Contribution à la conception, la mise en oeuvre et l'amélioration des algorithmes de calcul des intersections de carreaux NURBS

Dominique Michel

Université de Metz
Faculté des sciences
U.F.R. M.I.M

Laboratoire de Recherche en Informatique de Metz

"CONTRIBUTION A LA CONCEPTION, LA MISE EN OEUVRE ET L'AMELIORATION DES ALGORITHMES DE CALCUL DES INTERSECTIONS DE CARREAUX NURBS"

THESE

soutenue le 16 Mars 1992
en vue d'obtenir le titre de

DOCTEUR DE L'UNIVERSITE DE METZ

SPECIALITE INFORMATIQUE

par

Dominique MICHEL

Composition du jury

P. BEZIER (examinateur)	: professeur honoraire au Conservatoire National des Arts et Métiers
I. COSTEA (examinatrice)	: professeur à California State University USA
Y. GARDAN (directeur de thèse)	: professeur à l'Université de Metz
A. ROUX (rapporteur)	: professeur à l'Université de Metz
M. VERON (rapporteur)	: professeur à l'Université de NancyI

REMERCIEMENTS

Arrive aujourd'hui l'épilogue d'un long parcours qui ne s'est pas déroulé dans l'uniformité. Il a été ponctué d'interrogations, de retours en arrière, d'attentes et de veilles inquiètes.

Aussi je tiens à remercier chacun qui par sa collaboration ou ses encouragements m'a permis de réaliser ce travail:

A Yvon Gardan à qui j'exprime ma profonde et sincère reconnaissance pour son immense patience, ses précieux conseils et la confiance qu'il m'a témoignée.

A madame Iléana Costéa et à messieurs André Roux et Michel Véron qui ont bien voulu juger ce travail.

A Pierre Bézier, précurseur de notre discipline dont la participation au jury m'émeut profondément.

A Ahmed, Myriam, Christian et Sylvain pour leurs conseils avisés et à tous les membres du L.R.I.Metz qui m'ont aidé dans cet exercice difficile.

INTRODUCTION

La modélisation des solides est un des aspects fondamentaux de la CFAO. Les solides désignent généralement les objets destinés à être fabriqués dans l'industrie. La modélisation des solides consiste à représenter virtuellement les objets avant leur fabrication. Plusieurs caractéristiques sont simulées: la forme géométrique, les propriétés physiques, l'aspect visuel (couleur , texture, etc...),... Ce dernier permet une représentation réaliste du produit fini; les propriétés physiques sont nécessaires entre autres à la simulation d'essais mécaniques. A la base de toutes les caractéristiques se trouve la forme de l'objet.

Les formes à représenter sont très variées; elles peuvent être des plus simples: sphères, cubes jusqu'aux plus complexes: fuselage d'avion, carrosserie de voiture, visage humain (imagerie médicale).

Les surfaces les plus simples, plans, quadriques sont insuffisantes pour représenter des formes complexes; les surfaces définies par équations cartésiennes (surfaces implicites) [SED 85] ne sont pas d'un grand intérêt pour la modélisation interactive car elles n'offrent pas une grande souplesse d'utilisation.

Des modèles mathématiques de surfaces ont donc été créés dans le double but de pouvoir définir une grande variété de formes et d'offrir des caractéristiques qui se prêtent bien au calcul numérique.

Ce sont les surfaces biparamétriques polynomiales $(u,v) \longmapsto (x(u,v),y(u,v),z(u,v))$ [BEZ 86] [COO87]. Les carreaux de Bézier sont les plus utilisés ainsi que les carreaux B-spline qui sont polynomiaux par morceaux et qui généralisent les premiers. Ces surfaces conviennent bien à la modélisation des formes libres mais les polynômes sont incapables de décrire exactement les quadriques, très utilisées en CFAO (cylindres, sphères, etc ...) [LEV 76].

Les carreaux B-spline rationnels ou NURBS (Non Uniform Rational B-Spline) qui généralisent les carreaux B-spline polynomiaux résolvent le problème; par conséquent ils permettent de représenter et de traiter de manière uniforme les formes libres et les quadriques [GAR 89] [GAR90] [MIL 86].

Cet exposé traite principalement des carreaux NURBS et des surfaces induites par ceux-ci, les carreaux de Bézier.

La représentation des surfaces ne constitue qu'une partie de la modélisation de la forme des solides. Les surfaces doivent être assemblées de façon cohérente pour former les solides.

Les modeleurs solides utilisés dans les systèmes de CAO sont divisés en deux catégories. Ils utilisent soit un arbre CSG (Constructive Solid Geometry) soit un modèle B-REP (Boundary Representation) ou représentation par les limites [FOL 82] [GAR 87] [GAS 90].

Dans ce dernier modèle les solides sont connus par leur "peau". Cette peau est constituée de facettes adjacentes, en grossière comparaison comme un assemblage de tôles forme la coque d'un bateau.
La liberté de créer des objets complexes implique la nécessité de pouvoir combiner plusieurs solides pour en former de nouveaux. Les combinaisons sont les opérations booléennes (réunion, intersection, différence, ...). Celles ci impliquent de savoir découper et "retailler" les facettes. Ces dernières, dans le cas général, sont donc des carreaux restreints (trimmed patches), c'est à dire des parties de carreaux NURBS et non pas des carreaux complets qui ne se prêtent pas à la modélisation de facettes présentant plus de quatre cotés ou des trous (car topologiquement ce sont des carrés)[CAR 82] [CAS 87].
Le problème de l'intersection des surfaces apparaît donc comme un élément clé de la modélisation des solides. Dans cet exposé nous proposons une solution au problème de l'intersection de deux carreaux NURBS.

Dans la première partie sont définis les principaux types de surfaces, les carreaux de Bézier, les carreaux B-spline et les carreaux rationnels. Les définitions mathématiques essentielles sont énoncées; toutes les propriétés nécessaires à la suite de la présentation sont rappelées et les algorithmes fondamentaux sont décrits. Le lecteur familiarisé avec les carreaux biparamétriques n'y retrouvera que des résultats classiques mis à part une étude de la propagation d'erreur de plusieurs algorithmes par une méthode probabiliste.

La formulation mathématique du problème est décrite dans la deuxième partie. Les hypothèses et les notations sont précisées. Certaines propriétés fondamentales des courbes et des surfaces sont rappelées et les différentes représentations du résultat (les courbes d'intersection) sont exposées. Enfin la notion très importante de point d'intersection singulier est introduite.

Les approches classiques du problème de l'intersection surface-surface sont abordées dans la troisième partie. Les notions de filtre et de volume englobant qui sont universellement utilisées pour optimiser les algorithmes de calcul d'intersection sont introduites. La méthode associant subdivisions récursives et intersections de facettes planes est décrite. Des solutions spécifiques aux problèmes intersection surface paramétrique-surface paramétrique et intersection surface paramétrique-surface implicite sont ensuite exposées.

Les méthodes proposées sont décrites dans la quatrième partie. Un algorithme qui associe simplification des carreaux et dichotomie récursive est exposé; il décompose simultanément

deux carreaux NURBS en carreaux de Bézier biquadratiques. Ensuite est présenté le calcul de l'intersection des carreaux biquadratiques, puis des carreaux bicubiques à l'aide des résolvants. Les propriétés des courbes de Bézier sont utilisées pour séparer les racines des équations réelles. Une méthode de suivi de courbe est ensuite décrite; elle correspond à la solution du problème intersection surface paramétrique-surface paramétrique. Les différentes méthodes sont finalement associées pour former un algorithme général.

Enfin dans la cinquième partie les différentes phases de la résolution du problème sont résumées. Un bilan sur les différentes approches du problème est dressé; les difficultés majeures sont soulignées. L'intérêt et l'originalité des solutions que nous proposons y sont discutées. Enfin l'orientation future de nos travaux est évoquée.

I SURFACES BIPARAMETRIQUES

I.1 DEFINITIONS ET PROPRIETES DES CARREAUX DE SURFACE

I.1.1 Définition d'un carreau de Bézier

I.1.1.1 Définition des polynômes de Bernstein

Soit m un entier fixé non nul, soit $i \in [0,1]$
On appelle polynôme de Bernstein la quantité:

$B_i^m(u) = C_m^i u^i (1 - u)^{m-i}$ où u varie dans l'intervalle [0,1]

La famille $\{ B_0^m, \ldots, B_m^m \}$ forme une base de l'espace vectoriel des polynômes de degré inférieur ou égal à m

Propriétés des polynômes de Bernstein

1) $\sum_{i=0}^{m} B_i^m(u) = 1 \quad \forall u$

2) $B_i^m(u) \geq 1 \quad \forall u, \forall i$

3) $B_i^m(0) = 0$ si $i > 0$ et $B_0^m(0) = 1$
$B_i^m(1) = 0$ si $i < m$ et $B_m^m(1) = 1$

4) Dérivation

$[B_i^m]'(u) = m [B_{i-1}^{m-1}(u) - B_i^{m-1}(u)]$

5) Récurrence

$B_i^m(u) = u \, B_{i-1}^{m-1}(u) + (1- u) B_i^{m-1}(u)$

I.1.1.2 Définition d'un carreau de Bézier

On appelle carreau de Bézier la surface $\mathcal{B}$ définie par

$$\sigma: [0, 1] \times [0, 1] \longrightarrow \mathbf{R}^3$$

$$(u, v) \longmapsto \sigma(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{P}_{ij}\ B_i^m(u).\ B_j^n(v)$$

où les $\mathbf{P}_{ij}$ [1] sont des points de $\mathbf{R}^3$. L'ensemble des $\mathbf{P}_{ij}$ est appelé réseau ou polyèdre des points de contrôle.
La surface est de degré m en u et n en v. Par la suite nous dirons qu'elle est de degré mxn [BEZ 86].

Remarque : Courbe de Bézier

On appelle courbe de Bézier la courbe définie par

$$\alpha: [0, 1] \longrightarrow \mathbf{R}^3$$

$$t \longmapsto \alpha(t) = \sum_{i=0}^{m} \mathbf{P}_i\ B_i^m(u)$$

où les $\mathbf{P}_i$ sont des points de $\mathbf{R}^3$. Ils sont appelés points de contrôle de la courbe.

Propriétés des carreaux de Bézier

Les propriétés (1) et (2) des polynômes de Bernstein entraînent deux propriétés importantes des carreaux de Bézier

1) Propriété de l'enveloppe convexe

pour tout couple (u,v), $\sigma(u,v)$ appartient toujours à l'enveloppe convexe du réseau de contrôle $\{\mathbf{P}_{ij}\}$

2) Invariance par changement affine

Soit f : une application affine de $\mathbf{R}^3$ dans $\mathbf{R}^3$ alors

---

[1] Les grandeurs vectorielles sont notées en gras

$$f(\sigma(u, v)) = \sum_{i=0}^{m} \sum_{j=0}^{n} f(\mathbf{P}_{ij}) B_i^m(u). B_j^n(v)$$

la propriété 3) des polynômes de Bernstein entraîne

3) $\sigma(0,0) = \mathbf{P}_{00}$, $\sigma(1,0) = \mathbf{P}_{m0}$

$\sigma(0,1) = \mathbf{P}_{0n}$ et $\sigma(1,1) = \mathbf{P}_{mn}$

Le réseau et le carreau se touchent en les quatre coins.

4) Courbes isoparamétriques

Si l'on fixe un des paramètres dans l'expression $\sigma(u,v)$ on obtient une courbe de Bézier inscrite sur le carreau.
Exemple: u fixé $= \underline{u}$

La courbe est: $v \longmapsto \sigma(\underline{u}, v) = \sum_{j=0}^{n} \left(\sum_{i=0}^{m} \mathbf{P}_{ij} B_i^m(\underline{u}) \right). B_j^n(v)$

Lorsque u est fixé, on dit que la courbe est une isoparamétrique à u fixé. Si v est fixé, c'est une isoparamétrique à v fixé. L'ensemble de toutes les isoparamétriques à u fixé(resp. à v fixé) recouvre entièrement le carreau.

5) Dérivées partielles

Grâce à la relation (4), on montre facilement que:

$$\sigma_u(u, v) = m \sum_{i=0}^{m-1} \sum_{j=0}^{n} (\mathbf{P}_{i+1,j} - \mathbf{P}_{ij}) B_i^{m-1}(u). B_j^n(v)$$

$$\sigma_v(u, v) = n \sum_{i=0}^{m} \sum_{j=0}^{n-1} (\mathbf{P}_{i,j+1} - \mathbf{P}_{ij}) B_i^m(u). B_j^{n-1}(v)$$

Les dérivées partielles sont encore des carreaux de Bézier dont les polyèdres de contrôle sont immédiatement déduits du polyèdre de la surface initiale.

I.1.2 Définition d'une surface B-spline

I.1.2.1 Définition des fonctions de base

Soit $m \geq 1$ un entier fixé, soit k un entier $\in \{1,\ldots,m+1\}$.
Soit une suite finie de nombres réels $\{u_i\}_{i=0..m+k}$ telle que:

. $u_i \leq u_{i+1}$ pour tout i
. $u_{k-1} < u_{m+1}$
. $u_i < u_{i+k}$ pour tout i

Les u_i sont appelés les noeuds (ou valeurs nodales), $\{u_i\}_{i=0..m+k}$ est appelé vecteur nodal (ou table des noeuds) et est noté tndu.
Si $u_{i-1} < u_i = \ldots = u_{i+r-1} < u_{i+r}$, chacun des noeuds $u_i \ldots u_{i+r-1}$ est de multiplicité r.
D'après la définition de tndu, la multiplicité maximale d'un noeud est k.
Lorsque $u_{i+1}-u_i$ =constante pour tout i, le vecteur nodal est dit uniforme.

Lorsque . $u_0 = \ldots = u_{k-1} = 0$
. $u_{i+1} - u_i$ = constante pour $(k-1) \leq i \leq m$
. $u_{m+1} = \ldots = u_{m+k}$
Le vecteur nodal est dit quasi uniforme.

Les fonctions de base ou B-spline sont notés N_i^r où r est l'ordre de la B-spline et i varie de 0 à m+k-r.

On peut les définir suivant une relation de récurrence. [BOE 84]

Soit u un réel quelconque

$N_i^1(u) = 1$ si $u \in [u_i, u_{i+1}[$

$N_i^1(u) = 0$ si u n'appartient pas à $[u_i, u_{i+1}[$

pour $1 \leq r \leq k$ $$N_i^r(u) = \frac{u - u_i}{u_{i+r-1} - u_i} N_i^{r-1}(u) + \frac{u_{i+r} - u}{u_{i+r} - u_{i+1}} N_{i+1}^{r-1}(u)$$

convention: lorsqu'un des dénominateurs intervenant dans la formule est nul, le quotient correspondant est nul.

Propriétés des fonctions de base

1) support: $N_i^r(u) = 0$ si u n'appartient pas à $[u_i, u_{i+r}[$

2) positivité: $N_i^r(u) > 0$ si u appartient à $[u_i, u_{i+r}[$
si $u_i = u_{i+r}$ N_i^r est identiquement nulle

3) partition de l'unité: Si u appartient $]u_{r-1}, u_{m+k-r+1}[$

$$\sum_{i=0}^{m+k-r} N_i^r(u) = 1$$

4) Dérivation [BAR 88]:

$$[N_i^r]'(u) = (r-1)\left(\frac{1}{u_{i+r-1} - u_i} N_i^{r-1}(u) - \frac{1}{u_{i+r} - u_{i+1}} N_{i+1}^{r-1}(u)\right)$$

avec la même convention en cas de nullité du dénominateur.

5) Ordre de continuité: une fonction de base N_i^k est $(k-1-d_j)$ fois dérivable en un noeud u_j de multiplicité d_j $(i<j<i+k)$ (c-à-d appartenant à $]u_i, u_i+k[$).

6) Comportement des fonctions de base au voisinage d'un noeud de multiplicité maximale

Soit u_i un noeud de multiplicité k.
On suppose que $u_{i-1} < u_i = u_{i+1} = \ldots\ldots = u_{i+k-1} < u_{i+k}$.
N_{i-1}^k vaut 1 en u_i^- et 0 en u_i+, N_i^k vaut 0 en u_i^- et 1 en u_i+. Les autres fonctions de base s'annulent en ce noeud.

Si u_i n'est que de multiplicité k-1, par exemple si
$u_{i-1} < u_i = u_{i+1} = \ldots\ldots = u_{i+k-2} < u_{i+k-1}$, alors N_{i-1}^k vaut 1 en u_i.
Les autres fonctions de base s'annulent en u_i.

I.1.2.2 Définition d'une surface B-spline

soient m,n,k et l quatre entiers fixés tels que $m \geq 1$, $n \geq 1$, $1 \leq k \leq m+1$ et $1 \leq \ell \leq n+1$. Soient $tndu=\{u_0, \ldots, u_{m+k}\}$, $tndv=\{v_0, \ldots, v_{n+\ell}\}$ deux vecteurs nodaux fixés. Soient $(m+1)x(n+1)$ points de $\mathbf{R}^3$ notés $\mathbf{P}_{ij}$ pour $i=0..m, j=0..n$.

On appelle surface B-spline la surface $\mathcal{C}$ [BOO 78] définie par:

$\sigma: [\, u_{k-1}, u_{m+1}[\times [\, v_{\ell-1}, v_{n+1}[\longrightarrow \mathbf{R}^3$

$$(u, v) \longmapsto \sigma(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{P}_{ij}\, N_i^k(u).\, N_j^\ell(v)$$

$\{\mathbf{P}_{ij}\}$ est le polyèdre (ou réseau) de contrôle de la surface.
Les N_i^k sont associés au vecteur nodal $\{u_i\}$, les N_j^ℓ sont associés au vecteur nodal $\{v_j\}$.
La surface est dite d'ordre k x ℓ, d'ordre k dans la direction u, d'ordre ℓ dans la direction v.
La surface est entièrement définie par m,n,k,ℓ,tndu,tndv et le réseau $\{\mathbf{P}_{ij}\}$.
Le domaine de définition de σ, $[\, u_{k-1}, u_{m+1}[\times [\, v_{\ell-1}, v_{n+1}[$ est appelé carreau paramétrique de $\mathcal{C}$.

Remarque: courbe B-spline

Soient un entier $m \geq 1$ fixé et k un entier tel que $0 \leq k \leq m+1$.
Soient $tndu = \{u_0, \ldots, u_{m+k}\}$ un vecteur nodal fixé et $\{\mathbf{P}_i\}_{i=0\ldots m+1}$ une suite de points de $\mathbf{R}^3$.
On appelle courbe B-spline la courbe définie par

$\alpha: [\, u_{k-1}, u_{m+1}[\longrightarrow \mathbf{R}^3$

$$t \longmapsto \alpha(t) = \sum_{i=0}^{m} \mathbf{P}_i\, N_i^k(t)$$

Les $\mathbf{P}_i$ sont appelés points de contrôle.

Propriétés des surfaces B-spline.

1) Polynomiale par morceaux.

Sur chaque pavé non vide $[u_g, u_{g+1}[\times [v_d, v_{d+1}[$ $\sigma(u,v)$ est un polynôme en (u,v) de degré (k-1) x (ℓ-1).

2) Comportement local.

La propriété (1) des fonctions de base entraîne les résultats suivants:
Si (u,v) appartient à $[u_g,u_{g+1}[\times [v_d,v_{d+1}[$,

$$\sigma(u,v) = \sum_{i=g-k+1}^{g} \sum_{j=d-l+1}^{d} P_{ij}\, N_i^k(u).\, N_j^l(v)$$

donc $\sigma(u,v)$ ne dépend que des points représentés par la matrice A_{gd}:

$$\begin{bmatrix} P_{g-k+1,d-l+1} & \cdots\cdots & P_{g-k+1,d} \\ \vdots & & \vdots \\ P_{g,d-l+1} & \cdots\cdots & P_{g,d} \end{bmatrix}$$

Inversement le point de contrôle P_{ij} n'a d'influence que sur le pavé $[u_i,u_{i+k}[\times [v_j,v_{j+l}[$.

3) Enveloppe convexe

Les propriétés (1) (2) et (3) des fonctions de base entraînent que la restriction de la surface à un pavé paramétrique $[u_g,u_{g+1}[\times [v_d,v_{d+1}[$ est entièrement incluse dans l'enveloppe convexe des points de la matrice A_{gd}.

4) Invariance par transformation affine.

Soit F une transformation affine de $\mathbf{R}^3$ vers $\mathbf{R}^3$.
La propriété (3) des fonctions de base entraîne que

$$\forall\ (u,v) \qquad F(\sigma(u,v)) = \sum_{i=0}^{m} \sum_{j=0}^{n} F(P_{ij})\, N_i^k(u).\, N_j^l(v)$$

5) Courbes isoparamétriques
Les courbes isoparamétriques sont définies de la même façon que pour les carreaux de Bézier; les propriétés sont complètement analogues.

6) Dérivation

La forme des dérivées des fonctions de base nous permet de démontrer que

$$\sigma_u(u, v) = (k\text{-}1) \sum_{i=1}^{m} \sum_{j=0}^{n} \frac{\mathbf{P}_{i,j} - \mathbf{P}_{i-1,j}}{u_{i+k-1} - u_i} N_i^{k-1}(u).\ N_j^{\ell}(v)$$

$$\sigma_v(u, v) = (\ell\text{-}1) \sum_{i=0}^{m} \sum_{j=1}^{n} \frac{\mathbf{P}_{i,j} - \mathbf{P}_{i,j-1}}{v_{j+\ell-1} - v_j} N_i^{k}(u).\ N_j^{\ell-1}(v)$$

Les dérivées partielles sont donc encore des surfaces B-spline dont les caractéristiques sont facilement déduites de celles de σ.

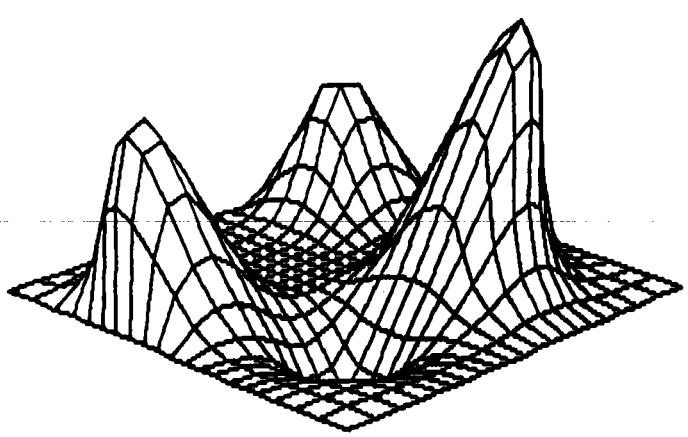
7) Influence de l'ordre k x ℓ

Nous exprimons brièvement à l'aide d'un tableau, la manière dont évoluent les propriétés de la surface lorsque l'ordre croît.

ordre	caractère local	capacité de la surface à bien approcher son réseau de définition	ordre de continuité	régularité de la surface
↗	↘	↘	↗	↗

De manière générale l'élévation de l'ordre "tend" la surface.

Exemple de surface B-spline d'ordre 3x3:



Remarques importantes

1) Supposons que les noeuds terminaux dans tndu et tndv soient de multiplicité maximale. C'est à dire que

$$u_0 = u_1 = \ldots = u_{k-1} < u_k$$
$$v_0 = v_1 = \ldots = v_{\ell-1} < v_\ell$$
$$u_m < u_{m+1} = u_{m+2} = \ldots = u_{m+k}$$
$$v_n < v_{n+1} = v_{n+2} = \ldots = v_{n+\ell}$$

2) La propriété (6) des fonctions de base entraîne que la surface passe par les quatre "coins" du réseau de définition, plus précisément nous avons:

$$\sigma(u_{k-1}, v_{\ell-1}) = \mathbf{P}_{00}$$

$$\sigma(u_{m+1}, v_{\ell-1}) = \mathbf{P}_{m0}$$

$$\sigma(u_{k-1}, v_{n+1}) = \mathbf{P}_{0n}$$

$$\sigma(u_{m+1}, v_{n+1}) = \mathbf{P}_{mn}$$

Dans la suite de cet exposé tous les carreaux B-spline vérifient cette condition sauf mention contraire.

2) Lien avec les carreaux de Bézier

Si les conditions suivantes sont vérifiées:

* $k = m+1$
* $\ell = n+1$
* $u_0 = u_1 = \ldots = u_{k-1} < u_{m+1} = u_{m+2} = \ldots = u_{m+k}$
* $v_0 = v_1 = \ldots = v_{\ell-1} < v_{n+1} = v_{n+2} = \ldots = v_{n+\ell}$

Alors le carreau B-spline $\mathcal{C}$ **est un carreau de Bézier** de degré (k-1) x (ℓ-1),
ayant $\{\mathbf{P}_{ij}\}_{i=0\ldots m\,;\ j=0\ldots n}$ pour réseau de définition.
Plus précisément si $\mathcal{C}'$ est le carreau de Bézier défini par:

$$\sigma' : [\,0,\,1[\,\times\,[\,0,\,1[\longrightarrow \mathbf{R}^3$$

$$(u, v) \longmapsto \sigma'(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{P}_{ij}\, B_i^m(u).\, B_j^n(v)$$

(Les B_i^m et les B_j^n sont les polynômes de Bernstein)

alors pour tout couple (u,v) de $[\,0,\,1[\,\times\,[\,0,\,1[$ on a:
$\sigma'(u, v) = \sigma(u_{k-1} + (u_{m+1}-u_{k-1}).u,\ v_{l-1} + (v_{n+1}-v_{l-1}).v)$.

Dans ce cas on le traite comme un carreau de Bézier.

I.1.3 Définition d'une surface B-spline rationnelle

La définition d'un carreau rationnel est rendue plus commode par l'introduction de deux fonctions mathématiques:
On appelle H la fonction: $\mathbf{R}^4 \longrightarrow \mathbf{R}^3$

$$(x, y, z, t) \longmapsto H(x, y, z, t) = \left(\frac{x}{t}, \frac{y}{t}, \frac{z}{t} \right)$$

Et T l'application: $\mathbf{R}^3 \longrightarrow \mathbf{R}^4$

$$(x, y, z) \longmapsto T(x, y, z) = (x, y, z, 1)$$

Soit $\{ \mathbf{P}_{ij} \}_{i=0\ldots m;\, j=0\ldots n;}$ un réseau de points de contrôle
en 3D .Soient deux tables de noeuds tndu $= \{u_0,\ldots,u_{m+k}\}$ et tndv $= \{v_0,\ldots,v_{n+l}\}$.

A chaque point $\mathbf{P}_{ij}$ est associé un poids t_{ij} qui est un réel > 0 (appelé aussi coefficient de pondération).

Nous définissons l'application

$$s : [\,u_{k-1},\, u_{m+1}[\,\times\,[\,v_{l-1},\, v_{n+1}\,[\longrightarrow \mathbf{R}^4$$

$$(u, v) \longmapsto s(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{Q}_{ij}\, N_i^k(u).\, N_j^l(v)$$

avec $\mathbf{Q}_{ij} = t_{ij}\, T(\mathbf{P}_{ij})$. Donc si $\mathbf{P}_{ij} = \begin{bmatrix} P^x_{ij} \\ P^y_{ij} \\ P^z_{ij} \end{bmatrix}$ alors $\mathbf{Q}_{ij} = \begin{bmatrix} t_{ij} P^x_{ij} \\ t_{ij} P^y_{ij} \\ t_{ij} P^z_{ij} \\ t_{ij} \end{bmatrix}$

Elle représente la paramétrisation d'un carreau B-spline polynomial $\mathcal{C}$ dans l'espace homogène.
Un carreau B-spline rationnel $\mathcal{N}$ est alors défini par la paramétrisation

$$\sigma: [a, b[\times [c, d[\longrightarrow \mathbf{R}^3$$
$$(u, v) \longmapsto \sigma(u, v) = H(s(u, v))$$

En d'autres termes

$$\sigma(u, v) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} t_{ij}\, \mathbf{P}_{ij}\, N_i^k(u).\, N_j^l(v)}{\sum_{i=0}^{m} \sum_{j=0}^{n} t_{ij}\, N_i^k(u).\, N_j^l(v)}$$

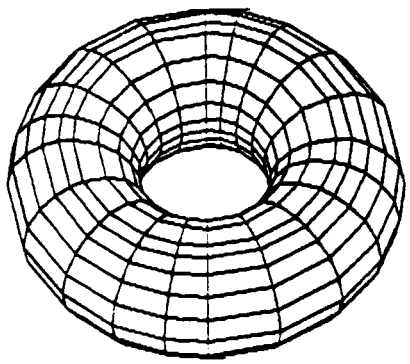
Remarques

Le carreau $\mathcal{N}$ est la projection du carreau $\mathcal{C}$, ce dernier est suffisant pour représenter complètement $\mathcal{N}$. C'est donc sous cette forme que $\mathcal{N}$ est représenté. Les informations conservées sont donc: m, n, k, l, $\{u_i\}$, $\{v_j\}$, $[\mathbf{Q}_{ij}]$.

Toutes les transformations géométrique affines sur $\mathcal{N}$ sont effectuées sur $\mathcal{C}$. Le calcul de points ou de dérivées partielles est d'abord effectué dans $\mathbf{R}^4$, là où sont appliquées toutes les propriétés relatives aux carreaux polynomiaux; puis les données calculées sont projetées dans l'espace euclidien.

Lorsque les vecteurs nodaux associés à une surface B-spline rationnelle
sont non uniformes, celle-ci est appelée surface B-spline rationnelle non uniforme ou NURBS (pour Non Uniform Rational B-Spline).

Exemple: Tore représenté par un carreau NURBS



Propriétés des surfaces B-spline rationnelles

1) Rationnelle par morceaux

Sur chaque pavé paramétrique non vide $[u_i,u_{i+1}[\times [v_j,v_{j+1}[$, la surface est une fraction rationnelle en ses paramètres.

2) 3) 4) Les propriétés de comportement local, d'enveloppe convexe et d'invariance par transformation affine des B-spline polynomiales sont intégralement conservées par les B-spline rationnelles.

Les deux remarques importantes concernant les B-spline polynomiales sont encore valables pour les rationnelles car elles sont liées aux vecteurs nodaux.

Cas particulier: carreau de Bézier rationnel

La définition d'un carreau de Bézier rationnel suit la même démarche que celle d'un carreau NURBS rationnel.
Soit $[\mathbf{P}_{ij}]_{i=0...m;\ j=0...n;}$ une matrice de $(m+1) \times (n+1)$ points de $\mathbf{R}^3$. A chaque point $\mathbf{P}_{ij}$ est associé un poids t_{ij} qui est un réel > 0
Nous définissons l'application

$$s : [0, 1[\times [0, 1[\longrightarrow \mathbf{R}^4$$
$$(u, v) \longmapsto s(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{Q}_{ij} \, B_i^m(u) . \, B_j^n(v)$$

avec $\mathbf{Q}_{ij} = t_{ij} \, T(\mathbf{P}_{ij})$.

Elle représente la paramétrisation d'un carreau Bézier polynomial $\mathcal{C}$ dans l'espace homogène.
Un carreau Bézier rationnel $\mathcal{B}$ est alors défini par la paramétrisation

$$\sigma: [0, 1[\times [0, 1[\longrightarrow \mathbf{R}^3$$
$$(u, v) \longmapsto \sigma(u, v) = H(s(u, v))$$

Comme pour les carreaux NURBS, ce sont les informations relatives au carreau dans l'espace homogène qui sont conservées.
Les informations conservées sont donc: m, n, $[\mathbf{Q}_{ij}]$.
L'ensemble des remarques que nous avions formulées pour les carreaux Bspline rationnels est encore valable pour les carreaux de Bézier rationnels.

I.2 ALGORITHMES FONDAMENTAUX

I.2.1 Algorithmes de calculs pour les courbes de Bézier

I.2.1.1 Algorithme de De Casteljau

L'algorithme de De Casteljau est l'outil de base de manipulation d'une courbe de Bézier. Il sert à calculer pour une valeur u* fixée du paramètre, le point correspondant sur la courbe.

Grâce aux propriétés de récurrence des polynômes de Bernstein, Paul De Casteljau [CAS 85] a établi cet algorithme simple et efficace.

Description de l'algorithme

Soit T une courbe de Bézier : T: $u \longmapsto \mathbf{R}(u)$, nous calculons $\mathbf{R}(u^*)$ où u* est une valeur quelconque de l'intervalle [0,1].

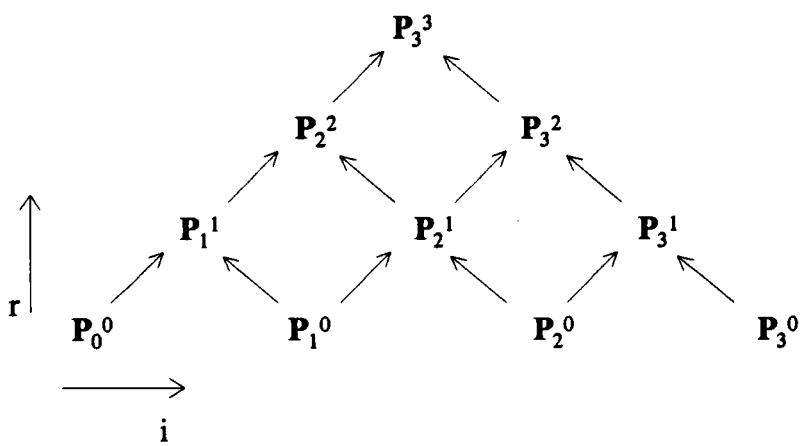
Initialisation

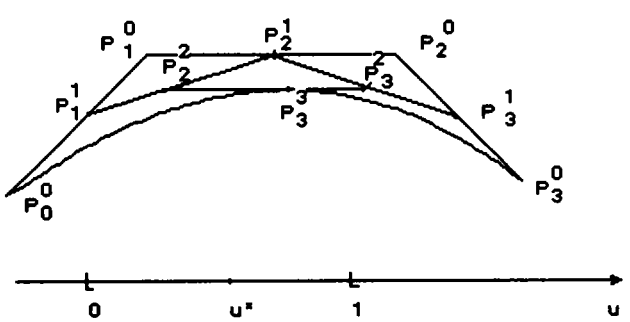
$$\mathbf{P}_i^0 = \mathbf{P}_i \quad \text{pour } i = 0..m \qquad (1)$$

récurrence

$$\mathbf{P}_i^r(u) = u^* \mathbf{P}_{i-1}^{r-1} + (1-u^*) \mathbf{P}_i^{r-1} \quad \text{pour } r \text{ variant de } 1 \text{ à } m$$
$$\text{et } i \text{ de } r \text{ à } m$$

Ce processus est parfaitement représenté sous la forme d'un schéma triangulaire.
Exemple pour m = 3





- Propriétés fondamentales de l'algorithme

Cet algorithme est également un très bon outil de subdivision. La courbe {**R**(u), u:0..1} est la réunion des arcs {**R**(u), u:0..u*} et {**R**(u),u:u*..1}. Ces deux arcs sont des courbes de

Bézier de degré m dont les polygones de contrôle sont les deux pentes du schéma triangulaire. Plus précisément:

$$\{ \mathbf{R}(u),\ u:0..u^*\} = \{ \mathbf{P}_0^0\, B_0^m(t) + \mathbf{P}_1^1\, B_1^m(t) + \ldots + \mathbf{P}_m^m\, B_m^m(t),\ t:0..1\}$$

$$\{ \mathbf{R}(u),\ u:u^*..1\} = \{ \mathbf{P}_m^m\, B_0^m(t) + \mathbf{P}_m^{m-1}\, B_1^m(t) + \ldots + \mathbf{P}_m^0\, B_m^m(t),\ t:0..1\}$$

Propagation de l'erreur

Nous étudions l'incertitude sur le calcul de $\mathbf{R}(u^*)$ en fonction des imprécisions sur les données.
Nous interprétons les erreurs commises sur les termes comme des variables aléatoires.
Notons $\underline{\mathbf{P}}_i^r$ le terme général de l'algorithme calculé réellement et $\mathbf{P}_i^r$ le terme théorique exact correspondant. L'erreur commise est $\mathcal{E}_i^r = \underline{\mathbf{P}}_i^r - \mathbf{P}_i^r$ (2)
Considérons $\mathcal{E}_i^r$ comme une variable aléatoire; notons $E(\mathcal{E}_i^r)$ son espérance mathématique et $Var(\mathcal{E}_i^r)$ sa variance. $\mathcal{E}_i^0$ est la variable aléatoire représentant l'imprécision sur le terme initial $\mathbf{P}_i^0$. On peut admettre que les erreurs sur les termes initiaux, donc les $\mathcal{E}_i^0$ sont des v. a. de même loi et indépendantes.
Donc $E(\mathcal{E}_i^0) = \text{constante}_1\ \forall i$ et $Var(\mathcal{E}_i^0) = \text{constante}_2\ \forall i$ (3)
Grâce à (1) et (2) on montre que $\mathcal{E}_i^r = u^* \mathcal{E}_{i-1}^{r-1} + (1-u^*)\mathcal{E}_i^{r-1}$
On en déduit que $E(\mathcal{E}_i^r) = u^*\, E(\mathcal{E}_{i-1}^{r-1}) + (1-u^*)\, E(\mathcal{E}_i^{r-1})$
Par récurrence on obtient que $E(\mathcal{E}_i^r) = E(\mathcal{E}_i^0) = \text{constante}_1$

Ce qui signifie que l'erreur moyenne sur le calcul de $\mathbf{R}(u^*)$ est la même que l'erreur moyenne sur les points de contrôle. On montre aussi que
$Var(\mathcal{E}_i^r) = (u^*)^2\, Var(\mathcal{E}_{i-1}^{r-1}) + (1-u^*)^2\, Var(\mathcal{E}_i^{r-1})$
On peut facilement en déduire par récurrence que:
$(1/2)^n\, Var(\mathcal{E}_i^0) \le Var(\mathcal{E}_r^r) \le Var(\mathcal{E}_i^0) = \text{constante}_2$

En particulier si $u = u^*$ $\quad Var(\mathcal{E}_r^r) = (1/2)^n\, Var(\mathcal{E}_i^0)$

Ce qui signifie que l'erreur sur le terme final est bien plus proche de l'erreur moyenne que ne l'étaient les terme initiaux.
Ces propriétés traduisent la grande stabilité numérique de cet algorithme.

I.2.1.2 Elévation du degré

Soit T une courbe de Bézier de degré m, définie comme dans l'introduction. Il est possible de définir T' une courbe de Bézier de degré (m+1) et de polyèdre de contrôle $\{\mathbf{P'}_i\}_{i=0...m+1}$; telle que pour tout u dans [0..1], T(u)=T'(u).

Description de l'algorithme

Initialisation

$$\mathbf{P'}_0 = \mathbf{P}_0$$

Récurrence

pour i=1..m+1 $\mathbf{P'}_i = a_i \mathbf{P}_{i-1} + (1-a_i)\,\mathbf{P}_i$

où $a_i = i/(m+1)$

Remarques:
Cet algorithme permet de faire coïncider les degrés de deux courbes.
Il est repris en détail dans le chapitre subdivision et diminution de degré (IV.1).

I.2.1.3 Traduction d'une courbe de Bézier dans la base canonique des polynômes

On désire écrire une courbe de Bézier **R**(u) sous la forme

$$\mathbf{R}(u) = \mathbf{A}_0 + \mathbf{A}_1 u + \mathbf{A}_2 u^2 + .. + \mathbf{A}_m u^m$$

Description de l'algorithme

Initialisation

$\Delta_i^0 = \mathbf{P}_i$ pour i=0...m

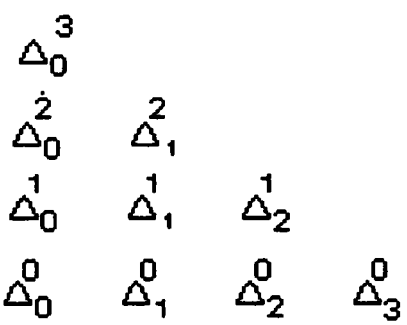
Récurrence

$\Delta_i^r = \Delta_{i+1}^{r-1} - \Delta_i^{r-1}$ pour r = 1...m et i = 0...m-r

Finalement

$$\mathbf{A}_i = C_m^i \, \Delta_0^i \quad \text{pour } i=0...m$$

Exemple pour m = 3 :



Remarque

On montre facilement que $\mathbf{A}_i = \sum_{j=0}^{i} C_i^j (-1)^j \, \mathbf{P}_{i-j}$

Propagation de l'erreur

Notre étude est analogue à celle effectuée pour l'algorithme de De Casteljau.

Nous étudions l'incertitude sur les termes finaux Δ_0^r en fonction des imprécisions sur les termes initiaux Δ_i^0 . Notons $\underline{\Delta}_i^r$ le terme général de l'algorithme calculé réellement et Δ_0^r le terme théorique exact.
L'erreur sur le terme général $\mathcal{E}_i^r = \underline{\Delta}_i^r - \Delta_i^r$ est considérée comme une variable aléatoire.
On admet que les $\mathcal{E}_i^0$(erreurs sur les termes initiaux) sont des variables aléatoires indépendantes et de même loi.
Donc $E(\mathcal{E}_i^0)$ = constante$_1$ $\forall i$ et $Var(\mathcal{E}_i^0)$ = constante$_2$ $\forall i$

On montre que $E(\mathcal{E}_0^r) = 0$ si $r > 0$
Ce qui signifie que l'erreur moyenne est nulle pour les termes résultats,
mais $Var(\mathcal{E}_0^r) = 2^r \, Var(\mathcal{E}_i^0)$

Ce qui signifie que la marge d'erreur moyenne augmente de façon exponentielle avec l'indice r du terme final.
Cet algorithme montre donc une amplification catastrophique de l'erreur pour des degré m élevés. Si l'on utilise des réels codés sur 32 bits, on peut convertir des polynômes jusqu'à un degré égal à 10, avec des réels codés sur 64 bits, on peut aller jusqu'à un degré égal à 20.

Changement de base réciproque

Il s'agit d'obtenir les Pi en fonction des Ai.
Ils sont calculés à l'aide de la formule

$$\mathbf{P}_i = \sum_{j=0}^{i} \frac{C_i^j}{C_m^j} \mathbf{A}_j \quad \text{pour } i = 0...m$$

Ils peuvent également être obtenus à l'aide du schéma triangulaire précédent à la différence que c'est la colonne de gauche qui est initialisée(au lieu de la base) et les termes finaux sont dans la ligne de base.

L'algorithme devient:

Initialisation

$$\Delta_0^r = \frac{1}{C_m^r} \mathbf{A}_r \text{ pour } r=0...m \text{ (colonne de gauche)}$$

Récurrence

$$\Delta_i^r = \Delta_{i-1}^r + \Delta_{i-1}^{r+1} \quad \text{pour } r = 1...m \text{ et } i = 0...m-r$$

Finalement

$$\mathbf{P}_i = \Delta_i^0 \qquad \text{pour } i=0...m \text{ (ligne de base)}$$

Propagation de l'erreur

La stabilité numérique n'est pas meilleure que pour l'algorithme précédent.

I.2.2 Algorithmes de calculs pour les courbes B-spline

I.2.2.1 Algorithme de Cox-De Boor

L'algorithme de Cox-De Boor est aux B-spline ce que l'algorithme de De Casteljau est aux Bézier. Son rôle est le calcul de points sur une courbe ou la subdivision. Ses propriétés sont complètement analogues.

Soit $\mathcal{C}$ une courbe B-spline d'ordre k, définie par (m+1) points $\{\mathbf{P}_0 .. \mathbf{P}_m\}$, de table nodale $\{u_0 .. u_{m+k}\}$. $\mathcal{C}$ est décrite par

$$\alpha: [\, u_{k-1}, u_{m+1}[\longrightarrow \mathbf{R}^3$$

$$t \longmapsto \alpha(t) = \sum_{i=0}^{m} \mathbf{P}_i \, N_i^k(t)$$

Nous décrivons le calcul de $\alpha(u^*)$ pour u^* quelconque appartenant à $[\, u_{k-1}, u_{m+1}[$

Description de l'algorithme

Initialisation

- recherche de l'unique indice d tel que

$$u_d \leq u^* < u_{d+1}$$

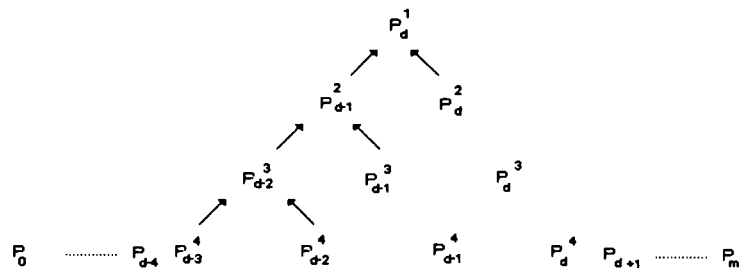
$$\mathbf{P}_i^k = \mathbf{P}_i \quad \text{pour } i = d-k+1 .. d$$

Récurrence

$$\mathbf{P}_i^r = a_{ir}\mathbf{P}_i^{r+1} + (1-a_{ir})\mathbf{P}_{i-1}^{r+1} \quad \text{pour r variant de (k-1) à 1}$$

$$\text{où } a_{ir} = (u^*-u_i)/(u_{i+r}-u_i) \quad \text{pour i variant de (d-r+1) à d}$$

Exemple pour k = 4



Cet algorithme trouve sa justification dans les propriétés de récurrence et de support borné des fonctions B-spline.

Redéfinition de la courbe

L'application de Cox-De Boor correspond à l'insertion de u^* (k-1) fois dans la table des noeuds. Par conséquent, la courbe peut être redéfinie. Les (m+k) points $\mathbf{P}_0$, .. , $\mathbf{P}_{d-k}$, $\mathbf{P}_{d+1}{}^k$, .. , $\mathbf{P}_d{}^1$, .. , $\mathbf{P}_d{}^k$, $\mathbf{P}_{d+1}$, .. ,$\mathbf{P}_m$ et les (m+2k) valeurs nodales

$v_0=u_0$, .. , $v_d=u_d$

$v_{d+1}=u^*$, .. , $v_{d+k-1}=u^*$

$v_{d+k}=u_{d+1}$, .. ,$v_{m+2k-1}=u_{m+k}$ constituent les nouveaux éléments de définition de la courbe T. On constate que l'ordre de T n'a pas été modifié. Le nouveau polygone de contrôle a été obtenu en substituant la base du schéma triangulaire par ses deux pentes. Cette modification rapproche le polygone de la courbe.

Remarque : Cet algorithme permet de diviser la courbe
$T=\{u \longmapsto \mathbf{R}(u),\ u:u_{k-1} .. u_{m+1}\}$ en deux nouvelles courbes
$T_g=\{u \longmapsto \mathbf{R}(u),\ u:u_{k-1} .. u^*\}$ et $T_d = \{u \longmapsto \mathbf{R}(u),\ u : u^* .. u_{m+1}\}$.
T_g est une courbe B-spline d'ordre k définie par les (d+1) points
$\mathbf{P}_0$, .. , $\mathbf{P}_{d-k}$, $\mathbf{P}_{d-k+1}{}^k$, .. , $\mathbf{P}_d{}^1$ et par le vecteur nodal
$tndw=\{w_0=v_0,\ ..\ , w_{d+k-1}=v_{d+k-1}\ ,\ w_{d+k}=u^*\}$.
T_d est une courbe B-spline d'ordre k définie par les (m+k-d) points
$\mathbf{P}_d{}^2$, .. , $\mathbf{P}_1{}^k$, $\mathbf{P}_{d+1}$, .. , $\mathbf{P}_m$ et par le vecteur nodal
$tnds = \{s_0=u^*,s_1=v_{d+1},\ ..\ ,s_{k-1}=v_{d+k-1},s_k=v_{d+k},\ ..\ ,s_{m-d+2k-1}=v_{m+2k-1}\}$

Une réalisation de cet algorithme (subdc) est décrite en annexe.

I.2.2.2 Algorithme de Boehm

C'est un cas particulier de l'algorithme d'Oslo [BAR 88]. Il insère une unique valeur nodale.

Remarque: On peut montrer que l'utilisation de l'algorithme de Cox - De Boor revient à répéter (k-1) fois l'algorithme de Boehm avec la même valeur nodale. On peut trouver une comparaison de ces différents algorithmes dans [DAN 89].

I.2.2.3 Algorithme de calcul des dérivées successives

Cet algorithme du à Boehm[BOE 84] permet le calcul simultané des dérivées jusqu'à l'ordre k-1 d'une courbe B-spline d'ordre k.
Soit $\mathcal{C}$ une courbe B-spline d'ordre k définie par les points $\mathbf{P}_0 \ldots \mathbf{P}_m$. et le vecteur nodal $\{u_0 \ldots u_{m+k}\}$. $\mathcal{C}$ est décrite par: $t \longmapsto \alpha(t) = \sum_{i=0}^{m} \mathbf{P}_i \, N_i^k(t)$

Soit u^* fixé appartenant à $[u_{k-1}, u_{m+1}[$, On désire calculer les dérivées de α en u.

Description de l'algorithme

Initialisation

- Recherche de l'unique indice d tel que u appartient à $[u_d, u_{d+1}[$.
puis initialisation "base triangle inférieur droit":

$$\mathbf{P}_i^0 = \mathbf{P}_i \text{ pour } i = d-k+1 \ldots d$$

Récurrence triangle inférieur droit:

$$\mathbf{P}_i^r = (\mathbf{P}_i^{r-1} - \mathbf{P}_{i-1}^{r-1})/(u_{i+k-r}-u_i) \qquad \text{pour } r = 1 \ldots k-1 \text{ et } i = d-(k-r)+1 \ldots d$$

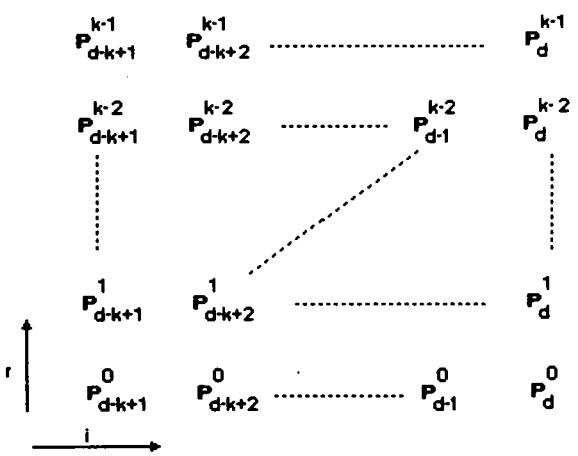
Récurrence triangle supérieur gauche:

$$\mathbf{P}_i^r = \mathbf{P}_i^{r-1} + (u^* - u_{d+r-(k-1)})\mathbf{P}_{i+1}^r \qquad \text{pour } r = 1\ldots k-1 \text{ et } i = d-k+1\ldots d-k+r$$

La dérivée de rang r est donnée par:

$$\alpha^{(r)}(u^*) = (k-1)(k-2)\ldots(k-r)\, \mathbf{P}_{d+r-(k-1)}^{k-1} \quad \text{pour } r = 0\ldots k-1$$

Il y a donc deux schémas triangulaires superposés qui forment un schéma carré. La plus haute ligne du carré fournit les dérivées successives.



Une réalisation de cet algorithme (ttesderiv) est décrite en annexe.

I.3 APPLICATIONS USUELLES

I.3.1 Traduction d'un carreau de Bézier dans la base canonique des polynômes

Soit δ un carreau de Bézier défini comme dans l'introduction par:

$$\sigma(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{P}_{ij}\, B_i^m(u).\, B_j^n(v)$$

On veut réécrire $\sigma(u,v)$ sous la forme: $\sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{A}_{ij}\ u^i.\ v^j$

or $\sigma(u, v) = \sum_{i=0}^{m} B_i^m(u)\left(\sum_{j=0}^{n} \mathbf{P}_{ij}\ B_j^n(v)\right)$ on pose $\mathbf{Q}_i(v) = \sum_{j=0}^{n} \mathbf{P}_{ij}\ B_j^n(v)$

Pour i fixé, $\mathbf{Q}_i(v)$ est une courbe de Bézier qui s'exprime à l'aide de l'algorithme de passage à la base canonique sous la forme $\sum_{j=0}^{n} \mathbf{D}_{ij}\ v^j$

Donc $\sigma(u, v) = \sum_{i=0}^{m} B_i^m(u)\left(\sum_{j=0}^{n} \mathbf{D}_{ij}\ v^j\right)$

ou encore $\sigma(u, v) = \sum_{j=0}^{n} v^j \left(\sum_{i=0}^{m} \mathbf{D}_{ij}\ B_i^m(u)\right)$ On pose $\mathbf{R}_j(u) = \sum_{i=0}^{m} \mathbf{D}_{ij}\ B_i^m(u)$

Pour j fixé, $\mathbf{R}_j(u)$ est également une courbe de Bézier qui s'écrit sous la forme

$\mathbf{R}_j(u) = \sum_{i=0}^{m} \mathbf{A}_{ij}\ u^i$

d'où $\sigma(u, v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{A}_{ij}\ u^i.\ v^j$

L'algorithme de passage à la forme canonique d'un carreau de Bézier consiste à calculer dans un premier temps les $\mathbf{D}_{ij}$ et ensuite les $\mathbf{A}_{ij}$.

I.3.2 Traduction d'un carreau B-spline dans la base canonique des polynômes

Passage à la base canonique pour une courbe B-spline

Soit une courbe B-spline d'ordre k, de vecteur nodal $\{u_0 \ldots u_{m+k}\}$ et de points de contrôle $\mathbf{P}_0 \ldots \mathbf{P}_m$. Sur chaque intervalle non vide $[u_d, u_{d+1}[$ la courbe est un polynôme de degré k-1. La détermination de ces polynômes est simple:

Pour chaque noeud u_d tel que k-1 $\leq d \leq m$ et tel que $[u_d, u_{d+1}[$ non vide, on calcule toutes les dérivées et donc grâce à la formule de Taylor-Young on obtient le développement polynomial de la courbe sur l'intervalle $[u_d, u_{d+1}[$.

Généralisation aux carreaux B-spline

Soit $\mathcal{C}$ un carreau B-spline, les notations sont les mêmes que celles utilisées dans l'introduction. Sur chaque pavé $[u_i,u_{i+1}[x[v_j,v_{j+1}[_{i=0..m; j=0..n;}$ notre surface est une polynomiale
en (u,v) de degré (k-1)x(ℓ-1).

Nous calculons cette polynomiale:

Soit (u,v) dans le pavé $[u_g,u_{g+1}[x[v_d,v_{d+1}[$. d et g sont tels que $u_g < u_{g+1}$ et $v_d < v_{d+1}$. Sur ce pavé, la surface s'écrit

$$\sigma(u, v) = \sum_{i = g-k+1}^{g} \sum_{j = d-\ell+1}^{d} P_{ij}\ N_i^k(u).\ N_j^\ell(v)$$

ou encore

$$\sigma(u, v) = \sum_{i = g-k+1}^{g} N_i^k(u).\ \left(\sum_{j = d-\ell+1}^{d} P_{ij}\ N_j^\ell(v) \right)$$

Le terme entre parenthèses noté $Q_i(v)$ est une courbe B-spline. En utilisant la méthode de passage à la base canonique pour une courbe B-spline, on obtient

$$Q_i(v) = \sum_{j = 0}^{\ell-1} C_{ij}(d)\ (v - v_d)^j$$

$$\text{donc}\quad \sigma(u, v) = \sum_{i = g-k+1}^{g} N_i^k(u).\ \left(\sum_{j = 0}^{\ell-1} C_{ij}(d)\ (v - v_d)^j \right)$$

$$\text{ou encore}\ \sigma(u, v) = \sum_{j = 0}^{\ell-1} (v - v_d)^j .\left(\sum_{i = g-k+1}^{g} C_{ij}(d)\ N_i^k(u) \right)$$

De même, le terme entre parenthèses peut être réécrit sous la forme

$$\sum_{i=0}^{k-1} A_{ij}(g,d)\,(u-u_d)^i$$

Sur chaque pavé non vide $[u_g,u_{g+1}[\times[v_d,v_{d+1}[$, on obtient par ce procédé une matrice $[A_{ij}(g,d)]_{i=0..k-1,j=0..l-1}$ des coefficients de la forme canonique.

I.3.3 Partition d'un carreau de Bézier

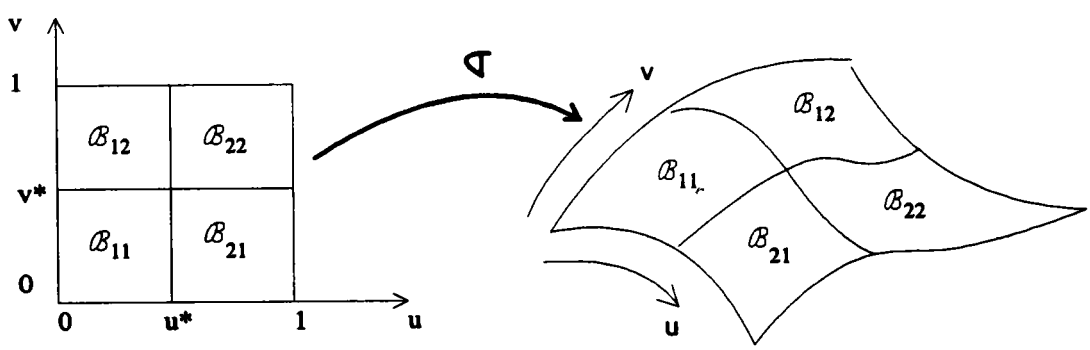
La subdivision d'un carreau de Bézier consiste à le découper en quatre carreaux de Bézier adjacents. Les lignes de coupe sont des lignes isoparamétriques. Les quatre sous-carreaux obtenus forment une partition du carreau initial et sont de même degré que lui.

Appelons $\mathcal{B}$ le carreau de Bézier m x n défini par:

$$\sigma: [0,1[\times[0,1[\longrightarrow \mathbf{R}^3$$

$$(u,v) \longmapsto \sigma(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{P}_{ij}\, B_i^m(u).\, B_j^n(v)$$

Le carreau $\mathcal{B}$ est "partitionné" en quatre sous-carreaux de Bézier de degré m x n :
$\mathcal{B} = \mathcal{B}_{11} \cup \mathcal{B}_{12} \cup \mathcal{B}_{22} \cup \mathcal{B}_{21}$ grâce à l'algorithme de De Casteljau.
Les deux lignes de coupe sont des courbes isoparamétriques définies à $u_{fixé}$ et à $v_{fixé}$.
Notons $u^* = u_{fixé}$ et $v^* = v_{fixé}$. Ces deux valeurs sont des valeurs quelconques de l'intervalle]0, 1[.
Les rectangles paramétriques des sous carreaux sont respectivement:
$[0,u^*[\times[0,v^*[,\ [0,u^*[\times[v^*,1[,\ [u^*,1[\times[v^*,1[,\ [u^*,1[\times[0,v^*[$.



Partition dans le plan paramétrique u-v — Partition dans l'espace

Calculs des nouveaux réseaux de contrôle

IL est commode d'écrire le réseau de contrôle de $\mathcal{B}$ sous la forme matricielle suivante.

$$P = \begin{bmatrix} P_{00} & \cdots\cdots & P_{0n} \\ \vdots & & \vdots \\ P_{i0} & & P_{in} \\ \vdots & & \vdots \\ P_{m0} & \cdots\cdots & P_{mn} \end{bmatrix}$$

Dans un premier temps, nous allons découper $\mathcal{B}$ en 2 carreaux $\mathcal{B}_1$ et $\mathcal{B}_2$ selon la courbe $\sigma(u,v^*)$.

$$\mathcal{B}_1 = \mathcal{B}_{11} \cup \mathcal{B}_{21} \qquad \mathcal{B}_2 = \mathcal{B}_{12} \cup \mathcal{B}_{22}$$

Chaque ligne de la matrice P détermine une courbe de Bézier en v que l'on découpe à l'aide de l'algorithme de De Casteljau en la valeur v^*.

Chaque ligne $[\mathbf{P}_{ij}]_{j=0..n}$ est donc prise à son tour comme la base du schéma triangulaire de Casteljau.

Chaque $[\mathbf{P}_{ij}]_{j=0..n}$ est remplacé par $[\mathbf{Q}_{ij}]_{j=0..2n}$

Nous obtenons 2 matrices de points qui sont les réseaux de $\mathcal{B}_1$ et $\mathcal{B}_2$.

$$\begin{bmatrix} Q_{00} & \cdots\cdots & Q_{0n} \\ Q_{m0} & \cdots\cdots & Q_{mn} \end{bmatrix} \begin{bmatrix} Q_{0n} & \cdots\cdots & Q_{02n} \\ Q_{mn} & \cdots\cdots & Q_{m2n} \end{bmatrix}$$

$\mathcal{B}_1$ et $\mathcal{B}_2$ sont découpés suivant la courbe $\sigma(u^*,v)$: pour j fixé
Chaque colonne $[\mathbf{Q}_{ij}]_{i=0..m}$ est remplacé à l'aide de l'algorithme de De Casteljau par $[\mathbf{R}_{ij}]_{i=0..2m}$

Finalement on obtient 4 nouvelles matrices

$$\begin{bmatrix} R_{0\,0} & \cdots & R_{0\,n} \\ R_{m0} & \cdots & R_{mn} \end{bmatrix} \quad \begin{bmatrix} R_{0\,n} & \cdots & R_{0\,2n} \\ R_{mn} & \cdots & R_{m\,2n} \end{bmatrix}$$

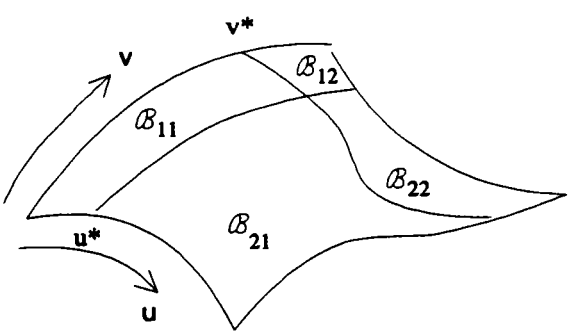
$$\begin{bmatrix} R_{m0} & \cdots & R_{mn} \\ R_{2m0} & \cdots & R_{2mn} \end{bmatrix} \quad \begin{bmatrix} R_{mn} & \cdots & R_{m\,2n} \\ R_{2mn} & \cdots & R_{2\,m\,2n} \end{bmatrix}$$

qui représentent les points de contrôle associés à $\mathcal{B}_{11}$, $\mathcal{B}_{12}$, $\mathcal{B}_{22}$, $\mathcal{B}_{21}$.

Remarque:

- On peut s'arrêter à $\mathcal{B}_1$ et $\mathcal{B}_2$ et ainsi découper uniquement en deux parties le carreau $\mathcal{B}$.

- Le choix de u* et v* est libre. Il est possible d'obtenir des carreaux de taille très différentes.



- Le point $\mathbf{R}_{mn}$, commun aux 4 matrices, est le point $\sigma(u^*,v^*)$.[BAR 88].

Note

La programmation de l'algorithme de De Casteljau conduit à un sous programme dont l'utilisation se résume à:

ENTREES: - Matrice des points de contrôle du carreau de Bézier initial
- Valeurs paramétriques de découpage: u^* et v^* comprises entre 0 et 1

SORTIES: - Les 4 matrices de points de contrôle des sous carreaux

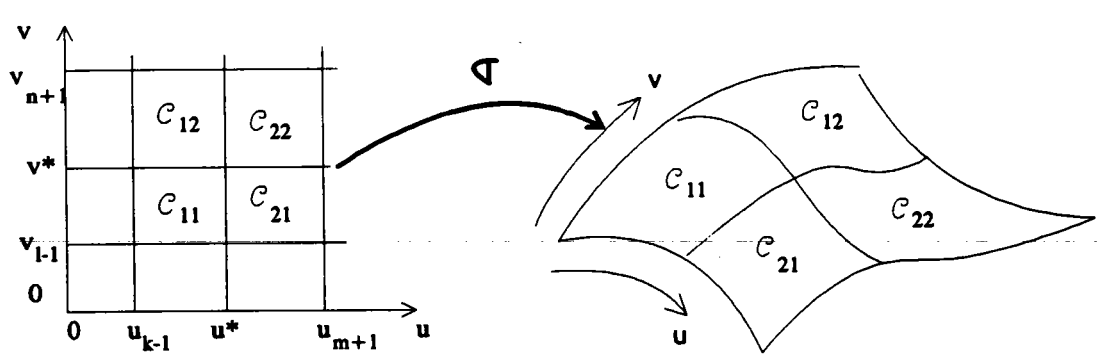
I.3.4 Application: décomposition arborescente d'un carreau de Bézier

Utilisé récursivement, ce "partitionnement" décompose donc un carreau de Bézier en un arbre 4-aire de carreaux de Bézier de même degré que le carreau initial.
Les sous-carreaux successifs sont de plus en plus petits et de plus en plus réguliers; les réseaux de contrôle successifs approchent de mieux en mieux leurs surfaces associées (propriété intrinsèque aux carreaux de Bézier). Cela constitue un moyen d'approximation polyédrique de la surface. La connaissance de la surface est préservée à chaque étape de la subdivision récursive.

I.3.5 Partition d'un carreau B-spline

Le processus de subdivision d'une surface B-spline est analogue à celui d'un carreau de Bézier. Soit $\mathcal{C}$ une surface B-spline notée comme dans l'introduction, nous allons construire une partition de $\mathcal{C}$, $\mathcal{C} = \mathcal{C}_{11} \cup \mathcal{C}_{21} \cup \mathcal{C}_{22} \cup \mathcal{C}_{12}$ grâce à l'algorithme de Cox-De Boor. Les deux lignes de coupe sont des isoparamétriques définies à $u_{fixé}$ et à $v_{fixé}$.
On note $u^* = u_{fixé}$ et $v^* = v_{fixé}$.
(u^*,v^*) est une valeur quelconque du pavé $[u_{k-1},u_{m+1}[\times[v_{l-1},v_{n+1}[$.

Les 4 sous-carreaux obtenus $\mathcal{C}_{11}$, $\mathcal{C}_{21}$, $\mathcal{C}_{22}$, $\mathcal{C}_{12}$ sont encore des surfaces B-spline de même ordre que $\mathcal{C}$.
Les carreaux paramétriques de $\mathcal{C}_{11}$, $\mathcal{C}_{21}$, $\mathcal{C}_{22}$, $\mathcal{C}_{12}$ sont respectivement:
$[u_{k-1},u^*[\times[v_{l-1},v^*[$, $[u^*,u_{m+1}[\times[v_{l-1},v^*[$, $[u^*,u_{m+1}[\times[v^*,v_{n+1}[$, $[u_{k-1},u^*[\times[v^*,v_{n+1}[$.



- Calcul des nouveaux réseaux de définition:

Le réseau de contrôle de $\mathcal{C}$ peut s'écrire sous la forme matricielle

$$P = \begin{bmatrix} P_{00} & \cdots\cdots & P_{0n} \\ \vdots & & \vdots \\ P_{i0} & & P_{in} \\ \vdots & & \vdots \\ P_{m0} & \cdots\cdots & P_{mn} \end{bmatrix}$$

On choisit de découper d'abord selon la courbe $\sigma(u,v^*)$, on obtient les surfaces $\mathcal{C}_1$ et $\mathcal{C}_2$.
$\mathcal{C}_1 = \mathcal{C}_{11} \cup \mathcal{C}_{21}$, $\mathcal{C}_2 = \mathcal{C}_{12} \cup \mathcal{C}_{22}$.

Soient g l'unique indice tel que $u_g \leq u^* \leq u_{g+1}$ et d l'unique indice tel que $v_d \leq v^* \leq v_{d+1}$

Chaque ligne $[\mathbf{P}_{ij}]_{j=0..n}$ est traitée comme suit:
L'algorithme de Cox - De Boor ayant pour données les ℓ points $\mathbf{P}_{i,d-\ell+1}, \ldots, \mathbf{P}_{i,d}$, la table nodale tndv et la valeur v^*, rend les $(2\ell-1)$ points $\mathbf{D}_{d-\ell+1}, \ldots, \mathbf{D}_d, \ldots, \mathbf{D}_{d+\ell-1}$.

La ligne $[\mathbf{P}_{ij}]_{j=o..n}$ est remplacée par la ligne $[\mathbf{Q}_{ij}]_{j=0..n+\ell-1}$

$\mathbf{Q}_{i0} = \mathbf{P}_{i0} \quad \ldots\ldots \quad \mathbf{Q}_{i,d-\ell} = \mathbf{P}_{i,d-\ell}$

$\mathbf{Q}_{i,d-\ell+1} = \mathbf{D}_{d-\ell+1} \quad \ldots\ldots \quad \mathbf{Q}_{i,d+\ell-1} = \mathbf{D}_{d+\ell-1}$

$\mathbf{Q}_{i,d+\ell} = \mathbf{P}_{i,d+1} \quad \ldots\ldots \quad \mathbf{Q}_{i,n+\ell-1} = \mathbf{P}_{i,n}$

Les matrices

$$\begin{bmatrix} Q_{00} & \cdots\cdots & Q_{0d} \\ Q_{m0} & \cdots\cdots & Q_{md} \end{bmatrix} \quad \begin{bmatrix} Q_{0d} & \cdots\cdots & Q_{0n+l-1} \\ Q_{md} & \cdots\cdots & Q_{m\,n+l-1} \end{bmatrix}$$

déterminent les réseaux de contrôle de $\mathcal{C}_1$ et $\mathcal{C}_2$

Les colonnes des matrices vont être traitées comme le furent les lignes. L'algorithme de Cox - De Boor remplace les k points $\mathbf{Q}_{g-k+1,j}$, ..,$\mathbf{Q}_{g,j}$ par les (2k-1) points $\mathbf{D}_{g-k+1},\ldots,\mathbf{D}_{g+k-1}$ Cox-De Boor est utilisé avec u* et la table des noeuds tndu.

On obtient alors les colonnes $[\mathbf{R}_{ij}]_{i=0..m+k-1}$

$$\mathbf{R}_{0,j} = \mathbf{Q}_{0,j} \quad \ldots\ldots \quad \mathbf{R}_{g-k,j} = \mathbf{Q}_{g-k,j}$$

$$\mathbf{R}_{g-k+1,j} = \mathbf{D}_{g-k+1} \quad \ldots\ldots \quad \mathbf{R}_{g+k-1,j} = \mathbf{D}_{g+k-1}$$

$$\mathbf{R}_{g+k,j} = \mathbf{Q}_{g+1,j} \quad \ldots\ldots \quad \mathbf{R}_{m+k-1,j} = \mathbf{Q}_{m,j}$$

Finalement, on obtient 4 nouvelles matrices

$$\begin{bmatrix} R_{0\,0} & \ldots\ldots\ldots & R_{0\,d} \\ R_{g\,0} & \ldots\ldots\ldots & R_{g\,d} \end{bmatrix} \quad \begin{bmatrix} R_{0\,d} & \ldots\ldots\ldots & R_{0\;n+l-1} \\ R_{g\,d} & \ldots\ldots\ldots & R_{g\;n+l-1} \end{bmatrix}$$

$$\begin{bmatrix} R_{g\,0} & \ldots\ldots\ldots & R_{g\,d} \\ R_{m+k-1\;0} & \ldots\ldots\ldots & R_{m+k-1\;d} \end{bmatrix} \quad \begin{bmatrix} R_{g\,d} & \ldots\ldots\ldots & R_{g\;n+l-1} \\ R_{m+k-1\;d} & \ldots\ldots & R_{m+k-1\;n+l-1} \end{bmatrix}$$

qui sont les réseaux de contrôle des surfaces $\mathcal{C}_{11}$, $\mathcal{C}_{21}$, $\mathcal{C}_{22}$, $\mathcal{C}_{12}$.

<u>Détermination des vecteurs nodaux des sous-carreaux</u>:

Les tables nodales associés à $\mathcal{C}_{11}$ sont:

$\{u_0, \ldots, u_g, u^*, \ldots u^*\}$ u^* est répété k fois

et $\{v_0, \ldots, v_d, v^*, \ldots, v^*\}$ v^* est répété l fois

Les tables nodales associés à $\mathcal{C}_{21}$ sont:

$\{u^*, .. u^*, u_{g+1}, .. , u_{m+k}\}$ u^* est répété k fois

et $\{v_0, .. , v_d, v^*, .. , v^*\}$ v^* est répété ℓ fois

Les tables nodales associés à $\mathcal{C}_{12}$ sont:

$\{u_0, .. , u_g, u^*, .. u^*\}$ u^* est répété k fois

et $\{v^*, .. , v^*, v_{d+1}, .. , v_{n+\ell}\}$ v^* est répété ℓ fois

Les tables nodales associés à $\mathcal{C}_{22}$ sont:

$\{u^*, .. u^*, u_{g+1}, .. , u_{m+k}\}$ u^* est répété k fois

et $\{v^*, .. , v^*, v_{d+1}, .. , v_{n+\ell}\}$ v^* est répété ℓ fois

Remarque: $\mathbf{R}_{g,d} = \sigma(u^*, v^*)$

Cas particulier important:

u^* peut être choisi parmi les valeurs nodales. C'est à dire que $u^* = u_d$ tel que $k-1 \leq d \leq m+1$. De même v^* peut être choisi parmi $v_{\ell-1}, \ldots, v_{n+1}$.

Pour chacun des sous-carreaux obtenus alors, le nombre de lignes (ou de colonnes) de la matrice des points de contrôle diminue: ils perdent des points de contrôle par rapport à $\mathcal{C}$ mais leur ordre est toujours $k \times \ell$. **Ils se rapprochent de la forme de Bézier.**
En outre l'algorithme de Cox- De Boor prend une forme plus simple; il est décrit en annexe.

Note
L'algorithme de Cox - De Boor appliqué à la subdivision des surfaces conduit à un sous programme dont l'utilisation se résume à:

ENTREES: - Matrice des points de contrôle du carreau initial
- Vecteurs nodaux du carreau initial
- Valeurs paramétriques de découpage: u^* et v^*

SORTIES: - Les 4 sous carreaux complets { Matrice de points de contrôle; Vecteurs nodaux}

I.3.6 Application: décomposition d'un carreau B-spline en carreaux de Bézier

Utilisé récursivement, ce "partitionnement" décompose donc un carreau B-spline en un arbre 4-aire de carreaux B-spline.
La connaissance de la surface est préservée à chaque étape de la subdivision récursive.

Si les valeurs u* et v* fixant les lignes de coupe de chaque sous-carreau de l'arbre sont des valeurs nodales non terminales, les fils de chaque sous-carreau perdent des lignes (ou des colonnes) de points de contrôle par rapport à leur père. Il en résulte qu'après un nombre fini de découpages successifs, on aboutit à des sous-carreaux qui sont des carreaux de Bézier. On obtient donc un arbre 4-aire dont les feuilles sont des carreaux de Bézier.

Nous proposons donc de découper chaque sous-carreau comme suit:
Soit $\mathcal{C}$ un sous-carreau quelconque d'ordre k x ℓ et de vecteurs nodaux $\{u_0,\ldots,u_{m+k}\}$ et $\{v_0,\ldots,V_{n+\ell}\}$
Si $m+1 > k$ et $n+1 > \ell$, alors $\mathcal{C}$ est découpé suivant les lignes de coupe déterminées par u* et v*.
On choisit $u^* = u_{(m+1+k-1)div2} = u_{(m+k)div2}$
et $v^* = v_{(n+1+\ell-1)div2} = {}_{(n+\ell)div2}$ où div = division entière
Ces valeurs de découpage sont les noeuds médians. Ce sont celles qui équilibrent le mieux la partition obtenue.
Si $m+1 = k$ et $n+1 = \ell$ alors $\mathcal{C}$ est déjà un carreau de Bézier; il n'est pas découpé.

Les valeurs de u^* et v^* indiquées minimisent le nombre d'étages de l'arbre et garantissent un arbre équilibré.

Cas particulier

Si $\mathcal{C}$ vérifie k = m+1 et ℓ < n+1 (par exemple), c'est déjà un carreau de Bézier dans la direction u (C'est à dire qu'une courbe isoparamétrique à v fixé est une courbe de Bézier). Le carreau est alors seulement partitionné en 2 sous carreaux.
La ligne de coupe est une isoparamétrique à $v_{fixé} = v_{(\ell+n)div2}$

Remarque:
Un carreau B-spline est polynomial par morceaux. Sur chaque pavé $[u_g, u_{g+1}[x[v_d, v_{d+1}[$ non vide de son rectangle paramétrique, c'est un polynôme de degré (k-1)x(ℓ-1). L'algorithme que nous venons de décrire découpe le carreau suivant tous les couples (u_g,v_d), par conséquent il fournit la forme polynomiale (écrite dans la base des polynômes de Bernstein) correspondant à chaque pavé.

I.3.7 Partitionnement d'un carreau rationnel

Soit un carreau rationnel $\mathcal{N}$ (Bézier ou B-spline). Appelons $\mathcal{C}$ le carreau polynomial tel que $\mathcal{N}$ = H($\mathcal{C}$) donc si nous avons une partition $\mathcal{C}_{11}$, $\mathcal{C}_{12}$, $\mathcal{C}_{22}$, $\mathcal{C}_{21}$ de $\mathcal{C}$,
H($\mathcal{C}_{11}$), H($\mathcal{C}_{12}$), H($\mathcal{C}_{22}$), H($\mathcal{C}_{21}$) est une partition de $\mathcal{N}$. Donc pour décomposer le carreau rationnel $\mathcal{N}$ il suffit de décomposer le carreau polynomial $\mathcal{C}$ dans l'espace homogène puis, de projeter les 4 sous-carreaux obtenus.

I.3.8 Décomposition arborescente d'un carreau B-spline rationnel

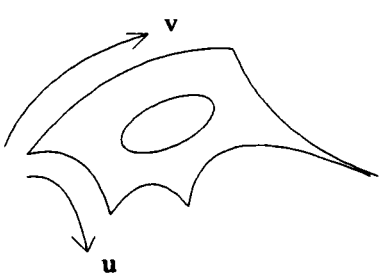
Un carreau rationnel est décomposé en un arbre 4-aire de la même façon qu' un carreau polynomial. Tous les calculs de partitionnement sont effectués dans l'espace homogène où elles sont polynomiales. Le processus de subdivision est alors strictement le même.

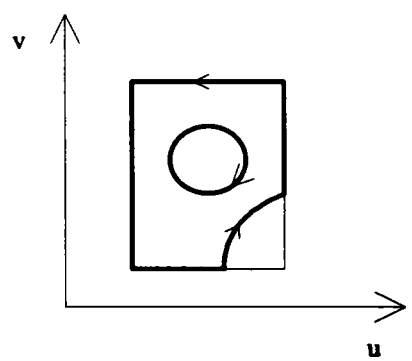
II FORMULATION MATHEMATIQUE

Dans ce chapitre nous exposons quelques rappels mathématiques, précisons les notations qui sont adoptées par la suite et formulons quelques remarques générales sur les intersections.

II.1 DEFINITION D'UN CARREAU RESTREINT

Les carreaux restreints sont des carreaux NURBS qui ont été découpés suivant un certain contour et dont on ne conserve qu'une partie.
Un carreau restreint est donc représenté par la donnée d'un carreau NURBS et d'une liste de contours 2D. Ces contours sont inclus dans le rectangle paramétrique du carreau NURBS; ils sont fermés ou bien il forment un contour fermé en étant associés aux bords. De plus ils sont et orientés et disjoints.
Ils peuvent être représentés par une succession de points ou par des arcs de courbe polynomiaux [ZID 90].





L'orientation des contours permet de savoir de quel coté se trouve la partie à conserver.

II.2 INTERSECTION DE DEUX CARREAUX RESTREINTS

Le calcul des courbes d'intersection entre deux carreaux restreints est effectué en deux étapes:

1) Intersection des carreaux complets
Dans un premier temps les carreaux complets sont étudiés. Les courbes correspondant à leur intersection entre ces deux derniers sont construites.

2) Intersection des carreaux restreints
Les courbes d'intersection de ces derniers sont les intersections des courbes précédentes avec le domaine utile de chaque carreau. La finalité de ces calculs est la réalisation de combinaisons booléennes des deux carreaux restreints (intersection, réunion, différence, ...) Les courbes d'intersection obtenues définissent pour chaque carreau une partie à conserver et une partie à éliminer. Elles modifient donc le domaine 2D du carreau restreint. Une reconstruction des contours et une mise à jour de la liste des contours de chaque carreau est donc effectuée.

Dans la suite de cet exposé nous décrivons les méthodes qui traitent l'intersection de deux carreaux NURBS complets.

II.3 FORMULATION MATHEMATIQUE D'UNE COURBE D'INTERSECTION

II.3.1 Définition et notation de deux carreaux complets

Soient δ_1 et δ_2 les deux carreaux dont on recherche la courbe d'intersection.
δ_1 est un carreau NURBS de degré $m_1 \times n_1$ dont une paramétrisation est:

$$\mathbf{X}: [a_1, b_1] \times [c_1,d_1] \subset \mathbf{R}^2 \longrightarrow \mathbf{R}^3$$
$$(u, v) \longmapsto \mathbf{X}(u, v)$$

$\mathcal{R}_1 = [a_1, b_1] \times [c_1,d_1]$ est appelé rectangle paramétrique de δ_1

De même δ_2 est un carreau NURBS de degré $m_2 \times n_2$ décrit par l'application:

$$\mathbf{Y}: [a_2, b_2] \times [c_2,d_2] \subset \mathbf{R}^2 \longrightarrow \mathbf{R}^3$$
$$(r, s) \longmapsto \mathbf{Y}(r, s)$$

$\mathcal{R}_2 = [a_2, b_2] \times [c_2,d_2]$ est le rectangle paramétrique de δ_2

Rappelons que **X** et **Y** doivent être des bijections de $\mathcal{R}_1$ (resp $\mathcal{R}_2$) sur δ_1 (resp δ_2) afin que ces surfaces soient correctement définies. Les carreaux NURBS dégénérés (à 2 cotés ou 3 cotés) sont des exemples de paramétrisations non injectives.

II.3.2 Définition d'une courbe d'intersection de deux carreaux de surface

Une courbe d'intersection de deux carreaux de surface est une courbe de l'espace qui est inscrite simultanément sur les deux carreaux. Dans le cas général elle est constituée d'un ensemble d'arcs connexes, disjoints, fermés (ou bien coupés par les bords de l'un des carreaux).

Hypothèses :
Nous supposerons que les deux carreaux ne présentent pas d'auto-intersection, ni de points anguleux.

II.3.3 Les différentes représentations d'une courbe d'intersection

II.3.3.1 Equation cartésienne d'une courbe

Supposons que l'on dispose pour l'un des deux carreaux d'une équation cartésienne, par exemple pour $\mathcal{S}_2$. Alors tout point $\mathbf{P}(x,y,z)$ appartenant à $\mathcal{S}_2$ vérifie une équation $F_2(x,y,z)=0$ où F_2 est un polynôme en x, y et z. Les points de $\mathcal{S}_1$ sont donnés par $\mathbf{X}(u,v) = (x_1(u,v),\ y_1(u,v),\ z_1(u,v))$. Par conséquent les points de la courbe d'intersection vérifient l'équation $F_2(x_1(u,v),\ y_1(u,v),\ z_1(u,v)) = 0$ que l'on peut noter $f(u,v) = 0$. C'est une équation de la courbe d'intersection dans le rectangle paramétrique $\mathcal{R}_1$.

II.3.3.2 Equation paramétrique d'un arc de courbe d'intersection

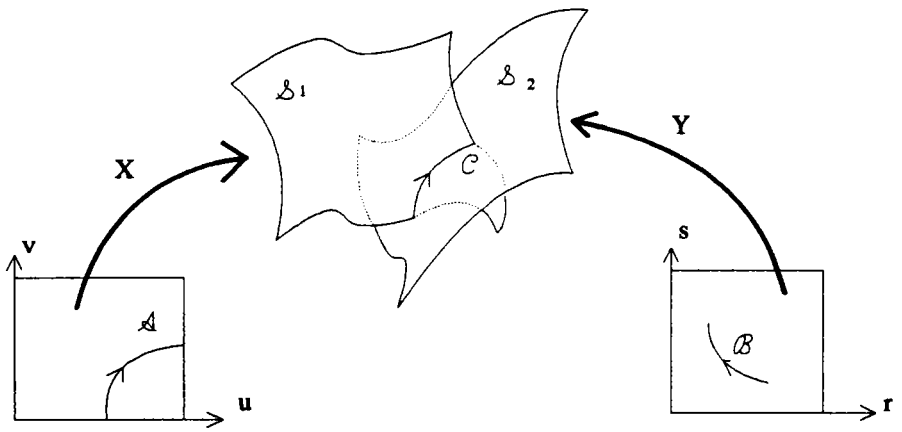
Soit $\mathcal{C}$ un arc de courbe décrivant toute (ou une partie de) la courbe d'intersection entre $\mathcal{S}_1$ et $\mathcal{S}_2$. Appelons $\mathcal{A}$ (resp $\mathcal{B}$) la courbe plane projection de $\mathcal{C}$ par $\mathbf{X}^{-1}$ (resp $\mathbf{Y}^{-1}$) sur le rectangle $\mathcal{R}_1$(resp $\mathcal{R}_2$). Notons α, β, γ les paramétrisations de $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$:

α: $[0, L] \subset \mathbf{R} \longrightarrow \mathcal{A} \subset \mathcal{R}_1$
$t \longmapsto \alpha(t) = (u(t), v(t))$ α est une bijection de [0, L] sur $\mathcal{A}$

β: $[0, L] \longrightarrow \mathcal{B} \subset \mathcal{R}_2$
$t \longmapsto \beta(t) = (r(t), s(t))$ β est une bijection de [0, L] sur $\mathcal{B}$

γ: $[0, L] \longrightarrow \mathcal{C} \subset \mathbf{R}^3$
$t \longmapsto \gamma(t) = (x(t), y(t), z(t))$ γ est une bijection de [0, L] sur $\mathcal{C}$

qui vérifient $\gamma(t) = \mathbf{X}(\alpha(t)) = \mathbf{Y}(\beta(t))$ pour tout t.
L'arc de courbe $\mathcal{C}$ définit en fait trois arcs de courbe, l'un spatial et les deux autres plans.



Les trois paramétrisations peuvent être très complexes et sont en général non rationnelles. Nous supposerons qu'elles sont C^2 voir C^3 sauf en des points isolés où elles sont C^0.
Certaines méthodes de calcul des courbes d'intersections utilisent cette formalisation. Nous utiliserons également les propriété du trièdre de Frénet que nous rappelons brièvement.

Trièdre de Frénet

Soit un arc courbe $\mathcal{C}$ spatial.
Notons σ: [0, L] $\longrightarrow \mathcal{C}$
$s \longmapsto \sigma(s)$ la paramétrisation intrinsèque de $\mathcal{C}$ [LEL 63]
C'est l'unique paramétrisation de $\mathcal{C}$ qui vérifie la condition:

s = longueur de l'arc $\overset{\frown}{(\sigma(0), \sigma(s))}$ pour tout s

Cette condition est équivalente à : $\| \sigma'(s) \| = 1$ pour tout s
($\sigma'(s)$ est le vecteur tangent unitaire)
Autrement dit σ parcourt la courbe $\mathcal{C}$ à une vitesse constante égale à 1. s est appelé abscisse curviligne.
Le vecteur dérivée seconde $\sigma''(s)$ est donc orthogonal à $\sigma'(s)$ pour tout s, donc normal à la courbe en tout point. On appelle courbure au point $\sigma(s)$ la quantité $\rho(s) = \| \sigma''(s) \|$ et on définit $\mathbf{n}$ vecteur unitaire normal à la courbe et orienté vers le centre de courbure par

$$\mathbf{n} = \frac{1}{\rho(s)} \sigma''(s).$$

La courbure a une signification géométrique immédiate:

Le cercle tangent à $\mathcal{C}$ en un point $\sigma(s)$ a pour rayon $R = \frac{1}{\rho(s)}$

On note également que ce cercle a pour centre le point **M** tel que $\overrightarrow{\sigma(s)M} = R.\mathbf{n}$

Un des intérêts de ce qui précède est de pouvoir approcher localement une courbe par un arc de cercle ou par un développement limité à l'ordre 2 ayant un sens géométrique (plutôt que par la droite engendrée par le vecteur tangent).

On définit le vecteur binormale $\mathbf{b} = \mathbf{t} \wedge \mathbf{n}$ où $\mathbf{t} = \sigma'(s)$ (2)

$\{\mathbf{t},\mathbf{n},\mathbf{b}\}$ est une base orthonormale directe associée à chaque point de $\mathcal{C}$. Cette base est le trièdre de Frénet.
$\mathbf{b}'$ vérifie $\mathbf{b}' = \tau\,\mathbf{n}$ $\quad s \longmapsto \tau(s)$ est appelée la torsion de la courbe.
On montre alors que $\mathbf{n}' = -\rho\,\mathbf{t} - \tau\,\mathbf{b}$

La connaissance du comportement local d'une courbe est facilitée par cette paramétrisation intrinsèque.
Lorsque l'arc de courbe $\mathcal{C}$ est connu par une paramétrisation quelconque $t \longmapsto \gamma(t)$ les quantités **t**, **n**, ρ sont toujours accessibles grâce aux relations:

$\mathbf{t} = \frac{1}{\| \gamma'(t) \|}\gamma'(t)$. (dans le cas général $\gamma'(t)$ est toujours un vecteur tangent à $\mathcal{C}$, orienté)

et $\rho(s)\,\mathbf{n} = \sigma''(s) = \frac{1}{\| \gamma'(t) \|^4}(\| \gamma'(t) \|^2 \gamma'(t) - \gamma'(t)(\gamma'(t)|\gamma''(t))$ (3)

d'où $\rho(s) = \| \sigma''(s) \| = \frac{\| \gamma'(t) \wedge \gamma''(t) \|}{\| \gamma'(t) \|^3}$ et $\mathbf{n} = \frac{1}{\rho(s)}\sigma''(s)$

II.3.3.3 Connaissance d'une courbe points par points

Les méthodes de calcul de courbe d'intersection fournissent des suites de points considérés comme appartenant à la courbe d'intersection. La courbe théorique est donc connue par un échantillonnage de points sous la forme d'une suite de quadruplets (u_i, v_i, r_i, s_i) tels que $\mathbf{X}(u_i, v_i) = \mathbf{Y}(r_i, s_i) = \mathbf{P}_i$. Pour un arc de courbe connexe on peut considérer que ces données sont des valeurs pour les paramétrisations α, β, γ de $\mathcal{A}$, $\mathcal{B}$ et $\mathcal{C}$. C'est à dire que:

---

(2) Le produit vectoriel est noté $\wedge$

(3) Le produit scalaire de deux vecteurs **a** et **b** est noté (a|b)

pour tout P_i, $\exists$ t_i appartenant à [0, L] tel que $(u_i, v_i) = (u(t_i), v(t_i)) = \alpha(t_i)$
et que $(r_i, s_i) = (r(t_i), s(t_i)) = \beta(t_i)$
et que $\mathbf{P}_i = \gamma(t_i)$

Nous verrons qu'on peut toujours compléter ces informations par la connaissance de vecteurs dérivée 1ère, 2ème, ou 3ème sur chacune des courbes en chaque point.
Il est donc possible pour chaque point de connaître exactement

$$\alpha(t_i)\ \alpha'(t_i)\ \alpha''(t_i)\ ,\ldots$$
$$\beta(t_i)\ \beta'(t_i)\ \beta''(t_i)\ ,\ldots$$
$$\gamma(t_i)\ \gamma'(t_i)\ \gamma''(t_i)\ ,\ldots$$

sans la connaissance des paramétrisations exactes.
Ces informations sont utiles pour classer ces points et passer de l'échantillonnage à un modèle de courbe théorique.

II.4 PROPRIETES LOCALES DES SURFACES: PLAN TANGENT ET COURBURES PRINCIPALES

Certaines méthodes de calcul de courbes d'intersection utilisent les résultats liés à la deuxième forme quadratique fondamentale des surfaces [LEL 63]. L'application de celle-ci s'étend à d'autres domaines, la création de carreaux de raccordement par exemple [VER 73] [VER 88]. Nous ne décrirons pas cette forme; nous nous contenterons d'énoncer les résultats principaux.

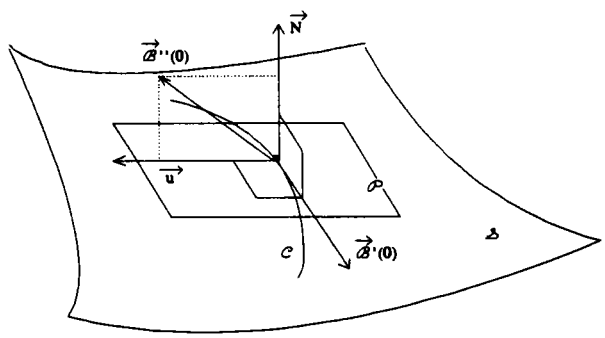
Soit un point P d'une surface $\mathcal{S}$ dont une paramétrisation est
$\mathbf{X}: \mathcal{U} \subset \mathbf{R}^2 \longrightarrow \mathcal{S} \subset \mathbf{R}^3$ $\mathcal{U}$ est une région connexe quelconque du plan
$(u, v) \longmapsto \mathbf{X}(u, v)$

Les développements qui suivent sont valables pour n'importe quelle surface C^2 et pas seulement pour un carreau NURBS. $\mathcal{S}$ représente donc une surface quelconque C^2
Soit $(\underline{u}, \underline{v})$ appartenant à $\mathcal{U}$ tel que $\mathbf{P} = \mathbf{X}(\underline{u}, \underline{v})$
Rappelons que $\mathcal{P}$ le plan tangent à $\mathcal{S}$ en **P** est engendré par les deux vecteurs dérivée partielles $\mathbf{X}_u(\underline{u}, \underline{v})$ et $\mathbf{X}_v(\underline{u}, \underline{v})$.

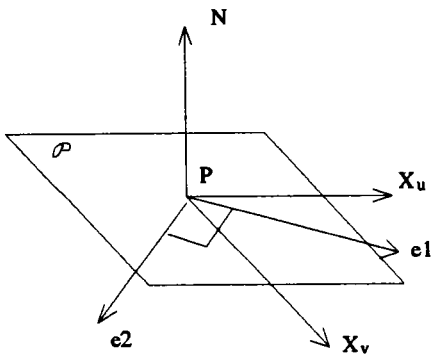
Par conséquent le vecteur $\mathbf{N} = \dfrac{1}{\| \mathbf{X}_u(\underline{u}, \underline{v}) \wedge \mathbf{X}_v(\underline{u}, \underline{v}) \|} \mathbf{X}_u(\underline{u}, \underline{v}) \wedge \mathbf{X}_v(\underline{u}, \underline{v})$ est un vecteur normal unitaire à ce plan.

Remarque:
Si $\mathbf{X}_u(\underline{u}, \underline{v}) \wedge \mathbf{X}_v(\underline{u}, \underline{v})$ s'annule la définition du plan tangent est impossible. Conformément aux hypothèses nous supposerons que cela n'est jamais le cas.

Notons $\mathbf{R}(u,v) = \{ \mathbf{P}, \mathbf{X}_u, \mathbf{X}_v, \mathbf{N} \}$ le repère lié à tout point de la surface.
La notion de courbure énoncée pour les courbes se généralise aux surfaces, mais de façon beaucoup plus complexe:
Soit $\mathcal{C}$ un arc de courbe inscrit sur $\mathcal{S}$ passant par **P**.
Notons σ: [-L, L] $\longrightarrow \mathcal{C} \subset \mathcal{S}$ sa paramétrisation intrinsèque.
On suppose que $\sigma(0) = \mathbf{P}$, par conséquent $\sigma'(0) \in \mathcal{P}$.
Choisissons $\mathbf{u} \in \mathcal{P}$ tel que $\{ \sigma'(0), \mathbf{u}, \mathbf{N} \}$ soit une base orthonormée.
$\sigma''(0)$ est un vecteur normal à la courbe en $\sigma(0)$ et $\| \sigma''(s) \|$ représente la courbure.
Donc puisque $\sigma'(0) \perp \sigma''(0)$
on a $\sigma''(0) = \rho_N \mathbf{N} + \rho_T \mathbf{u}$



La quantité ρ_N appelée courbure normale à $\mathcal{C}$ en **P** représente la courbure de $\mathcal{S}$ en **P** suivant la direction $\sigma'(0)$ c'est la courbure de la projection de $\mathcal{C}$ sur le plan $\{ \mathbf{P}, \sigma'(0), \mathbf{N} \}$. Cette courbure ne varie qu'avec l'orientation du vecteur tangent $\sigma'(0)$ dans le plan $\mathcal{P}$: elle ne dépend donc que de la forme de la surface au voisinage de **P**. ρ_N varie entre une courbure minimale k_1 et une courbure maximale k_2 qui sont appelées les **courbures principales.** Elles correspondent à deux directions particulières dans le plan tangent: **les directions principales de courbure** qui sont toujours orthogonales. Notons $\mathbf{e}_1$ et $\mathbf{e}_2$ deux vecteurs de $\mathcal{P}$ correspondant aux directions principales et donc à k_1 et k_2. La base $\{ \mathbf{e}_1, \mathbf{e}_2, \mathbf{N} \}$ est la généralisation aux surfaces du trièdre de Frénet.



Calcul de k_1, k_2, e_1, e_2 :

k_1 et k_2 sont les valeurs propres d'une matrice 2x2 particulière (représentant la deuxième forme quadratique); e_1 et e_2 sont les vecteurs propres associés à k_1 et k_2. En fait ces quantités ne nécessitent que la connaissance des dérivées partielles X_u, X_v, X_{uu}, X_{uv}, X_{vv} en **P**; les calculs sont très courts.

Le comportement d'une surface au voisinage d'un point peut être entièrement décrit à l'aide de la donnée du plan tangent, des courbures principales et des directions de courbure principale. Ces courbures principales fournissent par ailleurs une approximation quadratique très simple de la surface, localement.

Points singuliers

Les définitions précédentes nous permettent d'introduire la notion de point singulier.

Soit un point **P** d'une courbe d'intersection de deux surfaces $\mathcal{S}_1$ et $\mathcal{S}_2$. Si les plans tangents à $\mathcal{S}_1$ et $\mathcal{S}_2$ en **P** sont confondus, nous dirons que **P** est un **point singulier de 1er ordre.**
En un tel point la courbe $\mathcal{C}$ est irrégulière (généralement **P** est le carrefour de deux courbes). Presque tous les algorithmes de calcul de courbes d'intersection connus rencontrent des difficultés à traiter correctement la courbe au voisinage de **P.**
Si les surfaces admettent non seulement le même plan tangent en **P,** mais aussi les mêmes courbures principales et les mêmes directions de courbures principales nous dirons que **P** est **un point singulier de 2ème ordre.**
Bien sûr la courbe a un comportement, au voisinage de **P,** encore plus irrégulier que dans le cas précédent et par conséquent le niveau de difficultés rencontré par les algorithmes est accru.

III LES PRINCIPALES APPROCHES DU PROBLEME D'INTERSECTION

Il y a trois approches essentielles du problème d'intersection. La majorité des algorithmes implantés actuellement dans les systèmes de C.A.O s'inspire d'une ou plusieurs de ces trois méthodes. Auparavant sont introduites les notions de filtre et de volume englobant communes à plusieurs algorithmes.

III.1 VOLUMES ENGLOBANTS

Les concepts de filtre et de volume englobant sont largement utilisés dans tous les domaines de l'infographie, chaque fois que leur utilisation permet d'éviter de coûteux calculs d'intersection inutiles; le meilleur exemple est l' algorithme du tracé de rayon [VIV 93].

Un volume englobant d'un objet spatial est un domaine de l'espace dans lequel l'objet est entièrement inclus.
Le volume englobant est d'un grand intérêt dans la recherche des intersections entre deux carreaux. Lorsque chaque carreau est décomposé en plusieurs sous carreaux il permet de filtrer les sous carreaux intéressants:
Deux carreaux NURBS dont les volumes englobants sont disjoints le sont également. Les volumes englobants sont déterminés afin que les calculs engendrés par le test d'intersection de deux d'entre eux soient très rapides. Tester l'intersection des englobants de deux carreaux NURBS permet donc d'éviter de coûteux calculs inutiles dans la plupart des cas.

Le volume englobant le plus simple à obtenir pour le type de surfaces manipulées est un parallélépipède dont les arêtes sont parallèles aux 3 axes du repère (appelé aussi boite min-max). Il exploite une propriété des carreaux paramétriques: chaque point de la surface appartient à l'enveloppe convexe du polygone de contrôle. Il est donc défini par deux points de l'espace euclidien: point_min, point_max.

Notons les points de contrôle: $\mathbf{P}_{ij} = \begin{bmatrix} x_{ij} \\ y_{ij} \\ z_{ij} \end{bmatrix}$ Alors les deux points sont déterminés

comme suit:

$x_{point\_max} = \max_{ij} x_{ij}$ et $x_{point\_min} = \min_{ij} x_{ij}$

Et de même pour les autres coordonnées.

Le test d'intersection de deux tels englobants requiert au plus 3 comparaisons de réels.

D'autres volumes englobants peuvent être déterminés. Certains systèmes calculent des sphères ou des ellipsoïde ou encore des parallélépipède rectangles non parallèles aux axes du repère.

III.2 DICHOTOMIE ET INTERSECTIONS DE PLANS

Description générale

Cette méthode exploite le principe: diviser pour régner.
Elle généralise aux surfaces la méthode de dichotomie utilisée pour résoudre les équations numériques à une variable. Les deux carreaux sont décomposés récursivement jusqu'à obtenir des facettes quasi-planes. On recherche ensuite les intersections entre les portions planes obtenues; et enfin on reconstruit les courbes à partir des segments trouvés.
Toutes les facettes ne sont pas étudiées: elles sont filtrées à l'aide de volumes englobants.
Des algorithmes fonctionnant sur ce principe sont aussi décrits dans [AZI 90], [DOK 85], [HOU 85] et [MIC 87].

III.2.1 Décomposition récursive

Chacun des deux carreaux NURBS est décomposé en un arbre 4-aire de sous-carreaux. Tant que les sous-carreaux successifs ne sont pas sous la forme de Bézier, ils sont "partitionnés" grâce à l'algorithme de Cox-De Boor, puis dès que ce sont des carreaux de Bézier on poursuit les découpages grâce à l'algorithme de De Casteljau. L'algorithme réalise en fait la synthèse des deux décompositions évoquées en I.3.4 et I.3.6.

L'objectif de cette décomposition est multiple:
Un carreau NURBS est une polynomiale par morceaux (dans l'espace homogène); ses paramètres (u et v) évoluent dans des pavés délicats à manipuler. Par contre un carreau de Bézier est une polynomiale et ses paramètres évoluent toujours dans [0,1]x[0,1]. Les calculs d'intersection sont donc plus simples à entreprendre avec la forme de Bézier. Il est donc avantageux de décomposer un carreau B-spline en carreaux de Bézier. Ces carreaux de

Bézier sont eux-mêmes décomposés en arbres: La taille des sous-carreaux diminue à chaque étage et après un nombre suffisant de découpages, il est possible d'assimiler ces sous-carreaux à des plans.
Globalement cette partie consiste donc à ramener le problème de l'intersection de deux carreaux NURBS à l'intersection de deux plans.

III.2.3 Algorithme général

Les deux carreaux NURBS $\mathcal{N}_1$ et $\mathcal{N}_2$ sont décomposés simultanément en arbres 4-aires; à chaque découpage on ne conserve que les sous-carreaux provenant de $\mathcal{N}_1$ et de $\mathcal{N}_2$ dont les boites min-max sont sécantes. Les arbres sont donc incomplets. L'algorithme aboutit à la formation de couples de sous-carreaux "suffisamment plats" ($\mathcal{B}_1$ sous-carreau de $\mathcal{N}_1$, $\mathcal{B}_2$ sous-carreau de $\mathcal{N}_2$) impliqués dans l'intersection de $\mathcal{N}_1$ et $\mathcal{N}_2$. les intersections de ces couples sont calculées comme des segments. Ces segments sont rangés dans une liste qui est traitée plus tard.

L'algorithme se présente comme un module récursif DICHOTOMIE:

Description de DICHOTOMIE:

Arguments: $\mathcal{N}_1$, $\mathcal{N}_2$ (* 2 carreaux Nurbs ou Bézier rationnel sous la forme de noeuds d'arbres *)
L: (* liste des segments déjà calculés *)

(* cas terminal: Les volumes englobants de $\mathcal{N}_1$ et $\mathcal{N}_2$ sont disjoints ou les deux carreaux sont suffisamment plats *)

Début
Si boîte($\mathcal{N}_1$) $\cap$ boîte($\mathcal{N}_2$) $\neq \phi$
Alors si $\mathcal{N}_1$ est suffisamment plat *et* $\mathcal{N}_2$ est suffisamment plat
Alors
- Calculer le segment-intersection de $\mathcal{N}_1$ et $\mathcal{N}_2$
- ranger le segment dans la liste L

Sinon
Si Les 4 fils du noeud contenant $\mathcal{N}_1$ n'existent pas
Alors

- Calcul des paramètres de découpage u^* et v^*
- Subdiviser η_1 en η_{11} η_{12} η_{13} η_{14}

(* A l'aide de Cox-De Boor ou De casteljau selon la nature du carreau*)

- Créer les 4 carreaux-fils du noeud contenant η_1: fils N-W, fils N-E, fils S-E, fils S-W

Fin si

Si Les 4 fils du noeud contenant η_2 n'existent pas

Alors

- Calcul des paramètres de découpage r^* et s^*
- Subdiviser η_2 en η_{21} η_{22} η_{23} η_{24}
- Créer les 4 carreaux-fils du noeud contenant η_2: fils N-W, fils N-E, fils S-E, fils S-W

Fin si

- Appeler DICHOTOMIE(fils N-W(η_1), fils N-W(η_2))
- Appeler DICHOTOMIE(fils N-W(η_1), fils N-E(η_2))
- Appeler DICHOTOMIE(fils N-W(η_1), fils S-E(η_2))
- Appeler DICHOTOMIE(fils N-W(η_1), fils S-W(η_2))
- Appeler DICHOTOMIE(fils N-E(η_1), fils N-W(η_2))

.....

(* En tout 16 appels récursifs *)

.....

- Appeler DICHOTOMIE(fils S-W(η_1), fils S-W(η_2))

Fin si

Fin si

Fin

III.2.4 Calcul de l'intersection de deux carreaux "suffisamment plats"

Chacun des deux carreaux $\mathcal{B}_1$, $\mathcal{B}_2$ est approché par un quadrilatère plan.
Chacun de ces quadrilatères est déterminé par un plan $\mathcal{P}$ de l'espace et par un contour 2D à 4 cotés donné dans le plan $\mathcal{P}$.
L'intersection des deux quadrilatères fournit un segment $\mathcal{J}$ dans l'espace. Ce dernier est une approximation de l'arc de courbe $\mathcal{C}$ intersection exacte de $\mathcal{B}_1$, $\mathcal{B}_2$. Les deux extrémités $\mathbf{P}_{\text{début}}$, $\mathbf{P}_{\text{fin}}$ du segment $\mathcal{J}$ sont par conséquent des approximations de points $\mathbf{P}_{\text{début}}$, $\mathbf{P}_{\text{fin}}$ situés sur $\mathcal{C}$. Les points $\mathbf{P}_{\text{début}}$ et $\mathbf{P}_{\text{fin}}$ sont calculés, puis le segment qu'il forment est rangé dans la liste L.

III.2.4.1 Détermination du plan approchant le carreau

La façon la plus simple de déterminer $\mathcal{P}$ est de fixer un point **A** appartenant au plan et un vecteur normal (de préférence unitaire) **N** au plan. Le point **A** peut être le barycentre du réseau de contrôle. De même **N** peut être le barycentre d'une famille de normales aux points du réseau de contrôle. A cette fin on peut définir une "pseudo-normale" en un point du réseau en prenant le produit vectoriel de deux arêtes jointives en ce point (cette définition est ambigüe car il y a plusieurs possibilités). Le plan P ainsi déterminé est satisfaisant; cela est dû à deux raisons:
- Le carreau est "relativement plat" donc les pseudo-normales varient peu sur le réseau.
- Comme le carreau est issu de plusieurs partitions successives, le polyèdre de contrôle est très proche du carreau (propriété fondamentale de la subdivision récursive d'un carreau de Bézier).

Remarque: On peut simplifier la méthode en prenant pour **A** le barycentre des 4 coins du carreau et pour **N** le barycentre des normales aux quatre coins du carreau.

III.2.4.2 Détermination du contour 2D

Ce dernier est déterminé par projection sur $\mathcal{P}$ des 4 coins du carreau.

III.2.4.3 Calcul de l'intersection des deux quadrilatères plans

La droite d'intersection des deux plans approchant $\mathcal{B}_1$ et $\mathcal{B}_2$ est déterminée puis le segment $\mathcal{J}$ est obtenu en coupant la droite par les deux contours 2D.

III.2.4.4 Calcul de points exacts sur la courbe d'intersection

Les deux extrémités de $\mathcal{I}$ sont connues par leur coordonnées spatiales; $\underline{\mathbf{P}}_{\text{début}} = (x_1, y_1, z_1)$ et $\underline{\mathbf{P}}_{\text{fin}} = (x_2, y_2, z_2)$.

Notons $\mathbf{X}$: $[0, 1] \times [0,1] \longrightarrow \mathbf{R}^3$

$(u, v) \longmapsto \mathbf{X}(u, v)$ la paramétrisation de $\mathcal{B}_1$

et $\mathbf{Y}$: $[0, 1] \times [0,1] \longrightarrow \mathbf{R}^3$

$(r, s) \longmapsto \mathbf{Y}(r, s)$ la paramétrisation de $\mathcal{B}_2$

Les informations intéressantes relatives aux points $\mathbf{P}_{\text{début}}$ et $\mathbf{P}_{\text{fin}}$ sont leurs coordonnées paramétriques $(u_{\text{début}}, v_{\text{début}}), (r_{\text{début}}, s_{\text{début}}), (u_{\text{fin}}, v_{\text{fin}}), (r_{\text{fin}}, s_{\text{fin}})$.

Le quadruplet $(u_{\text{début}}, v_{\text{début}}, r_{\text{début}}, s_{\text{début}})$ peut être trouvé par une méthode Newton-Raphson (qui sera décrite dans IV.4.1) dont la mise en oeuvre exige une solution approchée $(\underline{u}, \underline{v}, \underline{r}, \underline{s})$.

Les valeurs $\underline{u}$ et $\underline{v}$ sont obtenues de la façon suivante:

On recherche le point de contrôle $\mathbf{P}_{i^* j^*}$ du réseau de contrôle $\{\mathbf{P}_{ij}\}_{i=0\ldots m,\, j=0\ldots n;}$ de $\mathcal{B}_1$ le plus proche de $\underline{\mathbf{P}}_{\text{début}}$ et on choisit $\underline{u} = i^*/m$ et $\underline{v} = j^*/n$.
En effet le carreau $\mathcal{B}_1$ est très régulier donc $\mathbf{X}(i^*/m , j^*/n)$ est un point de $\mathcal{B}_1$ très proche de $\mathbf{P}_{i^* j^*}$ et par conséquent de $\underline{\mathbf{P}}_{\text{début}}$. Ces valeurs sont très proches de la solution exacte et conviennent donc très bien à la méthode de Newton.
On obtient de façon analogue $\underline{r}$ et $\underline{s}$.

Le calcul du quadruplet $(u_{\text{fin}}, v_{\text{fin}} , r_{\text{fin}}, s_{\text{fin}})$ suit le même processus.

III.2.5 Evaluation du caractère "suffisamment plat"

Un sous-carreau $\mathcal{B}$ peut être approché par un quadrilatère plan si il est suffisamment plat et si ses quatre frontières peuvent être assimilées à des segments de droite.

Test du caractère "suffisamment plat":

Le test est extrêmement simple si le plan $\mathcal{P}$ approchant $\mathcal{B}$ est connu.
On détermine donc d'abord le plan approchant (c'est à dire **A** et **N** en fonction des quatre coins de $\mathcal{B}$) puis une équation $f(x,y,z) = a.x + b.y + c.z + d = 0$ de $\mathcal{P}$ à partir de **A** et **N**.
(Rappelons que la distance d'un point quelconque **M**(x,y,z) à $\mathcal{P}$ est tout simplement $f(x,y,z)$ si **N** est unitaire).
On calcule ensuite la distance de chaque point de contrôle au plan.

Critère:
Le carreau $\mathcal{B}$ est estimé suffisamment plat si:

distance($\mathbf{P}_{ij}$, $\mathcal{P}$) $< \varepsilon$ $\forall$ i, $\forall$j où ε est une constante arbitrairement petite.

Cette condition garantit que distance($\mathcal{P}$, $\mathcal{B}$) $< \varepsilon$ car le carreau est inclus dans l'enveloppe convexe de son réseau de contrôle $\{\mathbf{P}_{ij}\}_{i=0...m,\ j=0...n;}$

Test de linéarité des bords:

Ce test est effectué si le précédent a été effectué avec succès.
Le processus est analogue au précédent:
On projette les 4 coins de $\mathcal{B}$ sur $\mathcal{P}$, on détermine ainsi le quadrilatère approchant.
On détermine ensuite les équations (en 2D) des droites joignant les 4 sommets du quadrilatère.
On projette les points de contrôle périphériques sur $\mathcal{P}$ et on calcule leur distance aux droites correspondantes. Si la plus grande distance trouvée est inférieure à une tolérance donnée, les bords sont considérés comme linéaires.

III.2.6 Reconstruction des courbes d'intersection

Lorsque l'étape de dichotomie récursive est achevée la liste L contient tous les segments de taille négligeable impliqués dans la courbe d'intersection entre $\mathcal{N}_1$ et $\mathcal{N}_2$. Ces segments forment les pièces d'un puzzle qu'il faut assembler pour reconstituer la courbe d'intersection. Les segments sont mis bout à bout et réorientés si nécessaire pour former des sous listes qui, lorsqu'elles sont complètes sont les différents arcs connexes de la courbe.
Pour une sous-liste en cours de reconstruction:

On recherche dans la liste L le segment dont une extrémité est la plus proche d'une des extrémités de la sous-liste. Si la distance entre les deux extrémités (celle de la sous-liste et celle du segment trouvé) est inférieure à un certain seuil de tolérance, le segment trouvé et la sous-liste sont concaténés et le segment est retiré de la liste L.
Afin de diminuer le risque d'erreur le test de proximité peut être effectué dans les 3 domaines: l'espace euclidien, le rectangle paramétrique de η_1 et le rectangle paramétrique de η_2.
Il exploite la continuité des courbes α, β, γ (cf II.3.3.2).
On peut améliorer le test en intégrant un critère de continuité du vecteur tangent. Il est en effet possible de calculer, en tout point trouvé sur la courbe d'intersection, un vecteur tangent (cf IV.4.2).

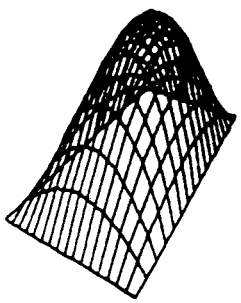
Note: Les extrémités de deux segments consécutifs sont d'autant plus proches l'une de l'autre que le test de linéarité des bords est exigeant.

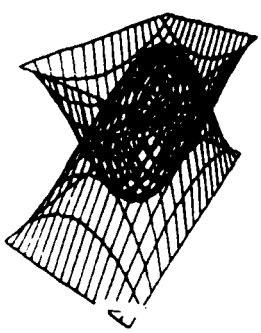
III.2.7 Conclusion

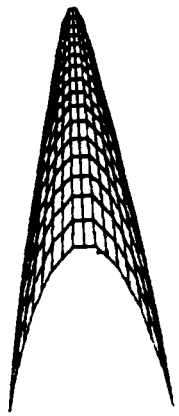
Cette méthode fut la première a être implantée dans les systèmes de C.A.O. La raison en est que les principes sur les quels elle s'appuie sont très simples et que sa mise en oeuvre est également très simple. Elle détecte également de façon infaillible **tous** les arcs de courbe fermés.
Cependant le fait qu'elle soit basée sur le calcul de l'intersection de plans sécants implique de mauvais résultats lorsque les deux carreaux sont tangents en un point de la courbe d'intersection. Les points trouvés sont alors imprécis et la reconstruction des courbes s'avère impossible. Le processus "subdivision récursive/approximation par facettes planes" est par ailleurs très coûteux en temps de calcul et en espace mémoire.
Nous proposons en IV.1 un algorithme de simplification des carreaux NURBS qui généralise cette première méthode.

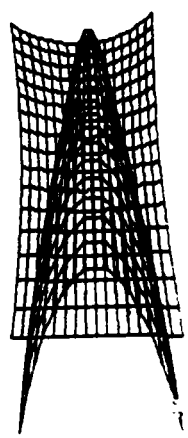
III.2.8 Exemples

Carreaux de Bézier biquadratiques sécants



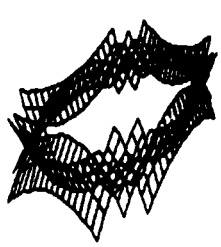


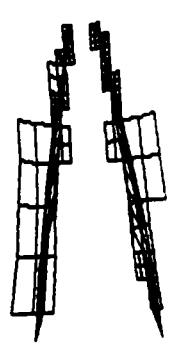




Courbes d'intersection

Listes des sous carreaux plats terminaux





III.3 INTERSECTION D'UNE SURFACE PARAMETRIQUE ET D'UNE SURFACE CONNUE PAR UNE EQUATION CARTESIENNE

Description générale

On recherche la courbe d'intersection entre deux surfaces.
La première surface est un carreau de Bézier rationnel noté $\mathcal{B}$.
La seconde surface $\mathcal{S}$ est connue par une équation cartésienne F(x,y,z) = 0. L'équation paramétrique de $\mathcal{B}$ et l'équation de $\mathcal{S}$ sont combinées pour obtenir une équation f(u,v) = 0 de la courbe d'intersection dans le domaine de $\mathcal{B}$. Cette équation permet une analyse complète puis une reconstruction de la courbe.
Cette méthode a été développée initialement par Farouki [FAR 87].

Hypothèses et notations

F est un polynôme de degré p défini par $F(x,y,z) = \sum_{i=0}^{p} \sum_{\alpha+\beta+\gamma=i} a_{\alpha,\beta,\gamma} \cdot x^{\alpha} \cdot y^{\beta} \cdot z^{\gamma}$ (1)

et le carreau $\mathcal{B}$ est défini par

$\sigma: [0, 1] \times [0, 1] \longrightarrow \mathbf{R}^3$

$$(u, v) \longmapsto \sigma(u, v) = \left(x(u,v) = \frac{X(u,v)}{W(u,v)}, y(u,v) = \frac{Y(u,v)}{W(u,v)}, z(u,v) = \frac{Z(u,v)}{W(u,v)} \right) \quad (2)$$

où X, Y, Z et W sont de degré m x n.

Remarque: Si $\mathcal{B}$ est initialement un carreau NURBS, il est d'abord décomposé en carreaux de Bézier afin de se ramener au cas présent.

III.3.1 Equation cartésienne de la courbe d'intersection

Un point $\sigma(u, v)$ de $\mathcal{B}$ appartenant à $\mathcal{S}$ vérifie F(x(u,v), y(u,v), z(u,v)) = 0.
Cette dernière équation est donc l'équation cartésienne de $\mathcal{C}$, la courbe d'intersection entre $\mathcal{B}$ et $\mathcal{S}$, donnée dans le rectangle paramétrique de $\mathcal{B}$.
Notons f(u,v) = F(x(u,v), y(u,v), z(u,v))
Compte tenu de (1) et (2), on a:

$$f(u,v) = \sum_{i=0}^{p} \sum_{\alpha+\beta+\gamma=i} a_{\alpha,\beta,\gamma} \cdot \left(\frac{X(u,v)}{W(u,v)}\right)^{\alpha} \cdot \left(\frac{Y(u,v)}{W(u,v)}\right)^{\beta} \cdot \left(\frac{Z(u,v)}{W(u,v)}\right)^{\gamma}$$

$$f(u,v) = \sum_{i=0}^{p} \sum_{\alpha+\beta+\gamma=i} a_{\alpha,\beta,\gamma} \cdot \frac{X^{\alpha}.\ Y^{\beta}.\ Z^{\gamma}}{W^{\alpha+\beta+\gamma}}$$

l'exposant de W est constant dans cette somme donc l'équation de $\mathcal{C}$ devient

$$\sum_{i=0}^{p} \frac{1}{W^{i}} \sum_{\alpha+\beta+\gamma=i} a_{\alpha,\beta,\gamma} \cdot X^{\alpha}.\ Y^{\beta}.\ Z^{\gamma} = 0$$

En multipliant chaque membre par W^p on obtient

$$\sum_{i=0}^{p} W^{p-i} \sum_{\alpha+\beta+\gamma=i} a_{\alpha,\beta,\gamma} \cdot X^{\alpha}.\ Y^{\beta}.\ Z^{\gamma} = 0$$

Cette équation peut se reconstituer sous la forme:

$$\sum_{\alpha+\beta+\gamma+\delta=p} b_{\alpha,\beta,\gamma,\delta} \cdot X^{\alpha}.\ Y^{\beta}.\ Z^{\gamma} \cdot W^{\delta} = 0$$

Le membre de gauche est un polynôme homogène de degré total p.

Ce développement montre que l'utilisation d'un carreau rationnel plutôt qu'un carreau polynomial n'accroît pas la complexité de l'équation de $\mathcal{C}$.

On en déduit également que cette dernière est une équation polynomiale de degré mp x np en (u,v).

L'exploitation de cette équation permet la détection, puis la reconstruction de tous les arcs de courbes connexes constituant la courbe $\mathcal{C}$. Les points de $\mathcal{C}$ où les deux surfaces $\mathcal{B}$ et $\mathcal{S}$ sont tangentes sont aussi déterminés avec précision.

III.3.2 Détermination du vecteur tangent en un point de la courbe

La détection de points caractéristiques de la courbe repose sur les propriétés du vecteur tangent à la courbe. Nous rappelons donc comment extraire un tel vecteur à partir de l'équation f(u,v)=0.

Soit $\mathcal{C}'$ un arc de courbe décrivant toute (ou une partie de) la courbe $\mathcal{C}$.

Appelons $\mathcal{A}$ la courbe plane projection de $\mathcal{C}'$ par σ^{-1} sur [0,1]x[0,1] et notons

α: $t \longmapsto \alpha(t) = (u(t), v(t))$ une paramétrisation de $\mathcal{A}$.

$\forall\ t$ on a donc $f(u(t)),v(t)) = 0$ (1)

On obtient en dérivant :

$\forall\ t\ \ f_u.u'(t) + f_v.v'(t) = 0$ (2)

Puis en dérivant une seconde fois:

$$\forall\ t\ \ f_{uu}.u'(t)^2 + 2\ f_{uv}.u'(t).v'(t) + f_{vv}.v'(t)^2 + f_u.u''(t)\ + f_v.v''(t) = 0 \quad (3)$$

L'équation (2) montre qu'en tout point de $\mathcal{A}$ le vecteur $\begin{bmatrix} f_u \\ f_v \end{bmatrix}$ (gradient de la courbe) est

orthogonal à la courbe; la déduction d'un vecteur tangent en est immédiate.

cas particulier points singuliers:

Si f_u et f_v s'annulent simultanément en un point, l'équation (2) n'a plus de sens; cela signifie qu'il est impossible de définir un vecteur normal en un tel point et par conséquent impossible d'en déduire un vecteur tangent. Par contre l'équation (3) se simplifie alors sous la forme:

$$f_{uu}.u'(t)^2 + 2\ f_{uv}.u'(t).v'(t) + f_{vv}.v'(t)^2\ = 0$$

on en déduit l'équation: $f_{uu}.\left(\frac{u'(t)}{v'(t)}\right)^2 + 2\ f_{uv}.\frac{u'(t)}{v'(t)} + f_{vv}\ = 0$ que nous pouvons exploiter.
Si cette équation admet deux solutions il y a deux directions de vecteurs tangents.
Géométriquement cela signifie qu'en un tel point deux arcs de courbes se croisent.

Une solution double correspond à un carrefour où les deux arcs sont tangents et l'absence de solutions correspond à un point d'intersection isolé.

On montre facilement que pour un point $\mathbf{P} = \sigma(u,v)$ appartenant à $\mathcal{C}$, la condition $f_u = f_v = 0$ et la condition ***P est un point singulier de 1er ordre*** sont équivalentes.

Il est aisé de montrer également qu'en un point de $\mathcal{C}$ où $f_u = f_v = f_{uu} = f_{uv} = f_{vv} = 0$ il y a en général trois directions de vecteurs tangents.

III.3.3 Détection des arcs de courbe connexes

Détection des arcs de courbe coupant un bord de $\mathcal{B}$

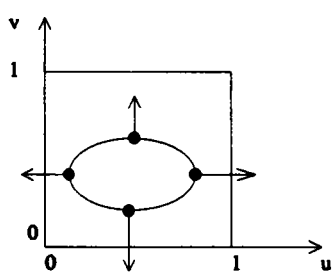
Un arc de courbe coupant un bord de $\mathcal{B}$ coupe nécessairement une des quatre isoparamétriques { u = 0 }, { u = 1 }, { v = 0 }, { v = 1 }. De tels arcs de courbe sont donc détectés en résolvant successivement les équations f(0,v) = 0, f(1,v) = 0, f(u,0) = 0, f(u, 1) = 0. Les solutions de ces équations fournissent des points sur les arcs de courbe recherchés.

Détection des arcs fermés

Un arc de courbe qui échappe à la recherche précédente est nécessairement fermé et ne touche aucun bord. Par conséquent le vecteur normal à la courbe passe nécessairement par les positions où il est parallèle aux axes du repère. On en déduit que chacune de ses coordonnées s'annule au moins deux fois au cours du parcours. Ces arcs de courbe sont donc détectés en résolvant les deux systèmes:

$$\begin{cases} f(u,v) = 0 \\ f_u(u,v) = 0 \end{cases} \quad (a) \qquad \begin{cases} f(u,v) = 0 \\ f_v(u,v) = 0 \end{cases} \quad (b) \qquad (4)$$

Ces systèmes fournissent sur les arcs recherchés les points où le vecteur normal est "horizontal" ou "vertical".



Détection des points singuliers de la courbe d'intersection

Les points singuliers sont les solutions communes aux deux systèmes précédents.

III.3.4 Construction des arcs de courbe

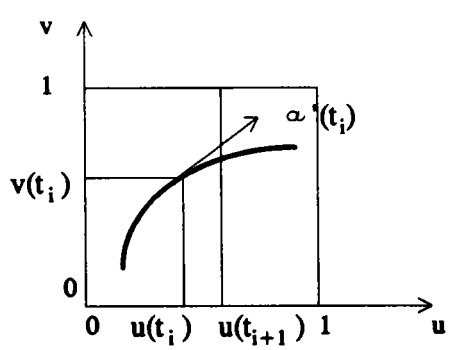
Tous les couples (u_i,v_i) trouvés dans les étapes précédentes, excepté les points d'intersection isolés, peuvent être choisis pour initialiser un processus de "suivi de courbe".

Un tel processus consiste à calculer des points sur la courbe $\mathcal{C}$ de proche en proche en suivant la courbe.
Plus précisément on obtient une suite de points ordonnés $\{ \alpha(t_i) = (u(t_i),v(t_i)) \}$ sur chaque arc $\mathcal{A}$ connexe.
La méthode s'appuie sur deux propriétés:
1) En tout point calculé sur la courbe, on connaît aussi un vecteur tangent (grâce aux équations (2) ou (3)); cela permet d'orienter la recherche d'un prochain point sur la courbe.
2) En fixant un des paramètres dans l'équation $f(u,v) = 0$, on obtient une équation à une inconnue qui est l'équation donnant les points d'intersection d'une courbe isoparamétrique de $\mathcal{B}$ avec la courbe $\mathcal{C}$.

Localisation du point suivant sur la courbe

Le point $\alpha(t_{i+1})$ est déterminé en fonction de $\alpha(t_i)$ comme suit:

Le vecteur tangent en $\alpha(t_i)$ permet de déterminer le champ d'isoparamétriques (à u fixé ou à v fixé) qui est le plus orthogonal à $\mathcal{A}$ au voisinage de $\alpha(t_i)$. Supposons par exemple que ce sont les isoparamétriques à u fixé qui sont choisies. On fixe alors $u(t_{i+1}) = u(t_i) + \text{pas}$; pas est une valeur arbitrairement petite qui peut dépendre de la courbure à $\mathcal{A}$ en $\alpha(t_i)$.
On recherche ensuite le point d'intersection de l'isoparamétrique $u_{fixé} = u(t_{i+1})$ et de la courbe $\mathcal{A}$, le plus proche du point $\alpha(t_i)$.



Il faut donc résoudre en v l'équation $f(u(t_{i+1}),v) = 0$ et choisir parmi les solutions v_j la plus proche de $v(t_i)$.
La méthode de Newton-Raphson, initialisée avec la valeur $v = v(t_i)$ remplit exactement cette tâche.

Dans la partie IV.4 la méthode de "suivi de courbe" est plus longuement développée dans un cadre plus général.

Ce processus est employé pour relier les points particuliers trouvés dans les étapes précédentes. Il achève la construction de la courbe $\mathcal{C}$.

III.3.5 Résolution des systèmes d'équations non linéaires à deux inconnues

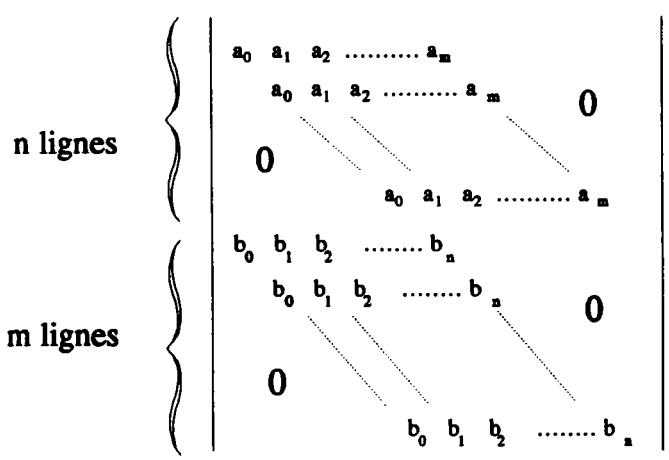
Les systèmes (4) évoqués dans cette discussion sont du type $\begin{cases} F(u,v) = 0 \\ G(u,v) = 0 \end{cases}$ (5)

où F et G sont des polynômes réels à deux variables.
Une application du théorème du résolvant en permet la résolution complète.

Rappel: Théorème du résolvant [KOR 61]

Soient
$f(x) = a_0 + a_1.x + a_2.x^2 + \ldots + a_m.x^m$ et $g(x) = b_0 + b_1.x + b_2.x^2 + \ldots + b_n.x^n$
deux polynômes réels. Les coefficients a_i et b_j sont réels et a_m et b_n sont non nuls.

On sait que f et g ont des zéros communs si et seulement si le déterminant



d'ordre n+m est nul.

Ce déterminant noté R(f,g) est appelé résolvant des deux polynômes f et g.

Application à la résolution du système proposé:

Supposons que $F(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{p} a_{ij} . u^i . v^j$

et que $G(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{q} b_{ij} . u^i . v^j$ où les a_{ij} et les b_{ij} sont des réels.

On considère que F est un polynôme en u dont les coefficients sont des polynômes en v:

$$F(u,v) = F_{(v)}(u) = \sum_{i=0}^{m} \left(\sum_{j=0}^{p} a_{ij} . v^j \right) . u^i = \sum_{i=0}^{m} a_i(v) . u^i$$

De même pour G:

$$G(u,v) = G_{(v)}(u) = \sum_{i=0}^{n} b_i(v) . u^i \qquad \text{avec } b_i(v) = \sum_{j=0}^{q} b_{ij} . v^j$$

Le résolvant des deux polynômes $F_{(v)}$ et $G_{(v)}$ est un polynôme en v; notons le r(v).
Du théorème nous pouvons déduire que pour toute solution v* de r(v) = 0 , les polynômes $F_{(v*)}$ et $G_{(v*)}$ ont un zéro commun en u.

La démarche consiste donc à ramener la résolution du système à deux inconnues à une équation à une inconnue.

Le processus complet de résolution du système (5) est décomposé en plusieurs étapes:

1) Les polynômes $\sum_{i=0}^{m} a_i(v) . u^i$ et $\sum_{i=0}^{n} b_i(v) . u^i$ sont déduits des polynômes F et G

2) Le résolvant r(v) est calculé. Cela consiste à évaluer un déterminant dont les coefficients sont des polynômes.

3) On recherche les solutions v_j de l'équation r(v) = 0 sur l'intervalle [0,1]

4) Pour chaque solution vj trouvée, on recherche le (ou les) zéro commun à $F(u,v_j)$ et à $G(u,v_j)$ sur l'intervalle [0,1]. Il n'est pas intéressant de résoudre simultanément ces deux équations et de comparer les solutions obtenues car les imprécisions sur les calculs gênent les comparaisons. Il est préférable de résoudre $F(u,v_j) = 0$ puis de choisir parmi les solutions trouvées celle qui minimise $|G(u,v_j)|$.

Intérêts et limites de la méthode du "résolvant"

Il est intéressant de comparer cette méthode de résolution de systèmes non linéaires aux techniques classiques, les méthodes de type Newton-Raphson.

La méthode décrite garantit que toutes les solutions du système (5) sont trouvées, contrairement aux méthodes itératives qui nécessitent toutes une solution approchée et convergent vers la solution la plus proche de la solution approchée.
Par contre il est possible de trouver "trop" de solutions. En effet les solutions communes v_k de $a_m(v) = 0$ et de $b_n(v) = 0$ sont des solutions de $r(v) = 0$. Pour ces valeurs particulières v_k la nullité du résolvant ne signifie pas que les polynômes $u \longmapsto F(u,v_k)$ et $u \longmapsto G(u,v_k)$ ont un zéro commun.
Il est donc nécessaire de rechercher ces valeurs v_k puis de diviser r(v) par $(v-v_k)$ pour toute valeur trouvée avant de résoudre $r(v) = 0$. Cela offre aussi l'avantage de simplifier le résolvant.
Les polynômes $u \longmapsto F(u,v_k)$ et $u \longmapsto G(u,v_k)$ doivent être traités comme un cas particulier.
Le degré de l'équation $r(v) = 0$ est connu et vaut $mq+np$; on peut donc majorer le nombre de solutions v_j et par conséquent le nombre de couples (u_j, v_j) solutions de (5).
On peut donc aisément déduire le nombre maximal de solutions des systèmes (4).
Or, à chaque arc fermé correspondent au moins deux solutions pour (a) et au moins deux solutions pour (b); il est donc possible de prévoir le nombre maximal d'arcs connexes composant $\mathcal{C}$.

Cependant l'expression du degré de r(v) montre qu'il est nettement plus élevé que les degrés des polynômes initiaux F et G. Une élévation du degré de F ou G entraîne une augmentation catastrophique du degré de r(v) et il devient numériquement impossible de calculer le résolvant ou encore de résoudre l'équation obtenue.
L'évaluation du résolvant, comme tous les calculs de déterminants, est très coûteuse en temps de calcul.

La mise en oeuvre de cette méthode est délicate car elle exige la programmation de toutes les opérations algébriques sur les polynômes impliquées par le calcul du résolvant.
Par ailleurs l'obtention de l'équation f(u,v) = 0 à partir des équations de $\mathcal{B}$ et $\mathcal{S}$ nécessite, elle, d'effectuer des calculs formels sur des polynômes à deux variables.

Lorsque les systèmes (4) sont de degré trop élevé, il peut être envisagé de les résoudre avec une méthode de Newton-Raphson. Cependant pour trouver toutes les solutions, il faut "lancer" le processus de résolution successivement avec plusieurs solutions approchées. Ces solutions approchées doivent être régulièrement réparties sur [0,1]x[0,1].

III.3.6 Conclusion

Le problème posé est du point de vue mathématique parfaitement résolu: tous les arcs connexes sont détectés et reconstruits, les points singuliers sont localisés et traités en tant que tels.
Cependant la mise en oeuvre n'est possible qu'avec des surfaces de faible degré.
Par exemple si $\mathcal{B}$ est un carreau bicubique et $\mathcal{S}$ une quadrique, l'équation f(u,v) est de degré 6x6 et $f_u(u,v)$ est de degré 5x6. Le résolvant obtenu est de degré 66; d'importants problèmes de stabilité numérique sont inhérents à la résolution de l'équation.
Par ailleurs cela implique que la courbe $\mathcal{C}$ peut présenter jusqu'à 33 arcs fermés disjoints; les degrés élevés traduisent simplement la complexité des courbes d'intersections entre surfaces gauches.
Comme on ne recherche que les solutions sur [0,1], le polynôme r(v) peut être simplifié et on peut alors diminuer son degré de façon non négligeable (cf IV.3.3). Il est possible aussi de chercher à diminuer le degré des polynômes avant le calcul du résolvant, cela offre l'avantage de simplifier le calcul du déterminant.

III.4 INTERSECTION DE DEUX CARREAUX PARAMETRIQUES

Hypothèses

Les deux surfaces $\mathcal{S}_1$ et $\mathcal{S}_2$ sont connues par les équations paramétriques:

$$\mathbf{X}: [a_1, b_1] \times [c_1,d_1] \subset \mathbf{R}^2 \longrightarrow \mathbf{R}^3$$
$$(u, v) \longmapsto \mathbf{X}(u, v)$$

et

$$\mathbf{Y}: [a_2, b_2] \times [c_2,d_2] \subset \mathbf{R}^2 \longrightarrow \mathbf{R}^3$$
$$(r, s) \longmapsto \mathbf{Y}(r, s)$$

où X et Y sont de la forme Bézier rationnel ou NURBS.
Notons $\mathcal{C}$ la courbe d'intersection entre $\mathcal{S}_1$ et $\mathcal{S}_2$.

Pour tout point $\mathbf{P} \in \mathcal{S}_1 \cap \mathcal{S}_2$ il existe un unique couple $(u,v) \in [a_1, b_1] \times [c_1,d_1]$ et un unique couple $(r,s) \in [a_2, b_2] \times [c_2,d_2]$ tel que $\mathbf{P} = \mathbf{X}(u,v) = \mathbf{Y}(r,s)$

Rechercher la courbe $\mathcal{C}$ équivaut donc à rechercher les quadruplets (u,v,r,s) vérifiant le système:

$$\mathbf{X}(u,v) = \mathbf{Y}(r,s) \qquad (1)$$

Remarquons que si $\mathcal{S}_1$ et $\mathcal{S}_2$ sont rationnels il n'est pas possible d'envisager de résoudre (1) dans l'espace homogène; en effet

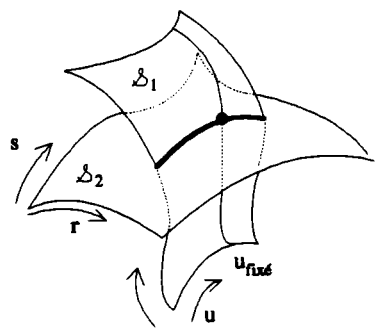
$$\mathbf{X}(u,v) = \frac{U_1}{d_1} = H\left(\begin{bmatrix} U_1 \\ d_1 \end{bmatrix}\right) \text{ et } \mathbf{Y}(r,s) = \frac{U_2}{d_2} = H\left(\begin{bmatrix} U_2 \\ d_2 \end{bmatrix}\right) \text{ où } U_i \in \mathbf{R}^3 \text{ et } d_i > 0$$

$$\text{L'égalité } \begin{bmatrix} U_1 \\ d_1 \end{bmatrix} = \begin{bmatrix} U_2 \\ d_2 \end{bmatrix} \text{ entraîne } \frac{U_1}{d_1} = \frac{U_2}{d_2}$$

mais la réciproque n'est pas vraie.

Pour résoudre ce système sous-déterminé il faut fixer une inconnue ou trouver une quatrième équation.

En fixant une variable, on ramène le problème au cas de l'intersection d'une isoparamétrique prise sur l'un des carreaux avec l'autre carreau.



C'est un cas particulier du problème intersection courbe paramétrique-carreau paramétrique.

Pour résoudre ce système non linéaire 3x3, il y a deux approches essentielles: algébrique ou analytique.

L'approche algébrique:

Le système est réduit à une équation à une inconnue par des calculs de résolvants et des manipulations d'équations. A cause de l'explosion du degré des résolvants on ne peut choisir cette démarche que pour des surfaces de faible degré.
La courbe $\mathcal{C}$ est obtenue en cherchant les points d'intersection d'une isoparamétrique de balayage prise sur $\mathcal{S}_1$ avec le carreau $\mathcal{S}_2$.

L'approche analytique:

Le système est résolu par une méthode itérative comme Newton-Raphson. La convergence de cette dernière exige une solution approchée. La méthode est donc intégrée dans un algorithme de " suivi de courbe" qui calcule des points de proche en proche sur $\mathcal{C}$. Un point calculé sur la courbe sert de solution approchée pour le calcul du prochain point. La difficulté majeure est l'initialisation du processus, il faut trouver un point-germe.
La solution la plus souvent retenue pour localiser un tel point est une recherche dichotomique récursive (cf III.1) associée à une facettisation des surfaces.

IV LES SOLUTIONS PROPOSEES

Les trois approches précédentes sont intéressantes et ne s'appliquent pas exactement aux mêmes conditions du problème. Nous étudions des solutions qui tentent de combiner les différents avantages.

La méthode de Farouki est la plus intéressante car elle offre une solution mathématique exacte mais elle nécessite des degré faibles et des équations cartésiennes. De façon plus générale toute méthode impliquant des calculs de résolvants exige des faibles degrés.

C'est pourquoi nous décrivons d'abord un algorithme de simplification des carreaux NURBS qui associe diminution de degré et subdivision récursive.

Nous montrons ensuite la difficulté de déterminer une équation cartésienne exacte ou encore une équation cartésienne approchée de faible degré pour un carreau de Bézier.

Il est possible toutefois d'obtenir l'équation cartésienne d'une courbe de Bézier rationnelle de degré 2 ou 3. Nous en déduisons une résolution algébrique du problème: intersection courbe rationnelle de degré 2-carreau rationnel de degré 2x2 et appliquons ce résultat au traitement de l'intersection de deux carreaux.

Puis nous décrivons la mise en oeuvre d'un algorithme de suivi de courbe et proposons plusieurs optimisations.

Enfin nous étudions une solution générale au problème de l'intersection de deux carreaux NURBS; l'algorithme proposé réalise la synthèse des parties précédentes.

IV.1 SIMPLIFICATION DU PROBLEME D'INTERSECTION

Nous proposons un algorithme qui décompose les deux carreaux NURBS impliqués en une collection de surfaces assez simples pour leur appliquer une méthode de résolution algébrique.
La méthode "Dichotomie et intersection de plans" exploite la propriété fondamentale des carreaux de Bézier: la subdivision récursive d'un carreau de Bézier crée des sous-carreaux successifs de plus en plus réguliers. L'ultime étape de cette évolution est le plan. Cette

régularité croissante suggère que ces sous-carreaux successifs peuvent être décrits par des paramétrisations de degré plus faible.
Nous montrons donc qu'il est possible d'associer subdivision récursive et diminution de degré. Nous en déduisons un algorithme de décomposition arborescente d'un carreau de Bézier où le degré des sous-carreaux successifs est progressivement diminué.
Nous utilisons cet algorithme pour décomposer un carreau de Bézier de degré quelconque en un arbre dont les feuilles sont des carreaux de Bézier de degré 2x2.

Par souci de clarté nous expliquons d'abord la méthode dans le cas particulier des courbes de Bézier polynomiales.

IV.1.1 Décomposition récursive associée à une diminution de degré pour une courbe de Bézier polynomiale dans R^3.

IV.1.1.1 Evolution d'une portion de courbe en fonction du nombre de subdivisions

Soit une courbe de Bézier ℓ^0 de degré m définie par

$$\alpha^0 : [0,1] \longrightarrow \mathbf{R}^3$$

$$t \longmapsto \alpha^0(t) = \sum_{i=0}^{m} \mathbf{P}_i^0 \, B_i^m(t)$$

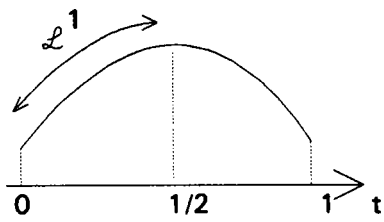
Les $\mathbf{P}_i^0$ forment le réseau de points de contrôle de la courbe, ce sont des points de $\mathbf{R}^3$.
Dans la base canonique des polynômes la paramétrisation est

$$\alpha^0(t) = \sum_{i=0}^{m} \mathbf{A}_i^0 \, t^i .$$

Rappelons que $\mathbf{P}_i^0 = \sum_{k=0}^{i} \frac{C_i^k}{C_m^k} \mathbf{A}_k^0$ Relation 1

Notons ℓ^1 la courbe de Bézier obtenue en prenant la subdivision gauche de la partition de ℓ^0 après un découpage en $t^* = 1/2$.

Notons $\alpha^1(t) = \sum_{i=0}^{m} \mathbf{A}_i^1 \, t^i$ sa paramétrisation sous la forme monomiale.



$\mathcal{L}^1$ est une subdivision gauche de $\mathcal{L}^0$

Pour tout t de [0,1] on a:

$$\alpha^1(t) = \alpha^0\left(\frac{t}{2}\right) = \sum_{i=0}^{m} A_i^0 \left(\frac{t}{2}\right)^i = \sum_{i=0}^{m} A_i^0 \frac{1}{2^i} . t^i$$

Donc $A_i^1 = \frac{1}{2^i} . A_i^0$ pour tout i.

Faisons p subdivisions successives de $\mathcal{L}^0$ en découpant à chaque fois en $t = \frac{1}{2}$ et en prenant à chaque fois la sous courbe gauche. C'est à dire que nous prenons la portion de courbe la plus à gauche du $p^{ième}$ étage de l'arbre binaire issu de la décomposition récursive de $\mathcal{L}^0$. Notons $\mathcal{L}^p$ la sous courbe obtenue et

$\alpha^p(t) = \sum_{i=0}^{m} A_i^p t^i$ sa paramétrisation

Elle s'écrit $\alpha^p(t) = \sum_{i=0}^{m} P_i^p B_i^m(t)$ dans la base de Bernstein.

On montre facilement que $A_i^p = \frac{1}{2^{pi}} . A_i^0$ pour tout i = 0...m.

Donc si p croît

$A_i^p \longrightarrow 0$ pour $i > 0$ et $A_0^p = A_0^0$

et grâce à la relation 1 on montre aussi que $P_i^p \longrightarrow P_0^0$ pour tout i=0...m

Donc la courbe $\mathcal{L}^p$ tend vers une courbe de degré de plus en plus faible et finalement vers le point A_0^0.

Donc pour p assez grand on peut approcher $\alpha^p(t)$ par $\underline{\alpha}^p(t) = \sum_{i=0}^{m-1} A_i^p t^i$

IV.1.1.2 Test d'erreur

L'erreur commise est

$$\varepsilon^p(t) = \| \alpha^p(t) - \underline{\alpha}^p(t) \| = t^m . \| A_m^p \|$$

donc $\mathcal{E}^p(t) \leq \mathcal{E}^p(1) = \| A_m{}^p \| = \frac{1}{2^p} \| A_m{}^0 \|$

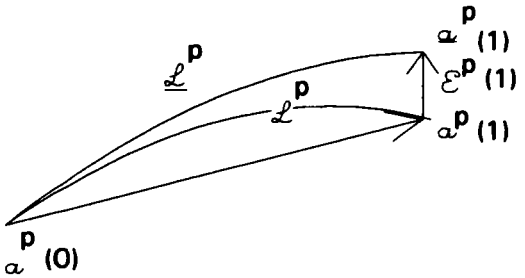
donc $\mathcal{E}^p(t) \longrightarrow 0$ si p croît suffisamment. On choisit de remplacer $\alpha^p(t)$ par $\underline{\alpha}^p(t)$ dès que l'erreur relative maximale

$$\mathcal{E}_{rel}{}^p = \frac{\| A_m{}^p \|}{\| \alpha^p(1) - \alpha^p(0) \|}$$ est inférieure à une précision choisie (de l'ordre de 10^{-3})

Interprétation géométrique:

$$\mathcal{E}_{rel}{}^p > \frac{\text{distance maximale entre } \mathcal{L}^p \text{ et } \underline{\mathcal{L}}^p}{\text{Longueur}(\mathcal{L}^p)}$$

Où $\underline{\mathcal{L}}^p$ est la courbe de Bézier dont la paramétrisation est $\underline{\alpha}^p$



Estimation de l'erreur relative entre $\mathcal{L}^p$ et $\underline{\mathcal{L}}^p$

Cette étude est valable pour toute sous courbe du $p^{ième}$ étage de l'arbre car le découpage est symétrique entre courbe gauche et courbe droite.

IV.1.1.3 Optimisation de la valeur paramétrique de découpage

La discussion précédente a été effectuée en prenant à chaque découpage la valeur $t^* = 1/2$ pour valeur paramétrique de découpage, mais on peut choisir une valeur qui diminue le nombre d'étages nécessaires pour diminuer le degré. Nous définissons d'abord cette valeur dans le cas des courbes planes non paramétriques puis tentons de généraliser aux courbes de l'espace.

Optimisation de la valeur paramétrique de découpage pour une courbe de Bézier polynomiale non paramétrique

Soit une courbe de Bézier plane non paramétrique $\mathcal{P}$.

Notons $\alpha : [0,1] \longrightarrow \mathbf{R}$

$$t \longmapsto \alpha(t) = \sum_{i=0}^{m} p_i \, B_i^m(t)$$ sa paramétrisation dans la base de Bernstein,

et $\alpha(t) = \sum_{i=0}^{m} a_i \, t^i$ sa traduction dans la base canonique.

Les a_i (p_i) sont des réels.

La valeur qui minimise le nombre d'étages de l'arbre binaire de décomposition de $\mathcal{P}$ nécessaires pour approcher α par une courbe de degré m-1 est la solution de l'équation $\frac{d^{m-1}\alpha}{dt^{m-1}}(t) = 0$

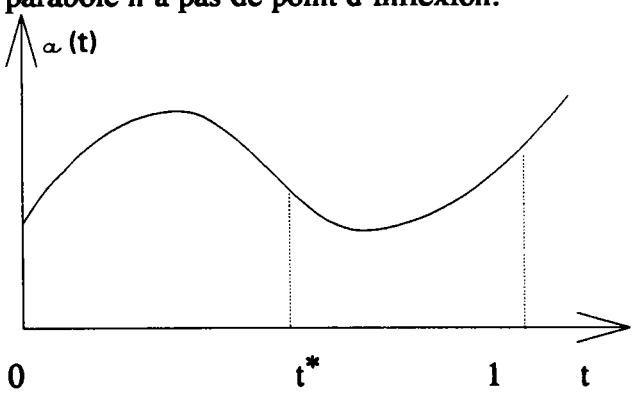
or $\frac{d^{m-1}\alpha}{dt^{m-1}}(t) = m!\, a_m\, t + (m-1)!\, a_{m-1}$ donc la valeur est $t^* = -\frac{a_{m-1}}{m.\, a_m}$

La courbe $\mathcal{P}$ est découpée en t^* si t^* appartient à $]0,1[$ sinon elle est découpée en $t = 1/2$

Interprétation géométrique

Exemple pour $m = 3$:

$t \longmapsto \alpha(t)$ est un arc de courbe de degré 3, on désire approcher cette cubique par une réunion de deux arcs de parabole. On découpe donc au point d'inflexion, En effet une parabole n'a pas de point d'inflexion.



On découpe une cubique au point d'inflexion

Optimisation de la valeur paramétrique de découpage d'une courbe de Bézier dans l'espace

Le résultat précédent est difficile à généraliser car l'équation $\frac{d^{m-1}\alpha}{dt^{m-1}}(t) = 0$ est maintenant une équation vectorielle. La solution t^* doit vérifier:

$m!\ A_m\ t + (m-1)!\ A_{m-1} = 0$ Cette équation n'a en général pas de solution.
On résout donc l'équation pour chaque coordonnée et on prend la la valeur la plus proche de 1/2; si elle n'appartient pas à [0,1] on choisit 1/2.

Note

On peut aussi chercher t^* qui minimise $\left\| \frac{d^{m-1}\alpha}{dt^{m-1}}(t) \right\|^2$

IV.1.1.4 Mise en oeuvre de la décomposition récursive associée à une diminution du degré lorsque la courbe est écrite dans la base de Bernstein

Les subdivisions successives d'une courbe de Bézier sont effectuées grâce à l'algorithme de De Casteljau qui opère avec des courbes données dans la base de Bernstein. Dans un processus de subdivision récursive la courbe initiale ainsi que toutes les portions de courbe issues de découpage sont connues dans la base de Bernstein. Il faut donc reformuler les parties précédentes, principalement la diminution du degré.

Rappelons d'abord l'algorithme d'élévation du degré, qui consiste à effectuer une augmentation fictive du degré d'une courbe de Bézier [BOE 84].

1) Algorithme d'élévation du degré

Soit $\mathcal{L}$ une courbe de Bézier dans l'espace définie par

$t \longmapsto \alpha(t) = \sum_{i=0}^{m} P_i\ B_i^m(t)$ dans la base de Bernstein et par $\alpha(t) = \sum_{i=0}^{m} A_i\ t^i$ dans la base canonique.

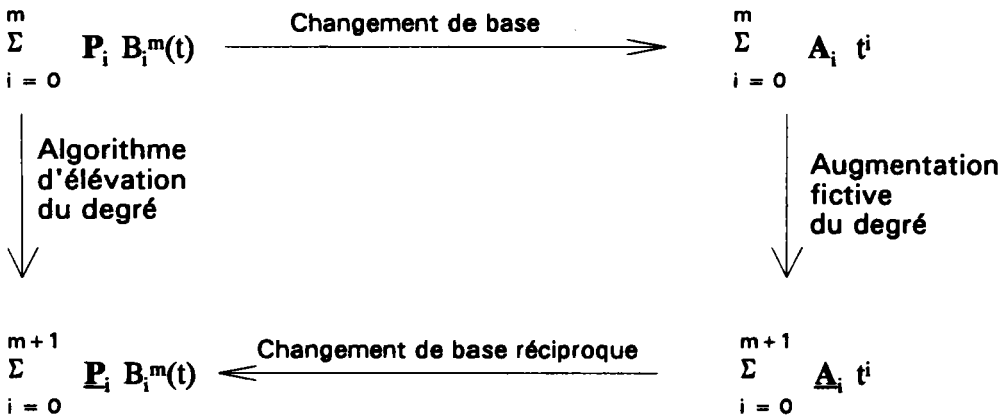
Soit $\underline{\mathcal{L}}$ la courbe de Bézier de degré $m+1$ (fictif) définie par

$t \longmapsto \underline{\alpha}(t) = \sum_{i=0}^{m+1} \underline{P}_i\ B_i^m(t)$ et par $\underline{\alpha}(t) = \sum_{i=0}^{m+1} \underline{A}_i\ t^i$

Les $\underline{A}_i$ sont définis comme suit :

$\underline{A}_i = A_i$ pour $i = 0 \ldots m$ et $\underline{A}_{m+1} = 0$

Les deux courbes $\mathcal{L}$ et $\underline{\mathcal{L}}$ sont évidemment confondues.
L'algorithme d'élévation du degré calcule directement les $\underline{P}_i$ en fonction des P_i sans passer par la forme canonique:

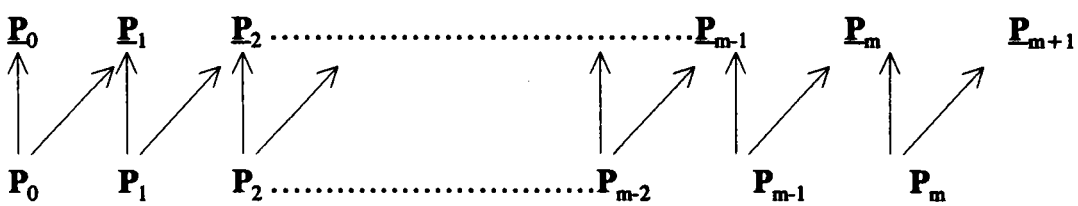


Les $\underline{P}_i$ sont calculés avec les relations (2):

$$\underline{P}_0 = P_0$$

$$\underline{P}_i = \alpha_i P_{i-1} + (1 - \alpha_i) P_i \quad \text{pour } i = 1 \ldots m \text{ avec } \alpha_i = \frac{i}{m+1}$$

$$\underline{P}_{m+1} = P_m$$



Relations entre les P_i et les $\underline{P}_i$

2) Algorithme de diminution du degré

Pour diminuer le degré d'une courbe de Bézier quelconque nous utilisons cet algorithme à l'envers et calculons les P_i en fonction des $\underline{P}_i$.

Conservons les mêmes notations:

Soit $\mathcal{L}$ la courbe de Bézier de degré **réel** m+1 dont on cherche à diminuer le degré.

Elle est définie par $t \longmapsto \underline{\alpha}(t) = \sum_{i=0}^{m+1} \underline{P}_i B_i^m(t)$ ou bien par $\underline{\alpha}(t) = \sum_{i=0}^{m+1} \underline{A}_i t^i$

Soit $\mathcal{L}$ la courbe de Bézier que l'on obtient après diminution du degré de $\mathcal{L}$. Supposons la définie par:

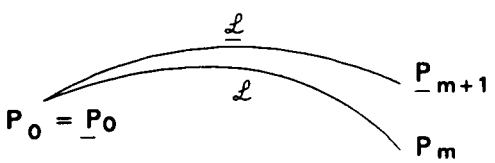
$t \longmapsto \alpha(t) = \sum_{i=0}^{m} P_i B_i^m(t)$ dans la base de Bernstein et par

$\alpha(t) = \sum_{i=0}^{m} A_i\ t^i$ dans la base canonique.

Le problème est de calculer les m+1 coefficients P_i avec les m+2 relations (2) prises à l'envers et généralement incompatibles.

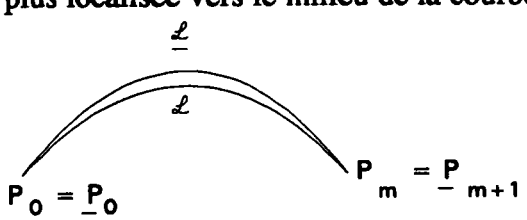
Plusieurs choix sont possibles:

1) On initialise l'algorithme avec $P_0 = \underline{P}_0$
et on calcule de proche en proche $P_1, P_2, \ldots\ldots\ldots\ldots P_m$ avec les relations (2). La condition $P_m = \underline{P}_{m+1}$ ne sera pas réalisée:
Les courbes α et $\underline{\alpha}$ divergent en t =1.



2) On initialise l'algorithme avec $P_m = \underline{P}_{m+1}$
et on calcule de proche en proche $P_{m-1}, P_{m-2}, \ldots\ldots\ldots\ldots P_0$ avec les relations (2).
Mais alors la condition $P_0 = \underline{P}_0$ ne sera pas réalisée et
Les courbes α et $\underline{\alpha}$ divergent en t =0.

3) On utilise simultanément les deux algorithmes précédents et on les interrompt après le calcul de la valeur P_{i*} médiane de la suite $\{P_i\}$.
On calcule $P_0, P_1, \ldots\ldots\ldots\ldots P_{i*}$ comme en (1)
Et on calcule $P_m, P_{m-1}, \ldots\ldots\ldots\ldots P_{i*+1}$ comme en (2). Les deux courbes coïncident aux extrémités ainsi que leurs dérivées jusqu'à l'ordre $\simeq$ i*
Contrairement aux algorithmes précédents la déformation est nulle aux extrémités; elle est plus localisée vers le milieu de la courbe ℓ.



Nous retenons cette dernière méthode car lors d'une décomposition récursive associée à une diminution du degré, elle garantit la continuité jusqu'à un ordre élevé entre plusieurs portions de courbe d'un même étage de l'arbre après diminution du degré.

Synthèse

Nous associons à une subdivision récursive une diminution du degré pour les sous courbes. On obtient ainsi un arbre binaire dont les feuilles sont de degré aussi faible que désiré; l'erreur due aux approximations est strictement contrôlée.

IV.1.2 Généralisation aux carreaux de Bézier polynomiaux

IV.1.2.1 Etude du lien entre subdivision récursive et diminution de degré pour un carreau de Bézier polynomial défini dans la base canonique

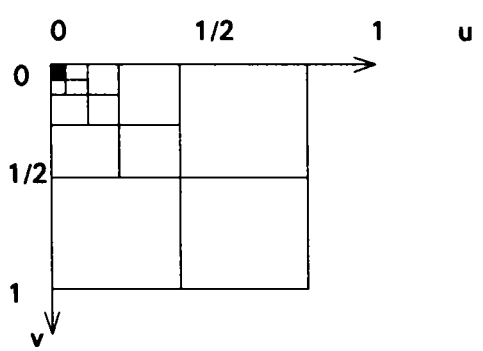
Appelons $\mathcal{B}$ le carreau de Bézier m x n défini par:

$$\sigma: [0, 1[\times [0, 1[\longrightarrow \mathbf{R}^3$$

$$(u, v) \longmapsto \sigma(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{P}_{ij}\ B_i^m(u).\ B_j^n(v)$$

et par $\sigma(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{A}_{ij}\ u^i.\ v^j$ dans la base canonique.

Faisons p subdivisions successives du carreau $\mathcal{B}$; les sous carreaux successifs sont à chaque fois découpés en $u^* = 1/2$ et $v^* = 1/2$. Appelons $\mathcal{B}^p$ le sous carreau obtenu en conservant à chacun des p découpages la restriction à $[0, 1/2[\times [0, 1/2[$ du carreau en amont:



Notons $\sigma^p: [0, 1[\times [0, 1[\longrightarrow \mathbf{R}^3$

$$(u, v) \longmapsto \sigma(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{P}_{ij}{}^p\ B_i^m(u).\ B_j^n(v)$$

sa paramétrisation dans la base de Bernstein et

$\sigma^p(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} A_{ij}{}^p \, u^i . v^j$ celle dans la base canonique.

Pour tout couple (u,v) de [0,1] x [0,1] on a $\sigma^p(u, v) = \sigma\left(\frac{u}{2^p}, \frac{v}{2^p}\right)$

Donc $\sigma^p(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \frac{1}{2^{p.(i+j)}} A_{ij} \, u^i . v^j$

Par conséquent pour tout i=0...m et tout j=0...n on a l'identité: $A_{ij}{}^p = \frac{1}{2^{p.(i+j)}} A_{ij}$

On en déduit que si p croît, $A_{ij}{}^p \longrightarrow 0$ sauf pour i=0 et j=0.
Cette étude montre qu'on peut négliger après plusieurs subdivisions les termes $A_{ij}{}^p$ tels que i+j soit supérieur à un certain degré: on peut remplacer par des zéros le triangle inférieur droit dans la matrice des coefficients:

$$\begin{pmatrix} A_{00}{}^p & \cdots & A_{0n}{}^p \\ \cdots & \cdots & \cdots \\ A_{m0}{}^p & \cdots & A_{mn}{}^p \end{pmatrix}$$

Mais cela est délicat à réaliser lorsque les coefficients sont donnés dans la base de Bernstein, nous préférons donc "éliminer " la dernière colonne ou la dernière ligne dans cette matrice, ce qui est équivalent à diminuer le degré en u ou en v.

Evolution des coefficients de σ^p dans la base de Bernstein

L'étude précédente nous permet de déduire le comportement des $P_{ij}{}^p$ lorsque p croît.

Rappelons que $P_{ij}{}^p = \sum_{k=0}^{i} \sum_{l=0}^{j} \frac{C_i^k}{C_m^k} \frac{C_j^l}{C_n^l} A_{kl}{}^p$ **Relation 2**

Cette expression montre que $P_{ij}{}^p \longrightarrow P_{00}{}^p = A_{00}{}^p$ lorsque p croît suffisamment. Ce qui signifie que tous les points de contrôle du sous carreau $\mathcal{B}^p$ tendent à se confondre. A l'extrême le sous carreau converge vers un point.

Cette discussion est valable pour tout sous carreau du $p^{ième}$ étage de l'arbre 4-aire issu de la décomposition de $\mathcal{B}$ car le découpage est symétrique.

IV.1.2.2 Test d'erreur

Appelons $\mathcal{B}$ le carreau de Bézier m x n défini par:

σ: [0, 1[x [0, 1[$\longrightarrow \mathbf{R}^3$

$$(u, v) \longmapsto \sigma(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} A_{ij}\, u^i . v^j \quad \text{dans la base canonique.}$$

Etudions l'erreur commise lorsqu'on diminue le degré de σ suivant la direction u.

σ est approché par le polynôme ϱ : $(u, v) \longmapsto \varrho(u,v) = \sum_{i=0}^{m-1} \sum_{j=0}^{n} \underline{A}_{ij}\, u^i . v^j$

L'erreur commise en (u,v) est:

$$\mathcal{E}(u,v) = \| \sigma(u, v) - \varrho(u,v) \| = \| \sum_{j=0}^{n} A_{mj}\, u^m . v^j \| = u^m . \| \sum_{j=0}^{n} A_{mj} v^j \|$$

L'erreur est nulle pour u = 0 et maximale pour u = 1.

On étudie donc $e(v) = \mathcal{E}(1,v) = \| \sum_{j=0}^{n} A_{mj}\, v^j \|$

La généralisation de l'erreur maximale relative pour les courbes devient:

$e_{rel}(v) = \dfrac{e(v)}{\| \sigma(1, v) - \sigma(0,v) \|}$ mais le dénominateur de cette expression étant difficile à minorer, on est contraint à utiliser l'erreur absolue e(v).

Majorons e(v):

$e(v) \leq n . \max_j \| A_{mj} \|$ Cette dernière quantité est facilement calculée.

On évalue donc cette dernière quantité, si elle est inférieure à un certain $\mathcal{E}$ on remplace $\sigma(u, v)$ par $\varrho(u, v)$.

La définition de σ est symétrique en u et v, on en déduit donc le test:

On diminue le degré en v si: $m . \max_i \| A_{in} \| \leq \mathcal{E}$

Comme toutes les subdivisions récursives sont effectuées dans la base de Bernstein, les diminutions de degré ainsi que les tests d'erreur sont effectués dans la base de Bernstein. On calcule donc les A_{mj} à partir des P_{ij} à l'aide de l'algorithme de changement de base.

IV.1.2.3 Algorithme de diminution du degré exprimé par rapport à la base de Bernstein

Nous décrivons comment réaliser l'approximation d'un carreau de Bézier par un carreau de degré inférieur lorsque ses coefficients sont connus dans la base de Bernstein.

Soit donc $\mathcal{B}$ le carreau de Bézier m x n défini par:

σ: [0, 1[x [0, 1[$\longrightarrow \mathbf{R}^3$

$$(u, v) \longmapsto \sigma(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{P}_{ij}\ B_i^m(u).\ B_j^n(v) \quad \text{dans la base de Bernstein.}$$

Nous décrivons comment diminuer le degré suivant la direction u, on en déduira, par symétrie, la méthode pour diminuer le degré suivant la direction v.

Nous l'approchons donc par le carreau $\mathcal{Q}$ de degré (m-1) x n défini par

le polynôme g : $(u, v) \longmapsto g(u,v) = \sum_{i=0}^{m-1} \sum_{j=0}^{n} \underline{\mathbf{P}}_{ij}\ B_i^{m-1}(u).\ B_j^n(v)$

Le problème est le calcul des $\underline{\mathbf{P}}_{ij}$ en fonction des $\mathbf{P}_{ij}$:

On sait que $\sigma(u, v) = \sum_{j=0}^{n} B_j^n(v) \left(\sum_{i=0}^{m} \mathbf{P}_{ij}\ B_i^m(u) \right)$ de même

$$g(u, v) = \sum_{j=0}^{n} B_j^n(v) \left(\sum_{i=0}^{m-1} \underline{\mathbf{P}}_{ij}\ B_i^m(u) \right)$$

Donc en fait pour tout j=0...n il suffit de diminuer le degré de $u \longmapsto \sum_{i=0}^{m} \mathbf{P}_{ij}\ B_i^m(u)$

pour obtenir la courbe $u \longmapsto \sum_{i=0}^{m-1} \underline{\mathbf{P}}_{ij}\ B_i^m(u)$

L'algorithme de diminution des courbes est appliqué à la suite $[\mathbf{P}_{0j}\ \mathbf{P}_{1j}\ \ldots\ldots\mathbf{P}_{mj}]$ pour obtenir la suite $[\underline{\mathbf{P}}_{0j}\ \underline{\mathbf{P}}_{1j}\ \ldots\ldots\underline{\mathbf{P}}_{m-1j}]$ pour toute valeur de j=0...n.

IV.1.2.4 Optimisation du point de découpage

Il est difficile de généraliser la notion de point de découpage optimisé déterminé dans le cas des courbes. La valeur t^* qui convenait était celle qui annulait la dérivée d'ordre n-1 lorsque la courbe était de degré n. Pour un carreau de degré mxn, afin de diminuer plus

rapidement le degré en u, il faudrait prendre la valeur u^* qui annule le vecteur dérivée partielle $\frac{\partial^{m-1}\sigma}{\partial u^{m-1}}(u,v)$ pour tout v; ce qui est impossible.

La valeur u^* choisie est donc la plus proche de 1/2 parmi celles qui annulent une des composantes de $\frac{\partial^{m-1}\sigma}{\partial u^{m-1}}(u,0)$ et de $\frac{\partial^{m-1}\sigma}{\partial u^{m-1}}(u,1)$. Le calcul de la valeur d'optimisation est donc basé sur le comportement des courbes frontières $u \longmapsto \sigma(u, 0)$ et $u \longmapsto \sigma(u, 1)$ qui sont faciles à obtenir lorsque le carreau est écrit dans la base de Bernstein.
En effet un traitement d'optimisation ne doit pas entraîner un volume de calculs supérieur à celui qu'il est supposé éviter. Ce choix est arbitraire mais il est aussi valable qu'un autre car il n'existe pas de choix dont on puisse dire qu'il soit le meilleur.

IV.1.2.5 Synthèse

Un carreau de Bézier initial de degré quelconque est décomposé en arbre 4-aire de carreaux de Bézier. Dès qu'un sous carreau vérifie une des conditions définies au paragraphe (2), on le remplace par un carreau approchant, de degré inférieur; les diminutions de degré en u et en v sont indépendantes.

Cas terminal de récursivité:
Un sous carreau n'est plus subdivisé dès qu'il est de degré 2x2.

IV.1.3 Décomposition d'un carreau de Bézier rationnel en carreaux de Bézier rationnels de degré 2x2

Diminution du degré d'un carreau Bézier rationnel

Tous les traitements utilisés dans la diminution de degré du carreau (Tests, Algorithme de diminution de degré) sont appliqués dans l'espace homogène là où le carreau est polynomial (cf I.1.3)

Décomposition récursive associée à une diminution de degré

La démarche est strictement identique à celle employée pour les carreaux polynomiaux. Tous les traitements sont effectués dans l'espace homogène. Le carreau de Bézier rationnel initial est décomposé en un arbre 4-aire de carreaux rationnels dont les feuilles sont des carreaux de Bézier rationnels de degré 2x2.

Note

La subdivision récursive d'un carreau de Bézier rationnel permet non seulement de diminuer le degré des portions successives mais aussi d'approcher celles-ci par des carreaux polynomiaux (cf annexe).

IV.1.4 Synthèse: Décomposition d'un carreau NURBS en carreaux de Bézier rationnels de degré 2x2

Un carreau NURBS est décomposé en un arbre 4-aire de sous carreaux de plus en plus simples et dont les feuilles sont des carreaux de Bézier rationnels de degré 2x2. Les deux étapes essentielles sont la décomposition du carreau NURBS initial en carreaux de Bézier rationnels puis la décomposition de chaque carreau de Bézier rationnel en carreaux terminaux. Nous avons choisi une structure de données arborescente. La concision des données est un objectif majeur car les carreaux paramétriques "consomment" beaucoup de mémoire.

IV.1.4.1 Structure de données

1) Représentation des surfaces

Carreau NURBS

Ce type de carreau (cf définition I.1.3) est connu par:
- sa matrice de points de contrôle dans l'espace homogène $[Q_{ij}]$ exprimée dans la base de Bernstein
- m, n
- k, l
- Les vecteurs nodaux $[u_i]$ et $[v_j]$; pour chaque noeud on dispose de sa valeur (réel) et de sa multiplicité (entier)

Il est représenté séquentiellement (enregistrement):

k
ℓ
m
n
$[u_i]$
$[v_j]$
$[Q_{ij}]$

Carreau de Bézier rationnel

Ce type de carreau (cf définition I.1.3) est connu par:
- les degrés suivant u et v: m, n
- sa matrice de points de contrôle dans l'espace homogène $[Q_{ij}]$ exprimée dans la base de Bernstein

Il est représenté séquentiellement (enregistrement):

m
n
$[Q_{ij}]$

Carreau de Bézier rationnel de degré 2x2
Ce dernier type de carreau est représenté par une matrice de 9 points de contrôle en 4D

2)Structure de l'arbre

L'arbre de décomposition du carreau NURBS contient des noeuds représentant des surfaces qui peuvent être de trois natures différentes, chaque noeud contient aussi un englobant du sous carreau, ainsi qu'une information pour le situer dans le domaine paramétrique du carreau racine.

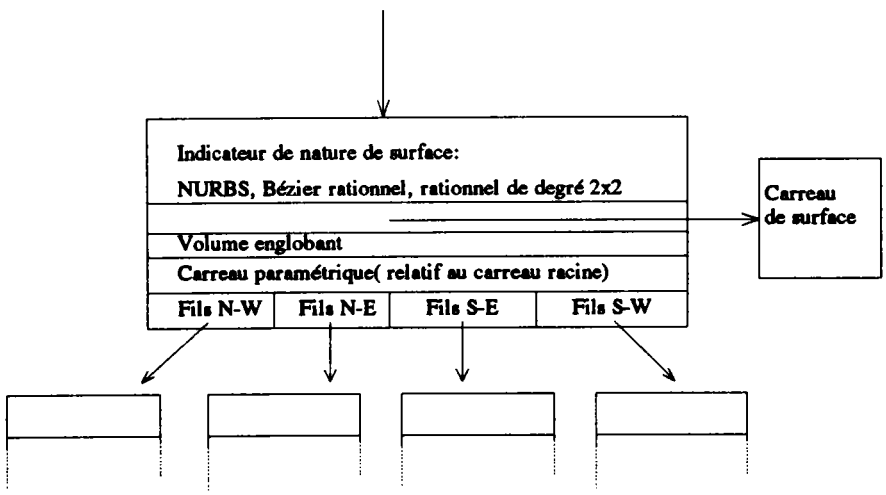
Volume englobant

Il est défini comme dans l'algorithme de dichotomie et intersection de plans, par la boîte Min-Max.

Carreau paramétrique

Chaque noeud contient un rectangle 2D qui indique quelle portion de surface le sous carreau représente par rapport au carreau racine.

La structure de données retenue pour représenter un noeud de l'arbre a la forme:



IV.1.4.2 Algorithme

L'algorithme complet se résume à deux modules récursifs: Simplifie_Nurbs et Simplifie_Bézier

Description de Simplifie_Nurbs:
Argument: $\mathcal{C}$: un carreau Nurbs (*sous la forme d'un noeud de l'arbre*)

(* Ce module décompose récursivement le carreau $\mathcal{C}$ en quatre sous carreaux;
cas terminal: c'est un carreau de Bézier*)

Début
Si k = m+1 *et* l = n+1 (* c'est un carreau de Bézier *)
Alors
- Changer la structure de représentation de $\mathcal{C}$
- Appeler Simplifie_Bézier($\mathcal{C}$)

Sinon (* On décompose $\mathcal{C}$ avec Cox-De Boor *)

- Calcul des paramètres de découpage u^* et v^*
- Subdiviser $\mathcal{C}$ en $\mathcal{C}_{11}$ $\mathcal{C}_{12}$ $\mathcal{C}_{22}$ $\mathcal{C}_{21}$
- Créer les 4 carreaux-fils: fils N-W, fils N-E, fils S-E, fils S-W
- Appeler Simplifie_Nurbs(fils N-W)
- Appeler Simplifie_Nurbs(fils N-E)
- Appeler Simplifie_Nurbs(fils S-E)
- Appeler Simplifie_Nurbs(fils S-W)

Fin_si
Fin

Remarque: On a omis les cas particuliers où k = m+1 et l < n+1 (et k < m+1 et l = n+1) pour simplifier.

Description de Simplifie_Bézier:
Argument: $\mathcal{C}$: un carreau de Bézier rationnel (*sous la forme d'un noeud de l'arbre*)

(* Ce module décompose récursivement le carreau $\mathcal{C}$ en quatre sous carreaux;
cas terminal: c'est un carreau de Bézier polynomial et de degré 2 x 2 *)

Début
Tant que m > 2 *et* test indiquant qu'il est possible de diminuer le degré en u
faire
- Diminuer le degré de $\mathcal{C}$ suivant la direction u

fin_tant que
.....
(* Instruction identique pour diminuer le degré en v*)
.....

Si m = 2 *et* n = 2
Alors
- Changer la structure de représentation de $\mathcal{C}$

Sinon (* On décompose $\mathcal{C}$ avec De Casteljau *)
- Calcul des paramètres de découpage u^* et v^*
- Subdiviser $\mathcal{C}$ en $\mathcal{C}_{11}$ $\mathcal{C}_{12}$ $\mathcal{C}_{22}$ $\mathcal{C}_{21}$
- Créer les 4 carreaux-fils: fils N-W, fils N-E, fils S-E, fils S-W
- Appeler Simplifie_Bézier(fils N-W)

- Appeler Simplifie_Bézier(fils N-E)
- Appeler Simplifie_Bézier(fils S-E)
- Appeler Simplifie_Bézier(fils S-W)

Fin_si
Fin

Remarque: On a omis certains cas particuliers (par exemple: m = 2 et n > 2) pour simplifier.

IV.1.5 Application à la simplification du problème: Intersection Carreau Nurbs-Carreau Nurbs

Nous recherchons les courbes d'intersection de deux carreaux Nurbs $\mathcal{N}_1$ et $\mathcal{N}_2$. L'utilisation de l'algorithme décrit en I.1.4 permet de simplifier de façon importante le problème. Nous appliquons simultanément aux deux carreaux l'algorithme précédent; et nous ne conservons à chaque découpage que les sous carreaux provenant de $\mathcal{N}_1$ et $\mathcal{N}_2$ dont les boîtes min-max sont sécantes, comme dans l'algorithme de recherche dichotomique (cf III.2).
L'algorithme complet s'exprime sous la forme d'un module récursif Intersection:

Description du module Intersection:

Arguments: $\mathcal{C}_1$, $\mathcal{C}_2$ (* 2 carreaux Nurbs ou Bézier rationnel ou rationnel de degré 2x2 sous la forme de noeuds*)

(* cas terminal: Les volumes englobants de $\mathcal{C}_1$ et $\mathcal{C}_2$ sont disjoints ou les 2 carreaux sont de degré 2x2 *)

Début
Si boîte($\mathcal{C}_1$) $\cap$ boîte($\mathcal{C}_2$) $\neq \phi$
Alors si $\mathcal{C}_1$ est un carreau terminal *et* $\mathcal{C}_2$ est un carreau terminal
Alors
- Appeler Recherche_directe_d'intersections($\mathcal{C}_1$, $\mathcal{C}_2$)

Sinon
- Simplifier $\mathcal{C}_1$ éventuellement
- Simplifier $\mathcal{C}_2$ éventuellement
- Calcul des paramètres de découpage u^* et v^*

- Subdiviser $\mathcal{C}_1$ en $\mathcal{C}_{11}$ $\mathcal{C}_{12}$ $\mathcal{C}_{13}$ $\mathcal{C}_{14}$

(* A l'aide de Cox-De Boor ou De casteljau selon la nature du carreau*)

- Créer les 4 carreaux-fils du noeud contenant $\mathcal{C}_1$:

fils N-W, fils N-E, fils S-E, fils S-W

- Calcul des paramètres de découpage r^* et s^*
- Subdiviser $\mathcal{C}_2$ en $\mathcal{C}_{21}$ $\mathcal{C}_{22}$ $\mathcal{C}_{23}$ $\mathcal{C}_{24}$
- Créer les 4 carreaux-fils du noeud contenant $\mathcal{C}_2$:

fils N-W, fils N-E, fils S-E, fils S-W

- Appeler Intersection(fils N-W($\mathcal{C}_1$), fils N-W($\mathcal{C}_2$))
- Appeler Intersection(fils N-W($\mathcal{C}_1$), fils N-E($\mathcal{C}_2$))
- Appeler Intersection(fils N-W($\mathcal{C}_1$), fils S-E($\mathcal{C}_2$))
- Appeler Intersection(fils N-W($\mathcal{C}_1$), fils S-W($\mathcal{C}_2$))
- Appeler Intersection(fils N-E($\mathcal{C}_1$), fils N-W($\mathcal{C}_2$))

.....

(* En tout 16 appels récursifs *)

.....

- Appeler Intersection(fils S-W($\mathcal{C}_1$), fils S-W($\mathcal{C}_2$))

Fin si

Fin si

Fin

Cet algorithme rend la liste des couples ($\mathcal{C}_1$, $\mathcal{C}_2$) de sous carreaux simplifiés ($\mathcal{C}_1 \subset \mathcal{N}_1$ et $\mathcal{C}_2 \subset \mathcal{N}_2$) impliqués dans l'intersection de $\mathcal{N}_1$ et $\mathcal{N}_2$. Les arbre 4-aires sont incomplets, seules les branches impliquées dans l'intersection sont développées. Cependant l'algorithme conduit à une croissance quasi exponentielle (en le nombre d'étages) du volume de données et de calcul. Pour économiser de la mémoire, on détruit les informations géométriques sur un sous carreau dès que ses fils sont créés. De plus lorsqu'un noeud représente un carreau de Bézier rationnel de degré 2 x 2, on change de représentation géométrique: les degrés ne sont plus stockés et seule la place pour 9 points en 4D est utilisée.

IV.1.6 Conclusion

La méthode exposée traite deux carreaux Nurbs dont on cherche les courbes d'intersection. Elle répond à deux tâches:
1) Décomposer les deux carreaux en parcelles et ne sélectionner que celles impliquées dans l'intersection.
2) Les parcelles retenues sont simplifiées progressivement jusqu'à qu'elles soient des Bézier biquadratiques rationnels.

Elle sert de prétraitement en simplifiant considérablement les objets dont on étudie l'intersection. Les couples de sous carreaux obtenus sont suffisamment simplifiés pour être traités avec des méthodes de résolution directe de systèmes d'équations polynomiales (technique de Résolvant). L'idée de la dichotomie récursive est reprise mais l'algorithme s'arrête avant d'obtenir des approximations planes. La récursivité est donc moins profonde d'où un gain de temps et de place. Il se distingue aussi par le fait que les sous carreaux sont progressivement simplifiés. Leur degré diminue régulièrement au cours des des découpages successifs. La complexité des calculs diminue donc aussi à chaque découpage.
Une réalisation a montré qu'il faut de trois à cinq subdivisions successives pour décomposer un carreau bicubique en carreaux biquadratiques; l'erreur relative en un point de la surface est alors de l'ordre de 10^{-3}.

IV.2 RECHERCHE D'EQUATIONS CARTESIENNES

IV.2.1 Recherche d'équation cartésienne exacte pour un carreau rationnel

Soit un carreau $\mathcal{C}$ paramétrique défini par: $(u, v) \longmapsto \sigma(u, v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix}$ (1)

Le problème est d'obtenir une équation polynomiale en x,y,z , $F(x,y,z) = 0$ (2)
telle que $\forall\ u, \forall\ v$ on ait $F(x(u,v),y(u,v),z(u,v)) = 0$
On recherche donc une condition sur un point (X,Y,Z) de $\mathbf{R}^3$ pour qu'il existe un unique

$$(u,v) \text{ tel que } \begin{cases} x(u,v) - X = 0 \\ y(u,v) - Y = 0 \\ z(u,v) - Z = 0 \end{cases} \quad (3)$$

La méthode générale consiste à éliminer les variables u et v du système [GOL 85].
Sederberg[SED84] montre qu'il est possible d'obtenir, pour un carreau de degré mxm une équation cartésienne de degré $2m^2$.
Pour un carreau biquadratique elle est donc de degré 8 et pour un bicubique, elle est de degré 18.
Le nombre de coefficients du polynôme F est
$N(p) = \frac{1}{6}(p+1)(p+2)(p+3)$ où p est le degré total en x,y,z (cf III.3.1)
aussi l'équation d'un carreau bicubique compte 1330 termes!
L'encombrement de ce polynôme est considérable, l'équation cartésienne est donc d'un intérêt uniquement théorique pour $m > 2$.

Si cette équation est utilisée pour appliquer la méthode de Farouki (cf III.3) à l'intersection de deux carreaux biquadratiques la dernière équation à résoudre $r(v) = 0$ (cf III.3.5) est de degré 496.

Ces difficultés insurmontables nous conduisent plutôt à rechercher des équations cartésiennes approchées de plus faible degré.

IV.2.2 Recherche d'équation cartésienne approchée pour un carreau paramétrique

Equation globale

Nous étudions la possibilité pour un carreau de Bézier $\mathcal{B}$ polynomial biquadratique d'obtenir une équation cartésienne approchée de degré 2, autrement dit une quadrique qui approche $\mathcal{B}$.
Cette démarche est justifiée par deux faits:
1) Il est possible de décomposer un carreau NURBS en carreaux de ce type.
2) Le calcul de l'intersection d'un carreau de ce type et d'une quadrique par la méthode de Farouki produit un résolvant de degré 28, ce qui est raisonnable.

Rechercher une quadrique correspond à trouver les 10 coefficients de l'équation

$$F(x,y,z) = \sum_{i+j+k \leq 2} a_{ijk} \cdot x^i \cdot y^j \cdot z^k = 0$$

La méthode la plus simple consiste à faire passer la quadrique par plusieurs points (x_j,y_j,z_j) régulièrement répartis sur $\mathcal{B}$. On obtient un système d'équations $F(x_j,y_j,z_j) = 0$ linéaires en les coefficients.

Mais aucune quadrique solution n'est assez proche (même grossièrement) du carreau $\mathcal{B}$, cela semble naturel lorsqu'on connaît la complexité de l'équation exacte.

Equation locale

Il est possible d'approcher n'importe quelle surface paramétrique localement par une quadrique.
Soit **P** un point du carreau $\mathcal{C}$. Notons k_1 et k_2 les courbures principales en ce point et $\mathbf{e}_1$, $\mathbf{e}_2$ les vecteurs propres associés (cf II.4). Supposons $\mathbf{e}_1$, $\mathbf{e}_2$ normés et orientés de telle sorte que $R = \{\mathbf{P}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{N}\}$ soit un repère orthonormé direct de l'espace associé à **P**. **N** est un vecteur normal à $\mathcal{C}$ en **P**.

Dans ce repère, nous considérons le paraboloïde $\mathcal{P}$ d'équation $Z = \frac{k_1}{2} X^2 + \frac{k_2}{2} Y^2$.

Ce dernier passe par **P**, possède le même plan tangent que $\mathcal{C}$ en **P**; ses courbures principales et ses directions de courbures principales en **P** sont également confondues avec celles de $\mathcal{C}$.
$\mathcal{P}$ constitue une approximation géométrique d'ordre 2 de $\mathcal{C}$ au voisinage de **P**.
Le caractère local de cette équation en limite cependant l'intérêt, elle peut contribuer toutefois à améliorer un algorithme de suivi de courbe (cf IV.4.2).

Des méthodes de calcul d'équations cartésiennes locales de degré variable sont développées dans [HOF 89] et dans [MON 86]; l'intérêt est la possibilité de choisir l'ordre d'approximation géométrique.

L'impossibilité d'obtenir une équation cartésienne "satisfaisante" exprime en fait la grande complexité des carreaux paramétriques polynomiaux.
Ces difficultés nous conduisent à renoncer à adapter la méthode exposée en III.3 aux carreaux NURBS.

IV.3 RESOLUTION ALGEBRIQUE DU PROBLEME D'INTERSECTION DE DEUX CARREAUX DE BEZIER RATIONNELS BIQUADRATIQUES

L'algorithme présenté correspond à l'approche algébrique évoquée en III.4.

Description générale

Soient deux carreaux de Bézier rationnels de degré 2x2, $\mathcal{B}_1$ et $\mathcal{B}_2$, définis par $(u,v) \longmapsto \mathbf{X}(u,v)$ et par $(r,s) \longmapsto \mathbf{Y}(r,s)$.

La courbe recherchée, l'intersection de $\mathcal{B}_1$ et $\mathcal{B}_2$ est notée $\mathcal{C}$.

L'algorithme général s'exprime très simplement:
- On couvre l'un des deux carreaux, par exemple $\mathcal{B}_1$ par un quadrillage de courbes isoparamétriques.
Notons $\{\mathcal{L}_j\}_{j=0...N;}$ la suite d'isoparamétriques à u fixé et $\{\mathcal{C}_j\}_{i=0...N;}$ la suite d'isoparamétriques à v fixé.
$\mathcal{L}_j$ est définie par $v \longmapsto \mathbf{X}(\frac{j}{N},v)$ et $\mathcal{C}_i$ est définie par $u \longmapsto \mathbf{X}(u,\frac{i}{N})$
- Pour chaque courbe on calcule le (ou les) point d'intersection entre elle et le carreau $\mathcal{B}_2$
- La courbe $\mathcal{C}$ est reconstituée à partir des points successifs calculés.

Les deux principales étapes sont le calcul de l'intersection:courbe-carreau et la reconstruction de la courbe.
Ce traitement a déjà été exposé en III.2.6, cependant il est ici simplifié par le fait que les points calculés sont "presque classés". Les liaisons entre les points sont d'autant plus faciles à déterminer que la densité de points est grande, c'est à dire que le nombre N d'isoparamétriques dans chaque direction est élevé.

L'élément essentiel de l'algorithme est donc la résolution du problème: intersection courbe-carreau.

IV.3.1 Résolution algébrique du problème: Intersection courbe de Bézier rationnelle de degré 2-carreau de Bézier rationnel de degré 2x2

Description générale:

La courbe de Bézier, notée $\mathcal{C}$ est définie par :

$\gamma : [0,1] \longrightarrow \mathbf{R}^3$

$t \longmapsto \gamma(t) = \frac{1}{d(t)} (A_0 + A_1 t + A_2 t^2)$ où $Ai \in \mathbf{R}^3$ pour $i = 0...2$ et

$d(t) = d_0 + d_1 t + d_2 t^2 > 0, \ \forall t \in [0,1]$

dans la base canonique des polynômes.

Le carreau est défini par $(u,v) \longmapsto \sigma(u,v) = \frac{1}{w(u,v)} \begin{bmatrix} a(u,v) \\ b(u,v) \\ c(u,v) \end{bmatrix}$

où a, b, c, w sont des polynômes de degré 2x2.

Le système à résoudre, $\gamma(t) = \sigma(u,v)$ est résolu par élimination successive des inconnues; la méthode est décomposée en plusieurs étapes:

1) Obtention des équations cartésiennes de deux surfaces dont la courbe $\mathcal{C}$ est l'intersection.
2) Utiliser ces équations pour réduire le système 3x3 en un système 2x2 polynomial en u,v.
3) Résoudre ce dernier système à l'aide des résolvants.
4) Trouver la 3ème inconnue t.

La méthode proposée apparaît donc comme une généralisation de celle décrite par Kajiya [KAJ 82] pour résoudre le problème: Intersection droite - Carreau bicubique.

IV.3.1.1 Equation cartésienne de la courbe

Supposons que la famille $L = \{ A_0 , A_1 , A_2 \}$ soit libre;
c'est à dire que la matrice $M = [A_0 \ A_1 \ A_2]$ soit inversible.

Dans la base L on a: $M^{-1}.\gamma(t) = \frac{1}{d(t)} \begin{bmatrix} 1 \\ t \\ t^2 \end{bmatrix}$

Cherchons une équation de $\mathcal{C}$:

$$\begin{cases} x = \frac{1}{d} \\ y = \frac{t}{d} \\ z = \frac{t^2}{d} \\ d = d_0 + d_1 t + d_2 t^2 \end{cases} <=> \begin{cases} x = \frac{1}{d} \\ y = \frac{t}{d} \\ z = \frac{t^2}{d} \\ d_0 x + d_1 y + d_2 z = 1 \end{cases}$$

$$<=> \begin{cases} x = \frac{1}{d} \\ y = \frac{t}{d} \\ x z = y^2 \\ d_0 x + d_1 y + d_2 z = 1 \end{cases}$$

Les deux dernières équations montrent que $\mathcal{C}$ est l'intersection d'une quadrique et d'un plan.

IV.3.1.2 Séparation des inconnues

Dans la base L on a: $\mathbf{M}^{-1}.\sigma(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix}$ qui s'écrit aussi $\frac{1}{w(u,v)} \begin{bmatrix} X(u,v) \\ Y(u,v) \\ Z(u,v) \end{bmatrix}$

or

$\gamma(t) = \sigma(u,v) <=> \mathbf{M}^{-1}.\gamma(t) = \mathbf{M}^{-1}.\sigma(u,v) <=>$

$$\begin{cases} x(u,v) = \dfrac{l}{d} & (\text{équ. 1}) \\ y(u,v) = \dfrac{t}{d} & (\text{équ. 2}) \\ x(u,v)\, z(u,v) = y(u,v)^2 & (\text{équ. 3}) \\ d_0\, x(u,v) + d_1\, y(u,v) + d_2\, z(u,v) = 1 & (\text{équ. 4}) \end{cases}$$

on déduit le système $\begin{cases} X(u,v).Z(u,v) = Y^2(u,v) \\ d_0\, X(u,v) + d_1\, Y(u,v) + d_2\, Z(u,v) = w(u,v) \end{cases}$ (1)

C'est un système polynomial en u et v.

IV.3.1.3 Résolution du système

Le système (1) est résolu par la méthode décrite en III.3.5.
La première équation est de degré 4x4 et la seconde est de degré 2x2.
par conséquent le résolvant obtenu r(v) est de degré 16.
Pour chaque couple (u_j,v_j) solution de (1), la (ou les) solution t correspondante est obtenue en résolvant l'équation (équ. 1).

Remarques
Le carreau et la courbe ont donc au maximum 16 points d'intersection; ce résultat est aussi montré dans [CHA 87].
Par ailleurs en combinant l'équation cartésienne du carreau (de degré 8 en x,y,z) et l'équation paramétrique de la courbe (de degré 2 en t) on obtient également une équation en t de degré 16. Les deux voies sont donc mathématiquement cohérentes.
Dans les cas particuliers où la famille L est liée, on montre aussi que $\mathcal{C}$ est l'intersection d'une quadrique et d'un plan (cf de deux plans). Par conséquent dans tous les cas le degré du résolvant est inférieur ou égal à 16.
Généralement le polynôme r(v) peut être approché avec une grande précision sur [0,1] par un polynôme de degré plus faible (entre 10 et 12 pour les intersections calculées); l'équation pratique est donc généralement plus simple que l'équation théorique.

IV.3.2 Généralisation au cas cubique

La démarche est analogue à la précédente quoique rendue plus complexe à cause de l'élévation du degré.

Description générale:

La courbe de Bézier, notée $\mathcal{C}$ est définie par :

$\gamma : [0,1] \longrightarrow \mathbf{R}^3$

$t \longmapsto \gamma(t) = \frac{1}{d(t)} (A_0 + A_1 t + A_2 t^2 + A_3 t^3)$ où $Ai \in \mathbf{R}^3$ pour $i = 0...3$ et

$$d(t) = d_0 + d_1 t + d_2 t^2 + d_3 t^3 > 0,$$

dans la base canonique des polynômes.

Le carreau est défini par $(u,v) \longmapsto \sigma(u,v) = \frac{1}{w(u,v)} \begin{bmatrix} a(u,v) \\ b(u,v) \\ c(u,v) \end{bmatrix}$

où a, b, c, w sont des polynômes de degré 3x3.

IV.3.2.1 Transformation du système

Supposons que la famille $L = \{ A_1 , A_2 , A_3 \}$ soit libre;
c'est à dire que la matrice $M = [A_1 \ A_2 \ A_3]$ soit inversible.

Le système $\gamma(t) = \sigma(u,v)$ peut s'écrire $\frac{1}{d(t)} \left(A_0 + M. \begin{bmatrix} t \\ t^2 \\ t^3 \end{bmatrix} \right) = \sigma(u,v)$

En multipliant par M^{-1}, on obtient:

$$\frac{1}{d(t)} \left(M^{-1}.A_0 + \begin{bmatrix} t \\ t^2 \\ t^3 \end{bmatrix} \right) = M^{-1}. \sigma(u,v)$$

Ce qui s'écrit encore $\frac{1}{d(t)} \begin{bmatrix} a_1 + t \\ a_2 + t^2 \\ a_3 + t^3 \end{bmatrix} = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix}$

avec $x(u,v) = \frac{X(u,v)}{w(u,v)}$, $y(u,v) = \frac{Y(u,v)}{w(u,v)}$, $z(u,v) = \frac{Z(u,v)}{w(u,v)}$ où X,Y et Z sont de degré 3x3.

On considère le système

$$\begin{cases} x = \frac{1}{d}(a_1 + t) \\ y = \frac{1}{d}(a_2 + t^2) \\ z = \frac{1}{d}(a_3 + t^3) \\ d = d_0 + d_1 t + d_2 t^2 + d_3 t^3 \end{cases} \quad <=> \quad \begin{cases} x = \frac{1}{d}(a_1 + t) \\ y = \frac{1}{d}(a_2 + t^2) \\ z = \frac{1}{d}(a_3 + t^3) \\ d_1 x + d_2 y + d_3 z = \frac{K}{d} + 1 \end{cases}$$

avec $K = d_1a_1 + d_2a_2 + d_3a_3 - d_0$ supposé non nul

$$<=> \quad \begin{cases} \frac{t}{d} = x - \frac{a_1}{d} \\ \frac{t^2}{d} = y - \frac{a_2}{d} \\ \frac{t^3}{d} = z - \frac{a_3}{d} \\ d_1 x + d_2 y + d_3 z - 1 = \frac{K}{d} \end{cases}$$

Comme $\left(\frac{t}{d}\right)^2 = \frac{1}{d}\frac{t^2}{d}$ et $\frac{t}{d}\frac{t^2}{d} = \frac{1}{d}\frac{t^3}{d}$ on en déduit

$$\begin{cases} \frac{t}{d} = x - \frac{a_1}{d} & (1) \\ \left(x - \frac{a_1}{d}\right)^2 = \frac{1}{d}\left(y - \frac{a_2}{d}\right) & (2) \\ \left(x - \frac{a_1}{d}\right)\left(y - \frac{a_2}{d}\right) = \frac{1}{d}\left(z - \frac{a_3}{d}\right) & (3) \\ \frac{1}{K}\left(d_1 x + d_2 y + d_3 z - 1\right) = \frac{1}{d} & (4) \end{cases}$$

Les équations (2) et (3) montrent que $\mathcal{C}$ est l'intersection de deux quadriques. Des résultats identiques sont montrés de façon différente dans [CHA 87].

En remplaçant x, y et z par leurs expressions, dans ces mêmes équations, on obtient un système en u et v constitué de deux équations polynomiales de degré 6x6.

IV.3.2.2 Résolution du système

La démarche est identique à celle décrite en IV.3.1.3
Cependant le degré élevé (6x6 + 6x6 = 72) du résolvant r(v) rend la détermination de ses zéros très délicate. Le problème majeur est d'ordre de stabilité numérique (cf IV.3.3) mais l'encombrement et le temps de calcul sont importants aussi. Cette extension au cas cubique n'a pas encore été développée. Toutefois on peut extrapoler à partir du cas quadratique, que le degré du résolvant peut être affaibli jusqu'à 50-55 dans le cas général.
Par ailleurs cette dernière équation montre qu'une courbe cubique traverse un carreau bicubique en 72 points au maximum. Ce résultat est exhaustif puisque si on combine l'équation cartésienne du carreau (de degré 18 en x, y, z) et l'équation paramétrique de la courbe (en t^3) on obtient une équation de degré 54 en t, d'où 54 points d'intersection au maximum. Le résolvant est donc nécessairement réductible.

Cas particuliers

Si la quantité K est nulle, la démonstration est plus longue. On parvient à montrer que $\mathcal{C}$ est l'intersection d'une surface de degré 4 et d'un plan. Par conséquent les deux équations en u et v déduites sont de degré 12x12 et 3x3. Le résolvant est donc aussi de degré 72; la complexité de l'équation reste la même.

Par contre lorsque la matrice **M** est de rang 2, la séparation des variables est plus difficile, nous ne réussissons pas à généraliser les résultats précédents. De longs calculs montrent que $\mathcal{C}$ est située à l'intersection d'une quadrique et d'une surface dont l'équation cartésienne est de degré 3. On aboutit alors à un résolvant r(v) de degré 108.
Une implantation de l'extension au cas cubique ne peut donc prétendre à traiter tous les cas théoriques.

IV.3.3 Résolution d'une équation polynomiale

Les algorithmes précédents aboutissent à la recherche des zéros d'un polynôme
$f(t) = \sum_{i=0}^{m} a_i t^i$ sur [0,1].
Cet intervalle particulier permet la mise-en-oeuvre d'une méthode de séparation des racines originale basée sur les propriétés des courbes de Bézier [GAR 91].

Conditionnement préalable du polynôme

Le polynôme f(t) étant issu de calculs de résolvants, peut être de degré très élevé et présenter des coefficients d'ordre de grandeur très différents. Cette dernière condition perturbe considérablement les algorithmes de résolution. Pour améliorer le comportement du polynôme celui-ci est normé: Chaque coefficient a_i est remplacé par $a_i / \max_{0 \le j \le m} |a_j|$

Les zéros n'appartenant pas à [0,1] n'ont aucun intérêt, on recherche donc un polynôme de plus faible degré possible $\underline{f}(t)$ qui soit confondu (à une précision près) avec f(t) sur [0,1] (cf algorithme de diminution de degré IV.1.1.1). f(t) est alors remplacé par $\underline{f}(t)$.

Note: Ces traitements peuvent être appliqués aux polynômes bivariables avant calcul de résolvants.

Méthode de Newton

Les zéros sont déterminés à l'aide de la méthode de Newton [RAL 65] [CHE 85].
Rappelons les conditions de son utilisation:
Soit un intervalle [a,b] tel que
1) la courbe de f a un seul point d'intersection avec la droite { y = 0 } sur]a,b[

2) La courbe ne présente pas de point d'inflexion sur [a,b]

alors la méthode de Newton converge vers l'unique solution sur]a,b[de $f(x) = 0$. La suite est initialisée avec une des deux bornes a ou b, celle qui vérifie $f(x).f''(x) > 0$.

Par conséquent pour chaque zéro x_0 de f sur [0,1] il faut déterminer un intervalle contenant x_0 et vérifiant ces conditions.

Cette étape de séparation des racines est effectuée dans la base des polynômes de Bernstein, par conséquent on calcule la forme de Bézier $\sum_{i=0}^{m} p_i \, tB_i^m(t)$ de f(t).

La courbe de Bézier plane ($x = t$, $y = f(t)$) a pour polygone de contrôle
$\mathbf{P} = \{ \mathbf{P}_i(i/m, p_i) \}_{i=0...m;}$

L'algorithme de détermination des intervalles s'appuie sur des propriétés importantes des polynômes de Bernstein, des courbes de Bézier et de l'algorithme de De Casteljau, que nous rappelons brièvement.
- Le point de contrôle p_i a le plus d'influence sur la fonction $t \longmapsto f(t)$ en $t = i/m$ car $t \longmapsto B_i^m(t)$ atteint son maximum en cette valeur.
- Une courbe de Bézier est toujours contenue dans l'enveloppe convexe formée par ses points de contrôle et l'on a: $\min_{0 \le j \le m} p_j \le f(t) \le \max_{0 \le j \le m} p_j$

- Le nombre de points d'intersection d'une droite avec le polygone de contrôle d'une courbe de Bézier est supérieur ou égal au nombre de points d'intersection de cette même droite avec la courbe elle-même.
- L'algorithme de De Casteljau est utilisé comme outil de subdivision récursive. Les polygones de contrôle issus de subdivisions successives convergent rapidement vers la courbe.

Algorithme de séparation des racines
Le principe est de subdiviser récursivement la courbe de f jusqu'à obtenir tous les intervalles satisfaisant les conditions (1) et (2).

On considère le polygone de contrôle **P** et le polygone de contrôle **Q** de la courbe de f''. On situe les segments $[P_i, P_{i+1}]$ par rapport à la droite { y = 0 }, quatre cas sont distingués:

*cas*1 Aucun segment ne coupe { y = 0 }, alors il n'y pas de zéros sur [0,1].

*cas*2 **P** coupe une seule fois { y = 0 } et **Q** ne coupe pas { y = 0 }, alors il y a une seule racine dans [0,1] qui peut être trouvée avec la méthode de Newton.

*cas*3 Le polygone **P** coupe plusieurs fois l'axe des x, alors on découpe la courbe en t = (i+1)/m où $[P_i, P_{i+1}]$ est le premier segment qui coupe { y = 0 } (on essaie d'isoler la première racine).
Le même algorithme est appliqué aux deux courbes-filles.

*cas*4 Le polygone **P** coupe une seule fois { y = 0 } et le polygone **Q** coupe au moins une fois cette droite. Il faut séparer la racine et le point d'inflexion: on localise grossièrement (comme dans le cas 3) le zéro de f et le zéro de f'' et on découpe au milieu. Le même algorithme est appliqué aux courbes-filles.

Après plusieurs découpages, on aboutit aux cas terminaux 1 ou 2, sauf si il y a des racines multiples (dans ce cas il est impossible d'obtenir un intervalle sur lequel **P** ne rencontre qu'une seule fois { y =0 }) ou si f et f'' ont un zéro commun.
On prévoit donc un cas terminal de "secours":
Si les bornes a,b de la sous-courbe sont trop rapprochées (ce qui arrive nécessairement après plusieurs subdivisions) on approche la sous-courbe par son polygone de contrôle. Et les intersections de celui-ci avec { y = 0 } constituent des bonnes approximations des zéros de f sur [a,b].

Remarques
Les sous-courbes sont approchées dès que possibles par des courbes de degré inférieur.

La méthode de Newton utilise la base canonique des polynômes et non pas la base de Bernstein car l'algorithme de Horner est de complexité m alors que celui de De Casteljau est de complexité m(m+1)/2.
Cet algorithme peut être utilisé pour la résolution de toute équation polynomiale sur un domaine borné.

Ses avantages sont la simplicité de mise-en-oeuvre, la rapidité de convergence et la stabilité numérique. Il est mal adapté aux équations de degré élevé car le passage de la base canonique des polynômes à la base de Bernstein devient imprécis (cf I.2.1.3).

IV.3.4 Conclusion

Performances

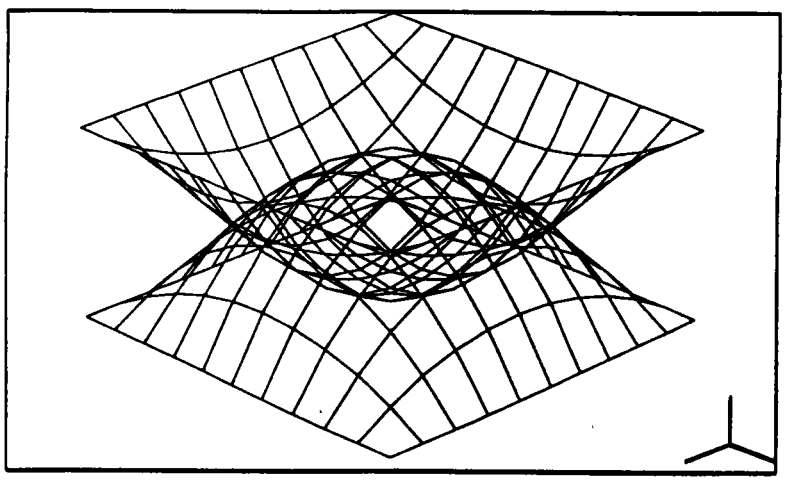
L'analyse de l'algorithme montre que tous les arcs de courbe disjoints sont détectés.
La programmation dans le cas quadratique montre que les solutions sont trouvées avec une très bonne précision même lorsque que sont des points singuliers.
Ces résultats dépendent essentiellement des performances de la méthode de résolution des équations polynomiales sur [0,1].
A précision identique, le temps de calcul engendré par l'algorithme dans le cas quadratique est nettement inférieur à celui nécessaire à la méthode "Dichotomie et Intersections de plans" (cf III.2).

Limites

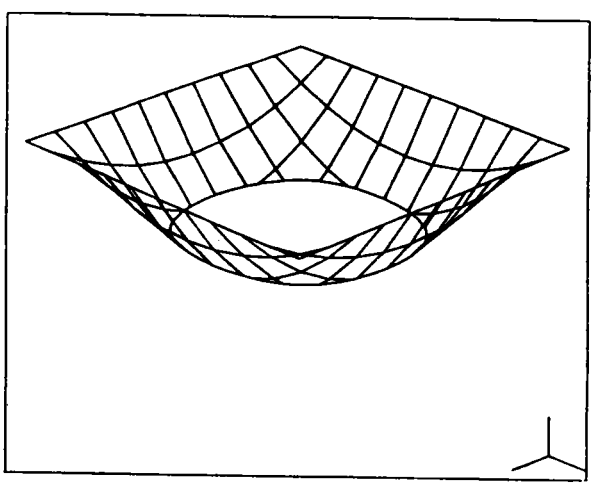
Relier les points pour former la courbe est très simple lorsque les points sont réguliers mais beaucoup plus ambigu au voisinage d'un point singulier. Une erreur de liaison conduit à une incohérence topologique sur la courbe (cf IV.5.2).
Cette difficulté provient du fait que les points singuliers ne sont pas localisés; elle ne remet pas en question la méthode mais montre que celle-ci est insuffisante à traiter complètement le problème.

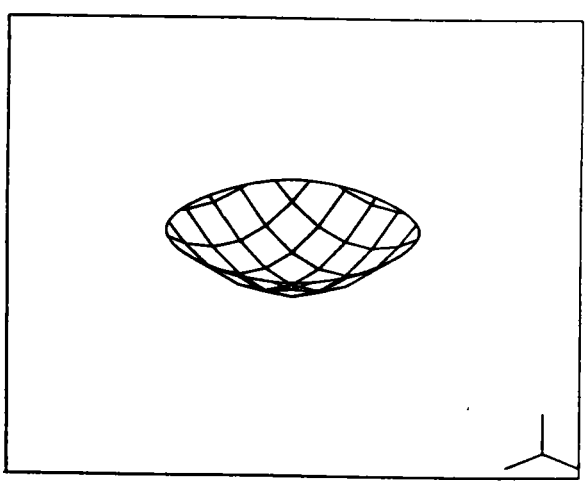
Par ailleurs on voit mal comment généraliser à des degrés supérieurs à 3 cette technique, même théoriquement. En effet on manque alors de dimensions (espace euclidien) pour séparer les puissances successives de t.

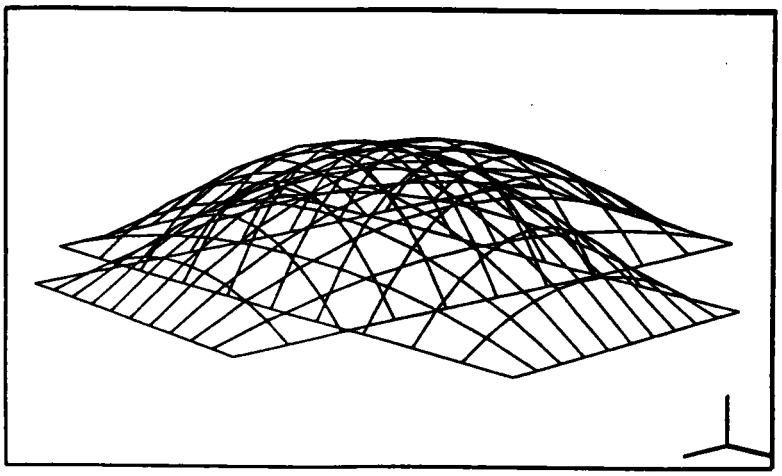
IV.3.5 Exemples



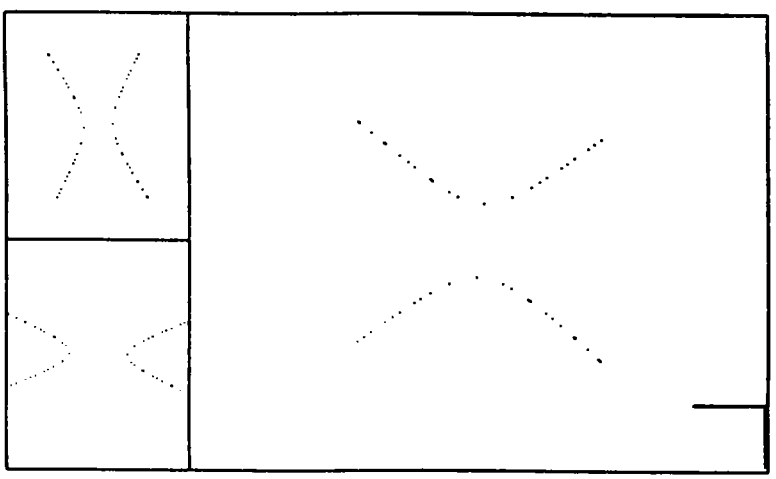
Les deux carreaux restreints déduits du second carreau



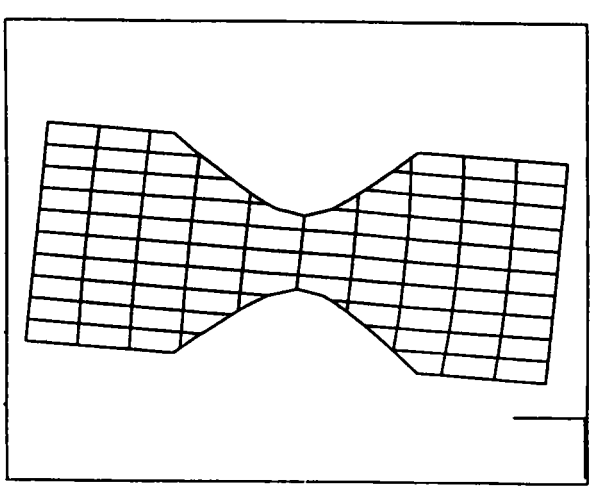


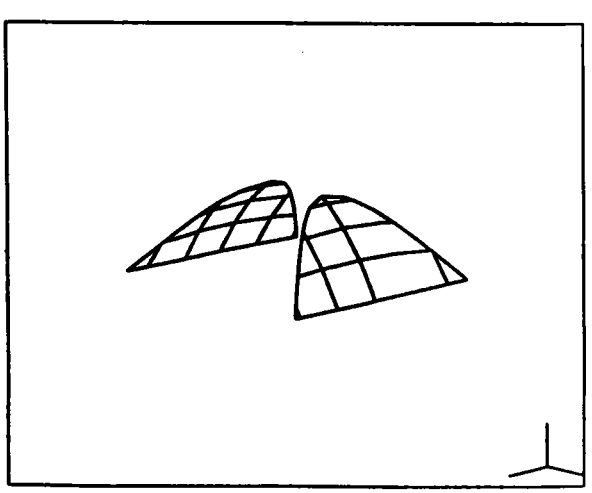


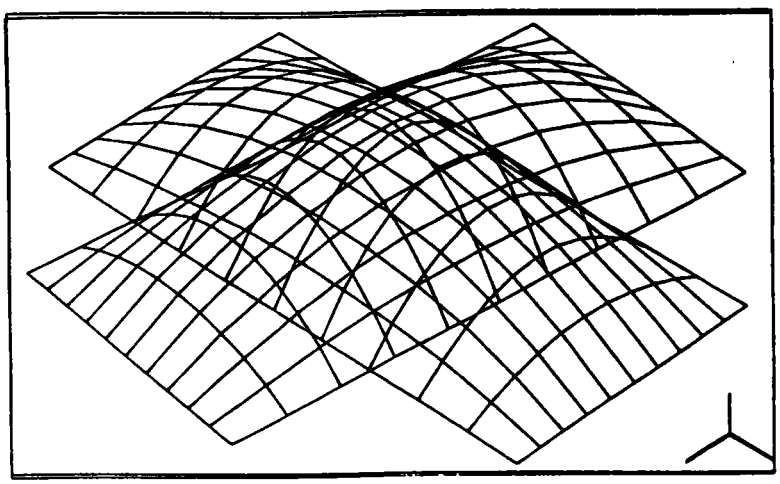
Courbe d'intersection et ses projections dans les domaines de définition des deux carreaux

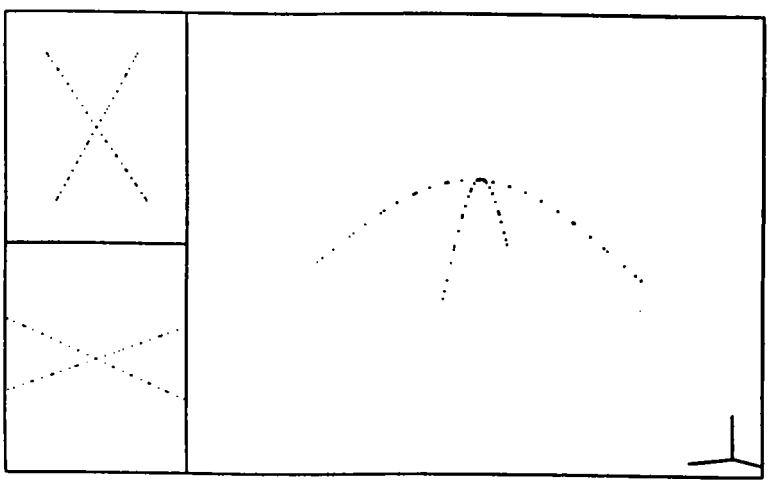


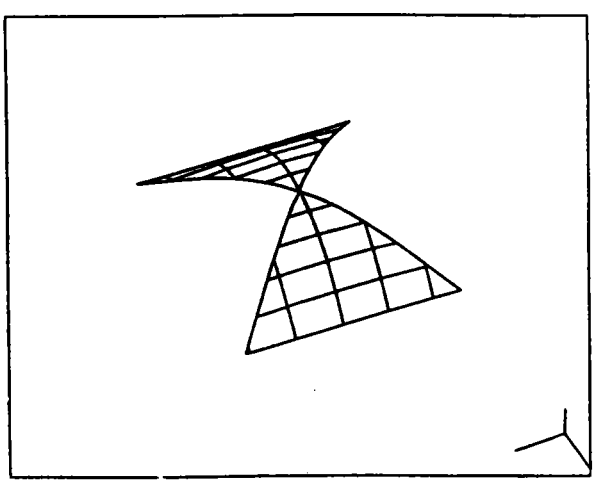
Les deux carreaux restreints déduits du second carreau

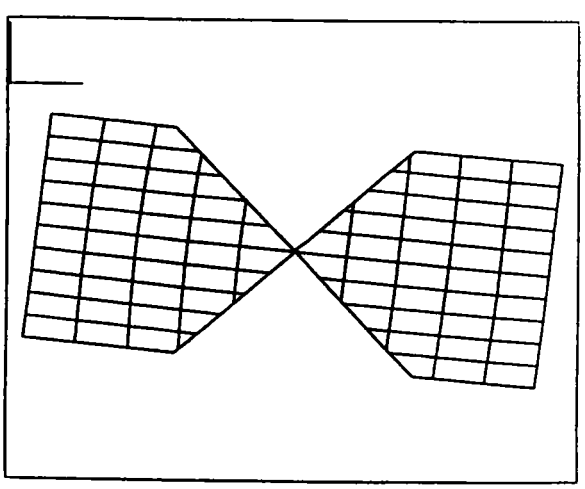


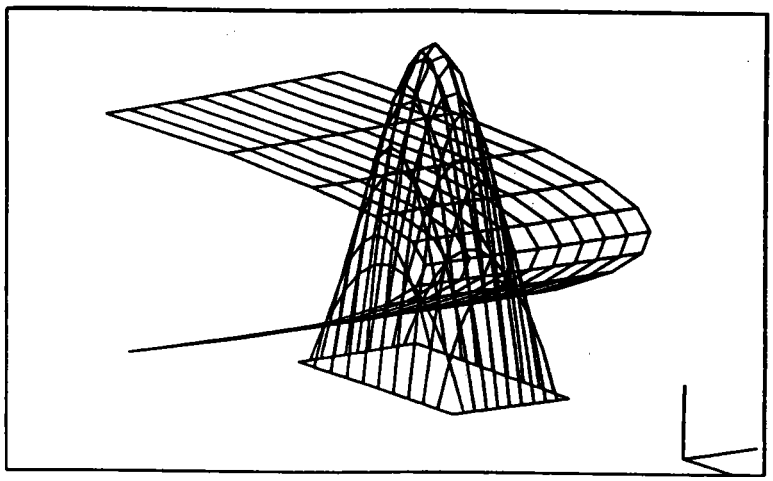


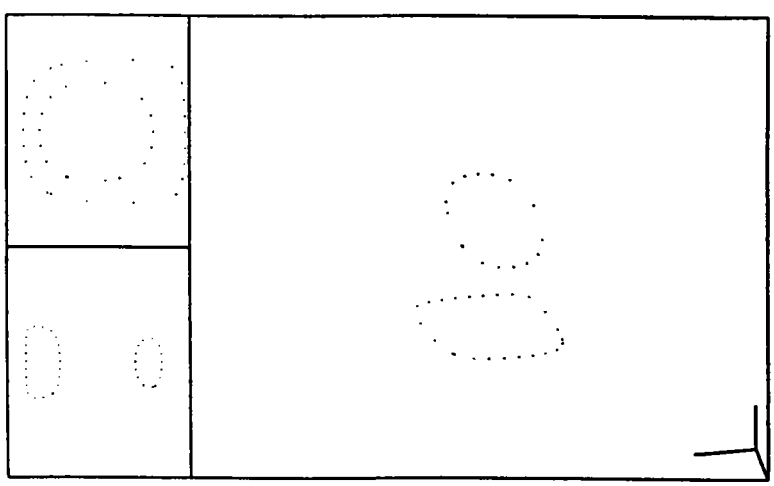


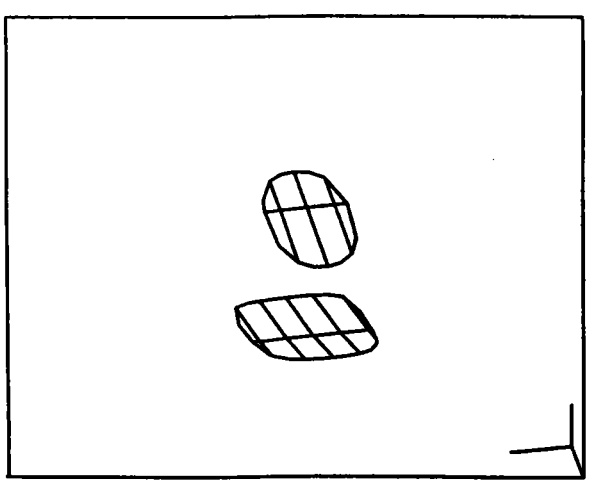












IV.4 METHODE DE SUIVI DE COURBE

L'algorithme exposé correspond à l'approche analytique évoquée en III.4

Description générale

La méthode de "suivi de courbe" calcule une suite de points $\{\mathbf{P}_i\}$ de proche en proche le long de la courbe d'intersection entre deux carreaux NURBS.
Elle est particulièrement adaptée à la reconstruction d'un arc de courbe connexe $\mathcal{C}$.
La principale étape de l'algorithme est le calcul d'un prochain point $\mathbf{P}_{i+1}$ sur $\mathcal{C}$ en fonction de $\mathbf{P}_i$. Cette action s'appuie sur deux propriétés:
1) En $\mathbf{P}_i$ il est possible de calculer de façon exacte des informations géométriques sur la courbe(vecteur tangent, vecteur normal,...) qui permettent la déduction du comportement de $\mathcal{C}$ au voisinage de $\mathbf{P}_i$.
Ces résultats sont utilisés pour guider la recherche du prochain point $\mathbf{P}_{i+1}$.

2) Pour le calcul de $\mathbf{P}_{i+1}$, on dispose d'une solution approchée: $\mathbf{P}_i$ (ou encore une solution approchée plus précise obtenue à l'aide des résultats de (1)). La méthode de Newton utilise de façon optimale cette donnée pour calculer (avec un faible coût de calcul) les coordonnées exactes de $\mathbf{P}_{i+1}$. Les points $\mathbf{P}_i$ et $\mathbf{P}_{i+1}$ doivent être voisins; cela implique la détermination d'un pas d'incrémentation.

Puisque l'algorithme progresse à "tâtons" sur la courbe, sans connaissance globale de $\mathcal{C}$, le contrôle du processus est difficile.
Il faut notamment s'assurer de la validité du dernier point obtenu, arrêter l'algorithme dès que l'arc sort du domaine de définition d'un des deux carreaux ou bien encore savoir détecter un arc fermé et ne pas "boucler" indéfiniment dans ce cas.

Nous décrivons chacune des étapes de l'algorithme puis montrons ses limites et proposons quelques améliorations.

Notations
Les deux carreaux NURBS sécants $\mathcal{S}_1$ et $\mathcal{S}_2$ sont définis comme en II.3.1, l'arc de courbe $\mathcal{C}$ et les deux courbes planes $\mathcal{A}$ et $\mathcal{B}$ associées sont définis comme en III.3.3.2. Nous reprenons également les notations et les hypothèses exposées en II.3.3.3 sur les points calculés.

IV.4.1 Calcul du point suivant sur la courbe

Déterminer un point suivant $\mathbf{P}_{i+1}$ sur $\mathcal{C}$ signifie calculer ses coordonnées locales (u_{i+1},v_{i+1}) et (r_{i+1},s_{i+1}) telles que $\mathbf{X}(u_{i+1},v_{i+1}) = \mathbf{Y}(r_{i+1},s_{i+1}) = \mathbf{P}_{i+1}$.
Nous supposons connue une solution approchée $(\underline{u}_{i+1}, \underline{v}_{i+1}, \underline{r}_{i+1}, \underline{s}_{i+1})$.
Le système à résoudre ($\mathbf{X}(u,v) = \mathbf{Y}(r,s)$) étant sous-déterminé il faut fixer une inconnue, u par exemple.
Par conséquent $u_{i+1} = u_{fixé}$ et le système est réduit au système 3x3: $\mathbf{X}(u_{fixé},v) = \mathbf{Y}(r,s)$
$\mathbf{P}_{i+1}$ est donc un point d'intersection entre la courbe isoparamétrique $v \longmapsto \mathbf{X}(u_{fixé}, v)$ et le carreau δ_2.
La détermination de $\mathbf{P}_{i+1}$ est donc ramenée à la résolution du problème intersection courbe-carreau connaissant une solution approchée.

Résolution du problème Intersection courbe-carreau par la méthode de Newton

Hypothèses
Soit $\mathcal{L}$ une courbe NURBS définie par $t \longmapsto \gamma(t)$ et δ un carreau NURBS défini par $(u,v) \longmapsto \sigma(u,v)$.
Nous cherchons le triplet (t* , u* , v*) solution de $\gamma(t) = \sigma(u,v)$ connaissant une solution approchée (t_0 , u_0 , v_0).

Formulation mathématique

On pose: $\mathbf{F}(t, u, v) = \gamma(t) - \sigma(u,v)$
$\mathbf{X} = (t, u, v)$
$\mathbf{X}_0 = (t_0, u_0, v_0)$
$\mathbf{X}^* = (t^*, u^*, v^*)$
Le problème devient donc:
Trouver $\mathbf{X}^*$ solution de $\mathbf{F}(\mathbf{X}) = \mathbf{0}$ (1)
connaissant la solution approchée $\mathbf{X}_0$.

Résolution
$\mathbf{X}^*$ est la limite de la suite classique de la méthode de Newton-Raphson [CHE 85] définie par
$\mathbf{X}_{n+1} = \mathbf{X}_n - \mathbf{Y}$ Y étant la solution du système 3x3 linéaire $\mathbf{F}(\mathbf{X}_n) = \mathbf{F}'(\mathbf{X}_n).\mathbf{Y}$ (2)
où F' est la jacobienne.

Cette méthode exige à chaque pas la résolution de (2) qui est possible uniquement si le carreau et la courbe ne sont pas parallèles aux points $\gamma(t_n)$ et $\sigma(u_n,v_n)$.

Conditions de convergence

Les conditions mathématiques rigoureuses qui garantissent la convergence de la suite $\mathbf{X}_n$ sont très délicates à vérifier; leur mise en oeuvre est complexe et conduit à des calculs très longs.

On "lance" donc la méthode sans savoir au préalable si il y a convergence et on utilise un test de convergence très simple, qui exploite la propriété:

La convergence de la suite $\mathbf{X}_n$ implique la stricte décroissance de la suite $\|\mathbf{F}(\mathbf{X}_n)\| = d_n$.

Le test déduit est donc:

Tant que $d_n < d_{n-1}$ (3) on calcule $\mathbf{X}_{n+1}$.

Bien entendu l'algorithme est aussi interrompu si le système (2) n'est pas inversible.

Critère de succès

La solution est atteinte dès que $\| \mathbf{X}_n - \mathbf{X}_{n-1} \| <$ précision_choisie (4)

Remarque

Si la solution est un point où $\mathcal{b}$ et $\mathcal{l}$ sont tangents (point singulier ou voisin d'un point singulier) ou bien si il n'y a pas de solution, la suite d_n peut devenir stationnaire. Dans ce cas le test (3) n'est plus significatif et par conséquent l'algorithme "boucle".

Le test (3) (théorique) est donc remplacé par le test (pratique) $d_n < d_{n-1} - \mathcal{E}$ (5) où $\mathcal{E}$ est fixé à une valeur négligeable devant précision_choisie.

Ce dernier critère a été établi après plusieurs essais sur machine.

En général l'algorithme converge vers le point d'intersection de $\mathcal{b}$ et $\mathcal{l}$ le plus proche de la solution approchée initiale (le critère (4) est alors satisfait) sinon l'algorithme est stoppé par la mise en défaut du critère (5).

Dans ce cas le dernier triplet calculé $\mathbf{X}_n$ correspond aux points $\gamma(t_n)$ et $\sigma(u_n,v_n)$ qui minimisent la distance carreau-courbe, les plus proches de la solution initiale ou bien il correspond à un point singulier. Les deux possibilités sont distinguées en mesurant $\|\mathbf{F}(\mathbf{X}_n)\|$. Si cette quantité est proche de zéro, $\mathbf{X}_n$ est considéré comme point d'intersection.

Remarque

La programmation de la méthode de Newton (en dimension 1 ou en dimension 3) montre que l'algorithme n'est jamais stoppé par l'impossibilité d'inverser le système (2) même si la solution est un point singulier.

Interprétation géométrique

$\mathbf{X}_{n+1}$ est la solution du système $\mathbf{F}(\mathbf{X}_n) + \mathbf{F}'(\mathbf{X}_n).(\mathbf{X} - \mathbf{X}_n) = \mathbf{0}$
qui peut s'écrire:
$\alpha(t_n) + \alpha'(t_n)(t - t_n) = \sigma(u_n, v_n) + \sigma_u(u_n, v_n)\,(u - u_n) + \sigma_v(u_n, v_n)\,(v - v_n)$

Le membre de gauche de l'équation est une équation paramétrique de la droite tangente à $\mathcal{L}$ en $\alpha(t_n)$ et le membre de droite est une équation du plan tangent à $\mathcal{S}$ en $\sigma(u_n, v_n)$. Par conséquent les différents termes de la suite sont les points d'intersection de droites tangentes et de plans tangents successifs pris sur $\mathcal{L}$ et $\mathcal{S}$.

Conclusion

Cette méthode a été comparée à la méthode de résolution algébrique dans le cas quadratique. Les différences observées entre les solutions non singulières pour chaque méthode sont de l'ordre de 10^{-8}, les valeurs étant bien sûr comprises entre 0 et 1. Cela prouve l'excellente précision de l'une ou l'autre des méthodes.

IV.4.2 Calcul des informations géométriques en un point de la courbe

Hypothèses

Nous supposons que $t \longmapsto \gamma(t)$ est le paramétrage intrinsèque de l'arc de courbe théorique $\mathcal{C}$.
Soit **P** un point de $\mathcal{C}$ connu par ses coordonnées locales
$(u(t^*), v(t^*)) = \alpha(t^*)$ et $(r(t^*), s(t^*)) = \beta(t^*)$.
Notons $\mathcal{P}_1$ (resp $\mathcal{P}_2$) le plan tangent à $\mathcal{S}_1$ (resp $\mathcal{S}_2$) en **P**. Ces deux plans sont supposés sécants.

Nous décrivons comment les vecteurs $\gamma'(t^*)$, $\alpha'(t^*)$, $\beta'(t^*)$, $\gamma''(t^*)$, $\alpha''(t^*)$, $\beta''(t^*)$ peuvent être calculés successivement sans la connaissance des paramétrages exacts.

Détermination de $\gamma'(t^*)$

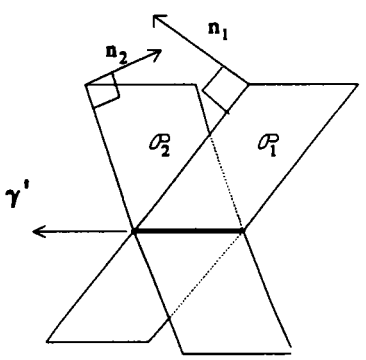
Un vecteur normal unitaire $\mathbf{n}_1$ (resp $\mathbf{n}_2$) au plan $\mathcal{P}_1$ (resp $\mathcal{P}_2$) est obtenu en normant $\mathbf{X}_u \hat{} \mathbf{X}_v$ (resp $\mathbf{Y}_r \hat{} \mathbf{Y}_s$).
$\mathbf{n}_1 \hat{} \mathbf{n}_2$ est un vecteur directeur de la droite $\mathcal{P}_1 \cap \mathcal{P}_2$ or le vecteur $\gamma'(t^*)$ a la même direction que $\mathcal{P}_1 \cap \mathcal{P}_2$ puisque $\mathcal{C}$ est inscrite simultanément sur $\mathcal{S}_1$ et $\mathcal{S}_2$.
Par conséquent $\mathbf{n}_1 \hat{} \mathbf{n}_2$ et $\gamma'(t^*)$ sont colinéaires.
$\gamma'(t^*)$ est donc obtenu en normant $\mathbf{n}_1 \hat{} \mathbf{n}_2$ et en l'orientant à l'aide du point précédent calculé sur $\mathcal{C}$.

Détermination de $\alpha'(t^*) = (u'(t^*), v'(t^*))$

On sait que $\gamma' = \mathbf{X}_u.u' + \mathbf{X}_v.v'$ (1)
donc pour obtenir (u',v') il suffit de calculer les coordonnées (qui sont (u',v',0)) de γ' dans la base $\{ \mathbf{X}_u , \mathbf{X}_v , \mathbf{n}_1 \}$.

$\beta'(t^*) = (r'(t^*), s'(t^*))$ est obtenu de la même façon.

Détermination de $\gamma''(t^*)$



$\{ \mathbf{n}_1 , \mathbf{n}_2 , \gamma' \}$ est une base de l'espace donc
$\gamma'' = x\, \mathbf{n}_1 + y\, \mathbf{n}_2 + z\, \gamma'$
mais comme $\gamma' \perp \gamma''$, on a $\gamma'' = x\, \mathbf{n}_1 + y\, \mathbf{n}_2$ (2)

Il suffit donc de calculer les composantes x et y:

On sait que
$\gamma'' = \mathbf{X}_u.u'' + \mathbf{X}_v.v'' + \mathbf{X}_{uu}.(u')^2 + 2\, \mathbf{X}_{uv}.u'v' + \mathbf{X}_{vv}.(v')^2$

On en déduit:
$(\mathbf{n}_1 \mid \gamma'') = (\mathbf{n}_1 \mid \mathbf{X}_{uu}.(u')^2 + 2\, \mathbf{X}_{uv}.u'v' + \mathbf{X}_{vv}.(v')^2)$

Cette dernière quantité ne dépend pas de u'' et v''; elle peut donc être calculée, notons la a_1.
De la même façon on calcule $a_2 = (\mathbf{n}_1 \mid \gamma'')$.

x et y sont obtenues en résolvant le système $\begin{cases} a_1 = x + (\ n_1 \mid n_2\ )y \\ a_2 = (\ n_1 \mid n_2\ )x + y \end{cases}$ qui est déduit de (2)

Détermination de $\alpha''(t^*)$

L'équation (2) s'écrit aussi

$$\gamma'' - [\ X_{uu}.(u')^2 + 2\ X_{uv}.u'v' + X_{vv}.(v')^2\ ] = X_u.u'' + X_v.v''$$

Le membre de gauche est maintenant connu, nous calculons les coordonnées de ce vecteur dans la base $\{ X_u , X_v , n_1 \}$. Elles valent respectivement u'', v'' et 0.

$\beta''(t^*)$ est calculé de la même façon.

Détermination des vecteurs dérivée d'ordre supérieur

La méthode précédente est facilement généralisée aux dérivées d'ordre supérieur. Le coût en calculs s'élève alors considérablement.
Pour les dérivées à l'ordre n, il faut calculer toutes les dérivées partielles de **X** et **Y** jusqu'à l'ordre n.

L'ordre 2 est suffisant pour le calcul des vecteurs normaux et des courbures sur les trois courbes, informations nécessaires au calcul du prochain point sur $\mathcal{C}$.
Le développement de la méthode de suivi a intégré ces calculs dans ce cas particulier. Les résultats numériques ont permis de vérifier la validité du choix de l'abscisse curviligne pour le parcours de $\mathcal{C}$.
Ce choix présente aussi un autre intérêt.
Si le pas $\| P_{i+1} - P_i \|$ est petit , on a $\| P_{i+1} - P_i \| \simeq t_{i+1} - t_i$. Ce qui permet d'associer facilement une valeur paramétrique t_i à chaque point P_i. Cette information est utile à la modélisation des courbes $\mathcal{A}$ et $\mathcal{B}$.

Markot et Magedson obtiennent les vecteurs dérivée 1ère et 2ème par une approche différente [MAR 91].

Cas particulier: **P** est un point singulier de 1er ordre

Dans ce cas $\mathbf{n}_1$ et $\mathbf{n}_2$ sont colinéaires et la démarche précédente devient impossible.
De même si **P** est voisin d'un point singulier, la quantité $\| \mathbf{n}_1 \wedge \mathbf{n}_2 \|$ est proche de zéro et la méthode précédente donne des solutions très imprécises.

Cependant les carreaux $\mathcal{S}_1$ et $\mathcal{S}_2$, au point **P**, sont distincts au niveau de leurs courbures principales et de leur directions principales de courbure.
Exploitons cette propriété:
On calcule d'une part l'équation cartésienne du paraboloïde tangent à $\mathcal{S}_2$ en **P** (cf IV.2.2) et d'autre part on considère le développement limité à l'ordre 2 de $\mathcal{S}_1$ en **P**:

$$(u,v) \longmapsto \underline{\mathbf{X}}(u,v) = \mathbf{X}(u^*,v^*) + \mathbf{X}_u(u^*,v^*)U + \mathbf{X}_v(u^*,v^*)V + \frac{1}{2}\left(\mathbf{X}_{uu}(u^*,v^*)U^2 + 2\mathbf{X}_{uv}(u^*,v^*)UV + \mathbf{X}_{vv}(u^*,v^*)V^2 \right) \quad (1)$$

où $u^* = u(t^*)$ et $v^* = v(t^*)$ et $U = u - u^*$ et $V = v - v^*$

La combinaison de l'équation du paraboloïde et de (1) fournit une équation $f(u,v) = 0$ de degré 4x4, qui constitue une équation cartésienne approchée de $\mathcal{A}$ valable au voisinage de $\alpha(t^*)$, donc en particulier en $\alpha(t^*)$. De cette équation, on peut déduire $\alpha'(t^*)$ comme dans la méthode de Farouki (cf III.3.2). Cependant il n'est pas possible de calculer $\alpha''(t^*)$ de façon exacte car cela exige le calcul des dérivées partielles de f(u,v) à l'ordre 3 or ces quantités ne correspondent pas à la courbe réelle $\mathcal{A}$ puisque l'équation est issue de développements limités à l'ordre 2.
En échangeant les rôles de $\mathcal{S}_1$ et $\mathcal{S}_2$, on obtient $\beta'(t^*)$, puis finalement $\gamma'(t^*)$.

IV.4.3 Calcul du pas et de l'isoparamétrique fixée pour le prochain point

Le calcul d'un point $\mathbf{P}_{i+1}$ sur $\mathcal{C}$ en fonction du point précédent $\mathbf{P}_i$ nécessite (cf IV.4.1) une solution approchée ($\underline{u}_{i+1}$, $\underline{v}_{i+1}$, $\underline{r}_{i+1}$, $\underline{s}_{i+1}$) et le choix d'un réseau d'isoparamétriques parmi les quatre possibles ($u_{fixé}$, $v_{fixé}$, $r_{fixé}$, $s_{fixé}$). L'une des quatre valeurs approchées doit donc fixer une isoparamétrique.

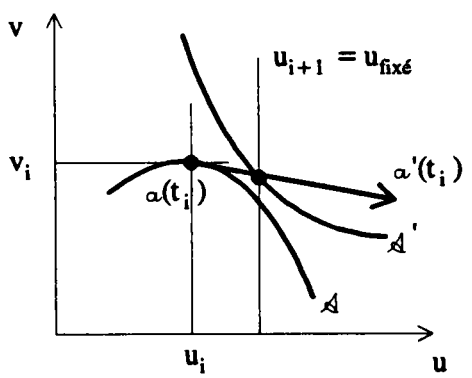
Données mathématiques

Pour déterminer ces deux éléments on dispose des informations suivantes associées à $\mathbf{P}_i$:

$\alpha(t_i) = (u_i , v_i)$, $\alpha'(t_i)$, $\alpha''(t_i)$
$\beta(t_i) = (r_i , s_i)$, $\beta'(t_i)$, $\beta''(t_i)$
$\gamma(t_i) = \mathbf{P}_i$, $\gamma'(t_i)$, $\gamma''(t_i)$
et des courbures et vecteurs normaux qui en sont déduits (cf II.3.3.2).

Contraintes

Le choix réalisé doit respecter plusieurs contraintes.
Le nombre d'itérations nécessaires à la méthode de Newton est d'autant plus faible que les solutions approchées $\underline{\alpha}_{i+1} = (\underline{u}_{i+1} , \underline{v}_{i+1})$ et $\underline{\beta}_{i+1} = (\underline{r}_{i+1} , \underline{s}_{i+1})$ sont proches de $\mathcal{A}$ et $\mathcal{B}$.
Une meilleure précision de $\underline{\alpha}_{i+1}$ et $\underline{\beta}_{i+1}$ accélère donc l'algorithme de suivi; elle diminue également le risque d'erreur.
La solution approchée pour $\mathbf{P}_{i+1}$ doit être assez proche de $\mathbf{P}_i$ pour que l'extrapolation sur la direction de la courbe au voisinage de $\mathbf{P}_i$ soit encore valable.
Une isoparamétrique choisie trop loin de $\mathbf{P}_i$ peut conduire à des erreurs comme celle-ci:



Bien entendu le volume de calculs engendré doit être faible afin de ne pas perdre le bénéfice de la rapidité de la méthode de Newton.

Cette prévision est réalisée en effectuant une extrapolation linéaire des courbes planes $\mathcal{A}$ et $\mathcal{B}$. Markot et Magedson [MAR 91] décrivent une méthode basée sur la courbe spatiale $\mathcal{C}$.
Nous décrivons ces méthodes et discutons l'intérêt des différentes solutions.

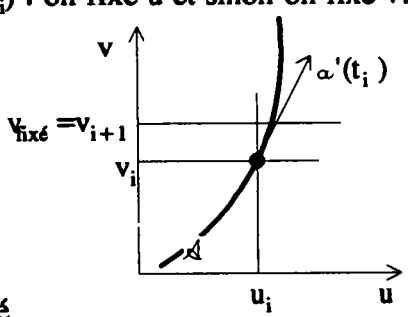
IV.4.3.1 Extrapolation linéaire dans le plan

Quel réseau d'isoparamétriques fixer?

Une première sélection est faite entre u-v et r-s ce qui correspond à réaliser les prévisions avec $\mathcal{A}$ ou $\mathcal{B}$.
De ces deux courbes c'est la plus régulière qui fournit les prévisions les plus fiables.
Or la plus régulière est celle qui a le comportement le plus linéaire possible, c'est donc celle qui a la plus faible courbure ou bien la "vitesse instantanée" la plus élevée.

Le critère retenu est donc:
si $\| \alpha'(t_i) \| > \| \beta'(t_i) \|$ on choisit $\mathcal{A}$, sinon on choisit $\mathcal{B}$.

Supposons que $\mathcal{A}$ soit retenue.
Parmi les deux réseaux d'isoparamétriques encore possibles, on choisit celui qui est le plus perpendiculaire à $\mathcal{A}$ en $\alpha(t_i)$. C'est celui qui respecte le mieux les contraintes.
Donc si $| u'(t_i) | > | v'(t_i) |$ on fixe u et sinon on fixe v.

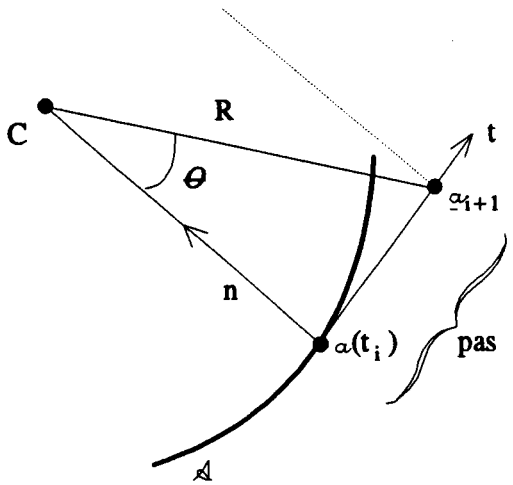


Supposons que v soit fixé

Détermination de α_{i+1}

Soit t le vecteur tangent unitaire orienté en $\alpha(t_i)$.
α_{i+1} est choisi par extrapolation linéaire: $\alpha_{i+1} - \alpha(t_i) =$ pas . t
La courbe isoparamétrique fixée est donc $v_{fixé} = v_{i+1}$
Calcul du pas
Pour que l'estimation précédente soit valable, il faut que le pas soit petit par rapport au rayon de courbure en $\alpha(t_i)$, c'est à dire que la variation angulaire (par rapport au cercle tangent) soit faible.
On fixe donc la variation angulaire θ à une petite valeur $\pi/8$, $\pi/16$, etc...
On en déduit pas = R sin θ

ou bien pas $= \frac{\sin \theta}{\rho}$



C centre de courbure
ρ courbure en $\alpha(t_i)$
$R = \frac{1}{\rho}$ rayon de courbure
n vecteur normal

Cependant pour assurer le contrôle de l'algorithme il faut fixer les bornes de variation du pas pas_min et pas_max, l'expression du pas est donc finalement:

$$\text{pas} = \frac{\sin \theta}{\rho + k} + \text{pas\_min} \quad \text{avec } k = \frac{\sin \theta}{\text{pas\_max} - \text{pas\_min}}$$

Choix de (r_{i+1}, s_{i+1})

Il n'est pas intéressant de calculer un pas sur la courbe $\mathcal{B}$. En effet celui-ci dans le cas général ne correspond pas à celui calculé sur $\mathcal{A}$ car les "unités de distance" sur les deux courbes sont différentes. Il est tout aussi difficile de déduire un pas sur $\mathcal{B}$ à partir de celui calculé sur $\mathcal{A}$ (cela nécessiterait d'établir des rapports entre les unités de distance instantanée sur $\mathcal{A}$, $\mathcal{B}$ et $\mathcal{C}$).
La valeur choisie est donc simplement (r_i, s_i).

Remarques
Si l'isoparamétrique fixée n'appartient pas au carreau paramétrique, il est probable que la courbe $\mathcal{C}$ sorte du carreau $\mathcal{S}_1$ entre $\mathbf{P}_i$ et $\mathbf{P}_{i+1}$. L'isoparamétrique qui a été calculée est donc remplacée par la courbe frontière traversée, ainsi le point suivant sur $\mathcal{C}$ appartient au bord de $\mathcal{S}_1$ et l'extrémité de la courbe est donc parfaitement déterminée.
Lorsque les plans tangents de $\mathcal{S}_1$ et $\mathcal{S}_2$ en $\mathbf{P}_i$ tendent à devenir parallèles, les courbes $\mathcal{A}$ et $\mathcal{B}$ peuvent présenter de grosses variations de courbure (cf même des points de rebroussement). Il est donc intéressant de modifier l'expression du pas par le facteur multiplicatif $\sin(\text{angle}(\mathbf{n}_1, \mathbf{n}_2)) = \| \mathbf{n}_1 \wedge \mathbf{n}_2 \|$.

Conclusion

La forme de cette démarche a été établie après plusieurs "essais" sur machine. Elle donne de bons résultats sauf si $\mathcal{A}$ et $\mathcal{B}$ présentent simultanément un point où la courbure est très forte: en ce cas l'algorithme de suivi "rebrousse chemin" (le point se comporte comme un point de rebroussement).
La densité des points calculés dépend de la courbure; si celle ci est importante il y a accumulation de points. Des points redondants doivent être éliminés avant la détermination d'une courbe d'interpolation qui approche $\mathcal{C}$. La vitesse de l'algorithme est également pénalisée.
Ces problèmes pourraient être réduits en allongeant le pas. Mais pour conserver la même précision, il faut utiliser un développement limité d'ordre supérieur (2 ou 3). Puisqu'une cubique peut admettre un point de rebroussement, il est possible qu'une extrapolation cubique puisse surmonter l'obstacle constitué par un point où la courbure est très importante. Cependant cette solution implique des calculs supplémentaires, les dérivées partielles d'ordre 3 de **X** et **Y**. Bien sûr le problème de la correspondance des pas choisis sur $\mathcal{A}$ et $\mathcal{B}$ se pose encore.

IV.4.3.2 Extrapolation dans l'espace

Markot et Magedson [MAR 89] proposent pour résoudre $\mathbf{X}(u,v) = \mathbf{Y}(r,s)$ (1) de rajouter une quatrième équation plutôt que de fixer une variable. Pour cela ils choisissent un plan $\mathcal{P}$ orthogonal à la courbe $\mathcal{C}$ au point $\mathbf{P}_i$. Ce plan doit être assez proche de $\mathbf{P}_i$, d'où la détermination d'un pas. En combinant l'équation du plan et l'équation paramétrique de $\mathcal{D}_1$ on obtient une équation $F(u,v) = 0$ (2) . Le quadruplet (u_{i+1} , v_{i+1} , r_{i+1} , s_{i+1}) recherché est celui qui vérifie (1) et (2).
Le système est bien sûr résolu par la méthode de Newton.

Cette méthode s'affranchit de l'étape de sélection d'un réseau d'isoparamétriques et l'algorithme conséquent est simplifié.Cependant si la courbe $\mathcal{C}$ est irrégulière, on rencontre les mêmes problèmes que dans le cas précédent. Pour le quadruplet-solution approchée il n'est pas aisé d'obtenir un choix plus précis que (u_i , v_i , r_i , s_i) car les rapports entre les unités de distance sur $\mathcal{A}$, $\mathcal{B}$ et $\mathcal{C}$ sont approximatifs.

Conclusion

Les choix retenus pour résoudre le problème posé sont discutables car heuristiques et non pas simplement calculatoires.
Quelque soit la méthode choisie, le pas est nécessairement issu d'un compromis. Trop grand il favorise les erreurs et ralentit la convergence de la méthode de Newton et trop petit il accroît la densité des points calculés, donc ralentit l'algorithme et augmente les "redondances" de points.
La garantie du bon déroulement de l'algorithme est liée à la régularité des courbes $\mathcal{A}$, $\mathcal{B}$ et $\mathcal{C}$.
Les meilleurs résultats sont sûrement obtenus en choisissant pour chaque nouvelle extrapolation, la courbe la plus régulière parmi les trois.

IV.4.4 Contrôle de l'algorithme et test d'arrêt

L'algorithme de suivi n'est pas déterministe. Il n'est pas possible de déduire d'après le dernier point calculé, le comportement ultérieur de la courbe. En particulier plusieurs questions se posent: le calcul du prochain point va -t-il aboutir? s'il aboutit, le résultat est il valide? quand l'algorithme doit il s'arrêter?
Si l'arc ne touche aucun bord de $\mathcal{D}_1$ ou de $\mathcal{D}_2$, il est nécessairement fermé et l'algorithme doit s'arrêter dès que la boucle est complète, sinon l'arc coupe un bord et l'algorithme doit s'arrêter sur le point limite. Nous décrivons l'étape de contrôle du résultat de la méthode de Newton puis les deux critères d'arrêt.

Contrôle de la validité du dernier point.

Le dernier point trouvé $\mathbf{P}_i$ n'est pas valide dans deux cas:
- Il appartient à un autre arc que $\mathcal{C}$.
- Il est situé trop loin de $\mathbf{P}_{i-1}$ par rapport à l'estimation qui a été faite, cela signifie que $\mathcal{C}$ a un comportement inattendu entre $\mathbf{P}_{i-1}$ et $\mathbf{P}_i$, qui n' a pas été détecté.

Le seul moyen de détecter ces erreurs est de comparer la distance trouvée entre (u_i, v_i) et (u_{i-1}, v_{i-1}) et le pas calculé (si la dernière estimation a été faite avec $\mathcal{A}$). Un critère basé sur la mesure de la différence de ces deux quantités permet donc d'accepter ou de refuser $\mathbf{P}_i$. Un test plus précis pourrait mesurer la distance entre α_i et $\alpha(t_i)$.

Si le dernier point n'est pas accepté la méthode de Newton est reconduite avec un pas plus petit. Plus généralement si le calcul du dernier point échoue, on procède identiquement. Un test plus exigeant augmente la densité des points calculés mais détecte mieux les anomalies. De façon générale le test est issu d'un compromis entre rapidité de l'algorithme et garantie de la validité de la courbe obtenue.

Critère d'arrêt: l'arc sort d'un des deux carreaux

Cela signifie que l'avant dernier point $\mathbf{P}_{i-1}$ appartient encore aux deux carreaux et que le dernier point $\mathbf{P}_i$ a dépassé l'un des deux. Cette éventualité est détectée en testant l'appartenance de (u_i, v_i) au domaine de définition de **X** et l'appartenance de (r_i, s_i) au domaine de définition de **Y**. L'un des deux segments $[\alpha(t_{i-1}), \alpha(t_i)]$, $[\beta(t_{i-1}), \beta(t_i)]$ coupe nécessairement l'isoparamétrique correspondant au bord coupé. Cette isoparamétrique est utilisée par la méthode de Newton pour calculer précisément l'extrémité de $\mathcal{C}$. Celle ci est substituée au dernier point dans la liste des points trouvés.

Critère d'arrêt: l'arc est fermé et ne touche aucun bord

L'arc est une courbe fermée. Par conséquent l'algorithme doit s'arrêter lorsque la boucle est entièrement parcourue et le dernier point doit être confondu avec le point initial. L'étude est faite simultanément sur les deux courbes $\mathcal{A}$ et $\mathcal{B}$. L'algorithme doit s'arrêter dès que l'une des deux forme une boucle complète (il se peut très bien que l'une soit fermée et que l'autre coupe les bords de son carreau).

Etudions l'arc $\mathcal{A}$.

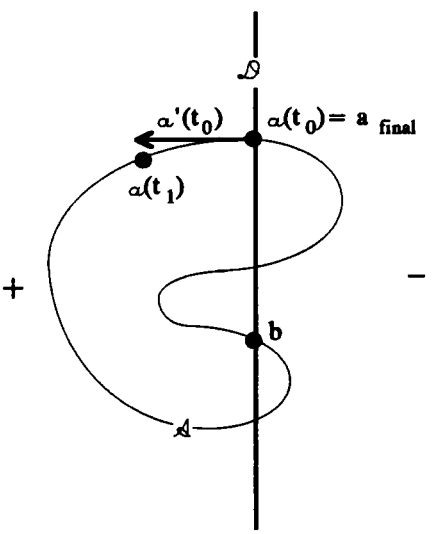
Une première approche est basée sur le fait que le vecteur tangent, au cours du parcours de la boucle, décrit nécessairement un angle multiple de 2π; il est malheureusement difficile d'en déduire un algorithme efficace.

Pour mesurer la variation angulaire du vecteur tangent, on calcule la somme des angles orientés $(\alpha'(t_{i-1}), \alpha('(t_i))$ entre deux vecteurs tangents successifs ou encore la somme des angles orientés entre deux segments successifs. Cette somme est mise à jour pour chaque nouveau point calculé. L'algorithme doit s'arrêter lorsque la somme "franchit" $+2\pi$ ou -2π. Malheureusement ce critère d'arrêt est peu précis. Le dernier point calculé est une approximation grossière (et même très éloignée si la courbe est linéaire au voisinage) de l'extrémité exacte et il n'y a aucun moyen de le préciser d'avantage.

Cet algorithme a été abandonné au profit d'une seconde approche beaucoup plus efficace.

Celle-ci consiste à calculer une droite $\mathcal{D}$ coupant la courbe fermée $\mathcal{A}$ en deux parties puis en classant les points successifs par rapport à $\mathcal{D}$.
Cette dernière est la droite passant par le point initial $\alpha(t_0)$ et qui est orthogonale à $\mathcal{A}$ en ce point; son équation $D(u,v) = 0$ est déterminée de telle sorte que $D(u(t_1),v(t_1)) > 0$.

Pour chaque point (u_i, v_i) calculé sur $\mathcal{A}$ on calcule le signe de $D(u_i,v_i)$.
Tant que l'arc reconstruit n'a pas fait une demi-boucle, le signe des points calculés est positif, puis dès que l'arc achève une demi-boucle, le signe des points change. Une éventuelle fin de boucle est détectée dès que le signe des points redevient positif. L'algorithme s'arrête donc dès que le signe associé au dernier point $\alpha(t_n)$ est positif alors que le signe du point précédent $\alpha(t_{n-1})$ est négatif.



Un processus dichotomique est alors engagé sur la portion de $\mathcal{A}$: $[\alpha(t_{n-1}), \alpha(t_n)]$ pour déterminer avec précision le point d'intersection avec $\mathcal{D}$. Ce point (noté **a**) est ensuite comparé à $\alpha(t_0)$ (les positions sont comparées et éventuellement les vecteurs tangents). Le critère final est donc:
Si **a** et $\alpha(t_0)$ sont confondus l'algorithme s'arrête et le point **a** est substitué à $\alpha(t_n)$ dans la liste des points.
Si **a** et $\alpha(t_0)$ sont distincts, l'algorithme repart à partir du point $\alpha(t_n)$.
L'extrémité finale est calculée avec une excellente précision numérique, ce qui rend le critère final très fiable. La précision du calcul dichotomique est due au fait que la droite et la courbe se coupent à angle droit en $\alpha(t_0)$.
Cependant dans le cas particulier où le point initial est un point singulier, la méthode peut échouer.

Les critères d'arrêt proposés garantissent que les extrémités des courbes sont très précises. Cette qualité est utile si plusieurs arcs de courbe obtenus par cette méthode doivent être recollés. Les techniques décrites sont facilitées par le fait que le système $X(u,v) = Y(r,s)$ est résolu en fixant une inconnue plutôt qu'en ajoutant une équation.

IV.4.5 Conclusion

L'algorithme complet est clairement résumé par un simple schéma itératif:

Données
Quadruplet-solution approchée et choix d'une isoparamétrique pour le calcul du point - germe $\mathbf{P}_o$

Initialisation
Calcul de $\mathbf{P}_o$
Calcul des informations géométriques liées à $\mathbf{P}_o$
Détermination des droites $\mathcal{D}$ et $\mathcal{D}'$ servant à contrôler si $\mathcal{A}$ et $\mathcal{B}$ forment des boucles

Algorithme

Tant que le calcul du dernier point $\mathbf{P}_i$ a abouti à un résultat valide
et $\mathbf{P}_i$ ne sort pas de δ_1 *et* $\mathbf{P}_i$ ne sort pas de δ_2
et la courbe $\mathcal{A}$ ne forme pas une boucle complète
et la courbe $\mathcal{B}$ ne forme pas une boucle complète
faire
Calcul de la solution approchée et de l'isoparamétrique fixée pour le calcul de $\mathbf{P}_{i+1}$
Calcul de $\mathbf{P}_{i+1}$
Calcul des informations géométriques liées à $\mathbf{P}_{i+1}$
Calcul de la position de $\mathbf{P}_{i+1}$ par rapport aux droites $\mathcal{D}$ et $\mathcal{D}'$ et éventuellement remplacer $\mathbf{P}_{i+1}$ par l'extrémité possible
$\mathbf{P}_i \longleftarrow \mathbf{P}_{i+1}$
Fin tant_que

Résultat
La liste des points trouvés.
Pour chaque point sont conservés les coordonnées locales, les vecteurs dérivée $1^{ère}$, $2^{ème}$, etc ... ainsi que la valeur paramétrique t_i estimée.

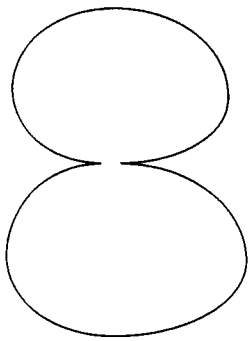
L'intérêt de cette méthode est multiple.
Son domaine d'application ne se limite pas aux seuls carreaux NURBS. Elle peut traiter toutes les surfaces paramétriques dont il est facile d'extraire les dérivées partielles. Il est

même montré qu'elle s'applique aux surfaces implicites. Bajaj, Hoffmann, Lynch et Hopcroft [BAJ 88] démontrent qu'il est possible de traiter de manière uniforme les cas intersection carreau paramétrique-carreau paramétrique et intersection surface implicite-surface implicite par cette méthode.
Elle utilise au mieux possible la méthode de Newton. Cette dernière est utilisée pour préciser une solution approchée déduite du point précédent. Ce processus minimise le volume de calculs nécessaire au calcul d'un nouveau point. En fait la vitesse de l'algorithme est essentiellement liée à la complexité des calculs de dérivées partielles.

Les étapes-clés de l'algorithme: méthode de Newton, calcul de vecteurs dérivées successives sont simples. Elles contribuent à simplifier sa mise en oeuvre cependant l'algorithme est encombré de nombreux cas particuliers fastidieux à traiter.

Les limites de la méthode sont bien connues: Connaissance d'un point initial et présence de points singuliers.
La solution approchée du point initial doit provenir d'une autre méthode. Par conséquent l'algorithme de suivi doit être précédé d'un prétraitement qui localise le point-germe (cf IV.5.2).
Les points singuliers sont assez mal traités par l'algorithme.
Au voisinage d'un tel point la convergence de la méthode de Newton est plus lente et perd de la précision, il devient aussi plus difficile de calculer les dérivées successives de la courbe.
C'est pourquoi il est préférable d'éviter ces points. Lorsque la convergence est trop lente, il suffit d'interrompre le processus et de recommencer avec une courbe isoparamétrique un peu déplacée par rapport à la précédente (obtenue par diminution du pas).
Cependant l'algorithme est incapable de détecter les deux différents arcs qui se croisent au point singulier, une seule parmi les trois branches est suivie. Cette ambigüité peut provoquer une incohérence topologique (cf IV.5.2)

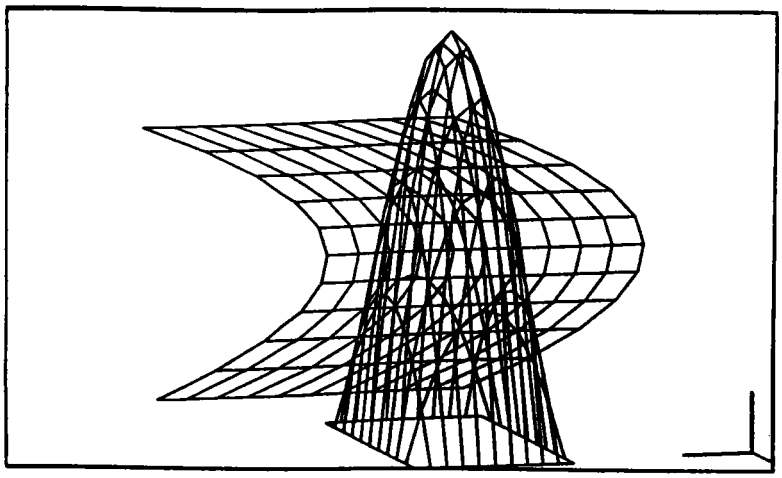
Lorsqu'une courbe admet "presque" un point singulier, l'algorithme se trouve également en difficulté. La courbe, au point critique possède une très forte courbure et l'extrapolation linéaire s'avère insuffisante, le processus peut alors rebrousser chemin.
Il devient nécessaire d'utiliser une extrapolation quadratique ou cubique. Cette dernière possibilité a été développée dans [BAJ 88] et dans [MON 86].



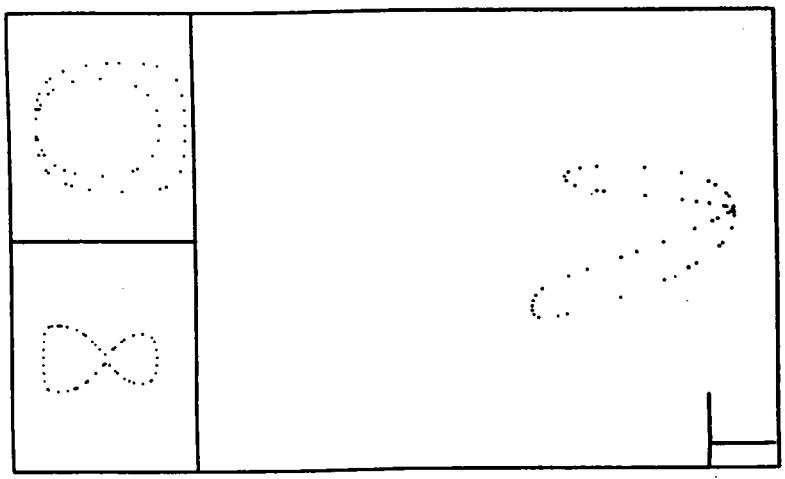
Exemple de point "presque singulier"

La méthode de suivi apparaît comme le moyen le plus économique de reconstruction d'un arc de courbe connexe, ne présentant aucun point singulier.
Les limites de la méthode montrent qu'elle ne peut être utilisée seule, elle doit être initialisée et contrôlée par un autre procédé.

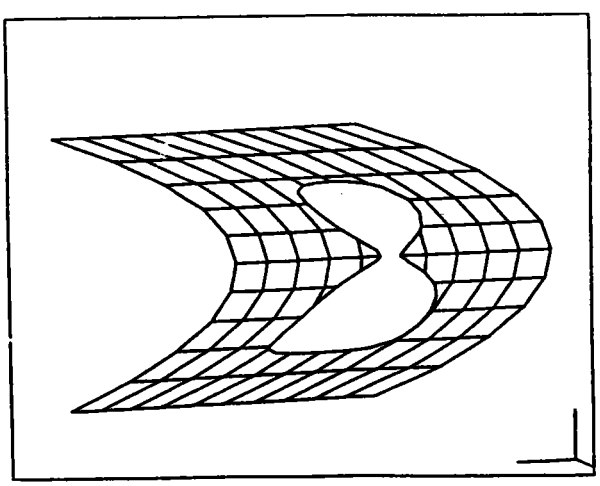
IV.4.6 Exemples

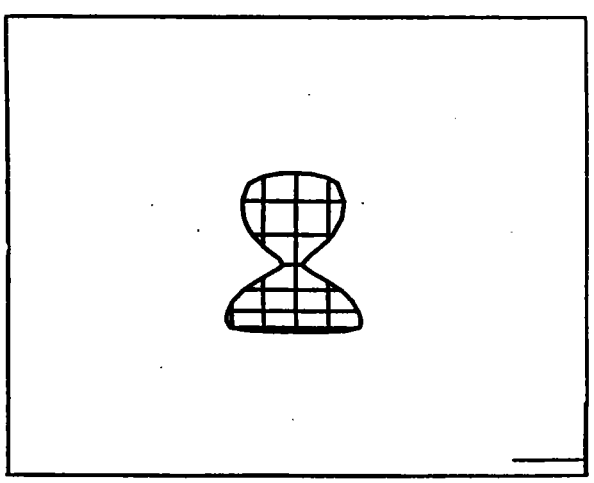


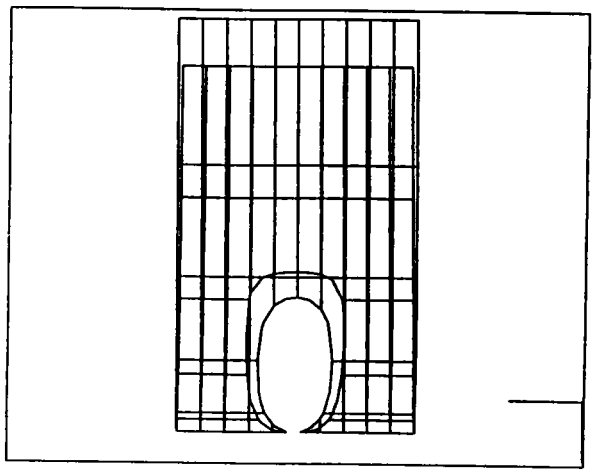
Courbe d'intersection et ses projections dans les domaines de définition des deux carreaux

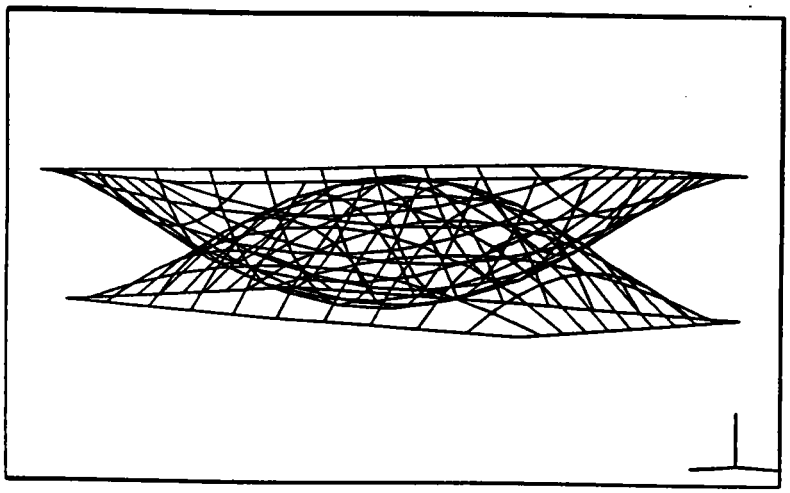


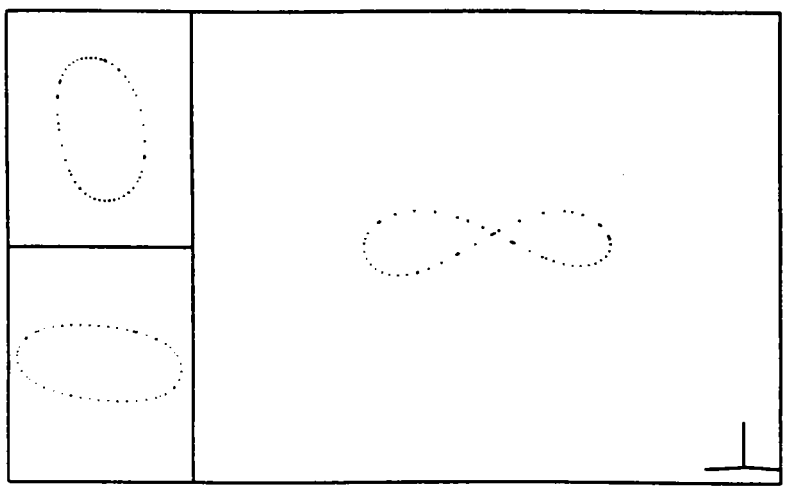
Les deux carreaux restreints déduits du second carreau

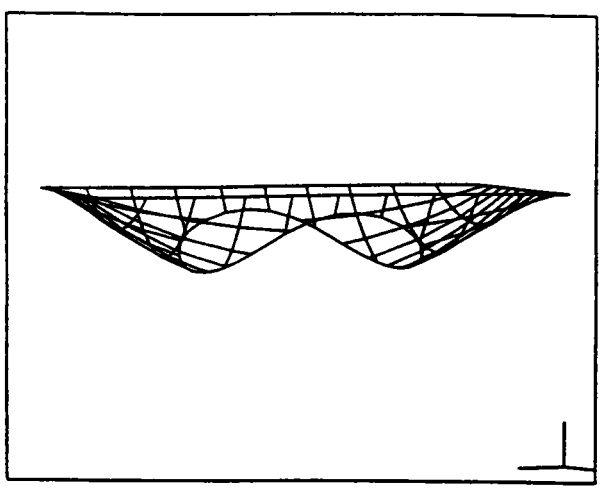












IV.5 SYNTHESE

Les techniques précédentes, étudiées séparément, ne fournissent que des solutions incomplètes; associées elles constituent un algorithme général très efficace.

La méthode de suivi de courbe est le moyen le plus rationnel pour reconstruire une courbe d'intersection connexe. Cependant pour garantir que toutes les courbes disjointes sont parcourues, il faut au moins un point germe par courbe connexe. Si ce point n'est pas unique il y a risque de parcourir plusieurs fois la même courbe. Or, sans la coopération d'un opérateur humain, aucune méthode de détection de points germes ne garantit l'unicité du point germe par courbe connexe. Le processus qui supervise l'algorithme de suivi de courbe doit donc détecter toutes les courbes disjointes et contrôler qu'elles sont parcourues une fois et une seule. Ce rôle est assuré par la synergie Algorithme de simplification des carreaux NURBS / Résolution algébrique de l'intersection courbe-carreau.
L'algorithme général comporte donc trois phases consécutives:
- Application de l'algorithme de simplification des carreaux NURBS (cf IV.1)
- Calculs de points germes à l'aide de la méthode de résolution algébrique du problème: intersection courbe de Bézier rationnelle quadratique - carreau de Bézier rationnel biquadratique.
- Reconstitution des courbes par la méthode de suivi de courbe.

Chacune de ces opérations a longuement été exposée il ne reste qu'à décrire l'organisation globale.

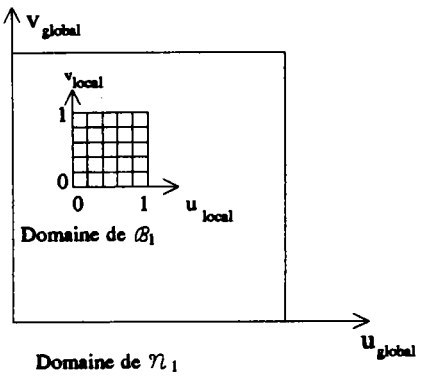
IV.5.1 Description de l'algorithme complet

Notons $\mathcal{N}_1$ et $\mathcal{N}_2$ les deux carreaux NURBS dont on recherche la courbe d'intersection $\mathcal{C}$ (non connexe dans le cas général).
$\mathcal{N}_1$ et $\mathcal{N}_2$ sont traités par l'algorithme de simplification. Celui-ci rend une liste de couples $(\mathcal{B}_1, \mathcal{B}_2)$ de sous carreaux appartenant respectivement à $\mathcal{N}_1$ et $\mathcal{N}_2$. Rappelons que les sous carreaux sont des carreaux de Bézier rationnels biquadratiques et que seuls les couples de sous carreaux dont les volumes englobants sont sécants sont retenus.
Pour chaque couple $(\mathcal{B}_1 , \mathcal{B}_2)$ obtenu sont reconstituées les portions de $\mathcal{C}$ incluses dans $\mathcal{B}_1 \cap \mathcal{B}_2$.
Cette reconstitution s'opère en deux étapes comme suit.

Etape 1 : Détermination des points germes

Chacun des deux sous carreaux est quadrillé par un réseau d'isoparamétriques comprenant les courbes frontières.

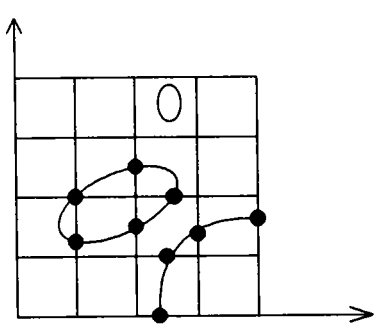
Pour chaque courbe frontière on calcule les points d'intersection avec le carreau adverse avec la méthode de résolution algébrique (cf IV.3.1)



Quadrillage de $\mathcal{B}_1$

Les points obtenus appartiennent à l'intersection de carreaux qui ne sont que des approximations de sous carreaux de $\mathcal{N}_1$ et $\mathcal{N}_2$, par conséquent ils ne sont que des approximations des points d'intersections exacts.
Ils sont donc raffinés avec la méthode de Newton; celle ci permet aussi d'éliminer les éventuelles solutions qui n'ont pas de signification géométrique.

Notons L la liste de tous les points obtenus. Toute portion de $\mathcal{C}$ coupant un bord de l'un des deux carreaux passe par un de ces points.
La détection des courbes fermées ne coupant aucun bord exige le calcul des points d'intersection entre les autres isoparamétriques prises sur $\mathcal{B}_1$ avec le carreau $\mathcal{B}_2$ et réciproquement. Ces points additionnels, après raffinement sont placés dans une liste L_1.



Calcul de points germes sur les différents arcs de courbe.

Toujours afin d'éviter de parcourir plusieurs fois la même courbe, il faut contrôler que chaque point n'apparaît qu'une seule fois dans L U L_1.

Etape2 : Construction des arcs de courbe

Chaque point de L sert à lancer un algorithme de suivi de courbe. Cet algorithme est arrêté temporairement au franchissement d'une isoparamétrique intérieure. Le point d'intersection de l'arc en cours de construction avec celle ci est retiré de la liste L_1. Le suivi de courbe est achevé lorsque l'arc forme une boucle complète ou bien lorsqu'il atteint une courbe frontière. Le point initial et le point final sont retirés de L.
Les reconstitutions d'arc se poursuivent jusqu'à ce que L soit vide.
Les points résiduels de L_1 appartiennent à des courbes fermées ne touchant aucune frontière de $\mathcal{B}_1$ ou $\mathcal{B}_2$. Chacun de ces points est utilisé pour initialiser un processus de suivi de courbe qui s'achève lorsque la boucle est complète. Les points d'intersection de la boucle avec les isoparamétriques des quadrillages ainsi que le point initial sont retirés de L_1.
Lorsque L_1 est vide toutes les courbes sont reconstruites.

Les portions de $\mathcal{C}$ provenant de tous les couples ($\mathcal{B}_1$, $\mathcal{B}_2$) sont finalement assemblées à l'aide de critères de continuité.

IV.5.2 Intérêt de l'algorithme proposé

L'aptitude de l'algorithme à résoudre le problème posé peut être mesuré à l'aide de deux critères: la qualité du résultat et le volume de calcul induit.
La qualité du résultat peut être définie en plusieurs points: identification de toutes les courbes disjointes, précision de celles ci et cohérence topologique.

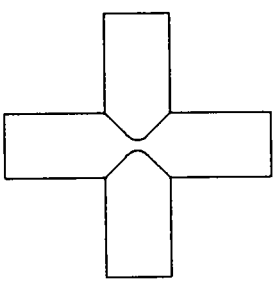
Analysons la qualité de la solution.
Les courbes qui échappent à la détection sont les courbes fermées qui "passent à travers" les mailles des deux quadrillages simultanément. Il est possible de faire varier la densité du quadrillage de façon à fixer la taille maximale d'une maille. Par conséquent il est possible de garantir que toutes les courbes fermées de taille supérieure à une certaine tolérance sont détectées.
Ce problème est inhérent à tous les algorithmes qui décomposent les surfaces et recherchent des points-germes pour initialiser des processus de suivi de courbe (cf [TIM 85], etc ...).

Remarquons que si le quadrillage est suffisamment resserré il n'est plus nécessaire d'utiliser la méthode de suivi. Les points germes calculés sont assez denses pour représenter les courbes d'intersection. Les points sont simplement reliés entre eux après raffinement et calcul de vecteurs tangents, vecteurs normaux et courbures. Dans ce cas limite, la méthode présentée est donc identique à celle exposée en IV.3, le raffinement des point est la seule différence.

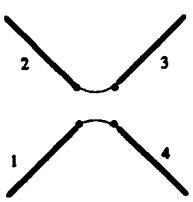
La précision des points calculés sur la courbe est celle de la méthode de Newton qui est excellente sauf peut être au voisinage des points singuliers.

Le problème de la cohérence topologique se pose au moment de relier les points d'intersection entre eux pour former les courbes. Les points sont reliés à l'aide de critères de continuité (de position, de direction, de courbure, ...). Ces critères sont implicitement pris en compte par la méthode de suivi et explicitement lors du recollement final des différents brins de $\mathcal{C}$. La fiabilité de ces critères, très bonne lorsque les points sont réguliers diminue au voisinage des points singuliers ou quasi-singuliers. Les points singuliers sont non seulement difficiles à localiser mais peuvent provoquer des incohérences topologiques. Illustrons ce dernier problème par un exemple:

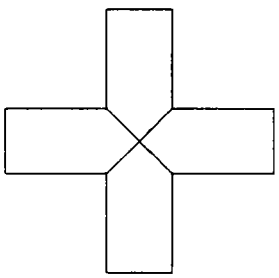
Soient deux cylindres sécants admettant presque un point singulier.



Supposons que les brins 1, 2, 3, 4 ont étés construits à l'aide de la méthode de suivi et qu'ils doivent être reliés. Les seules liaisons correctes sont 1 - 4 et 2 - 3.
L'imprécision des calculs et la proximité des quatre extrémités rend le choix incertain.

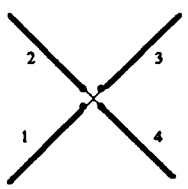


Le risque d'erreur augmente si les cylindres sont réellement tangents.



Dans ce cas les liaisons possibles entre les quatre brins sont

$\begin{cases} 1\text{-}2 \\ 3\text{-}4 \end{cases}$ ou $\begin{cases} 1\text{-}4 \\ 2\text{-}3 \end{cases}$.



Toute autre combinaison est absurde.

Sans **la localisation du point singulier et la détermination des deux directions de vecteurs tangents en ce point**, il est difficile d'éliminer l'ambigüité et de raccorder correctement les brins entre eux. Ce problème est universel; il est rencontré quelque soit la méthode employée.
L'algorithme exposé ne traite pas le cas particulier des points singuliers; il est cependant clair qu'une parfaite efficacité passe par la détection et l'analyse des points singuliers. A cette fin plusieurs solutions sont envisagées. Elles sont évoquées dans le chapitre suivant.

IV.6 CONCLUSION

Les principes des trois démarches classiques, de la plus simple, "dichotomie récursive" à la plus raffinée, "Méthode de Farouki", ont été approfondis. Leur étude a abouti à la mise au point de nouveaux algorithmes qui apportent des éléments de solution au problème général.
La première de ces techniques constitue un algorithme de prétraitement au calcul de l'intersection de deux carreaux NURBS. Elle généralise la méthode "Dichotomie récursive et intersection de plans".
Les deux carreaux sont simultanément décomposés récursivement en sous-carreaux de plus en plus petits; seuls les sous-carreaux dont les volumes englobants sont sécants sont conservés. La régularité croissante des sous-carreaux successifs permet d'approcher ceux-ci

par des carreaux de Bézier de degré de plus en plus faible et même par des carreaux non rationnels. La décomposition s'achève lorsque les sous-carreaux sont rationnels biquadratiques car ceux-ci sont suffisamment simples pour être soumis à une méthode de résolution algébrique. L'algorithme fournit donc une liste de couples de sous-carreaux sécants simplifiés. L'implantation de cette méthode a permis de vérifier que les erreurs d'approximation sont parfaitement contrôlées ainsi que la validité des approximations.

Le problème d'intersection est mieux formalisé lorsqu'une des surfaces est implicite (connue par une équation cartésienne) et l'autre est définie par une équation biparamétrique. La détermination d'une équation cartésienne exacte pour un carreau de Bézier est conceptuellement possible mais irréalisable pratiquement sauf pour les carreaux biquadratiques.
Nous avons recherché, pour les carreaux biquadratiques, une équation cartésienne simplifiée et par conséquent approchée. Malheureusement une réalisation a montré qu'il était impossible de trouver une équation de faible degré satisfaisante.

Il est beaucoup plus simple d'obtenir l'équation cartésienne d'une courbe de Bézier rationnelle quadratique ou cubique. La première se situe sur l'intersection d'une quadrique et d'un plan et la seconde est située sur l'intersection de deux quadriques. Ces propriétés sont utilisées pour résoudre le problème intersection courbe m_ique-carreau bi_m_ique (m = 2, 3).
La méthode consiste à combiner l'équation cartésienne de la courbe et l'équation biparamétrique du carreau; le système non linéaire obtenu est résolu par la méthode des résolvants. Cette méthode est appliquée, par le biais des isoparamétriques, à la résolution du problème intersection carreau bi_m_ique-carreau bi_m_ique; elle a été mise en oeuvre dans le cas quadratique.

Enfin nous avons étudié, puis implanté la méthode de "suivi de courbe" qui consiste à calculer des points de proche en proche le long d'une courbe d'intersection connexe.
En chaque point sont calculées des informations géométriques qui traduisent le comportement de la courbe au voisinage du point. Une valeur approchée pour le prochain point sur la courbe en est déduite. Ce point est ensuite raffiné par la méthode de Newton.

Les trois techniques développées sont complémentaires, elles ont été combinées pour concevoir un algorithme très efficace. La méthode de suivi constitue le moyen le plus économique pour construire une courbe connexe, aussi elle a été retenue pour cette tâche.

L'organisation des reconstructions ainsi que l'initialisation sont assurées par l'association des deux autres techniques.

Nous pouvons maintenant discuter des intérêts respectifs de ces techniques et les replacer par rapport aux méthodes existantes.

V BILAN ET PERSPECTIVES

Après un résumé des techniques développées, nous faisons le point sur les manières d'aborder le problème de l'intersection des carreaux NURBS. En particulier nous évoquons les difficultés majeures qui sont rencontrées et discutons l'apport et l'originalité de nos démarches.

Nous présentons ensuite l'orientation future de nos travaux dans ce domaine.
Enfin l'intersection des carreaux n'étant qu'un des aspects de la modélisation des surfaces nous montrons que nos méthodes peuvent être appliquées à une autre opération importante réalisée sur les carreaux NURBS: le tracé de rayon.

V.1 BILAN

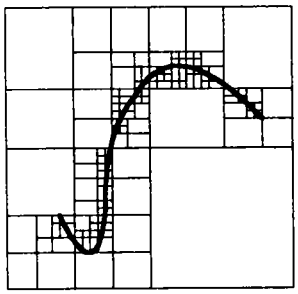
V.1.1 Résumé

La courbe d'intersection (non connexe dans le cas général) entre deux carreaux NURBS est recherchée. Notons $\mathcal{C}$ la courbe et $\mathcal{N}_1$ et $\mathcal{N}_2$ les deux carreaux.

L'algorithme présenté peut être décomposé en quatre étapes.

Etape 1
Les deux carreaux sont décomposés en carreaux simplifiés et la courbe est localisée.

Les deux carreaux sont subdivisés récursivement simultanément. Chacun d'eux génère donc un quadtree. Les premières subdivisions sont effectuées grâce à l'algorithme de Cox-De Boor, puis dès que les sous carreaux sont sous la forme de Bézier, les subdivisions sont poursuivies à l'aide de l'algorithme de De Casteljau. La régularité croissante des sous carreaux de Bézier successifs permet d'approcher ceux-ci par des carreaux de degré de plus en plus faible. Les points de découpage sont calculés de façon à accélérer les diminutions de degré. Le processus s'arrête lorsque les sous carreaux possèdent un carreau de Bézier approximatif biquadratique.
Un filtre optimise le nombre de décompositions utiles. Seuls les couples de sous carreaux adversaires dont les volumes englobants (boite min-max) sont sécants sont subdivisés. Les quadtree sont donc incomplets.



Les sous carreaux par les quels $\mathcal{C}$ ne passe pas ne sont plus décomposés

Le processus rend donc une liste de couples ($\mathcal{B}_1$, $\mathcal{B}_2$) de sous carreaux de $\mathcal{N}_1$ et $\mathcal{N}_2$ respectivement, susceptibles de contenir une partie de $\mathcal{C}$ et pour chacun desquels on possède un carreau de Bézier approximatif biquadratique.

Etape 2
Sur chaque couple ($\mathcal{A}_1$, $\mathcal{A}_2$) de sous carreaux sont calculés des points de passage de $\mathcal{C}$ (points germes). L'objectif étant d'obtenir au moins un point sur chaque courbe connexe restreinte à ($\mathcal{A}_1$, $\mathcal{A}_2$)

Chaque couple ($\mathcal{B}_1$, $\mathcal{B}_2$) obtenu est traité de la façon suivante. $\mathcal{B}_1$ et $\mathcal{B}_2$ sont quadrillés chacun par un réseau d'isoparamétriques. On calcule les points d'intersection de chaque isoparamétrique avec le carreau adverse.
Dans un premier temps ces calculs sont effectués sur les carreaux approximatifs qui sont suffisamment simples pour être soumis à des techniques de résolvant. La méthode consiste à ramener le problème intersection courbe rationnelle quadratique-carreau biquadratique rationnel à la résolution d'une équation réelle de degré 16. Les points trouvés sont très proches des points d'intersection exacts sur ($\mathcal{N}_1$, $\mathcal{N}_2$).
Dans un deuxième temps ces derniers sont donc trouvés à l'aide d'une itération de Newton initialisée grâce aux points approximatifs.

Etape 3
Les portions de $\mathcal{C}$ locales à chaque couple ($\mathcal{A}_1$, $\mathcal{A}_2$) sont reconstruites par une méthode de suivi de courbe.

Sur chaque couple ($\mathcal{B}_1$, $\mathcal{B}_2$):
Les points calculés servent à initialiser des processus de suivi de courbe. Ceux ci reconstituent les portions de courbes restreintes à ($\mathcal{B}_1$, $\mathcal{B}_2$).

La méthode de suivi de courbe consiste à calculer des points de proche en proche le long de la courbe. En chaque point sont calculées des informations géométriques (position, vecteur tangent, vecteur normal, courbure, ...) permettant par continuité d'estimer un point suivant sur la courbe; ce point est ensuite raffiné par la méthode de Newton.

Etape 4
Les portions de $\mathcal{C}$ provenant de tous les couples ($\mathcal{A}_1$, $\mathcal{A}_2$) sont assemblées pour former la courbe complète.

La courbe $\mathcal{C}$ est un puzzle qu'il faut reconstituer à partir des segments de courbe calculés à l'étape précédente. Les extrémités des fragments successifs coïncident avec une excellente précision. Les morceaux peuvent donc être recollés à l'aide de critères de continuité.

Des modifications de cet algorithme sont envisageables.
A l'étape 1 les subdivisions récursives peuvent être stoppées dès que les carreaux sont bicubiques. En effet il est possible de calculer à l'aide de résolvants tous les points d'intersection entre une courbe de Bézier rationnelle cubique et un carreau de Bézier rationnel bicubique. Le problème est ramené à la résolution d'une équation réelle de degré 72.
L'étape 3 peut être omise si les quadrillages sont assez resserrés; les points calculés sur ceux ci sont alors assez denses pour représenter efficacement les courbes.

Par rapport aux méthodes existantes cet algorithme présente une meilleure formalisation et une optimisation de l'étape de détection des points germes.
Les limites de l'algorithme sont connues et classiques. Il ne garantit pas que toutes les courbes sont détectées et la présence de points singuliers perturbent son fonctionnement. En effet les courbes fermées "assez petites" pour passer à travers les mailles des quadrillages sont oubliées et les points singuliers dégradent les performances de la méthode de Newton et peuvent provoquer des incohérences topologiques tant à la construction des courbes qu'au recollement des brins.
Tous ces élément sont repris dans le paragraphe suivant.

V.1.2 Bilan

Les différentes phases du traitement du problème d'intersection sont analysées. Les différentes solutions aux principaux problèmes sont discutées, avec une analyse de l'intérêt et de l'originalité des techniques que nous avons développées.
Le traitement de l'intersection peut être résumé en trois points, la construction des courbes connexes, la détection de celles-ci et le traitement des points singuliers.

Construction des courbes connexes

La méthode de suivi apparaît comme la plus largement employée pour reconstituer un arc de courbe. Elle est utilisée quelle que soit la nature des deux surfaces traitées (implicite-implicite, paramétrique-implicite, paramétrique-paramétrique) pourvu que les définitions soient suffisamment dérivables. Son intérêt essentiel est l'économie; c'est la méthode qui exige le moins de calculs pour obtenir un point supplémentaire sur une courbe car elle utilise de façon optimale la méthode de Newton. Pour un nombre identique de points calculés sur une courbe, le volume de calculs engendré par cette méthode est bien inférieur à celui engendré par une méthode utilisant les résolvants ou encore une méthode qui utilise subdivisions récursives et intersections de plans. De plus les points calculés par la méthode de suivi sont très précis car obtenus avec la méthode de Newton.

Cette méthode offre aussi l'avantage de classer et relier immédiatement les points calculés. Ainsi la courbe est construite au fur et à mesure que les points sont trouvés contrairement aux méthodes qui associent subdivisions récursives et dichotomie où les points sont trouvés dans l'ordre donné par l'arborescence et doivent être reliés explicitement ultérieurement.

Optimiser la méthode passe par une diminution de la densité des points calculés et par l'accélération de la méthode de Newton. Cela nécessite d'allonger le pas et d'augmenter la précision du prochain point estimé sur la courbe, ce qui est paradoxal. Une extrapolation quadratique ou cubique convient donc mieux qu'une extrapolation linéaire pour l'estimation du prochain point. Il est donc intéressant de pouvoir calculer en un point de la courbe les dérivées successives. Nous avons montré qu'en supposant que la courbe spatiale était parcourue par l'abscisse curviligne, il était possible de calculer de façon exacte ces quantités. Ces résultats ont été obtenus de manière différente dans [MAR 89], dans [BAJ 88] et dans [MON 86]. La programmation de cette méthode a permis de vérifier la validité de l'hypothèse de l'abscisse curviligne ainsi que les résultats.

Notons que le paramétrage intrinsèque de la courbe spatiale assure que la distance entre deux points successifs sur la courbe et la différence paramétrique entre eux sont sensiblement égales. Les points calculés peuvent donc être munis de dérivées successives et de valeurs paramétriques. Ces informations sont très utiles si l'on désire ultérieurement éliminer des points "superflus" ou modéliser la courbe d'intersection par des courbes d'interpolation.

La méthode de suivi a cependant des limites.
Pour être initialisé le processus a besoin d'un point germe sur chaque courbe connexe; celui-ci doit être obtenu par un autre moyen.
Les performances de la méthode se dégradent au voisinage d'un point singulier et enfin son contrôle est nécessaire: par exemple, une fois initialisé le processus peut boucler indéfiniment sur une courbe fermée. Nous avons résolu très simplement ce problème et montré qu'il est très facile de vérifier qu'une courbe passe par un point particulier et de stopper le processus au franchissement de ce point.

Détection des courbes connexes

Les approches du problème constitué par la détection des points germes sont très variées.
L'objectif est de trouver au moins un point sur chaque courbe connexe. Les courbes qui coupent une frontière de l'un des carreaux ne posent pas de difficultés; en revanche les courbes fermées sont plus difficiles à localiser.
Farouki propose à cette fin de résoudre les systèmes

$$\begin{cases} f(u,v) = 0 \\ f_u(u,v) = 0 \end{cases} (1) \qquad \text{et} \qquad \begin{cases} f(u,v) = 0 \\ f_v(u,v) = 0 \end{cases} (2)$$ dont les zéros sont les points où les

courbes sont parallèles aux axes du repère. Cette méthode suppose qu'une équation cartésienne $f(u,v) = 0$ de la courbe est connue mais cette dernière provient de la combinaison de l'équation paramétrique de l'un des carreaux et de l'équation cartésienne de l'autre carreau généralement trop complexe pour être calculée.

Toutefois les systèmes $\begin{cases} \mathbf{X}(u,v) - \mathbf{Y}(r,s) = \mathbf{0} \\ (\mathbf{Y}_r \wedge \mathbf{Y}_s \mid \mathbf{X}_u) = 0 \end{cases}$ (3) et $\begin{cases} \mathbf{X}(u,v) - \mathbf{Y}(r,s) = \mathbf{0} \\ (\mathbf{Y}_r \wedge \mathbf{Y}_s \mid \mathbf{X}_v) = 0 \end{cases}$ (4)

qui ne nécessitent que les équations paramétriques sont équivalents à (1) et (2). Malheureusement les systèmes (1) ... (4) sont de degré trop élevé pour être traités par une

technique de résolvants; seule une méthode itérative peut aboutir. Cette méthode nécessite une solution approchée pour chaque solution exacte; elle implique donc l'échantillonnage des surfaces et la recherche de points candidats.
Cette approche est délaissée au profit de solutions plus simples, mais empiriques.
Les deux carreaux sont quadrillés par des réseaux d'isoparamétriques, les points d'intersection entre les isoparamétriques et les carreaux sont recherchés; ceux ci constituent les points germes.
Le problème est donc ramené au calcul de l'intersection isoparamétrique-carreau. Ce calcul peut être réalisé de trois façons différentes: en polygonalisant la courbe et le carreau ou en recherchant à résoudre le système $\mathbf{X}(u_{fixé},v) - \mathbf{Y}(r,s) = \mathbf{0}$ non linéaire à l'aide d'une méthode de Newton ou encore une technique de résolvant.

La polygonalisation des deux carreaux peut être obtenue suivant deux méthodes différentes. Elle peut être effectuée à l'aide de subdivisions récursives; elles sont poursuivies jusqu'à obtenir des approximations planes et les sous carreaux utiles sont filtrés. Les points germes sont alors obtenus comme intersections de segments et de facettes planes. Cette méthode est cependant très lente du fait des subdivisions; elle est mal adaptée à cette tâche.
Une autre méthode consiste à effectuer un échantillonnage assez dense de points sur les deux carreaux; les isoparamétriques d'un des carreaux sont alors polygonalisés alors que l'autre carreau est triangularisé. Les points d'intersection entre les segments et les triangles sont alors recherchés. Cette méthode est développée dans [JOR 87] et dans [AZI 90].
Timmer [TIM 85] associe échantillonnage et méthode de Newton. Sur une isoparamétrique fixée $\mathcal{C}$ sur le carreau $\mathcal{S}_1$, pour chaque point **P** échantillonné sur celle-ci est calculé le point **Q** le plus proche de **P** sur le carreau adverse $\mathcal{S}_2$. Le point **Q** est obtenu avec une méthode de Newton qui est initialisée avec le point de l'échantillon sur $\mathcal{S}_2$ le plus proche de **P**. Grâce à la normale calculée sur $\mathcal{S}_2$ en **Q**, il est possible de savoir de quel coté de $\mathcal{S}_2$ se trouve le point **P**. En évaluant cette information pour les points successifs sur $\mathcal{C}$ on localise les points où la courbe traverse le carreau $\mathcal{S}_2$. Ces points sont ensuite raffinés avec une méthode de Newton.

La démarche que nous avons développée est très différente. Les points d'intersection entre une isoparamétrique et un carreau sont obtenus à l'aide d'une technique de résolvant qui d'un point de vue conceptuel est plus rationnelle (elle apporte la solution mathématique exacte et complète). Cependant celle-ci n'est applicable qu'à des carreaux de faible degré (biquadratiques, ou encore bicubiques).

Nous avons donc conçu un algorithme qui simplifie les carreaux NURBS jusqu'à ce degré. Il généralise la méthode dichotomie récursive et intersections de plans. Les sous carreaux sont filtrés et simplifiés progressivement. Globalement l'algorithme de détermination des points germes associe donc subdivision récursive, diminution de degré et technique de résolvant. Il réalise un compromis entre une méthode qui simplifie les surfaces mais qui augmente considérablement les données et une méthode mathématiquement rigoureuse qui exige des données simples. L'algorithme conserve donc les excellents principes de la dichotomie récursive: simplification, structuration hiérarchique des surfaces (quadtree) et localisation de l'intersection mais en limite les inconvénients: les subdivisions sont arrêtées bien plus tôt. Le concept subdivision/diminution de degré a été analysé puis un algorithme de diminution de degré fonctionnant dans la base de Bernstein a été mis au point. Les carreaux sont symétriquement modifiés de façon à garantir une continuité d'ordre élevé aux frontières.
La résolution du problème intersection courbe rationnelle quadratique-carreau rationnel biquadratique est ramenée à l'aide de résolvants à une équation réelle de degré 16 sur [0,1]. La méthode conçue généralise celle de Kajiya [KAJ 82] pour la résolution du problème intersection droite-carreau bicubique. Les avantages de cette résolution algébrique sur les méthodes associant échantillonnage et itération de Newton ou échantillonnage et intersection segment-triangle sont qu'elle trouve toutes les solutions et surtout qu'elle s'affranchit des fastidieux tests de proximité nécessaires à l'initialisation d'une itération de Newton ou à une comparaison segment-triangle. Une réalisation a montré que cette méthode peut être utilisée seule pour résoudre le problème intersection carreau biquadratique-carreau biquadratique.

Il est difficile de comparer avec précision les performances des trois méthodes de détermination des point germes. L'échantillonnage de points ou les subdivisions récursives nécessitent de gros volumes de calcul qui sont de même ordre de grandeur. Le nombre de points échantillonnés est très supérieur au nombre de subdivisions cependant l'échantillonnage est effectué dans la base canonique des polynômes et utilise l'algorithme de Horner ou même les différences successives de polynômes alors que les subdivisions sont effectuées dans la base de Bernstein avec l'algorithme de De Casteljau de complexité supérieure.
L'intérêt de la dernière méthode est plus net si les intersections sont calculées plusieurs fois sur des objets qui sont successivement déplacés. Dans ce cas il est intéressant de conserver les échantillons ou les quadtree pour les opérations répétées; il suffit de les mettre à jour en appliquant aux points les matrices de transformation. Pour les opérations ultérieures, les comparaisons impliquées dans la localisation de la courbe d'intersection avec la méthode

"subdivision/diminution de degré/résolution par les résolvants" sont négligeables puisqu'il s'agit d'un parcours d'arbre 4-aire équilibré alors que les comparaisons de points avec les autres méthodes sont très coûteuses car il n'y a pas de structuration de l'espace.

Du point de vue de la qualité du résultat, aucune de ces méthodes ne garantit la détection de toutes les courbes. Dans tous les cas cette dernière dépend de la densité du quadrillage.

Détection et analyse des points singuliers

La seconde importante difficulté rencontrée par la méthode de suivi, la présence de points singuliers, est inhérente à toute méthode de calcul d'intersection. Ces points ou les configurations de surfaces qui s'en approchent rendent les calculs imprécis et provoquent des incohérences topologiques. Une gestion correcte des problèmes d'intersection est impossible sans la détection et l'analyse de ces points.
Ceux-ci sont solutions du système

$$\begin{cases} f(u,v) = 0 \\ f_u(u,v) = 0 \\ f_v(u,v) = 0 \end{cases} \quad (5) \qquad \text{ou encore du système équivalent} \qquad \begin{cases} \mathbf{X}(u,v) - \mathbf{Y}(r,s) = \mathbf{0} \\ (\mathbf{Y}_r \wedge \mathbf{Y}_s \mid \mathbf{X}_u) = 0 \\ (\mathbf{Y}_r \wedge \mathbf{Y}_s \mid \mathbf{X}_v) = 0 \end{cases} \quad (6)$$

Ces systèmes sont surdéterminés. Dans [BAJ 88] (5) est résolu à l'aide d'une méthode combinant itération de Newton et approximation par les moindres carrés:

Appliquée à (5) la méthode de Newton revient à calculer à chaque itération $\begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix}$ qui

vérifie $\begin{bmatrix} f_u & f_v \\ f_{uu} & f_{uv} \\ f_{uv} & f_{vv} \end{bmatrix} \cdot \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} = - \begin{bmatrix} f(u,v) \\ f_u(u,v) \\ f_v(u,v) \end{bmatrix}$ encore noté $A.\ \Delta U = -B$

mais comme le système est surdéterminé on choisit ΔU qui minimise $\| A.\ \Delta U + B \|^2$, ΔU est donc solution de ${}^tA.A\ \Delta U = -{}^tA.B$

Dans le cas particulier où le point singulier vérifie $f_{uu}.f_{vv} - f_{uv}^2 = 0$, c'est à dire lorsque les deux arcs qui se croisent en ce point sont tangents la suite ne converge pas car dét(${}^tA.A$) tend à s'annuler. Le système (5) est singulier. Il faut une équation de plus. Il est alors remplacé par le système

$$\begin{cases} f(u,v) = 0 \\ f_u(u,v) = 0 \\ f_v(u,v) = 0 \\ f_{uu}(u,v) = 0 \end{cases}$$

et le même schéma est appliqué.

La méthode est généralisée aux points singuliers d'ordre supérieur à 1.

Markot et Magedson [MAR 89] définissent un champ de vecteurs dont les zéros sont les points singuliers.
Ces systèmes puisque résolus par des méthodes itératives nécessitent des solutions approchées; ils peuvent donc être associés à une méthode de suivi. En effet le long du parcours la rencontre prochaine d'un point singulier peut être décelée en mesurant l'angle que forment les plans tangents des deux carreaux.
Owen et Rockwood [OWE 87] ont une approche différente du problème. Au voisinage du point singulier ils calculent une équation cartésienne approchée de degré 2 de la courbe puis analysent la courbe localement comme une conique.
Nous avons également obtenu une équation cartésienne approchée de la courbe en tout point en combinant l'équation cartésienne d'un paraboloïde approchant pour l'un des carreaux et un développement limité d'ordre 2 pour l'autre carreau. L'équation plus précise et de degré plus élevé (en u^4 v^4) peut être utilisée aux mêmes fins.

La détection de points singuliers n'est pas suffisante; les directions possibles en un tel point doivent aussi être connues afin de raccorder correctement les différents arcs confluents en ce point. L'équation précédemment citée, calculée en ce point, convient parfaitement à ce calcul si le point singulier est d'ordre 1. Pour un ordre supérieur il faut la remplacer par une combinaison d'une équation cartésienne de degré supérieur à 2 (le paraboloïde est insuffisant) pour la première surface et un développement limité d'ordre supérieur à 2 pour la seconde surface.

Enfin certaines difficultés rencontrées par les algorithmes de calcul d'intersection sont liées à la définition même des surfaces. La modélisation des quadriques par les NURBS engendre certaines ambigüités: Une sphère NURBS est composée de huit carreaux de Bézier dégénérés à trois cotés (les octants); les pôles correspondent aux cotés disparus. La paramétrisation n'est plus injective au pôles.

Si deux sphères NURBS sont sécantes et que la courbe d'intersection passe par un pôle de l'une d'elles un processus de suivi utilisé pour reconstituer la courbe échoue au franchissement du pôle. Un cas particulier doit être considéré.

V.2 PERSPECTIVES

V.2.1 Intersections

Perspectives théoriques

Dans le traitement de l'intersection des surfaces subsistent de nombreuses difficultés mathématiques, parmi celles ci figurent les points singuliers.
Nous avons évoqué précédemment plusieurs méthodes dont l'objectif est la détection de ces points. Notre objectif est d'étudier et d'expérimenter ces différentes méthodes; notamment il nous semble intéressant d'appliquer la méthode de [BAJ 88] au système (6) qui a l'avantage sur le système (5) de se contenter des équations paramétriques.
Nous cherchons également à calculer une équation cartésienne locale de degré variable en tout point d'un carreau paramétrique qui généraliserait celle de la quadrique. L'objectif est de faciliter l'analyse des points singuliers d'ordre supérieur à 1.
Nous proposons de déterminer l'équation de la manière suivante.
Notons $F(x,y,z) = 0$ l'équation polynomiale recherchée; elle doit représenter le comportement de la surface au voisinage d'un point $X(u^*,v^*)$. Par conséquent les dérivées de $F(x(u,v), y(u,v), z(u,v))$ doivent s'annuler jusqu'à un ordre fixé en (u^*,v^*) le plus grand possible. Elles fournissent des équations linéaires homogènes en les coefficients de l'équation. Il faut que nombre_d'équations = nombre_de _coefficients - 1. Le nombre de dérivées partielles à calculer dépend donc du degré choisi pour F.
Cette idée sera approfondie. Nous recherchons en particulier à montrer qu'elle généralise l'équation de la quadrique. Une réalisation suivra, dont la principale difficulté est l'évaluation du système linéaire.

Réalisations en cours

Plusieurs méthodes doivent être implantées, principalement l'extrapolation cubique pour la méthode de suivi de courbe et la résolution par les résolvants du problème intersection courbe cubique-carreau bicubique. La principale difficulté liée à la mise en oeuvre de cette opération est la stabilité numérique.

Réalisations à venir

La courbe d'intersection de deux carreaux est calculée afin de permettre le calcul de l'intersection de deux carreaux restreints. L'étape suivante est la création de carreaux restreints. Ils doivent permettre la modélisation d'objets composés de surfaces gauches. Ces derniers doivent être intégrés au modeleur solide B-rep du logiciel SACADO qui est actuellement développé au LRIM.

V.2.2 Applications à d'autres opérations

Dans l'algorithme du tracé de rayon le calcul de l'intersection droite-objet intervient de façon omniprésente quelle que soit la modélisation solide choisie (B-rep ou arbre CSG) [ROT 82] [SAH 90] [SWE 88]. Ces calculs représentent la majeure partie des calculs induits par l'algorithme [PER 88]. Toute optimisation de ceux-ci se traduit par un gain de rapidité important pour l'algorithme.
Lorsque les solides sont modélisés à l'aide de carreaux NURBS, cette opération d'intersection est ramenée au calcul de l'intersection droite-carreau. Ce dernier constitue un problème bien plus simple que celui qui a été exposé. L'algorithme de subdivision récursive/diminution de degré, appliqué ici permet une réduction des calculs.
Il peut être considéré comme un prétraitement antérieur à toute visualisation. Chaque carreau NURBS composant la scène est décomposé en quadtree dont les feuilles sont des carreaux de Bézier biquadratiques rationnels. Pour les noeuds non terminaux, ne sont conservés que les volumes englobants. Avec cette structure l'intersection d'un rayon et d'un carreau est décomposé en trois étapes.

Etape 1

L'arbre est parcouru depuis la racine jusqu'à la feuille (ou les feuilles) sécante avec le rayon. La structure arborescente associée aux volumes englobants (boîtes min-max) optimise le nombre de comparaisons nécessaires.

Etape 2

Le point d'intersection entre le rayon et le carreau terminal est calculé avec la méthode de Kajiya [CAU 88] [KAJ 82]: combinaison des équations cartésiennes de la droite et de l'équation paramétrique du carreau puis résolution du système obtenu avec les résolvants. L'équation finale est de degré 8.

Etape 3
Le point d'intersection exact entre le rayon et le carreau-racine est calculé avec une méthode de Newton initialisée avec le point obtenu à l'étape 2. Le vecteur normal au carreau en ce point est ensuite calculé.

La majeure partie des calculs engendrés par cette méthode provient de l'évaluation du résolvant (déterminant d'ordre 4) et de la résolution de l'équation réelle.
L'algorithme proposé réalise un compromis entre la décomposition par facettes planes et la résolution directe par les résolvants du système non linéaire général. Cette dernière, appliquée à un carreau bi_m_ique conduit à l'évaluation d'un résolvant d'ordre 2m et à la résolution d'une équation de degré $2m^2$. Bien que cette méthode soit mathématiquement séduisante, le volume de calcul engendré est incompatible avec l'optimisation recherchée. A l'inverse la décomposition récursive jusqu'aux facettes planes réduit le calcul d'intersection au cas linéaire qui est le plus simple mais le volume de données est considérable et le prétraitement est long.

La méthode exposée, bien sûr, ne se substitue pas aux techniques qui se servent de la cohérence des rayons primaires [JOY 86], mais elle leur est complémentaire. Notamment la première intersection droite-carreau ne peut être calculée en utilisant la cohérence des rayons; la décomposition proposée apporte une solution.

CONCLUSION

L'exposé traite de l'intersection des carreaux NURBS, problème incontournable de la modélisation des solides par les surfaces gauches.
Les méthodes classiques ont été étudiées. Il s'agit de l'association "décomposition dichotomique récursive en facettes planes/intersections de facettes planes", de la méthode de Farouki et de la méthode de suivi de courbe. La méthode de Farouki consiste à analyser une équation de la courbe d'intersection $f(u,v) = 0$ déduite de l'équation paramétrique de l'une des surfaces et de l'équation cartésienne de l'autre surface. La dernière méthode calcule des points de proche en proche le long de la courbe d'intersection.
Les trois approches sont intéressantes et ne s'appliquent pas exactement aux mêmes conditions du problème. Nous avons étudié des solutions qui tentent de combiner les différents avantages.
La première approche a été approfondie et généralisée; elle a débouché sur un algorithme où les subdivisions successives sont progressivement simplifiées. Les carreaux initiaux sont décomposés en carreaux de Bézier biquadratiques qui sont suffisamment simples pour être traités par des méthodes de résolvants.
Ces derniers ont été utilisés pour traiter l'intersection courbe quadratique-carreau biquadratique. Ils ont permis la résolution complète du système 2x2 qui est obtenu par combinaison d'une équation cartésienne de la courbe et de l'équation paramétrique du carreau.
Un algorithme exploitant le principe du suivi de courbe a été réalisé, dans lequel la courbe construite est parcourue par l'abscisse curviligne. Ceci a permis en chaque point le calcul de vecteurs dérivées successives et par conséquent la détermination d'un développement limité de la courbe.
L'association de ces trois opérations constitue un algorithme général.

Les perspectives envisagées sont multiples.
La résolution du problème de l'intersection courbe quadratique-carreau biquadratique à l'aide des résolvants doit être étendue au cas cubique. La stabilité numérique est la principale difficulté car la méthode aboutit à la résolution d'une équation réelle de degré théorique égal à 72.
Un problème plus fondamental est la détection et l'analyse des points singuliers. Plusieurs méthodes sont étudiées; leur réalisation est nécessaire pour les valider.
Enfin la connaissance des courbes d'intersection entre des carreaux NURBS conduit à la création de carreaux restreints. Les surfaces gauches, par le biais de ces derniers, seront

finalement intégrées au modeleur solide du logiciel SACADO développé actuellement au Laboratoire de Recherche en Informatique de Metz.

BIBLIOGRAPHIE

[AZI 90] : Nadim M.AZIZ, Sudarshan BHAT
"Bezier Surface/Surface Intersection"
IEEE CGA Janvier 1990

[BAJ 88] : C.L. BAJAJ, C.M. HOFFMANN, R.E. LYNCH, J.E.H. HOCROFT
"Tracing surface intersections"
Computer Aided Geometric Design volume 5 pp 285-307 Avril 1988

[BAR 88] : Brian BARSKY, Richard BARTELS, John BEATTY
"B-SPLINES"
Mathématiques et CAO volume 6
éditions HERMES PARIS 1988

[BEZ 86] : Pierre BEZIER
"Courbes et Surfaces"
Mathématiques et CAO volume 4
éditions HERMES PARIS 1986

[BOE 84] : W. BOEHM, G. FARIN, J. KAHMANN
"A survey of curve and surface methods in CAGD"
Computer Aided Geometric Design volume 1 n°1 pp 1-601984

[BOO 78] : C . de BOOR
"A practical guide to splines"
éditions Springer Verlag New-York 1978

[CAR 82] : W.CARLSON
"An algorithm and data structure for 3D object synthesis"
Computer Graphics volume 16 n° 3 1982

[CAS 87] : M. S.CASALE, E. L.STANTON
"Free-form solid modeling with trimmed surface patches"
IEEE CGA janvier 1987

[CAS 85] : P. de CASTELJAU
"Formes à pôles"
Mathématiques et C.A.O volume 2
éditions HERMES PARIS 1985

[CAU 88] : R. CAUBET, Y. DUTHEN, R. PUJADO, M.Z. SANDOUK
"Lancer de rayons optimisé pour les surfaces gauches de formes libres"
Actes de la 7ème conférence internationale de MICAD
éditions HERMES PARIS 1988

[COO 87] : S A. COONS
"Méthode matricielle"
Mathématique et C.A.O volume 5
éditions HERMES PARIS 1987

[CHA 87]: Vijaya CHANDRU, Bipin S KOCHAR
"Analytic Techniques for Geometric Intersection Problems"
pp 305-318 Geometric Modeling: Agorithms and New Trends by Gerald E. FARIN
1987 SIAM

[CHE 85] : P. CHENIN, M. COSNARD, Y. GARDAN, F. ROBERT, Y. ROBERT, P.WITOMSKY
"Méthodes de base"
Mathématiques et C.A.O volume 1
éditions HERMES PARIS 1985

[DAN 89]: M. DANIEL
"Modélisation de courbes et surfaces par des B-splines. Application à la conception assistée par ordinateur et à la visualisation de formes."
Thèse de l'Université de Nantes 1989

[DOK 85] : Tor DOKKEN
"Finding intersections of B-spline represented geometries using recursive subdivision techniques"
Computer Aided Geometric Design volume 2 pp 189-195 Avril 1985

[FAR 87] : Rida FAROUKI
"Direct Surface Section Evaluation"
pp 319-334 Geometric Modeling: Agorithms and New Trends by Gerald E. FARIN
1987 SIAM

[FOL 82] : J.D. FOLEY, A. VAN DAM
"Fundamental of interactive computer graphics"
éditions ADDISON, WESLEY 1982

[GAR 87] : Y. GARDAN
" La C.F.A.O introduction, techniques et mise en oeuvre"
éditions HERMES PARIS 1987 2ème édition

[GAR 89] : Y.GARDAN, D. MICHEL, M. SAHNOUNE
" Etude comparée des formes paramétriques pour la modélisation des surfaces"
Rapport de recherche LRI METZ 1989

[GAR 90] : Y. GARDAN, D. MICHEL , M. SAHNOUNE
" Comparaison des propriétés des formes rationnelles et non rationnelles pour la modélisation des surfaces"
Actes de la 9éme conférence internationale de MICAD
éditions HERMES PARIS 1990

[GAR 91] : Y. GARDAN, D. MICHEL , A. ZIDNA
Note technique " Une nouvelle technique de séparation des racines d'un polynôme"
Revue internationale de CFAO et d'infographie Volume 6 n° 1
éditions HERMES PARIS 1991

[GOL 85] : R . N GOLDMAN
"The method of resolvants: A technique for the implicitization, inversion, and intersection of non planar parametric rational cubics curves."
Computer Aided Geometric Design volume 2 1985

[HOF 89] : Jung Hong CHUANG, Christoph M. HOFFMANN
"On Local Implicit Approximation and Its Applications"
A.C.M Transactions on Graphics volume 8 n°4 pp 298-321 octobre 1989

[HOU 85] : E. G.HOUGHTON, J. D.FACTOR, C. L.SABHARWAL
" Implementation of a divide and conquer method for intersection of parametric surfaces"
Computer Aided Geometric Design volume 2 1985

[JOR 87] : R.E. BARNHILL, G. FARIN, M. JORDAN and B.R. PIPER
" Surface/surface intersection"
Computer Aided Geometric Design volume 4 3-16 janvier 1987

[JOY 86] : Kenneth I. JOY, Murthy N. BHETANABHOTLA
"Ray Tracing Parametric Surface Patches Utilizing Numerical Techniques and Ray Coherence"
A.C.M. volume 20 n°4 1986

[KAJ 82] : J. T. KAJIYA
" Ray tracing parametric patches "
ACM computer graphics volume 16 n° 3 1982

[KOR 61] : J . KORGANOFF
"Méthodes de calculs numériques"
Tome 1 algèbre non linéaire éditions DUNOD PARIS 1961

[LEL 63] : J . LELONG FERRAND
"Géométrie différentielle"
Masson 1963

[LEV 76] : J. LEVIN
"A parametric algorithm for drawing picture of solid object composed of quadric surfaces"
Communication of A.C.M volume 19 n° 10 1976

[MAR 89] : R. P. MARKOT, R. L. MAGEDSON
"Solutions of tangential surface and curve intersections"
Computer-Aided-Design volume 21 n° 7 septembre 1989

[MAR 91]: R. P. MARKOT, R. L. MAGEDSON
"Procedural method for evaluating the intersection curves of two parametric surfaces"
Computer-Aided-Design volume 23 n° 6 juillet 1991

[MIC 87] : D. MICHEL
"Etude de l'intersection et du raccordement de deux carreaux de Bézier"
Rapport de DEA Université de Technologie de Compiègne/LRI Metz 1987

[MIL 86] : J. MILLER
" Sculptured Surfaces in solid modeling: Issues and alternative approachs"
IEEE CGA décembre 1986

[MON 86] : Yves de MONTAUDOIN, Wayne TILLER, Harvard VOLD
"Applications of power series in computational geometry"
Computer-Aided-Design volume 18 n° 10 décembre 1986

[OWE 87] : John C. OWEN, Alyn P. ROCKWOOD
"Intersection of General Implicit Surfaces"
pp 335-345 Geometric Modeling: Agorithms and New Trends by Gerald E. FARIN
1987 SIAM

[PER 88] : B. PEROCHE, J. ARGENCE, D. GHAZANFARPOUR, D. MICHELUCCI
"La synthèse d'images"
éditions HERMES PARIS 1988

[RAL 65] : A . RALSTON
"A first course in numerical analysis"
éditions Mac Graw Hill 1965

[ROT 82] : S. D. ROTH
" Ray casting for modeling solids"
Computer graphics and Image processing 18 1982

[SAH 90] : Myriam SAHNOUNE
"Contribution à l'intégration des surfaces gauches dans les modèles de solides"
Thèse de l'Université de Metz Décembre 1990

[SED 84] : T.W. SEDERBERG , D.C. ANDERSON, R.N. GOLDMAN
" Implicit representation of curves and surfaces"
Computer Vision and Image processing 28 1984

[SED 85] : T.W. SEDERBERG
"Piecewise algebraic surface patches"
Computer Aided Geometric Design 2 1985

[SWE 86] : M . A . J SWEENNEY, R.H BARTELS
"Ray tracing free form B-Splines surfaces"
IEEE 1984

[TIM 85] : TIMMER
"Surfaces Intersections"
pp 333-344 Geometric Modeling Michael E. Mortenson
éditions JOHN WILEY and sons New-York 1985

[VER 73] : M. VERON
"Contribution à l'étude des surfaces numériques Unisurf. Conditions de raccordement"
Thèse de Doctorat d'Etat, Université NANCY I 1973

[VER 88] : M. VERON
Séminaire "La modélisation des surfaces"
Journée du 21 Juin 1988 Paris
éditions HERMES PARIS 1988

[VIV 93] : Robin VIVIAN
"Contribution à l'optimisation des algorithmes de tracé de rayon par une décomposition intelligente et adaptative de la scène"
Thèse de l'Université de Metz 1993 (à paraître)

[ZID 90] : Ahmed ZIDNA
"Contribution à la modélisation des carreaux troués"
Thèse de l'Université de Metz Décembre 1990

ANNEXES

I APPROXIMATION D'UN CARREAU DE BEZIER RATIONNEL PAR UN CARREAU POLYNOMIAL

Définition du carreau de Bézier polynomial approchant

$\mathcal{B}$ est le carreau de Bézier rationnel défini par:

$$(u,v) \longmapsto \sigma(u,v) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} t_{ij} P_{ij} B_i^m(u).B_j^n(v)}{\sum_{i=0}^{m}\sum_{j=0}^{n} t_{ij} B_i^m(u).B_j^n(v)}$$

Notons t = moyenne(t_{ij}), c'est à dire que $t = \frac{1}{(m+1)x(n+1)} \sum_{i=0}^{m}\sum_{j=0}^{n} t_{ij}$

Nous approchons ce carreau par le carreau de Bézier polynomial $\mathcal{Q}$ défini par :

$$(u,v) \longmapsto \varrho(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \frac{t_{ij}}{t} P_{ij} B_i^m(u).B_j^n(v)$$

Estimation de l'erreur

Notons $\mathcal{E}_{rel}(u,v)$ l'erreur relative en (u,v).

$$\mathcal{E}_{rel}(u,v) = \frac{\| \sigma(u,v) - \varrho(u,v) \|}{\| \sigma(u,v) \|}$$

Un rapide calcul montre que: $\mathcal{E}_{rel}(u,v) \leq \frac{\max_{0 \leq i \leq m,\, 0 \leq j \leq n} | t_{ij} - t |}{t}$

Cette dernière quantité est facile à obtenir à partir du polygone de contrôle dans l'espace homogène. Si la valeur trouvée est inférieure à la précision choisie, on remplace $\mathcal{B}$ par $\mathcal{Q}$.

Démonstration de convergence

Si le carreau $\mathcal{B}$ est le résultat de plusieurs subdivisions successives, il devient proche du carreau $\mathcal{Q}$. En effet les points de contrôle Q_{ij} dans l'espace homogène convergent tous vers la même valeur (cf IV.1.2.1). Par conséquent (cf I.1.3) les poids t_{ij} convergent aussi vers une valeur commune, donc

$\forall i \forall j$, $| t_{ij} - t | \longrightarrow 0$ Donc $\mathcal{E}_{rel}(u,v) \longrightarrow 0$ lorsque le nombre de subdivisions successives est assez important.

II EXEMPLES DE REALISATIONS D'ALGORITHMES

Deux exemples de réalisations d'algorithmes sur les courbes B-spline sont présentés. Il s'agit de la subdivision récursive et du calcul des dérivées successives (ttesderiv). Le premier est programmé dans deux cas, lorsque la valeur insérée est quelconque (subdc) et lorsque celle-ci est un noeud de multiplicité égale à 1 (subdc1).
Les algorithmes ont été réalisés en langage C. Toutes les définitions nécessaires sont décrites au préalable.

```
/*-------------------------------------------------------------------------------------------*/
/*------- CONSTANTES POUR LES COURBES ET SURFACES BSPLINES --------    */
/*-------------------------------------------------------------------------------------------*/

#define Maxpoint 9                      /* Nbre max de pts de contrôle d'une courbe  */

#define Maxnoeud (2*Maxpoint)           /* Nbre max de noeuds d'une courbe           */
                                        /* il faut que Maxnoeud=<2*Maxpoints         */

#define Maxp (Maxpoint-1)               /* Maxp=Maxpoint-1                           */
#define Maxn (Maxnoeud-1)               /* Maxn=Maxnoeud-1                           */

#define Maxpoint1 (Maxpoint + 1)

/*-------------------------------------------------------------------------------------------*/
/*---------------- EPSILONNS ----------------------------------------------------------------*/
/*-------------------------------------------------------------------------------------------*/

#define eps 0.00001                     /* used in subdivision des courbes           */

/*-------------------------------------------------------------------------------------------*/
/*---------- TYPES POUR LES COURBES ET SURFACES ----------------------------    */
/*-------------------------------------------------------------------------------------------*/

typedef float   point4D[4];             /* point en 4 D     */

typedef struct noe {
                double val;             /* valeur ∈ R       */
                int multip;             /* multiplicité     */
                } noeud;                /* noeud            */

typedef noeud   vecnod[Maxnoeud];       /* vecteur nodal    */

typedef point4D polcurv[Maxpoint];      /* suite de pts de contrôle:polygône */
```

```
typedef point4D   polsurf1[Maxpoint1][Maxpoint];
                                        /* polygone de contrôle matriciel          */
                                        /* utilisé pour modéliser le schéma        */
                                        /* triangulaire de l'algo de Cox-De Boor   */

typedef struct Bcu {
                int k,m;                /* k est l'ordre de la courbe              */
                                        /* m est l'indice max des pts de contrôle  */
                                        /* Les pts de contrôle sont indexés de 0 à m */
                vecnod t;
                                        /* vecteur nodal                           */
                polcurv pol;            /* polygône de contrôle                    */
                } Bcurv;
                /* Le type Bcurv représente complètement une  courbe B_spline      */

/*-------------------------------------------------------------------------------------------*/
/*--------- FONCTIONS DE CALCUL VECTORIEL ---------------------------------------------------*/
/*-------------------------------------------------------------------------------------------*/

/*****************************************************************************/
/*                        SOM4D                                              */
/*    effectue  a <-- b + c  en 4 dimensions                                 */
/*****************************************************************************/

void som4D( a, b, c)
point4D       a, b, c;
{
float          *p,*q,*r;
int             i;

/*------som4D------*/

for ( p=&(a[0]), q=&(b[0]), r=&(c[0]), i=0; i<=3; p++, q++, r++, i++)
     *p=*q+*r;

/*------som4D------*/
}
```

```
/*****************************************************************/
/*                        DIFF4D                                 */
/*    effectue  a <-- b - c  en 4 dimensions                     */
/*****************************************************************/

void diff4D( a, b, c)
point4D       a, b, c;
{
float         *p,*q,*r;
int           i;

/*----- diff4D -------*/

for ( p=&(a[0]), q=&(b[0]), r=&(c[0]), i=0; i<=3; p++, q++, r++, i++)
    *p=*q-*r;

/*----- diff4D -------*/
}

/*****************************************************************/
/*                        MUL4D                                  */
/*    effectue  a <-- l * b  en 4 dimensions ,l ∈ R, a ∈ R⁴ , b ∈ R⁴  */
/*****************************************************************/

void mul4D( a, l, b)
point4D       a;
double        l;
point4D       b;
{
float          *p, *q;
int            i;

/*------mul4D-------*/

for ( p=&(a[0]), q=&(b[0]), i=0; i<=3; p++, q++, i++)
    *p=(*q)*l;

/*------mul4D-------*/
}
```

```
/*************************************************************************/
/*                          AFF4D                                        */
/*    effectue  a <--  b  en 4 dimensions ,l ∈ R, a ∈ R⁴ , b ∈ R⁴        */
/*************************************************************************/

void aff4D( a, b)
point4D     a,  b;
{

/*------aff4D-------*/

a[0] =b[0]; a[1] =b[1]; a[2] =b[2]; a[3] =b[3];

/*------aff4D-------*/
}

/*************************************************************************/
/*                          COMBLIN4D                                    */
/*    effectue  a <-- lam * b + mu * c                                   */
/*  a ∈ R⁴ , b ∈ R⁴ , c ∈ R⁴ , lam ∈ R, mu ∈ R                           */
/*************************************************************************/

void comblin4D( a, lam, b, mu, c)   /* a <-- lam*b + mu*c */
point4D          a;
double           lam;
point4D          b;
double           mu;
point4D          c;
{
float            *p,*q,*r;
int              i;

/*----- comblin4D ------*/

for( p=&(a[0]), q=&(b[0]), r=&(c[0]), i=0; i<=3; p++, q++, r++, i++)
   *p=(*q)*lam+(*r)*mu;

/*----- comblin4D ------*/
}
```

```
/*-------------------------------------------------------------------------------------------------*/
/*------- FONCTIONS de CALCUL sur LES COURBES BSPLINES ----------------     */
/*-------------------------------------------------------------------------------------------------*/

/*****************************************************************************/
/*                          OMEGA                                            */
/* calcule om=(x-Ui)/(Ui+r-Ui) et omb=(Ui+r-x)/(Ui+r-Ui)                     */
/*         si Ui+r-Ui=0 rend om=omb=0                                        */
/*  sert ds les algos de Boor_Cox                                            */
/*  ENTREES:x,i,r,noeuds(vecteur nodal)                                      */
/*  SORTIES:  pom,pomb  (pointeurs sur les réels)                            */
/*****************************************************************************/

void omega( pom, pomb, x, i ,r, noeuds)
double        *pom,*pomb, x;
int            i, r;
vecnod       noeuds;
{
double       d;

/*---------omega------------*/

d=(noeuds[i+r]).val-(noeuds[i]).val;   /* dénominateur */
if (d<eps)
   {
    printf("dénominateur nul\n");
    *pom=*pomb=0.0;
    }
else
    {
     *pom=(double)((x-(noeuds[i]).val)/d);
     *pomb=(double)(((noeuds[i+r]).val-x)/d);
     }

/*---------omega------------*/
}
```

```
/*************************************************************************/
/*                    CHERCHE                                            */
/* rend i tel que Ui< = x <Ui+1                                          */
/* ENTREES: noeuds(vecteur nodal), k(ordre de la courbe), x              */
/*  SORTIE: i                                                            */
/*************************************************************************/

int cherche( x, noeuds, k)
double       x;
vecnod       noeuds;
int           k;
{
int           i;

/*---------cherche-----------*/

for ( i=k-1; x> =(noeuds[i]).val; i++);     /* on suppose que x> =Uk-1 */
return(--i);

/*---------cherche-----------*/
}

/*************************************************************************/
/*              INITRIANG                                                */
/*  initialise la base  du schéma triangulaire de Cox_de Boor            */
/*  initialise la ligne n°r entre les colonnes n°imin et n°imax          */
/*  avec les éléments de pol compris entre n°imin et n°imax              */
/*  ENTREES: r, imax, imin, pol(polygone de controle unidimensionnel     */
/*  SORTIES: triangle (matrice de points de controle)                    */
/*************************************************************************/

void initriang( triangle, r, imax, imin, pol)
polsurf1      triangle;
int            r, imax, imin;
polcurv       pol;
{
int            i;

/*---------initriang-----------*/

for ( i=imax; i> =imin; i--)
    aff4D(triangle[r][i],pol[i]);

/*---------initriang-----------*/
}
```

```
/*************************************************************************/
/*                              SUBDC                                    */
/* subdivise la courbe pcourb en deux parties pcurvg(gauche) et pcurvd   */
/* (droite) en u=x. x ne doit pas déja étre un noeud. Utilise l'algo de  */
/* Boor.                                                                 */
/*************************************************************************/

void subdc( pcurvg, pcurvd, x, pcourb)
Bcurv       *pcurvg,*pcurvd;
double      x;
Bcurv       *pcourb;
{
noeud       extre;
polsurf1    triangle;
int          m, i, j, r, delta, imin, k;
double      om, omb;

/*----------subdc----------*/

k=pcourb->k;   /* ordre de la courbe */
m=pcourb->m;   /* indice max des pts de ctrle de la courbe initiale */

extre.val=x;      /* extrèmité commune des 2 courbes obtenues */
extre.multip=k;

if ((x < ((pcourb->t)[k-1]).val+eps1) ||
   (x >= ((pcourb->t)[m+1]).val-eps1) )
  {
   printf("argument x transmis à subdc ∈ pas à [ t[k-1],t[m+1] ]");
   }

delta=cherche(x,pcourb->t,k);       /* cherche delta / Udelta<=x<Udelta+1 */
imin=delta-k+1;
initriang(triangle,k,delta,imin,pcourb->pol);

/* algo de Boor */

  for ( r=k-1; r>=1; r--)
      {
       imin=delta-r+1;
       for ( i=delta; i>=imin; i--)
           {
            omega(&om,&omb,x,i,r,pcourb->t);
            comblin4D(triangle[r][i],om,triangle[r+1][i],omb,triangle[r+1][i-1]);
```

```
        }
    }

/* Initialisation des indices maximaux et des ordres des courbes filles */

pcurvg->k=pcurvd->k=k;
pcurvg->m=delta;
pcurvd->m=pcourb->m+k-delta-1;

/* Initialisation des polygones de ctrle */

for ( i=0; i<=delta-k; i++)
    aff4D((pcurvg->pol)[i],(pcourb->pol)[i]);

for ( i=delta-k+1, r=k; i<=delta; i++, r--)
    aff4D((pcurvg->pol)[i],triangle[r][i]);

for ( i=0; i<=k-1; i++)
    aff4D((pcurvd->pol)[i],triangle[i+1][delta]);

for ( i=k, r=delta+1; i<=pcurvd->m; i++, r++)
    aff4D((pcurvd->pol)[i],(pcourb->pol)[r]);

/* Initialisation des vecteurs nodaux */

for ( i=0; i<=delta; i++)
    (pcurvg->t)[i]=(pcourb->t)[i];

 for ( i=delta+1, j=0; i<=delta+k; i++, j++)
     {
     (pcurvg->t)[i]=extre;   /*noeud extrèmal de multiplicité k */
     (pcurvd->t)[j]=extre;
     }

for ( i=k, r=delta+1; i<=pcurvd->m+k; i++, r++)
    (pcurvd->t)[i]=(pcourb->t)[r];

/*--------- subdc ----------*/
}
```

```
/*****************************************************************************/
/*                          SUBDC1                                           */
/* subdivise la courbe pcourb en deux parties pcurvg(gauche) et pcurvd       */
/* (droite) en Udelta. Udelta doit être un noeud de multiplicité = 1         */
/* nécéssairement. Utilise l'algo de Boor.                                   */
/* Le triangle de Boor compte un étage de moins que ds le cas où x est       */
/* de multiplicité 0 . Il faut que k-1 < = delta < = m+1                     */
/*****************************************************************************/

void subdc1( pcurvg, pcurvd, delta, pcourb)
Bcurv       *pcurvg,*pcurvd;
int          delta;
Bcurv       *pcourb;
{
polsurf1    triangle;
double      om,omb;
int          i, j, r, m, k, imin;
noeud       extre;

/*---------- subdc1 ---------------*/
k=pcourb->k;
m=pcourb->m;
extre.val=((pcourb->t)[delta]).val;
extre.multip=k;

if ((delta < k-1)  || (delta > m+1) )
   printf("argument delta transmis à subdc1 ∈ pas à [ k-1,m+1 ]");

initriang(triangle,k,delta-1,delta-k+1,pcourb->pol);

/* Algo de Boor */

  for ( r=k-1; r> =2; r--)
      {
      imin=delta-r+1;
      for (i=delta-1; i> =imin;i--)
          {
          omega(&om,&omb,((pcourb->t)[delta]).val,i,r,pcourb->t);
          comblin4D(triangle[r][i],om,triangle[r+1][i],omb,triangle[r+1][i-1]);
          }
      }

/* initialisation des indices maximaux et de l'ordre des courbes filles   */

pcurvg->k=pcurvd->k=k;
```

```
pcurvg->m=delta-1;
pcurvd->m=m+k-delta-1;

/*  initialisation des polygones de ctrle    */

for (i=0; i<=delta-k; i++)
    aff4D((pcurvg->pol)[i],(pcourb->pol)[i]);

for ( i=delta-k+1, r=k; i<=delta-1; i++, r--)
    aff4D((pcurvg->pol)[i],triangle[r][i]);

for ( i=0; i<=k-2; i++)
    aff4D((pcurvd->pol)[i],triangle[i+2][delta-1]);

for ( i=k-1, r=delta; i<=pcurvd->m; i++, r++)
    aff4D((pcurvd->pol)[i],(pcourb->pol)[r]);

/* Initialisation des vecteurs nodaux */

for ( i=0; i<=delta-1; i++)
    (pcurvg->t)[i]=(pcourb->t)[i];

 for ( i=delta, j=0; i<=delta-1+k; i++, j++)
     {
     (pcurvg->t)[i]=extre;    /* noeud extrèmal de multiplicité = k */
     (pcurvd->t)[j]=extre;
     }


for ( i=k, r=delta+1; i<=pcurvd->m+k; i++, r++)
    (pcurvd->t)[i]=(pcourb->t)[r];

/*---------- subdc1 ----------*/
}
```

```
/*************************************************************************/
/*          TTESDERIV                                                    */
/* calcule les dérivées d'ordre compris entre 0 et k-1 à droite          */
/* en x pour la courbe pcourb. la variable deriv contient en sortie      */
/* les dérivées. deriv[i] contient la dérivée d'ordre i, pour i=0,..k-1  */
/* Il faut que Uk-1 <= x < Um+1                                          */
/* Utilise l'algorithme de Boehm                                         */
/*************************************************************************/

void ttesderiv( deriv, x, pcourb)
polcurv         deriv;
double          x;
Bcurv           *pcourb;
{
polsurf1         triangle;
int              i, k, r, imin, imax, delta;
point4D         q;
double          d, lambda;

/*--------ttesderiv--------*/

k=pcourb->k;
delta=cherche(x,pcourb->t,k);

/* initialisation base du triangle */

initriang( triangle,0,delta,delta-k+1,pcourb->pol);

/* récurrence triangle inf droit */

for ( r=1; r<=k-1; r++)
    {
     imin=delta-k+r+1;
     for ( i=delta; i>=imin; i--)
         {
         d=((pcourb->t)[i+k-r]).val-((pcourb->t)[i]).val;
         if (d<eps)
            aff4D(triangle[r][i],vecnul);
         else
             {
             diff4D(q,triangle[r-1][i],triangle[r-1][i-1]);
             mul4D(triangle[r][i],1.0/d,q);
             }
         }
  }
```

```
/* récurrence triangle sup gauche */

imin=delta-k+1;

for ( r=1; r<=k-1; r++)
    {
     imax=delta-k+r;
     lambda=x-((pcourb->t)[imax+1]).val;
     for ( i=imax; i>=imin; i--)
         {
          mul4D(q,lambda,triangle[r][i+1]);
          som4D(triangle[r][i],q,triangle[r-1][i]);
          }
   }

/* calcul des dérivées */

aff4D(deriv[0],triangle[k-1][delta-k+1]);
d=1.0;
for (i=1; i<=k-1; i++)
    {
    d*=k-i;
    mul4D(deriv[i],d,triangle[k-1][imin+i]);
    }

/*--------- ttesderiv ---------*/
}
```

Résumé:

La modélisation de la forme des solides est un des aspects fondamentaux de la CFAO. L'enveloppe d'un solide est souvent définie par assemblage de plusieurs surfaces (modélisation géométrique par les limites). De nombreux modèles mathématiques de surface, adaptés à la CFAO ont été élaborés.
Ce mémoire traite des surfaces NURBS (Non Uniform Rational B-Spline); elles généralisent les carreaux de Bézier et permettent la représentation exacte des quadriques. La création d'objets par assemblage de faces implique de pouvoir "retailler" celles ci, et par conséquent, de savoir calculer l'intersection de plusieurs surfaces
Dans cet exposé, sont étudiées les différentes approches du problème intersection surface paramétrique-surface paramétrique. Les méthodes classiques sont rappelées. Il s'agit de la méthode "Dichotomie/Subdivision récursive/Intersection de facettes planes" bien connue qui ramène le problème général à celui de l'intersection facette plane-facette plane, de la méthode de Farouki qui traite le cas intersection surface cartésienne-surface paramétrique et de l'algorithme de suivi de courbe qui consiste à calculer de proche en proche des points le long de la courbe d'intersection.
Les trois approches sont intéressantes et ne s'appliquent pas au mêmes conditions du problème. Des solutions qui tentent de combiner les différents avantages sont développées.
Un algorithme original de simplification qui généralise la première méthode est conçu; il associe subdivisions récursives et diminution de degré.
La théorie des résolvants permet une résolution complète du problème intersection courbe quadratique-carreau biquadratique; son application au calcul des courbes d'intersection est très efficace. Enfin un algorithme exploitant le principe du suivi de courbe est réalisé, dans lequel la courbe construite est parcourue par l'abscisse curviligne. Ceci permet en chaque point d'obtenir des informations géométriques qui précisent le comportement de la courbe. Elles contribuent à améliorer les performances de l'algorithme. L'association de ces trois opérations constitue un algorithme général. Les difficultés majeures de la construction des courbes d'intersection, la détection des courbes fermées et le traitement des points singuliers sont analysées et partiellement résolues dans certains cas.

Mots-Clés:

C.A.O.
Carreaux de Bézier, carreaux NURBS, B-spline, carreaux restreints
Résolvants
Intersections
Subdivision récursive