de cette fonction est de sauvegarder les poins d'intersection dans un fichier.

```
sauvgarder_intersection_Z_Constant()
Début
    Pour tout les plan
    Faire
        Pour tout les points d'intersection d'un plan
        Faire
            Sauvegarder le point dans un fichier
        Fait
    Fait
Fin
```



Distance point (p0, p1) :c'est une fonction qui calcule la distance entre deux points d'intersections.

```
Distance point (p0, p1)
Début
  dis=sqrt((p1.x-p0.x)*(p1.x-p0.x)+(p1.y-p0.y)*(p1.y-p0.y)+(p1.z-p0.z)*(p1.z-
  p0.z));
  retourne(dis);
Fin
```

Création des contours () : à partir des poins d'intersection, cette fonction crée les contours (séquence de points) après élimination des points doubles.

```
Création des contours ()
Début
    Elimination des répétitions ;
    Création des sous contours ;
    Création des contours ;
    Modifier le sens des contours ;
    Calculer le plus court chemin ;
Fin
```

```
Création _contours()
Début
    dis=5 ;
    dist_petit=11^e20 ;
    dist_petita=11^e20 ;
    nbre_contours=1 ;
    Pour tout les points de plan
    Faire
      Pour tous les points de plan à partir de deuxième point
      Faire
            Si (le point n'appartient a aucun contour)
            Alors
                  Distance =distans_point(point1, point2 ) ;
                  Si (Distance< dist_petita)
                  Alors
                        dist_petita= Distance;
                        indice_petita=indice de point2 ;
                  Fin si
                  Si (Distance<Dist)
                  Alors
                        Si (Distance<dist_petit)
                        Alors
                              dist_petit= Distance;
                              pemut =vrai ;
                              indice_petit=indice de point2 ;
                        Fin si
                  Fin si
            Fin si
      Fait
      Si (pemut==vrai)
      Alors
          Permuter entre le point de l'indice petit et le successeur de point1 ;
            Le numéro de contour de point de l'indice_petit= Le numéro de
            contour de point1 ;
      Fin si
      Si non
      Alors
            Permuter entre le point de l'indice petita et le successeur de
            point1 ;
            Le numéro de contour de point de l'indice_petita= Le numéro de
            contour de point1+1 ;
            nbre_contours++ ;
      Fin sinon
    Fait
```

```
Pour tout les contours
Faire
  Pour tout les contours
  Faire
      Si (la distance entre début et fin ou fin et fin ou début et
      début ou fin et début  de deux contours <Dist)
      Alors
          Pour tout les points de deuxième contour
          Faire
              Donner le même numéro de premier contour
              au pont de deuxième contour ;
          Fait
      Fin si
  Fait
Fait
Pour tout les contours
Faire
  Si (le contour n'est pas insérer dan la liste)
  Alors
      Pour tout les points de contour de début jusqu'à fin
      Faire
          Insérer le point en tête  de la liste des points ;
      Fait
      Si (la distance entre le début et fin de contour <Dist)
      Alors
          Réinsérer le premier point de contour a la fin de liste ;
      Fin si
      Pour tout les contours
      Faire
          Si (la distance entre le début de contour insérer et le
          début d'un autre contour non inséré)
          Alors
              Insérer les points de contour non inséré de
              début jusqu'à fin au début de la liste de ;
          Fin si
          Si (la distance entre le début de contour insérer et la
          fin d'un autre contour non inséré)
          Alors
              Insérer les points de contour non inséré de
              début jusqu'à fin a la fin de la liste de ;
          Fin si
          Si (la distance entre la fin de contour insérer et le
          début d'un autre contour non inséré)
          Alors
              Insérer les points de contour non inséré de fin
              jusqu'à fin au début de la liste de ;
          Fin si
```

```
                    Si (la distance entre la fin de contour insérer et la fin
                    d'un autre contour non inséré)
                    Alors
                        Insérer les points de contour non inséré de fin
                        jusqu'à début a la fin de la liste de ;
                    Fin si
                Fait
            Fin si
        Fait
Fin
```

Plus court chemin (plan, nbre contour) : cette fonction a pour but de trouver une séquence des contours qui forment le trajet d'usinage tel que ce trajet soit le plus court (trajet minimum).

```
plus court chemin (plan, nbre contour)
Début
        L=0;
 Pour tout les contours
 Faire
  Le contour existe dans le chemin =faux;
 Fait
 Pour tout les contours
 Faire
            Min=1^e10;
     Pour tout les contours
    Faire
      Si (le contour existe dans le chemin =faux ) Alors

        Si (la distance entre le contour et le contour L < Min ) Alors
          Indice petit =indice de contour;
        Fin si
      Fin si
    Fait
    le contour existe dans le chemin =vrai;
    Contour tmp;
    tmp=contour[L+1];
    contour[L+1]=contour[Indice petit];
    contour[Indice petit]=tmp;
    L=L+1;
 Fait
Fin
```

Changer_Sens (plan, contour) : cette fonction prend en compte les modes d'usinage en avalant et en opposition. Le sens de déplacement sur les différents contours dépend du sens de rotation de la broche.

```
Changer_Sens (plan, contour)
Début
    Pour tout les points de contours de fin au début
    Faire
        Insérer le point dans une liste des points;
    Fait

    Tant que (La liste n'est pas vide)
    Faire
        Insérer l'élément de la tête de liste dans un vecteur ;
        Pointer sur le prochain élément de la liste
    Fait
Fin
```

dessiner segment () : l'objectif de cette fonction est de visualiser les différents contours sur les différents plans d'usinage.

```
dessiner segment ()
Début
        Pour tout les plan
        Faire
                Pour tout les contours
                Faire
                        Pour tout les points de contour
                        Faire
                                Dessiner un sagement entre chaque deux
points
                        Fait
                Fait
        Fait
Fin
```

dessiner_triangulation_Z_Constant() : cette fonction permet de visualiser la triangulation d'une surface usinée avec la méthode Z-Constant.

```
dessiner_triangulation_Z_Constant ()
Début
    Pour tout le nombre de triangles
    Faire
        Dessiner un triangle entre les trois points de triangle (dessiner à
partir de chaque point de triangle un sagement) ;
    Fait
Fin
```

III.3.2. Implémentation de module « choix d'outils » :

Pour arriver à implémenter ce module nous définissons les classes suivantes :

- La classe Outil : cette classe représente l'objet Outil utilisé pour usiner une surface. Les outils sont stockés dans une base de données.
- La classe Combinaisons : l'objet Combinaison de cette classe représente la combinaison d'outils utilisés pour l'optimisation de temps d'usinage.
- La classe Temps_usinage : elle représente le temps d'usinage calculé pour une combinaison d'outils d'optimisation de choix d'outils.
- La classe T_rayon : contient les paramètres d'outil : rayon et le longueur.

Pour manipuler et utiliser les objets outil de la classe «Outil », nous avons utilisé la structure des listes chaînées.

```
struct list_Outil{
        Outil p;
        list_Outil *svt;
};
```

Nous utilisons aussi une liste chaînée «list_Temps » pour manipuler les objets de la classe Combinaisons.

```
struct list_Temps{
        Combinaisons elt;
        list_Temps *svt;
};
```

Calculer temps usinage z constant (plan, outil) : cette fonction calcule le temps d'usinage d'un plan avec un outil donné.

```
Calculer temps usinage z constant (plan, outil)
Début
        Distance=0 ;
      Temps total=0 ;
      Pour tout les contours de plan
      Faire
        Pour tout les points de contour
        Faire
          P1.x=point [plan][i].x+outil.D3/2* point [plan][i].surface normale.x ;
          P1.y=point [plan][i].y+outil.D3/2* point [plan][i].surface normale.y ;
          P1.z=point [plan][i].z+outil.D3/2* point [plan][i].surface normale.z ;
          P2.x=point [plan][i+1].x+outil.D3/2* point [plan][i+1].surface normale.x ;
          P2.y=point [plan][i+1].y+outil.D3/2* point [plan][i+1].surface normale.y ;
          P2.z=point [plan][i+1].z+outil.D3/2* point [plan][i+1].surface normale.z ;
          Distance = Distance+distance(P1,P2) ;
        Fait
        Distance= Distance+distance engagement +distance dégagement ;
        Temps total= Temps total+Distance /outil.vitesse d'avance ;
        Distance=0 ;
      Fait
Fin
```

Nombre_combinaison (int c, int n) : cette fonction calcule le nombre de combinaisons possibles en fonction du nombre d'outils disponibles et du nombre d'outils par combinaison.

```
Int Nombre_combinaison (int c, int n)
Début
    Int comb=0 ;
    comb =factorielle(n)/( factorielle n-c)* factorielle (c));
    Rotourn (comb);
Fin
```

Choix outil () : cette fonction établit les rayons d'outil pour chaque plan et chaque contour à partir d'un ensemble d'outils disponibles dans la base de données.

```
petit_Temps_usinage_surface(list_Temps)
Début
    temps_petit=10e20;
    Pour tout les elements de la liste list_Temps
    Faire
      Si (le temps d'usinage de l'élément < temps_petit)
      Alors
            temps_petit= le temps d'usinage de l'élément ;
      Fin si
      Pointer sur le prochain élément de la liste ;
    Fait
Fin
```

III.4. Interface utilisateur:

Notre interface utilisateur a pour rôle de guider l'utilisateur vers les différentes fonctions de l'application et d'apprécier en même temps le résultat de son travail.

Cette interface est composée de :

- Une fenêtre principale qui contient le menu principal, les boutons de traitements et la barre d'état ;
- Une deuxième fenêtre dite «fenêtre de visualisation» sur laquelle sont affichées les différentes formes générées (voir figure 6.6).

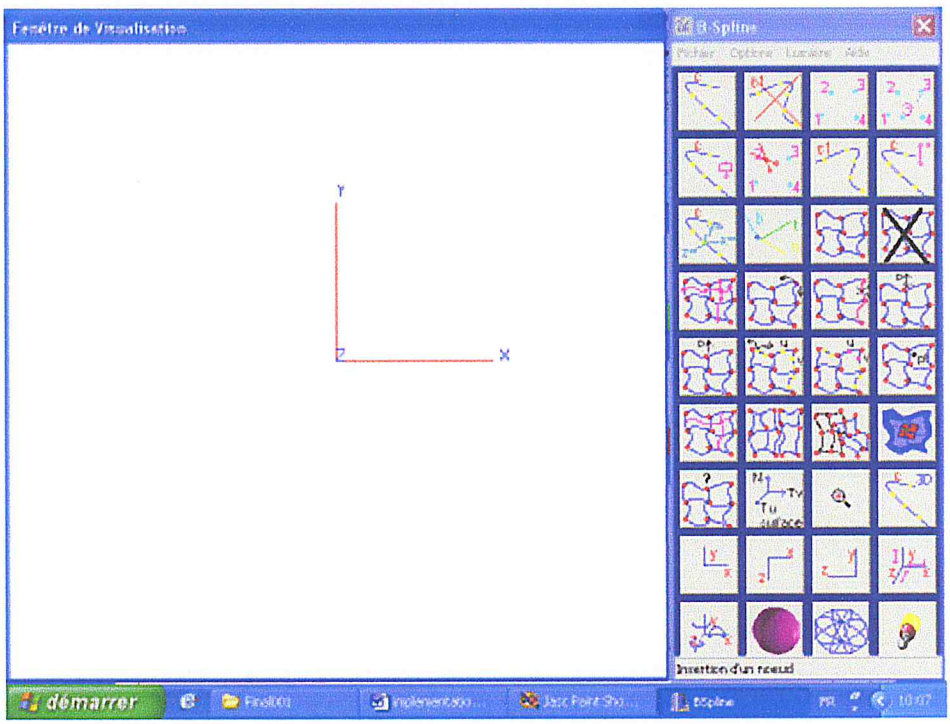


Figure 6.6: La fenêtre principale et la fenêtre de visualisation.

Le menu principal se compose de quatre rubriques: fichier, options, lumière et aide.

- Rubrique Fichier: cette rubrique contient toutes les fonctions de manipulation de fichier, c'est-à-dire, ouvrir un nouveau fichier, ouvrir un fichier existant, la sauvegarde du fichier courant afin de le réutiliser ultérieurement (figure 6.7).
- Rubrique Options : cette rubrique permet à l'utilisateur de modifier les paramètres des courbes, des surfaces, et de l'usinage des surfaces. Cette rubrique permet de modifier les paramètres d'usinage, la visibilité, les couleurs et épaisseurs d'usinage et également de simuler virtuellement l'usinage et ouvrir la fenêtre de choix automatique des outils (figure6.8).

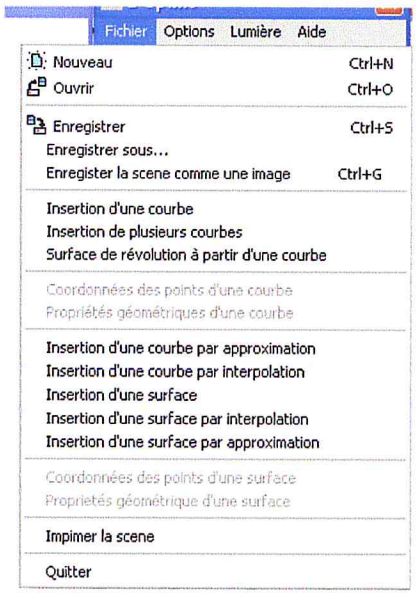


Figure 6.7: Rubrique fichier.

Figure 6.8: Rubrique options.

Le clic sur " paramètres d'usinage " affiche une fenêtre qui permet de lancer l'usinage des surfaces (voir figure 6.9). Dans la fenêtre "Usinage", l'utilisateur sélectionne les surfaces à usiner par la souris ou par introduction du numéro de la surface dans le champ des surfaces. Les surfaces sélectionnées peuvent être ajoutées ou supprimées de la liste d'usinage. L'utilisateur a la possibilité de modifier les paramètres de visualisation et de choisir la méthode d'usinage à partir d'une liste des méthodes d'usinage.

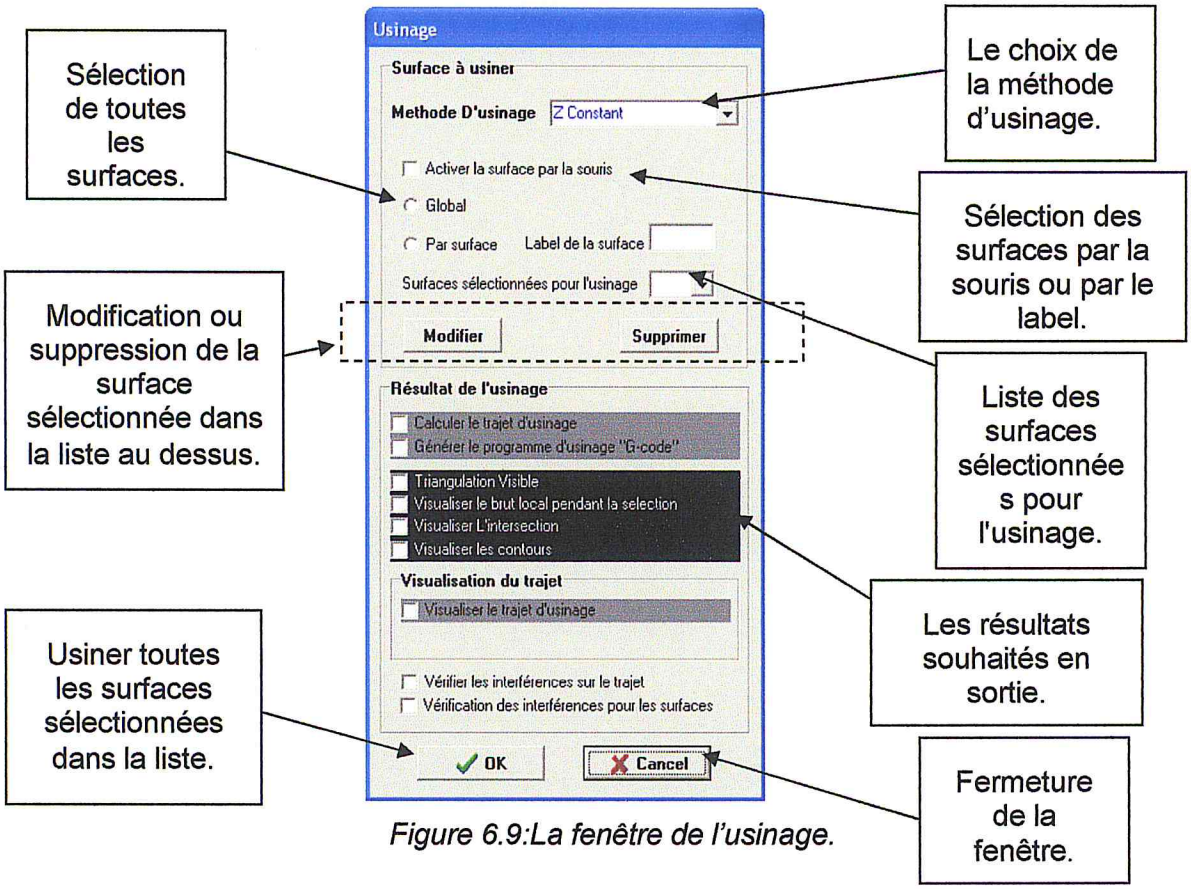


Figure 6.9:La fenêtre de l'usinage.

Pour usiner les surfaces sélectionnées par la méthode Z-Constant, il faut définir les paramètres d'usinage dans la fenêtre "Paramètres" qui apparaît dès que la sélection des surfaces est terminée.

La fenêtre "Paramètres" est décomposée en cinq pages :

- La page "Paramètres d'outil";
- La page "Visibilité";
- La page "Paramètres du G-Code";
- La page "Triangulation/Intersection ";
- La page "Tolérances d'usinage ".

Notre travail consiste à ajouter dans la page «Triangulation/Insertion» la possibilité de choisir le mode d'usinage (figure 6.10).

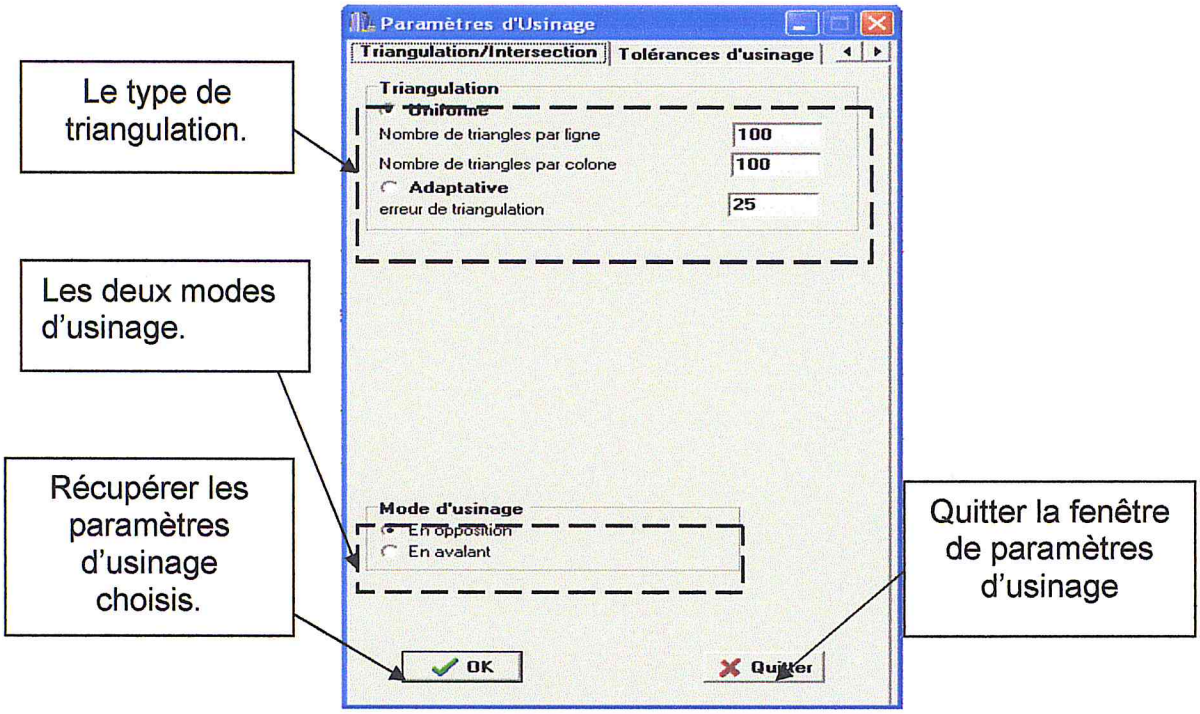


Figure 6.10:La fenêtre de paramètres d'usinage.

Notre tache réside dans l'ajout de «choix automatique des outils» dans la rubrique "Usinage" qui permet d'ouvrir la fenêtre de choix des outils (voir figure 6.11). Cette fenêtre affiche initialement les rayons d'outils calculables dans deus tableaux (un pour les plan et l'autre pour les contours). Cette fenêtre contient aussi deux boutons, un pour le choix automatique des outils à partir de la base de données et l'autre pour faire l'optimisation de choix d'outils ainsi que deux boutons pour visualiser les outils par plan et par contour.

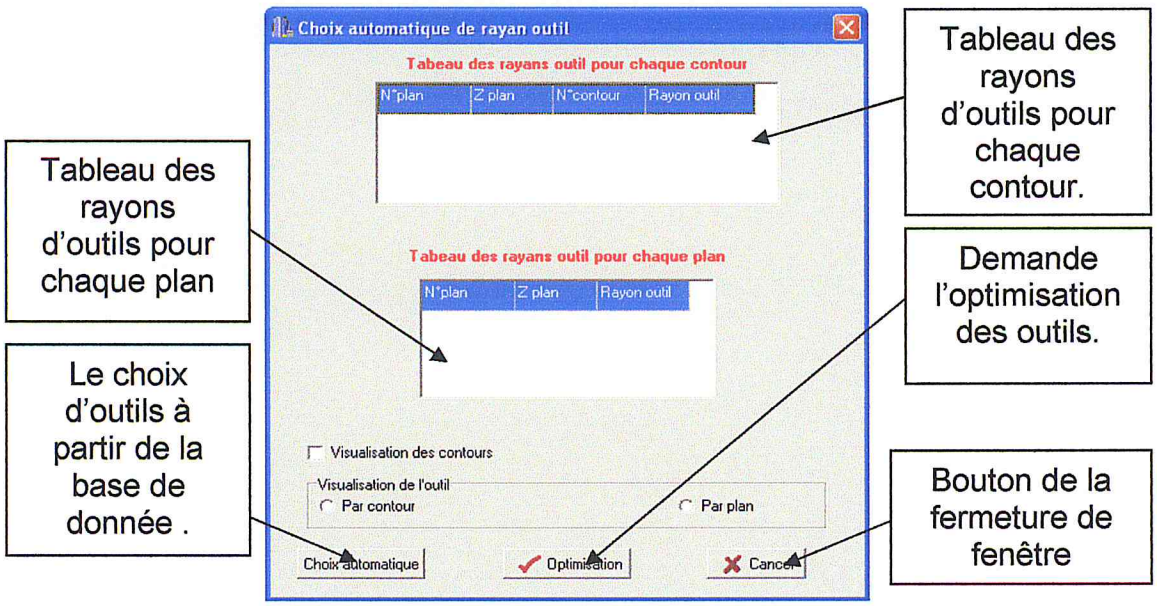


Figure 6.11. : La fenêtre de choix d'outils.

```
Choix outil ()
Début
    rayon=vrai ;
    Pour tout les plan
    Faire
        Pour tout les rayon d'outil disponible de plus grand jusqu'à plus petit
        Faire
            Si (rayon d'outil disponible <rayon d'outil de plan)et (rayon=vrai )
                rayon d'outil de plan = rayon d'outil disponible ;
                rayon=faux;
            Fin si
        Fait
        Si (rayon=vrai) Alors
            rayon d'outil de plan = le plus petit rayon d'outil disponible ;
        Fin si
        rayon=vrai ;
    Fait
```

Int Factorielle (n):c'est une petit fonction pour calculer le factorielle de n valeur.

```
Int Factorielle (n)
Début
    int fact=n;

    Pour (int i=1;i<=n-1;i++)
    Faire
        fact=fact*(n-i);
    Fait

    Rotourn (fact);

Fin
```

petit_Temps_usinage_surface(list_Temps) : pour une liste donnée des temps d'usinage, cette fonction détermine le plus petit temps d'usinage qui définit la combinaison optimale.

La fenêtre de choix des outils affiche initialement 4 boutons qui représentent les 4 types d'outils (voir figure 6.12). La sélection d'un de ces boutons permet d'afficher la liste des outils corresponds. La validation de choix se fait par le bouton ok (voir figure 6.13).

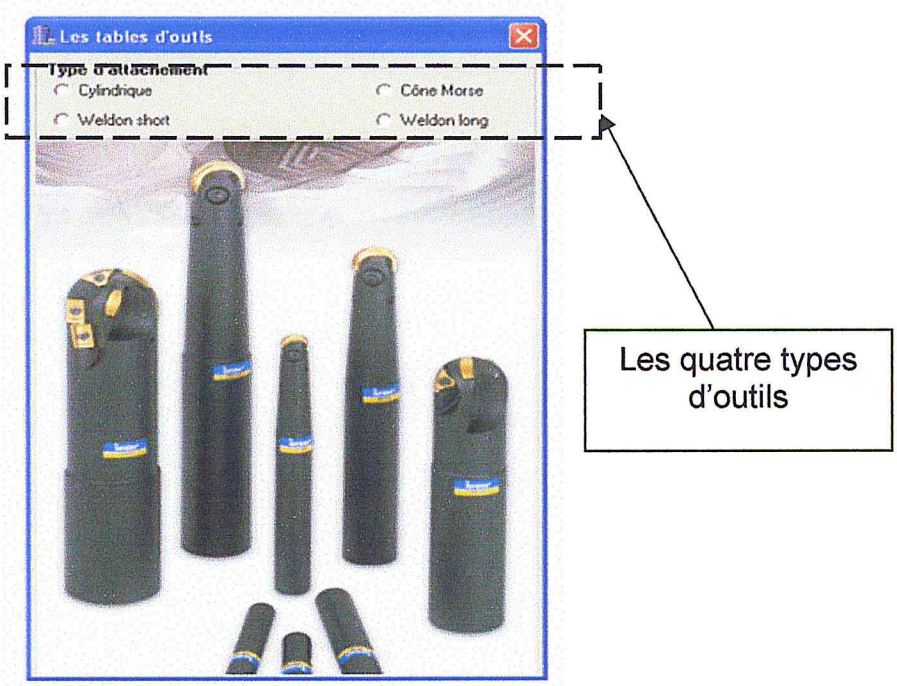


Figure 6.12. : La fenêtre de types d'outils.

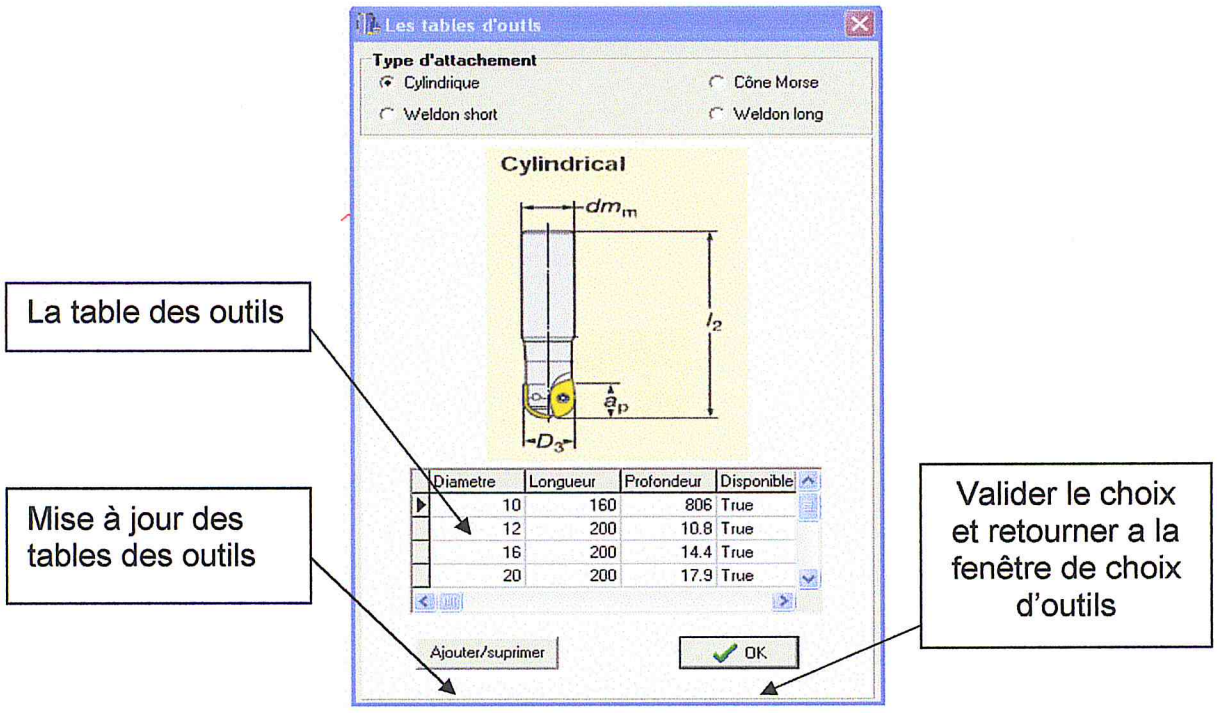


Figure 6.13. : La fenêtre de tables des outils.

Le clic sur le bouton «Ajouter/Supprimer» de la fenêtre «les tables d'outils» ouvre une fenêtre pour faire la mise à jour des tables d'outils. Cette fenêtre se compose de trois pages, une pour la suppression, une pour l'ajout et la dernière pour la modification.

La page d'insertion affiche la table des outils et des champs à remplir avec les informations du nouveau outil à insérer. La validation de l'insertion se fait avec un simple clic sur le bouton «insérer » (voir figure 6.14).

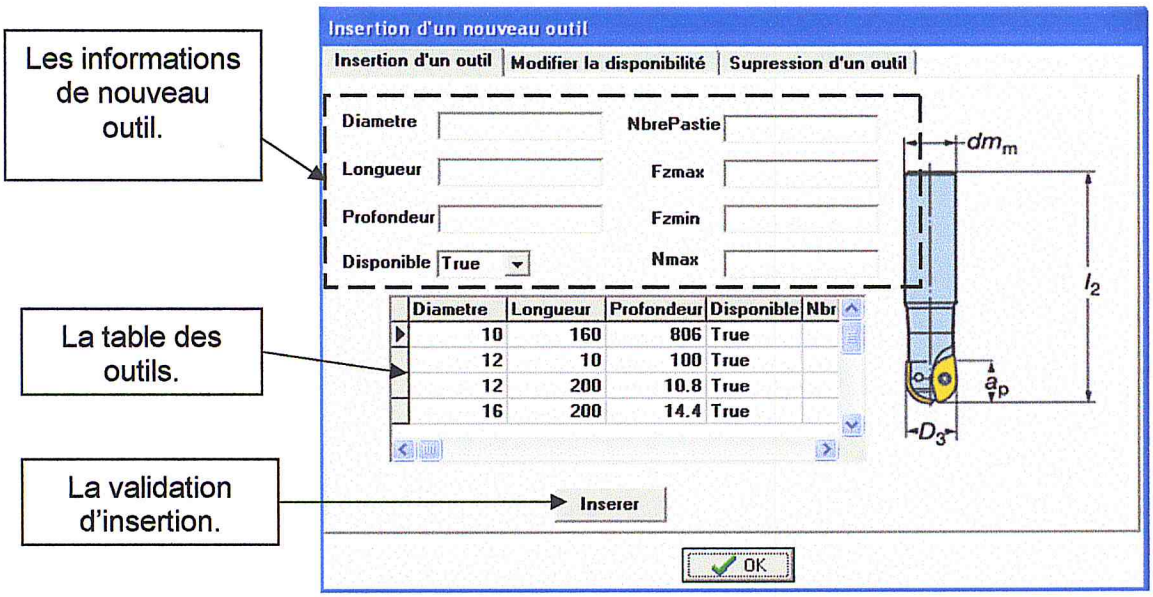


Figure 6.14. : La fenêtre d'insertion d'un outil.

La modification de la disponibilité d'un outil existant dans la base de données se fait dans la page «modifier la disponibilité ». Cette page donne à l'utilisateur la possibilité de modifier la disponibilité d'un outil ou non et valider la modification avec le bouton «Modifier» (voir figure 6.15).

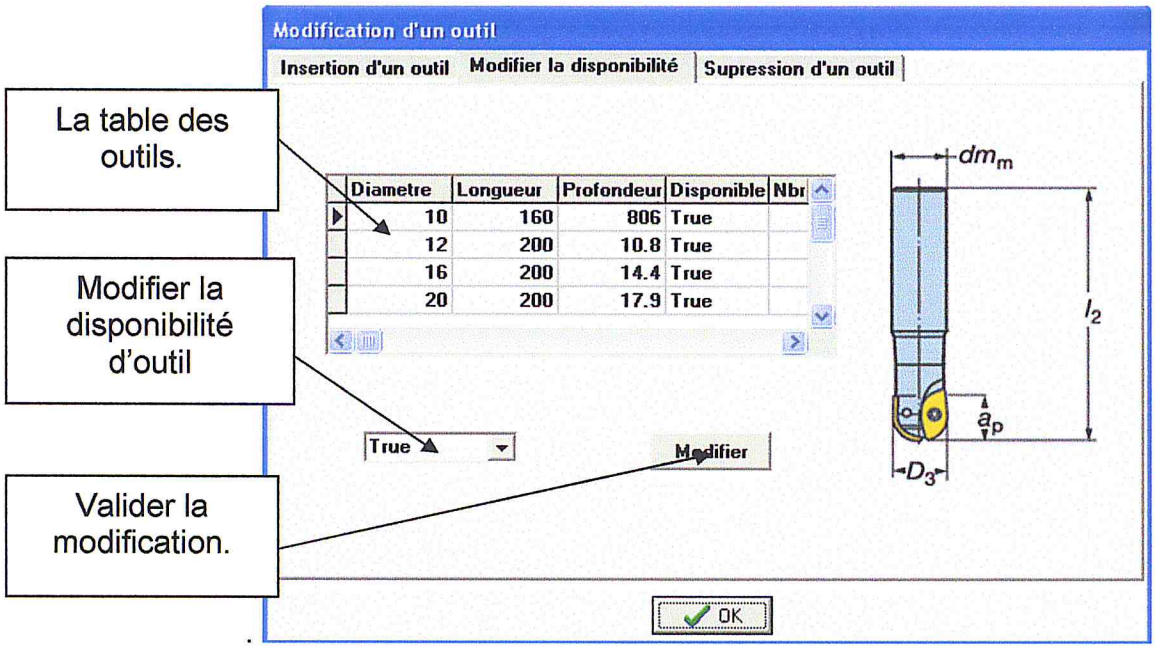


Figure 6.15. : La fenêtre de modification d'un outil.

La dernière page dans la fenêtre de mise à jour des tables des outils est la page de suppression (voir figure 6.16). Plus de la table des outils, dans cette page on ne dispose que d'un bouton pour supprimer l'outil sélectionné de la base de données.

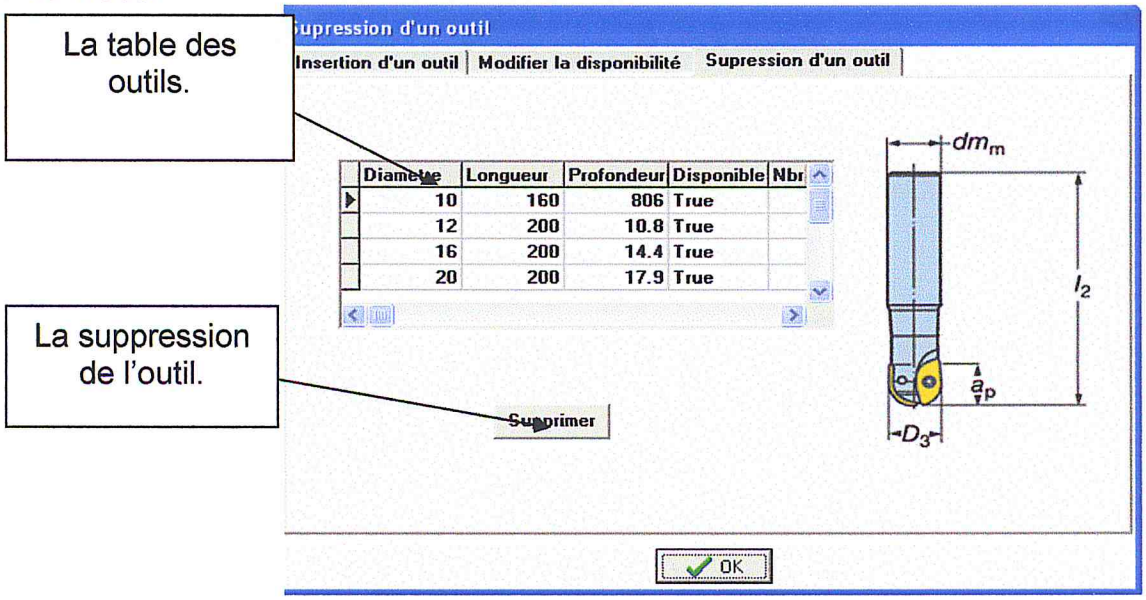


Figure 6.16. : La fenêtre de suppression d'un outil.

Le bouton «Optimisation» de la fenêtre de choix d'outils donne la main à la fenêtre d'optimisation de choix d'outils (figure 6.17). Les informations détaillées concernant la combinaison optimale sont affichées sur la fenêtre « La combinaison optimale » et cela en cliquant sur le bouton « Combinaison optimale » (figure 6.18).

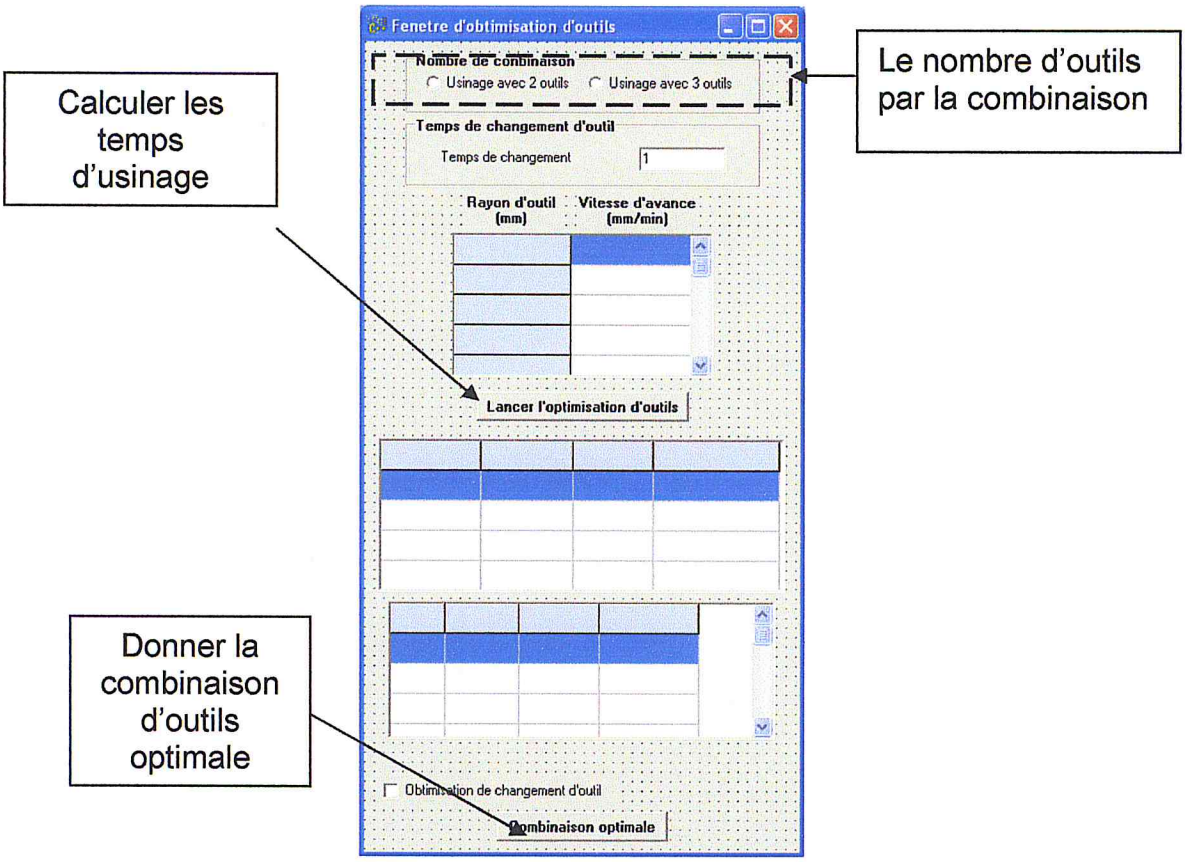


Figure 6.17 : La fenêtre de tables des outils.

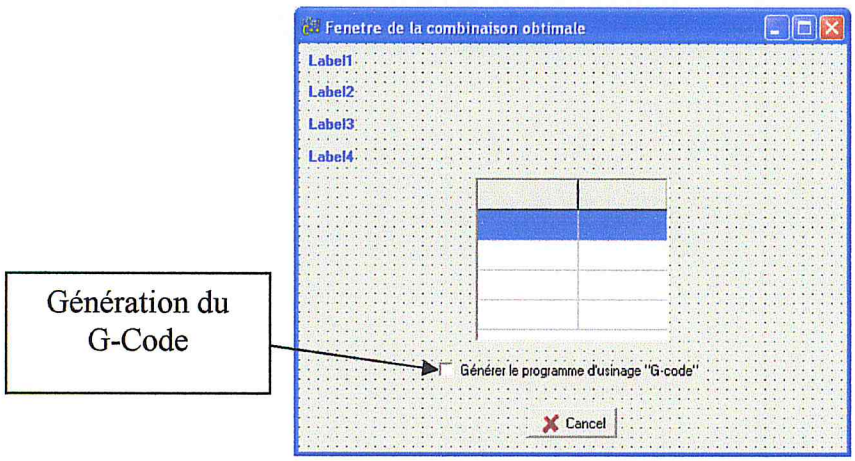


Figure 6.17 : La fenêtre de tables des outils.

IV. CONCLUSION :

Ce chapitre représente les résultat de notre travail ainsi que les solutions proposées pour résoudre la problématique posée par ce système, les choix d'implémentation et l'implémentation les algorithmes avec un langage de programmation.

Chapitre7

Tests et Validation

I. INTRODUCTION :

La dernière étape dans notre démarche de développement et d'écriture du code associé à notre système, c'est l'étape de tests et de validations des résultats des différentes étapes précédentes. Cette étape est très importante puisqu'elle permet de valider le logiciel par rapport au cahier de charges et de faire les corrections nécessaires en cas de problèmes qui peuvent apparaître pendant l'exécution.

II. TESTS ET VALIDATION :

Cette phase de validation consiste à simuler et tester les cas d'utilisations de l'usinage et le choix d'outils à l'aide de déroulement des scénarios représentés par les diagrammes de séquences.

- ✓ Cas d'utilisation d'usinage des surfaces gauches par la méthode Z-Constant :

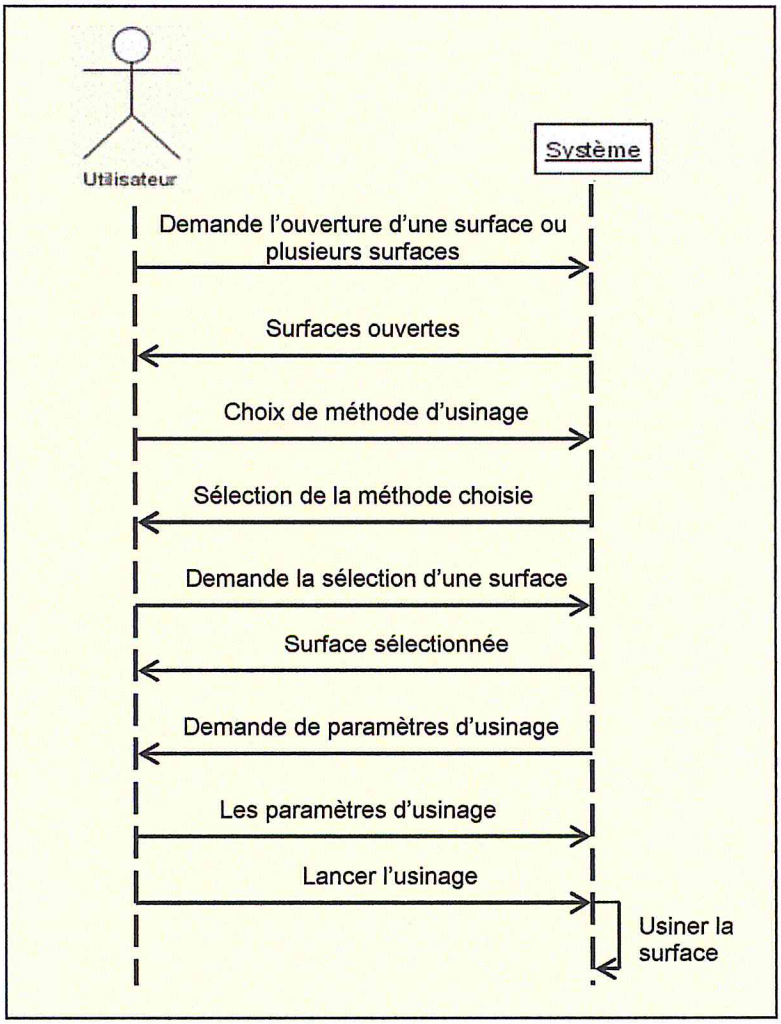


Figure 7.1: Le scénario d'usinage des surfaces gauche par la méthode Z constant.

La première étape consiste à ouvrir une surface à usiner. Dans notre cas, nous avons choisis la surface du demi vase (voir figure 7.1).

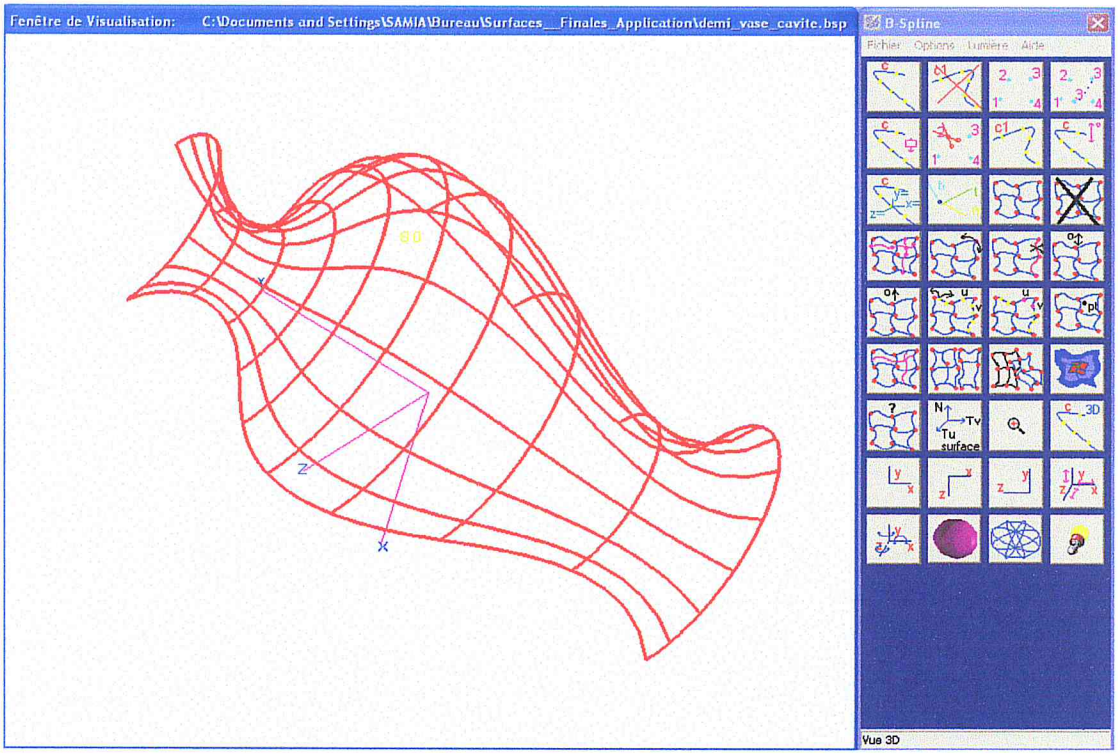


Figure 7.2: Ouverture d'une surface à usiner.

Pour usiner la surface ouverte (figure 7.2), nous devons ouvrir la fenêtre d'usinage (figure 7.3). Dans cette fenêtre, nous choisissons la méthode d'usinage, pour notre cas c'est la méthode Z-Constant, et par la suite nous sélectionnons la surface soit par la souris ou par le label de la surface pour arriver à l'introduction des différents paramètres d'usinage.

Pour ce test, nous utilisons les paramètres suivants :

- La distance entre plans est égale à 1 millimètre ;
- Une triangulation uniforme ;
- Le nombre de triangles par ligne est égal à 100 ;
- Le nombre de triangle par colonne est égal à 100 ;
- Un mode d'usinage on opposition ;
- Un sens trigonométrique pour la rotation de la broche
- Un engagement égale à 10 millimètres ;
- Un dégagement égale à 10 millimètres;
- Un outil de rayon égale à 10 millimètres et longueur égale à millimètres 20.

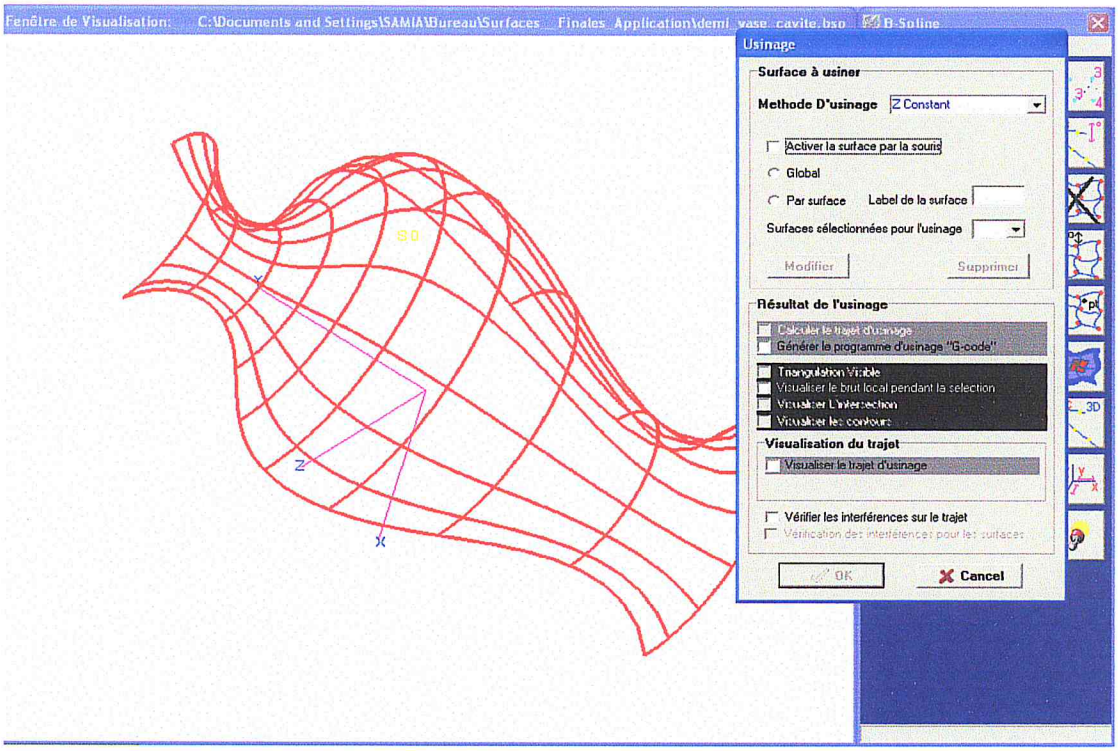


Figure 7.3: Demande l'usinage de la surface.

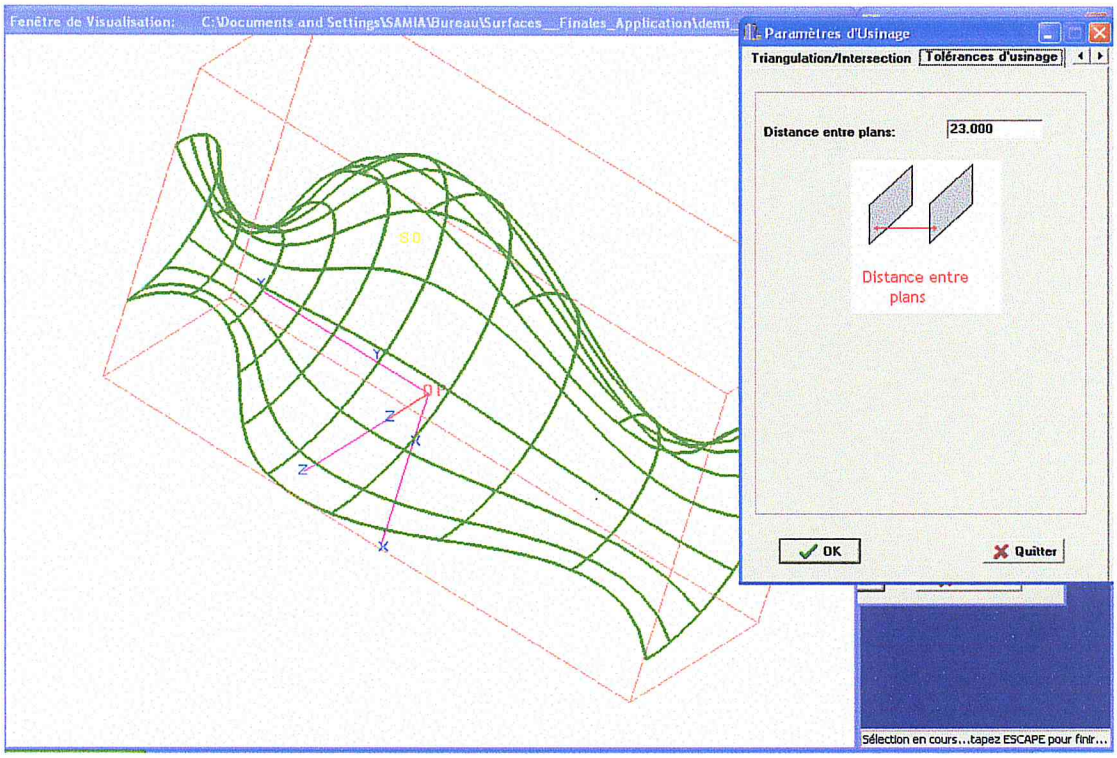


Figure 7.4: Définir les paramètres d'usinage.

Après la définition des différents paramètres d'usinage de la surface (figure 7.4), nous retournons à la fenêtre d'usinage pour choisir les différents résultats à visualiser et pour lancer l'usinage.

Si pendent l'usinage ils existent des interférences globale ou locale sur le trajet d'usinage, le système informe l'utilisateur automatiquement de l'existence de ces problèmes avec l'apparition d'une fenêtre qui donne la possibilité de visualiser ces interférences (figure7.5).

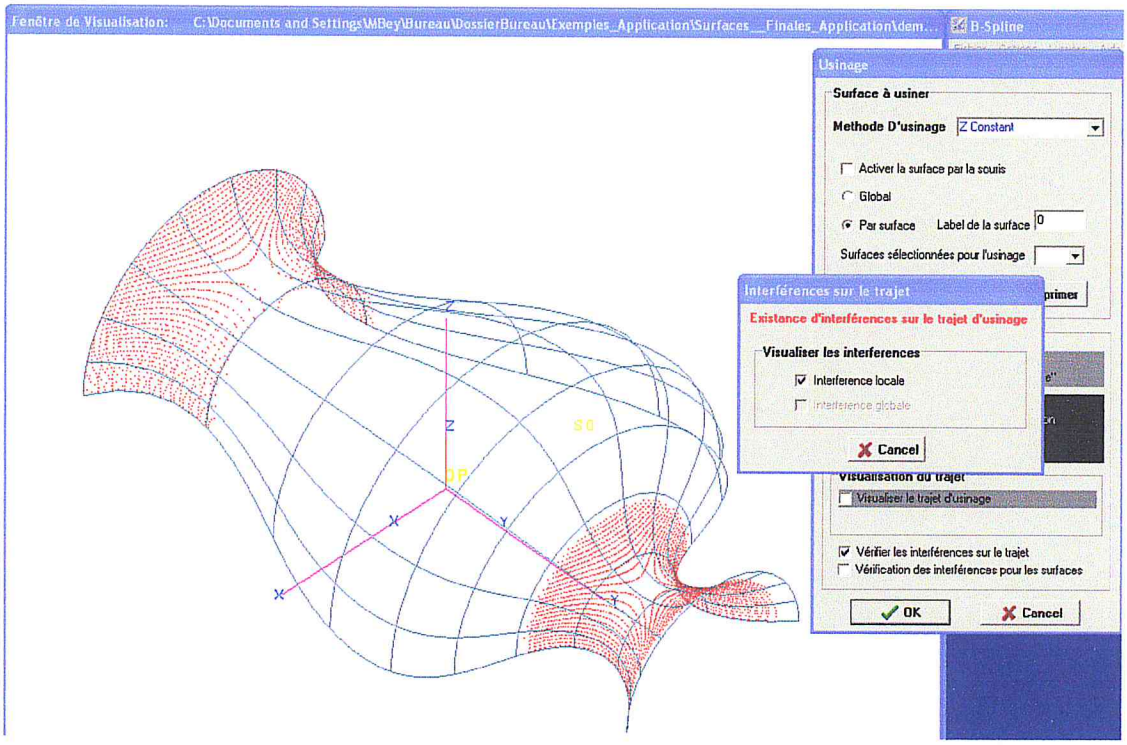


Figure 7.5: Visualisation des interférences.

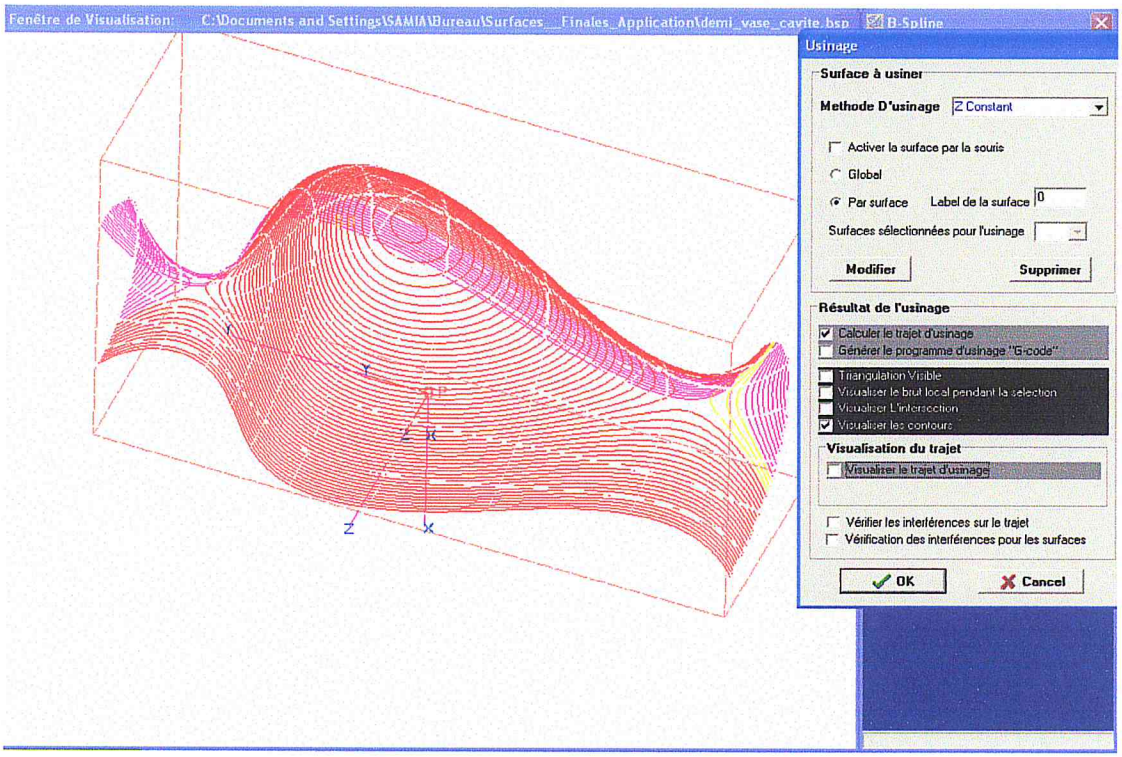


Figure 7.6: Visualisation des contours.

Le premier résultat de l'usinage de la surface est un ensemble de contours comme le montre la figure7.6. Le trajet d'usinage est calculé à partir de ces contours (figure 7.7).

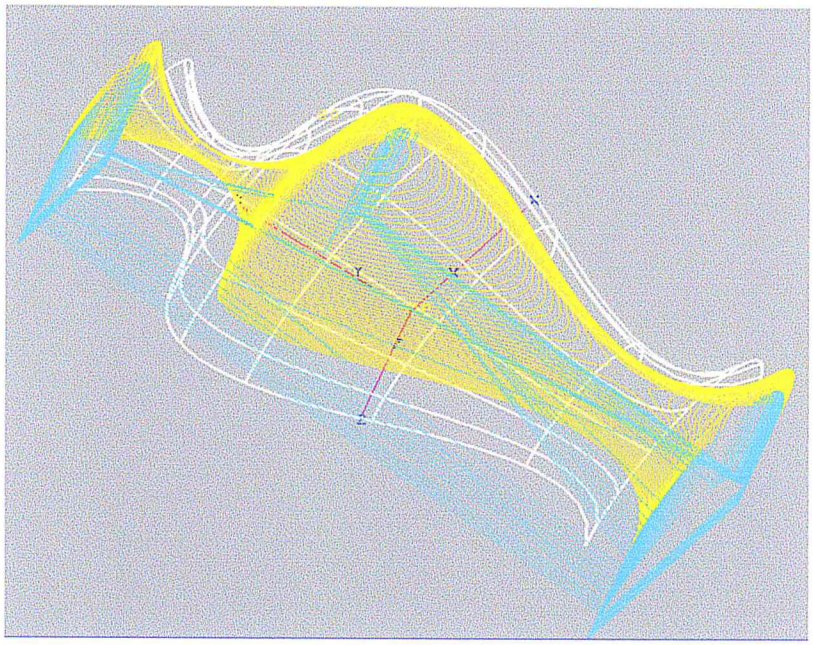


Figure 7.7: Le trajet d'usinage.

✓ Cas d'utilisation de simulation d'usinage :

La simulation d'usinage par plan ne peut se faire qu'après l'usinage de la surface. Pour tester l'implémentation, nous simulons le scénario de simulation (figure 7.8). La figure 7.9 montre le résultat de la simulation par plan.

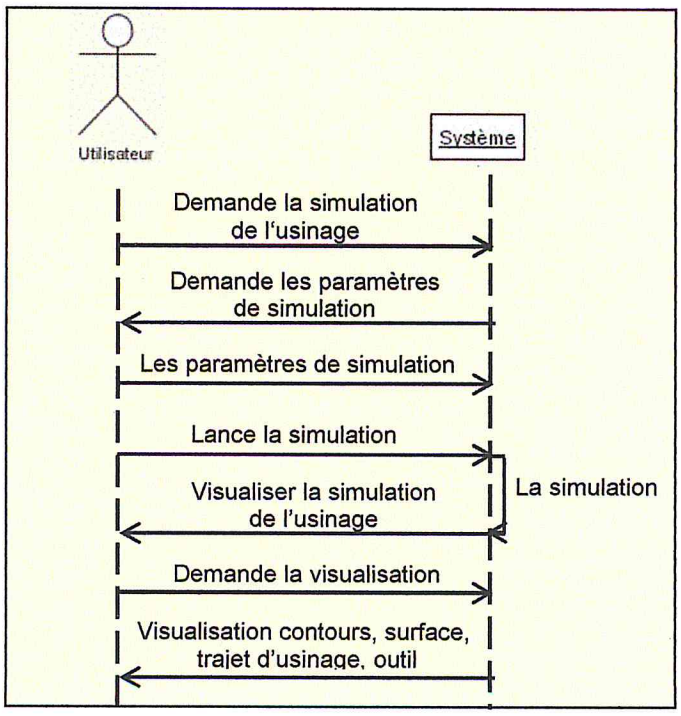


Figure 7.8: Le scénario de simulation.

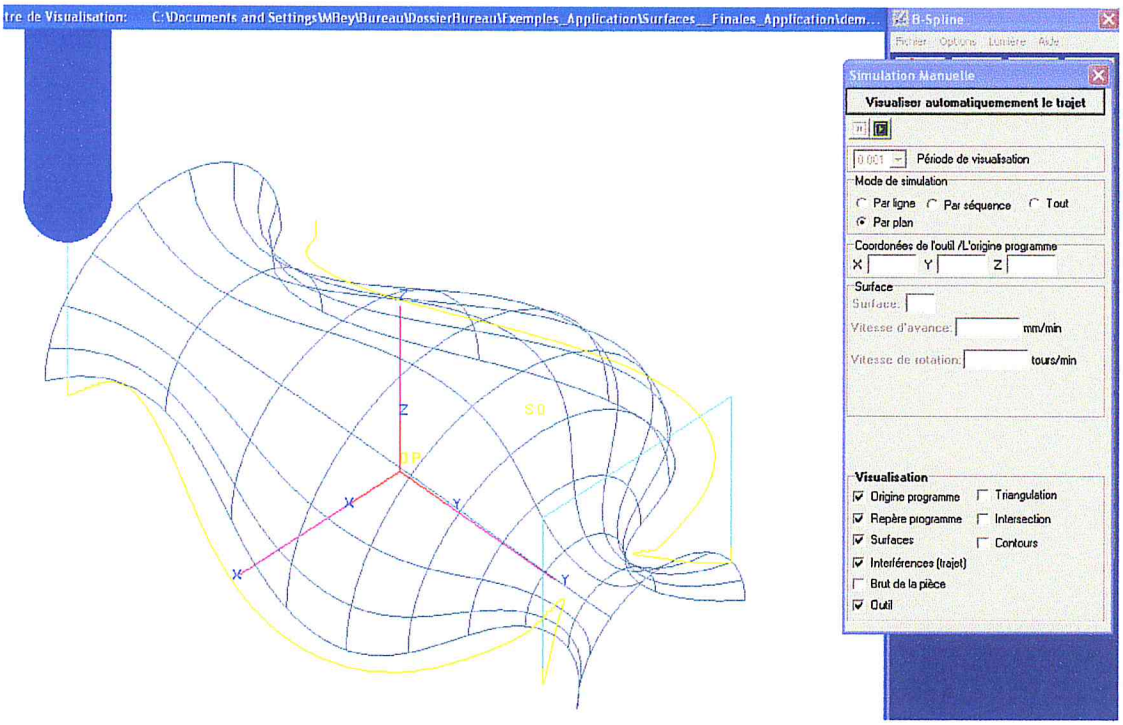


Figure 7.9: La simulation d'usinage par plan.

✓ Cas d'utilisation choix d'outils :

Le choix d'outil est demandé par un clic sur l'item de choix d'outils dans le menu principal (figure 7.11). Les résultats de choix d'outils sont deux tableaux des rayons d'outils théoriques (figure 7.12) qui peuvent être utilisés sans avoir des problèmes d'interférences locales, le premier pour chaque plan et le second pour chaque contour

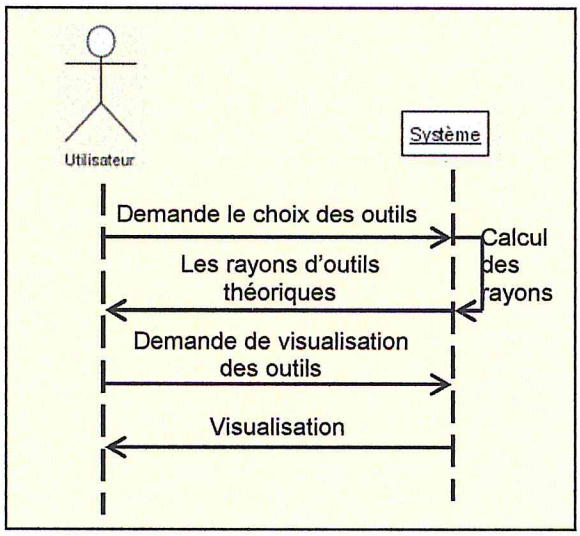


Figure 7.10: Le scénario de choix d'outils.

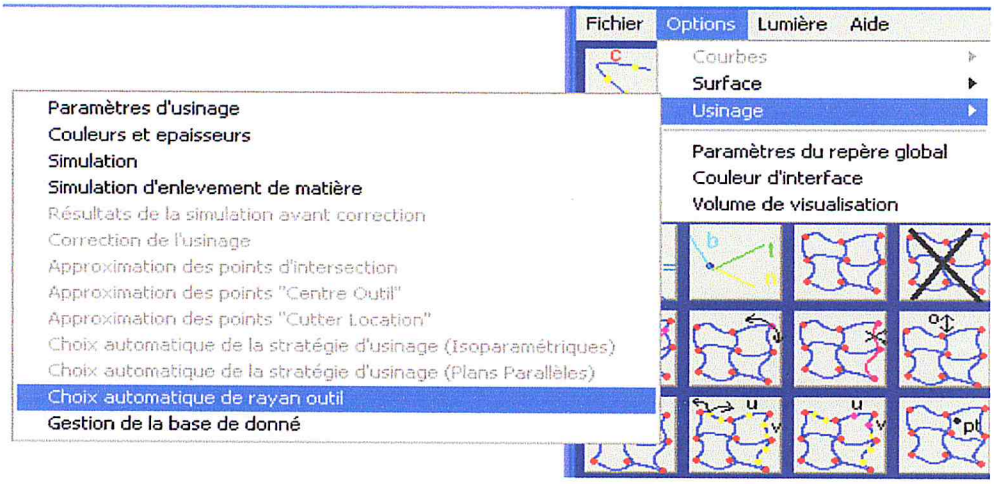


Figure 7.11:Demande le choix automatique d'outils.

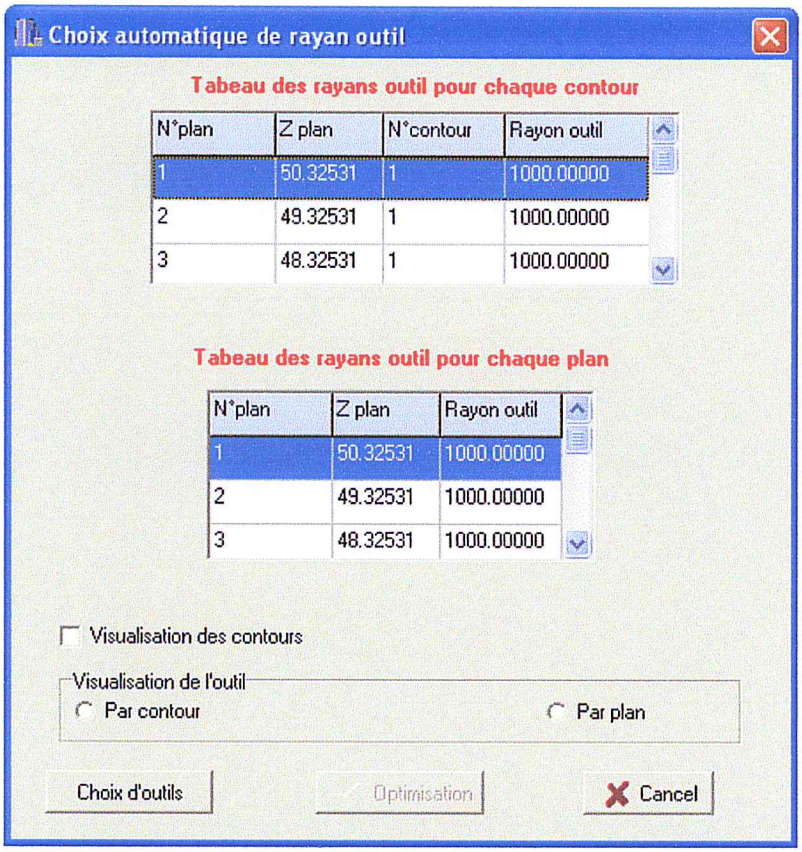


Figure 7. 12: Les rayons d'outils théoriques.

✓ Cas d'utilisation de choix d'outils à partir de la base de données :

Après la demande du choix d'outils à partir de la base de données, il faut choisir le type d'outils à utiliser (figures 7.14, 7.15). Dans notre cas, nous choisissons le type d'attachement cylindrique.

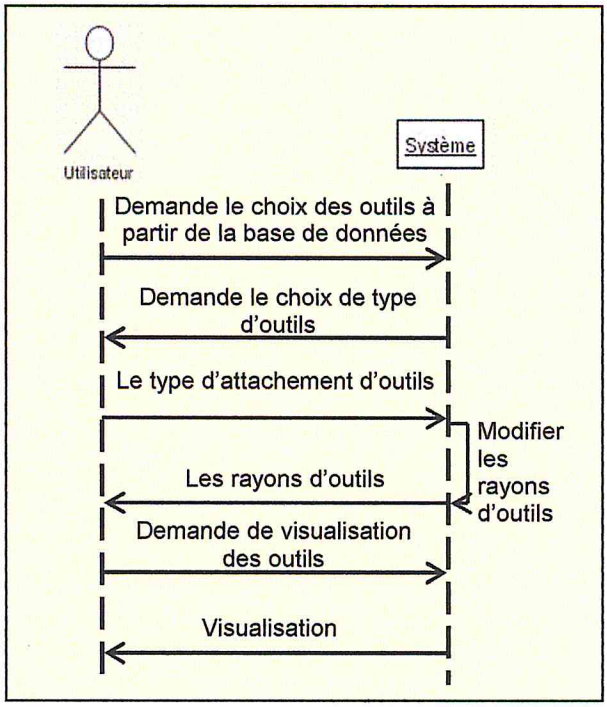


Figure 7.13: Le scénario de choix d'outils à partir de la base de donnée.

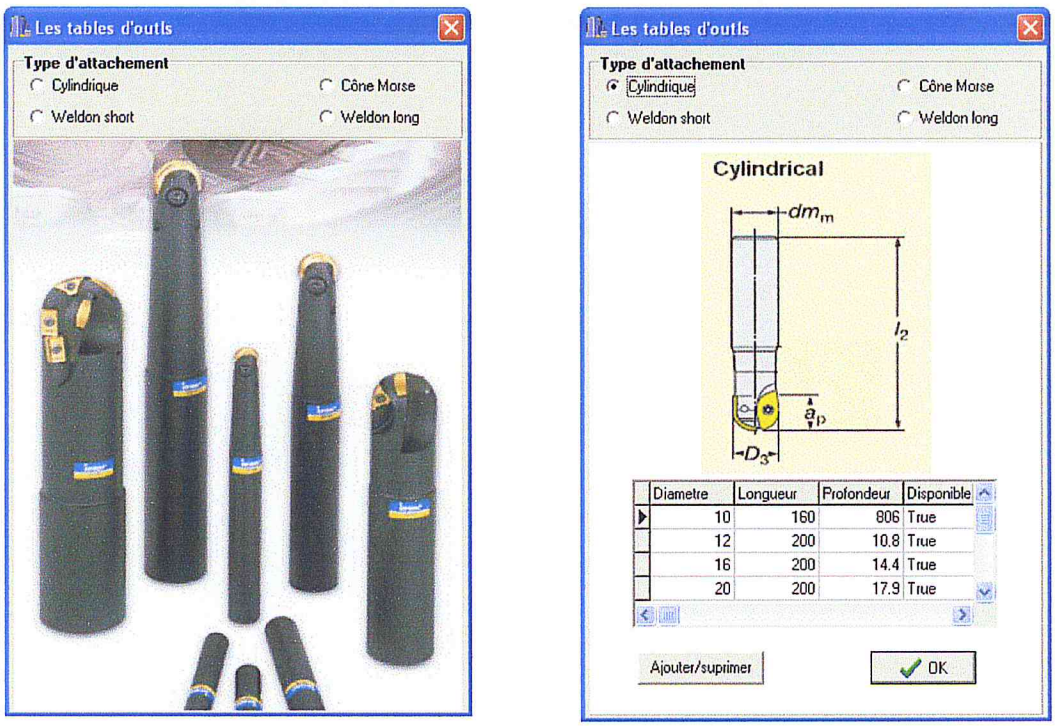


Figure 7.14: Le choix d'un type d'outils.

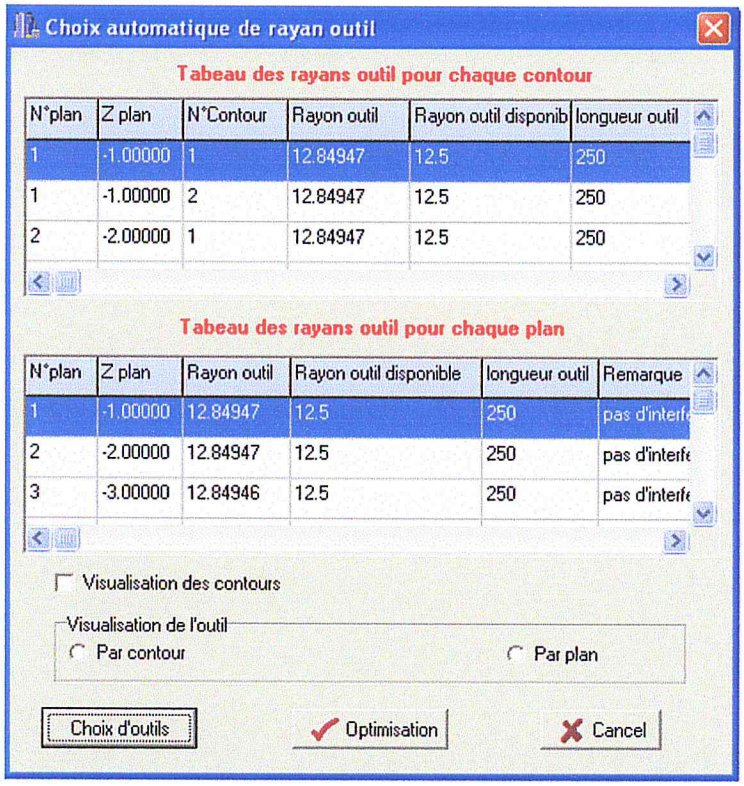


Figure 7.15: Les rayons d'outils disponibles dans la base de données.

✓ Cas d'utilisation d'optimisation de choix d'outils :

L'optimisation de choix d'outils (figure7.17) consiste à trouver la combinaison d'outils optimale(figure 7.18). Cette combinaison est déterminée après le calcul de temps d'usinage pour toutes les combinaisons possibles.

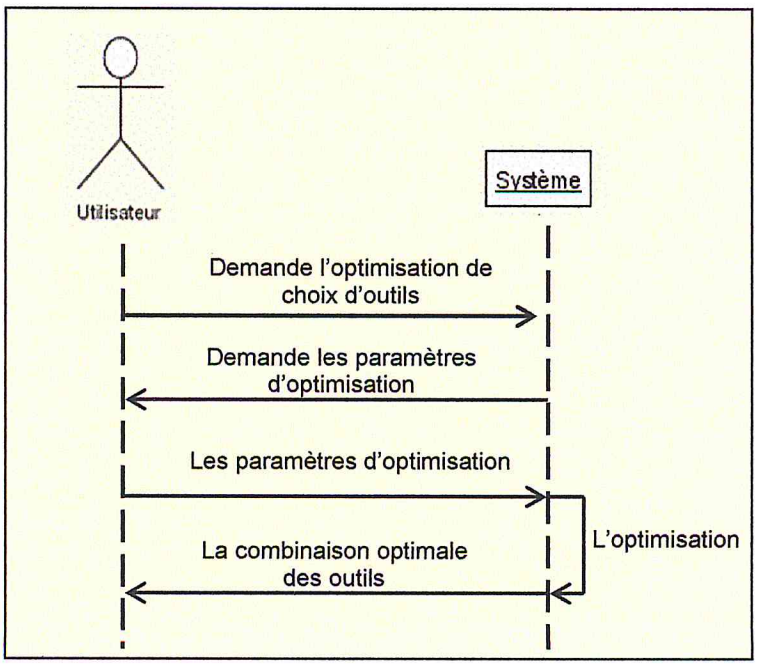


Figure 7.16: Le scénario de l'optimisation de choix d'outils.

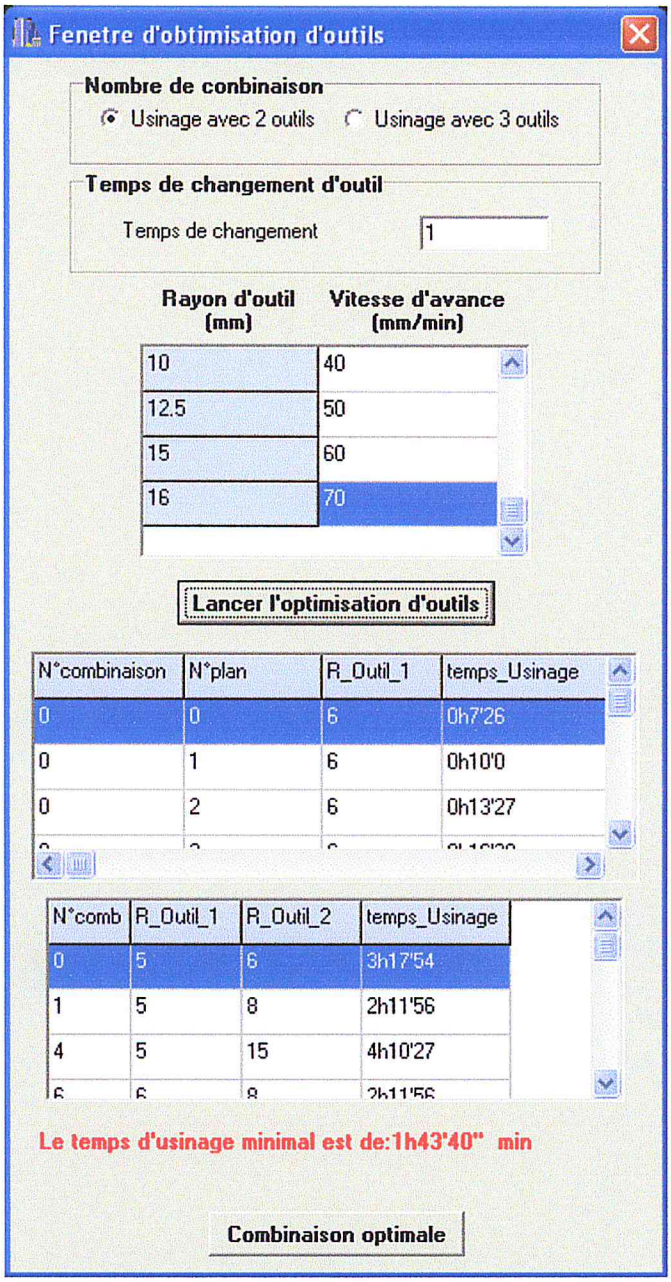


Figure 7.17: Lancer l'optimisation d'outils.

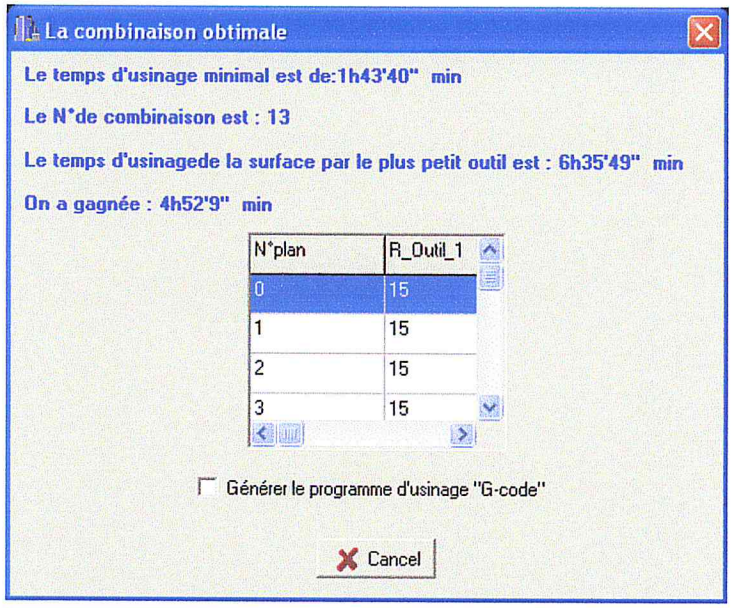


Figure 7.18: La combinaison optimale.

✓ Cas d'utilisation de gestion de la base de données :

Le déroulement de ce scénario consiste à faire des suppressions, des insertions et des modifications des tables d'outils (figures 7.20, 7.217,7.22).

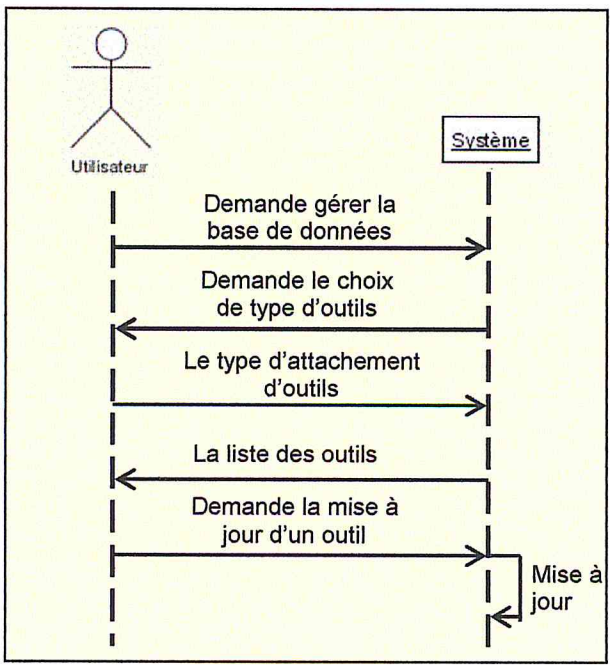


Figure 7.19 : Scénario pour gérer la base de données.

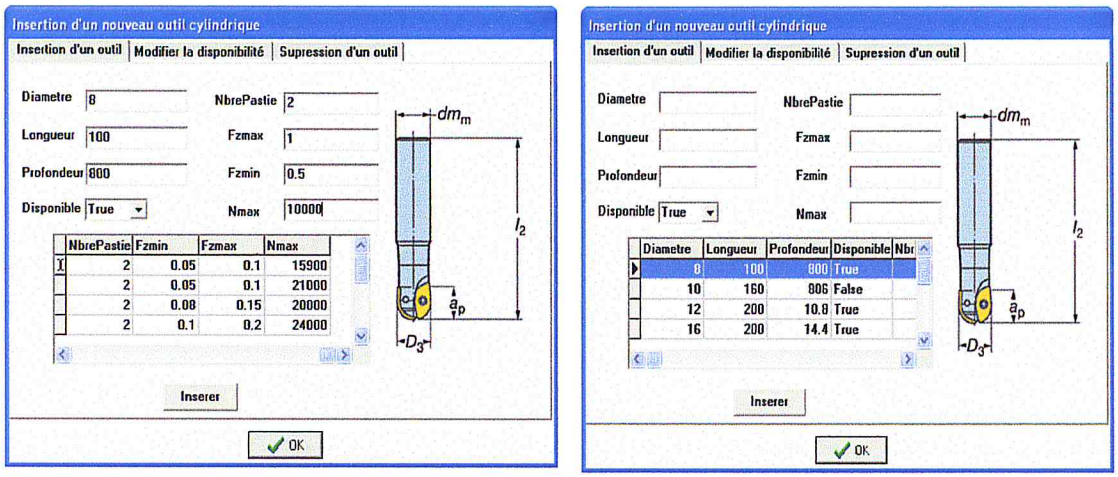


Figure 7.20 : Insertion d'un nouveau outil.

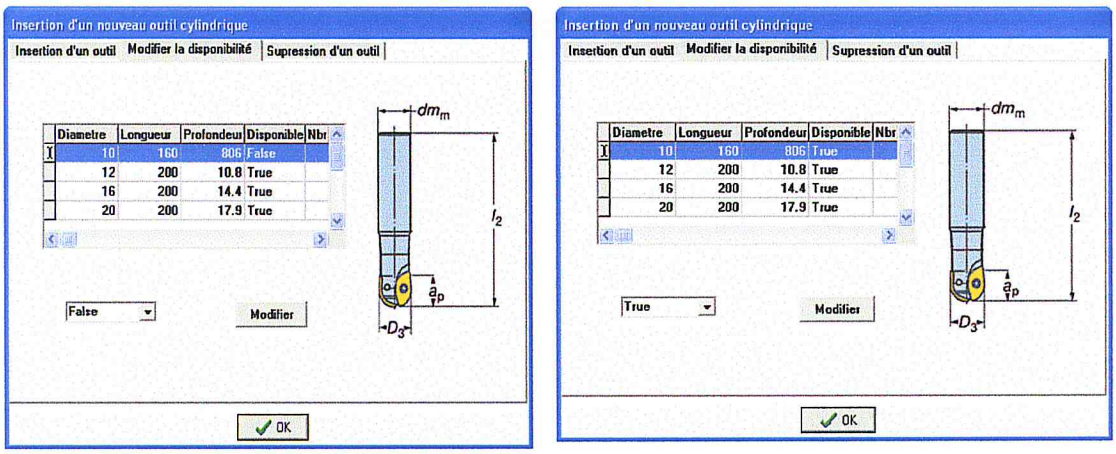


Figure 7.21 : Modification de la disponibilité d'un outil.

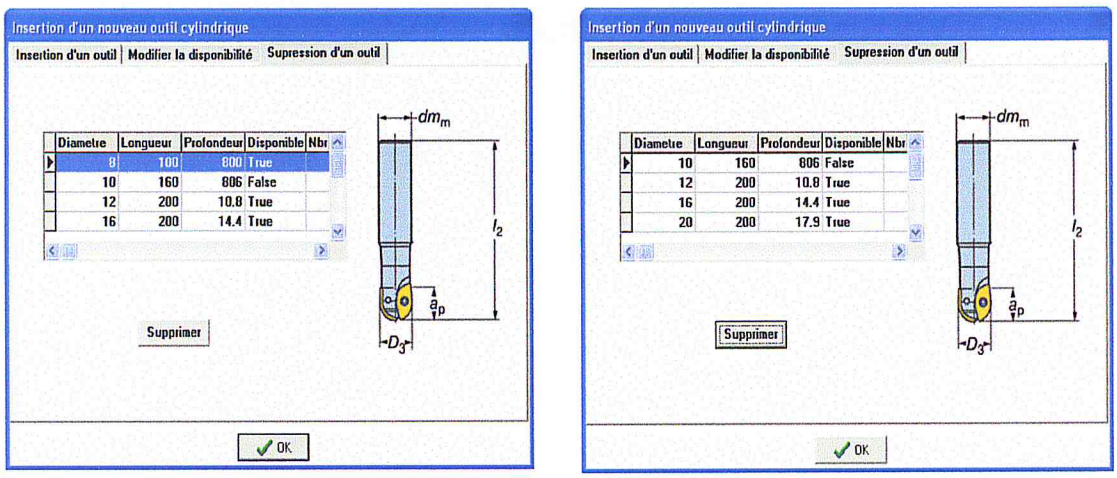


Figure 7.22 : Suppression d'un outil.

III. CONCLUSION :

Dans ce chapitre nous avons pris l'exemple d'une surface et nous avons tester et valider toutes les fonctions développées. Ces tests ont permis de montrer l'importance de l'outil informatique dans l'automatisation de l'usinage des surfaces gauches.

CONCLUSION GENERELE

Le travail que nous avons présenté dans ce mémoire consiste en une automatisation de l'opération d'usinage en finition des surfaces gauches avec la méthode Z-Constant sur des fraiseuses à commande numérique à 03 axes.

Pour cela, nous avons mené une étude bibliographique sur les différentes méthodes de conception et de représentation des surfaces gauches en appuyant notre analyse sur les surfaces B-Spline et NURBS. Ensuite nous avons présenté l'architecture des fraiseuses à commande numérique et leur langage de programmation ainsi que les différentes méthodes d'usinage et les caractéristiques de la méthode d'usinage en Z-Constant. A la fin, nous avons montré les différentes fonctions offertes par l'application logicielle développée.

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

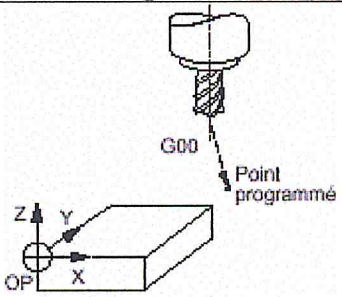
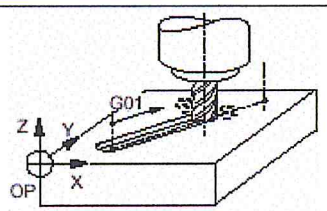
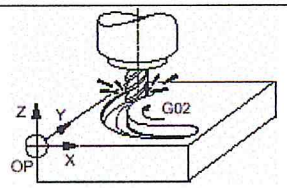
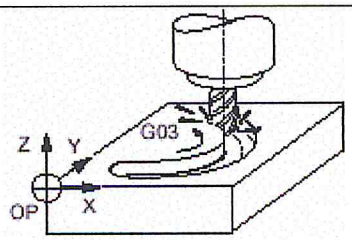
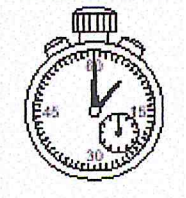
- ✓ D'automatiser la génération des fichiers G-Code.
- ✓ D'usiner les surfaces gauches avec la méthode Z-Constant.
- ✓ De détecter automatiquement les interférences el les collisions avant l'usinage.
- ✓ D'optimiser la trajectoire d'outils.
- ✓ De modifier les différents paramètres d'usinage.
- ✓ De simuler virtuellement les mouvements de l'outil par rapport à la surface et de calculer les temps d'usinage.
- ✓ De choisir automatiquement les outils optimums pour usiner chaque plan sans avoir des interférences.
- ✓ De choisir la combinaison optimale des outils permettant d'usiner la totalité de la surface en un minimum de temps.
- ✓ De gérer une base de données des outils.

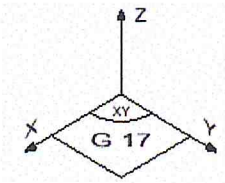
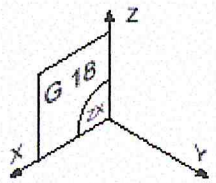
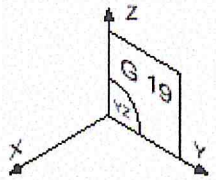
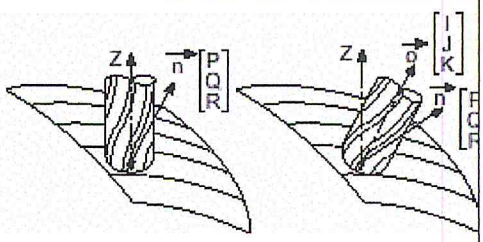
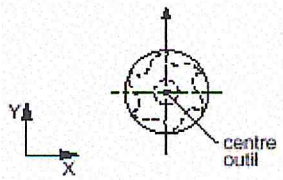
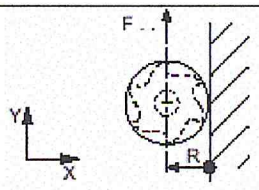
En perspective de notre travail, nous recommandons de traiter les points suivants :

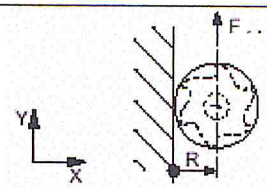
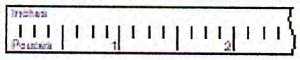
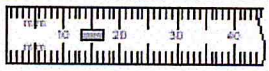
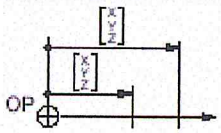
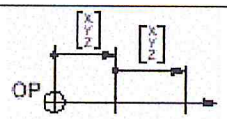
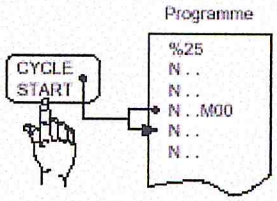
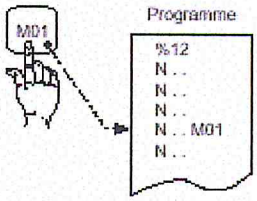
- ✓ Variation adaptative de la vitesse d'avance en fonction de la quantité de matière enlevée.
- ✓ Simulation d'enlèvement de matière lors de l'usinage avec des outils de différentes formes (cylindriques et toriques).
- ✓ Usinage des surfaces gauches sur des fraiseuses à 5 axes.
- ✓ Automatisation du choix des outils pour les différentes régions d'une surface.

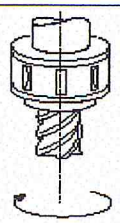
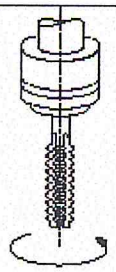
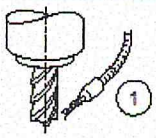
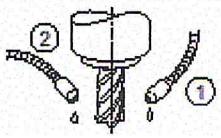
Annexe A

TABLEAU RECAPITULATIF DES PRINCIPALES FONCTIONS

Fonctions	Figures
G00 : Interpolation linéaire à vitesse rapide. **Fonction modale** **Syntaxe :** N.. [G90/G91] **G00** [□±] X.. Y.. Z..	 *Figure A.1 : La fonction G00.*
G01 : Interpolation linéaire à vitesse d'avance. Programmée **Fonction modale** **Syntaxe :** N.. [G90/G91] **G01** [□±] X.. Y.. Z.. [F..]	 *Figure A.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 A.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 A.4 : La fonction G03.*
G04 : Temporisation programmable. **Fonction modale** **Syntaxe :** N.. **G04** F..	 *Figure A.5 : La fonction G04.*

G17* : Choix du plan XY **Fonction modale** **Syntaxe :** N.. **G17**	 *Figure A.6. La fonction G17.*
G18 : Choix du plan ZX **Fonction modale** **Syntaxe :** N.. **G18**	 *Figure A.7 : La fonction G18.*
G19 : Choix du plan YZ **Fonction modale** **Syntaxe :** N.. **G19**	 *Figure A.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 A.9 : La fonction G29.*
G40 : Annulation de correction de rayon **Fonction modale** **Syntaxe :** N.. [G00/G01] **G40** X.. Y.. Z..	 *Figure A.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 A.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 A.12 : La fonction G42.*
G70 : Programmation en pouce **Fonction modale** **Syntaxe :** N.. **G70**	 *Figure A.13 : La fonction G70.*
G71 : Programmation en métrique **Syntaxe :** N.. **G71**	 *Figure A.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 A.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 A.16 :La fonction G91.*
M00 : Arrêt programmé . **Fonction non modale** **Syntaxe** : N.. [G40] **M00** [$0 ...]	 *Figure A.17 : La fonction M01.*
M01 : Arrêt programmé optionnel. **Fonction modale** **Syntaxe** : N.. [G40] **M01** [$0 ...]	 *Figure A.18 : La fonction M02.*
M02 : Fin de programme. **Fonction modale** **Syntaxe** : N.. **M02**	

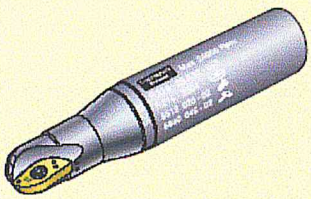
M03 : Rotation de broche sens antitrigonométrique **Fonction modale** **Syntaxe** : N.. **M03**	 *Figure A.19 : La fonction M03.*
M04 : Rotation de broche sens trigonométrique **Fonction modale** **Syntaxe** : N.. **M04**	 *Figure A.20 : La fonction M04.*
M05 : Arrêt de broche **Fonction modale** **Syntaxe** : N.. **M05**	
M08 : Arrosage numéro 1. **Fonction modale avant** **Syntaxe** : N.. **M08**	 *Figure A.21 : La fonctionM08.*
M09 : Arrêt des arrosages 1 et 2 **Fonction non modale après** **Syntaxe** : N.. **M09**	 *Figure A.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)	

Annexe B

Outils hémisphériques **Sandvik** utilisés dans l'usinage des surfaces gauches

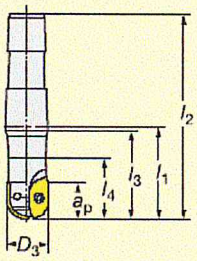
CoroMill® ball nose endmill

Diameter 12—50 mm

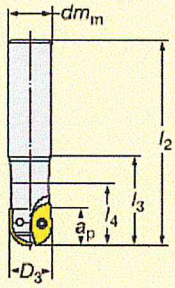


Machine tools: All types
Materials: All types
Inclination angle: –10°

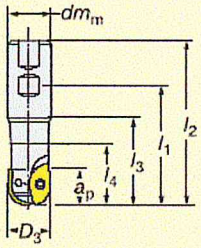
Morse Taper



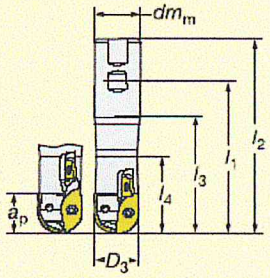
Cylindrical



Weldon



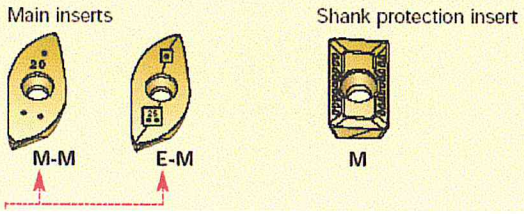
Weldon

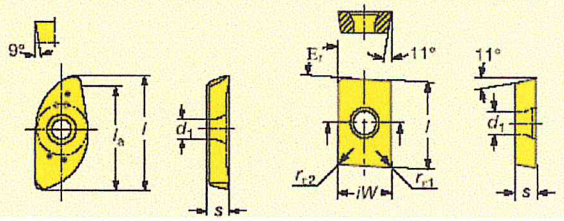


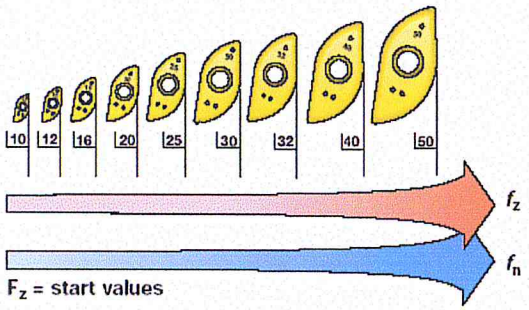
l_1 = programming length

Shank type	Ordering code	D_3 mm			Dimensions, mm								Inserts[1] Main insert	Shank protection insert
					kg	l_1	l_2	l_3	l_4	dm_m	Max a_p	n_{max} [2]		
Cylindrical	R216- 10A16-050	10		2	0,3	–	160	50	20	16	8,6	15900	R216-10 02 ...	–
	12A20-045	12		2	0,4	–	200	45	20	20	10,8	21000	12 02 ...	–
	16A20-045	16		2	0,4	–	200	45	27	20	14,4	20000	16 03 ...	–
	20A25-055	20		2	0,6	–	200	55	33	25	17,9	24000	20 T3 ...	–
	25A32-065	25		2	1,3	–	250	65	40	32	22,3	24000	25 04 ...	–
	30A32-070	30		2+1	1,4	–	250	70	50	32	26,9	19500	30 06 ...	APMT 160408-M
	32A32-070	32		2	1,4	–	250	70	50	32	28,6	18500	32 06 ...	–
Weldon/ short	R216- 12B20-040	12		2	0,2	66	91	40	20	20	10,8	21000	R216-12 02 ...	–
	16B20-040	16		2	0,2	66	91	40	27	20	14,4	20000	16 03 ...	–
	20B25-050	20		2	0,3	75	107	50	33	25	17,9	24000	20 T3 ...	–
	25B25-060	25		2	0,4	85	117	60	40	25	22,3	24000	25 04 ...	–
	30B32-070	30		2+1	0,6	95	131	70	50	32	26,9	19500	30 06 ...	APMT 160408-M
	32B32-070	32		2	0,6	95	131	70	50	32	28,6	18500	32 06 ...	–
	40B40-100	40		2+2	1,3	131	171	100	69	40	36,5	8000	40 07 ...	APMT 160408-M
	50B40-100	50		2	2,0	131	171	100	–	40	44,6	7000	50 07 ...	–
	50M51-125	50		2+2	2,5	161	207	125	95	50,8	44,6	7000	50 07 ...	APMT 160408-M
Weldon/ long	R216- 12B20-060	12		2	0,2	86	111	60	20	20	10,8	21000	R216-12 02 ...	–
	16B20-060	16		2	0,2	86	111	60	27	20	14,4	20000	16 03 ...	–
	20B25-070	20		2	0,3	95	127	70	33	25	17,9	24000	20 T3 ...	–
	25B25-080	25		2	0,4	105	137	80	70	25	22,3	24000	25 04 ...	–
	30B32-100	30		2+1	0,7	125	161	100	80	32	26,9	19500	30 06 ...	APMT 160408-M
	32B32-100	32		2	0,8	125	161	100	82	32	28,6	18500	32 06 ...	–
	40B40-150	40		2+2	1,7	181	221	150	120	40	36,5	8000	40 07 ...	APMT 160408-M
	50M51-175	50		2+2	3,1	211	257	175	145	50,8	44,6	7000	50 07 ...	APMT 160408-M
Morse	R216- 12E2-040	12		2	0,1	45	104	40	20	2	10,8	21000	R216-12 02 ...	–
	16E2-060	16		2	0,1	65	124	60	27	2	14.4	20000	16 03 ...	–
	20E2-070	20		2	0,2	70	134	70	–	2	17,9	24000	20 T3 ...	–
	20E3-070	20		2	0,3	70	151	70	33	3	17,9	24000	20 T3 ...	–
	25E3-080	25		2	0,4	80	161	80	70	3	22,3	24000	25 04 ...	–
	32E3-100	32		2	0,5	105	181	100	–	3	28,6	18500	32 06 ...	–
	32E4-100	32		2+1	0,8	105	202,5	100	82	4	28,6	18500	32 06 ...	APMT 160408-M

Inserts for Coromill® ball nose endmill







Start values in feed per insert (f_z) should be adjusted depending on tool length, overall stability and engagement of the cutter.

Recommended feed

Insert size	f_z (feed in mm/tooth) Starting value	(min – max.)
10	0,05	(0,05 – 0,10)
12	0,05	(0,05 – 0,10)
16	0,08	(0,08 – 0,15)
20	0,10	(0,10 – 0,20)
25	0,12	(0,12 – 0,25)
30	0,15	(0,15 – 0,35)
32	0,15	(0,15 – 0,35)
40	0,20	(0,20 – 0,40)
50	0,25	(0,25 – 0,40)

Terminology and units — milling

D_c	= Cutting diameter	mm	h_m	= Average chip thickness	mm
l_m	= Machined length	mm	z_c	= *Effective nubmer of teeth*	*piece*
Dap	= Max cutting diameter at a specific depth	mm	k_{c1}	= Specific cutting force (for h_{ex} = 1 mm)	N/mm²
a_p	= Cutting depth	mm	n	= Spindle speed	rev/min
a_e	= Working engagement	mm	P_c	= Cutting power net	kW
v_c	= Cutting speed	m/min	η_{mt}	= Efficiency	
Q	= Metal removal rate	cm³/min	κ_r	= Major cutting edge angle	degrees
T_c	= Period of engagement	min	v_{c0}	= Constant for cutting speed	
z_n	= Total number of edges in the tool	piece	c_{vc}	= Correction factor for cutting speed	
f_z	= Feed per tooth	mm	m_c	= Rise in specific cutting force (k_c) as a function of chip thickness	
f_n	= Feed per revolution	mm	*iC*	= inscribed circle	
v_f	= Table feed (feed speed)	mm/min			
h_{ex}	= Max chip thickness	mm			

General formulas

Cutting speed (m/min)

$$v_c = \frac{\pi \times D_c \times n}{1000}$$

Spindle speed (rev/min)

$$n = \frac{v_c \times 1000}{\pi \times D_c}$$

Table feed (feed speed) (mm/min)

$$v_f = f_z \times n \times z_n$$

Feed per tooth (mm)

$$f_z = \frac{v_f}{n \times z_n}$$

Feed per revolution (mm/rev)

$$f_n = \frac{v_f}{n}$$

Removal rate (cm³)

$$Q = \frac{a_p \times a_e \times v_f}{1000}$$

Specific cutting force (N/mm²)

$$k_c = k_{c1} \times h_m^{-m_c}$$

Average chip thickness (mm) (Side and facemilling) when $a_e/D_c \leq 0{,}1$

$$h_m \approx f_z \sqrt{\frac{a_e}{D_c}}$$

Average chip thickness (mm) when $a_e/D_c \geq 0{,}1$

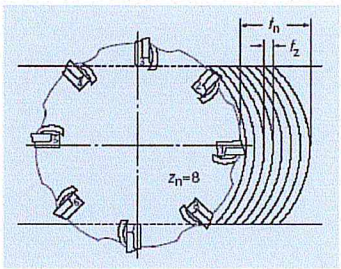
$$h_m = \frac{\sin \kappa_r \times 180 \times a_e \times f_z}{\pi \times D_c \times \arcsin\left(\frac{a_e}{D_c}\right)}$$

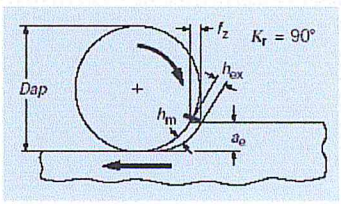
Machining time (min)

$$T_c = \frac{l_m}{v_f}$$

Net power (kW)

$$P_c = \frac{a_p \times a_e \times v_f \times k_c}{60 \times 10^6 \times \eta}$$





Annexe C

UML - Unified Modeling Language -

I. PRESENTATION D'UML :

I.1. Historique [22]:

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

Cette méthode représente un moyen de spécifier, représenter et construire les composantes d'un système informatique.

I.2. Définition [21]:

UML représente l'état de l'art des langages de modélisation objet. Il fournit les fondements pour spécifier, construire, visualiser et décrire les artefacts d'un système logiciel. Pour cela, UML se base sur une sémantique précise et sur une notation graphique expressive.

UML permet de modéliser de manière claire et précise la structure d'un système indépendamment de tout langage de programmation.

I.3. Avantages et inconvénients d'UML [19]:

I.3.1. Les points forts d'UML:

- UML est un langage formel et normalisé :
  - Gain de précision.
  - Gage de stabilité.
  - Encourage l'utilisation d'outils.
- UML est un support de communication performant :
  - Il cadre l'analyse.
  - Il facilite la compréhension de représentations abstraites complexes.
  - Son caractère polyvalent et sa souplesse en font un langage universel.

I.3.2. Les points faibles d'UML :

- La mise en pratique d'UML nécessite un apprentissage et passe par une période d'adaptation : même si l'Espéranto est une utopie, la nécessité de s'accorder sur des modes d'expression communs est vitale en informatique. UML n'est pas à l'origine des concepts objets, mais en constitue une étape majeure, car il unifie les différentes approches et en donne une définition plus formelle.
- Le processus (non couvert par UML) est une autre clé de la réussite d'un projet : or, l'intégration d'UML dans un processus n'est pas triviale et améliorer un processus est une tâche complexe et longue. Les auteurs d'UML sont tout à fait conscients de l'importance du processus, mais l'acceptabilité industrielle de la modélisation objet passe d'abord par la disponibilité d'un langage d'analyse objet performant et standard.

II. LA MODELISATION [21,19]:

II.1. Qu'est-ce qu'un modèle?

La modélisation consiste à créer une représentation simplifiée d'un problème: le modèle. Grâce au modèle il est possible de représenter simplement un problème, un concept et le simuler. La modélisation comporte deux composantes:

- L'analyse, c'est-à-dire l'étude du problème.
- La conception, soit la mise au point d'une solution au problème.

Le modèle constitue ainsi une représentation possible du système pour un point de vue donné.

II.2. La modélisation UML :

Le métamodèle UML fournit une panoplie d'outils permettant de représenter l'ensemble des éléments du monde objet (classes, objets, ...) ainsi que les liens qui les relie.

Toutefois, étant donné qu'une seule représentation est trop subjective, UML fournit un moyen astucieux permettant de représenter diverses projections d'une même représentation grâce aux vues.

Une vue est constituée d'un ou plusieurs diagrammes. On distingue deux types de vues:

- Les vues statiques, c'est-à-dire représentant le système physiquement :
  - diagrammes d'objets ;
  - diagrammes de classes ;
  - diagrammes de cas d'utilisation ;
  - diagrammes de composants ;
  - diagrammes de déploiement.
- Les vues dynamiques, montrant le fonctionnement du système
  - diagrammes de séquence ;
  - diagrammes de collaboration ;
  - diagrammes d'états-transitions ;
  - diagrammes d'activités.

II.2.1. 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 C.1).

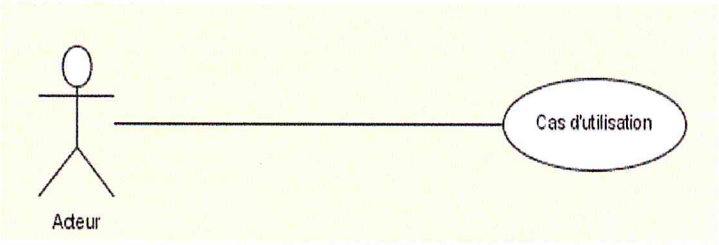


Figure C.1: cas d'utilisation.

Les cas d'utilisation peuvent être structurés et reliés par trois types principaux de relations :

- 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.

II.2.2. Diagramme de classes :

Le diagramme de classes représente l'architecture conceptuelle du système : il décrit les classes que le système utilisé (voir figure C.2), ainsi que leurs liens, que ceux-ci représentent un emboîtage conceptuel (héritage, marqué par une flèche terminée par un triangle) ou une relation organique (agrégation, marquée par une flèche terminée par un diamant).

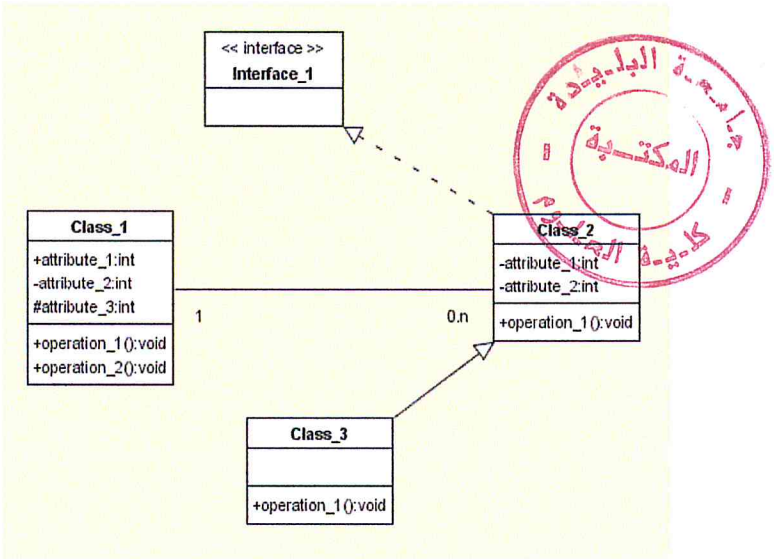


Figure C.2: Diagramme de classes.

II.2.3. Diagramme d'objets :

Ce type de diagramme UML montre des objets (instances de classes dans un état particulier) et des liens (relations sémantiques) entre ces objets. Ce diagramme sert essentiellement en phase exploratoire, car il possède un très haut niveau d'abstraction.

II.2.4. Diagramme de composants :

Les diagrammes de composants permettent de décrire l'architecture physique et statique d'une application en terme de modules (voir figure C.3) : fichiers sources, librairies, exécutables, ...etc. Ils montrent la mise en oeuvre physique des modèles de la vue logique avec l'environnement de développement.

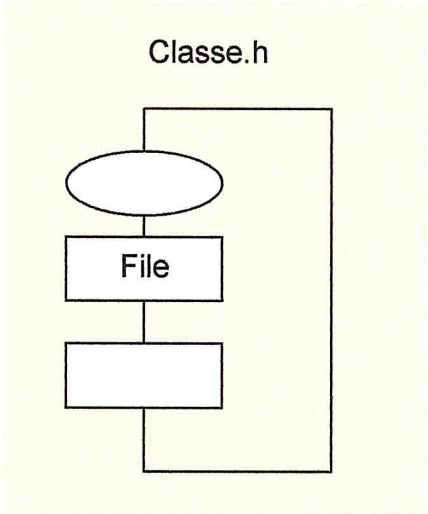


Figure C.3: Représentation d'un composant.

II.2.5. Diagramme de déploiement :

Les diagrammes de déploiement montrent la disposition physique des matériels qui composent le système et la répartition des composants sur ces matériels. Les ressources matérielles sont représentées sous forme de noeuds. Les noeuds sont connectés entre eux à l'aide d'un support de communication. La nature des lignes de communication et leurs caractéristiques peuvent être précisées.

II.2.6. Diagramme de collaboration :

Les diagrammes de collaboration montrent des interactions entre objets (figure C.4) (instances de classes et acteurs). Ils permettent de représenter le contexte d'une interaction, car on peut y préciser les états des objets qui interagissent.

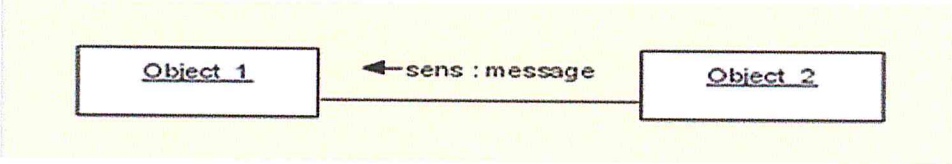


Figure C.4: Formalisme de base du diagramme de collaboration.

II.2.7. Diagramme de séquence :

Le diagramme de séquence représente la succession chronologique des opérations réalisées par un acteur : saisir une donnée, consulter une donnée, lancer un traitement ; il indique les objets que l'acteur va manipuler et les opérations qui font passer d'un objet à l'autre (figure C.5). On peut représenter les mêmes opérations par un diagramme de collaboration.

Diagramme de séquence et diagramme de collaboration sont deux vues différentes, mais logiquement équivalentes (on peut construire l'une à partir de l'autre), d'une même chronologie.

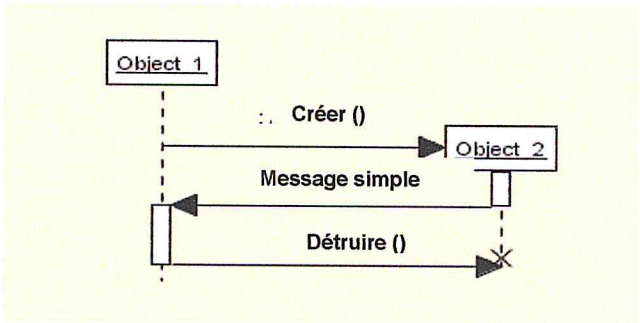
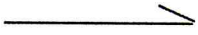
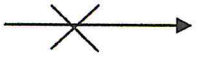
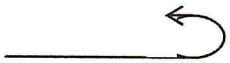


Figure C.5: Diagramme de séquence.

Le diagramme de séquence distingue 5 types de messages prédéfinis qui sont présentés dans le tableau suivant :

Tableau C.1 : Types de messages.

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. Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré.
 Délai	Bloque l'expéditeur pendant un temps donné, en attendant la prise en compte du message par le récepteur.

II.2.8.Diagramme d'etats-transitions :

Le diagramme d'état (voir figure C.6) représente la façon dont évoluent ("cycle de vie") durant le processus les objets appartenant à une même classe. La modélisation du cycle de vie est essentielle pour représenter et mettre en forme la dynamique du système.

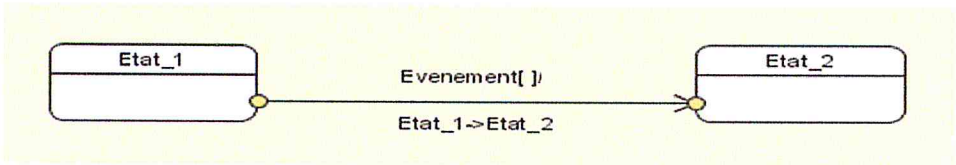


Figure C.6 : La notation des états, transition et événement.

II.2.9. Diagramme d'activités :

UML permet de représenter graphiquement (voir figure C.7) 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 une exécution d'un mécanisme, un déroulement d'étapes séquentielles. Le passage d'une activité vers une autre est matérialisé par une transition.

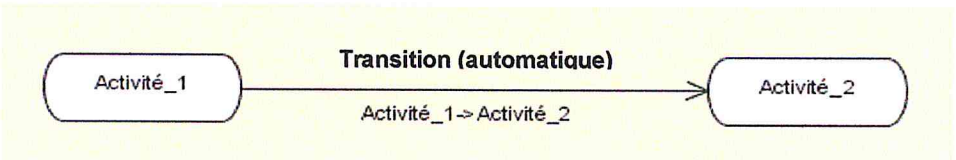


Figure C.7: La notation des activités et transition.

REFERENCES BIBLIOGRAPHIQUES

[1] M. BEY " ***Modélisation des courbes et des surfaces*** ". Rapport de Recherche 2000, CDTA.

[2] M. OUARET, F. BOUABDELLAH "***Conception et Réalisation d'un Outil de Modélisation des Surfaces B-Spline et NURBS*** ". Mémoire de fin d'étude à l'Institut d'Informatique. USTHB. Algérie 2002.

[3] S. HOUBANE, M. YALAOUI ***"Reconstruction des courbes et des surfaces B-Spline à partir d'un nuage de points"*** Mémoire du Projet de fin d'études. USTHB. Algérie 2002.

[4] A. LADJADJ, Ab_H. REZZAZ ***" Conception et réalisation d'un outil de modélisation des courbes B-Spline et NURBS"*** Mémoire du Projet de fin d'études. USTHB. Algérie 2001.

[5] Y. SAHMI A. SARIRETTE ***" Usinage_surfaces_gauches "*** Mémoire du Projet de fin d'études. USTHB. Algérie 2003.

[6] Dr. C.-K. Sheena ***"CS3621*** **Introduction to Computing with Geometry Notes"** Department of Computer Science Michigan Technological University, Décembre 2003.

[7] *GPA* " ***Fabrication Assistée Par Ordinateur***".

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

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

[10] C. Marty, C. Cassagnes, P. Marin. " ***La pratique des machine à commande numérique*** ". Edition Hermès, France 1993.

[11] E. Duc, E. Lefur. " ***Machines-outils à commande numérique, structure, modélisation et réglage*** ". Mémoire de DEA à l'école normale supérieur de Cachan, France, septembre 1997.

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

[13] E. DUC. ***" Evaluation des modes de génération des trajets outil pour usinage de formes gauches ".*** Mémoire de DEA à l'école normale supérieur de Cachan, France, juillet 1996.

[14] Ren. C. Luo and Yawei Ma. ***"Free Form Surface Representation and Machining for Parts".*** Centr for Robotics and Intelligent Machines. North Carolina State University.

[15] Susan X. Li Robert B. Jerard ***"Non-isoparametric Three Axis NC Tool Path Generation for Finish Machining of Sculptured Surfaces".*** Department of Mechanical Engineering. University of New Hampshire, Durham, NH 03824.

[16] S. BOUDJOUAD, N .TAFAT-BOUZID "***Méthode des plans parallèles pour l'usinage des surfaces gauches sur des fraiseuses à commande numérique à 3 axes"*** Mémoire du Projet de fin d'études. USTHB. Algérie 2004.

[17] Sang C. Park ***"Tool path generationf*or Z-constant contour machining ".***Cubic Technology Center , Ace Techno-Tower 1101 , Seoul, South Korea .2001.

[18] ***"Cycle de développement",*** un article de Wikipédia, l'encyclopédie libre.

[19] L.Piechocki ***"UML en français".***

[20] A. Strohmeier ***"CYCLE DE VIE DU LOGICIEL"*** Laboratoire de Génie Logiciel - Département d'Informatique. Ecole Polytechnique Fédérale de Lausanne Mars 2000.

[21] P.A. Muller, N. Gaertner***" Modélisation : objets avec UML "*** Edition : Osman Eyrolles Multimédia - OEM (23 mars 2000).

[22] www.commentçamarche.com , site web.