- La localité d'une surface NURBS est $L^{(p+1)(q+1)}$, c'est-à-dire qu'un point $P_{i,j}$ n'influence que $(p+1)(q+1)$ carreaux de la surface globale (*cf.* Figure 3.7).

- Le problème de la continuité de la surface à la frontière des carreaux est plus complexe que dans le cas des courbes, puisqu'il doit être géré dans deux directions différentes : celle décrite par la variation du paramètre u et celle décrite par la variation du paramètre v. Néanmoins, les mêmes formules peuvent s'appliquer [PIEG91] : pour un noeud u_i (respectivement v_j) de multiplicité μ_i (respectivement μ_j), le raccordement s'effectue selon une continuité $C^{p-\mu_i}$ (respectivement $C^{q-\mu_j}$).

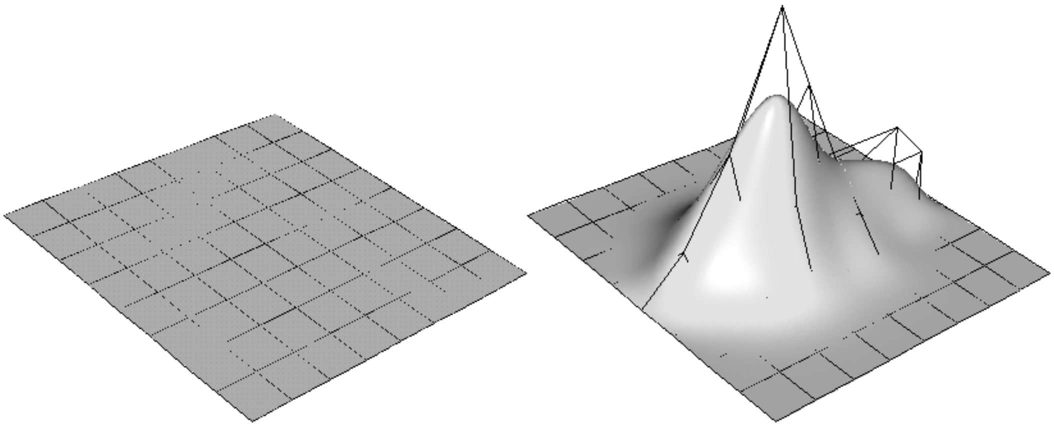


Figure 3.7: *Localité du contrôle pour une surface NURBS*

B - Manipulation des courbes et des surfaces

D'après ce que nous venons de voir on peut dire qu'une courbe et une surface NURBS disposent de nombreux degrés de liberté dont les variations permettent de modifier leur forme : position des points de contrôle, valeurs des poids, multiplicité des noeuds. Parmi ces degrés de liberté, la position des points de contrôle est le plus classique et ne nécessite pas d'avantage d'explication. Les deux autres, par contre, sont plus spécifiques et moins bien connus. C'est pourquoi, nous allons préciser ici de quelle manière (et avec quelles limitations) ces paramètres permettent d'influer sur la forme des courbes et surfaces résultantes.

B1 - Influence de la valeur des poids

Dans le cas d'une courbe NURBS, lorsqu'on augmente la valeur d'un poids w_k la courbe est attirée vers le point de contrôle P_k correspondant. Inversement, si on diminue w_k, la courbe s'éloigne du point P_k. Par exemple, la partie supérieure de la Figure 3.8 montre une courbe NURBS pour laquelle tous les poids sont égaux à 1 (il s'agit donc d'une courbe NUBS) et la partie inférieure, la courbe obtenue en positionnant le poids de P_4 à 6.

En fait, l'augmentation (respectivement la diminution) de l'influence du point de contrôle P_k lorsqu'on modifie son poids, se fait au détriment (respectivement au bénéfice) de l'influence de ses voisins dans le réseau de contrôle. Cette caractéristique s'explique de manière très simple au vu de l'Equation 3.1 qui représente une somme pondérée des points P_k dans laquelle w_k intervient dans le coefficient de pondération.

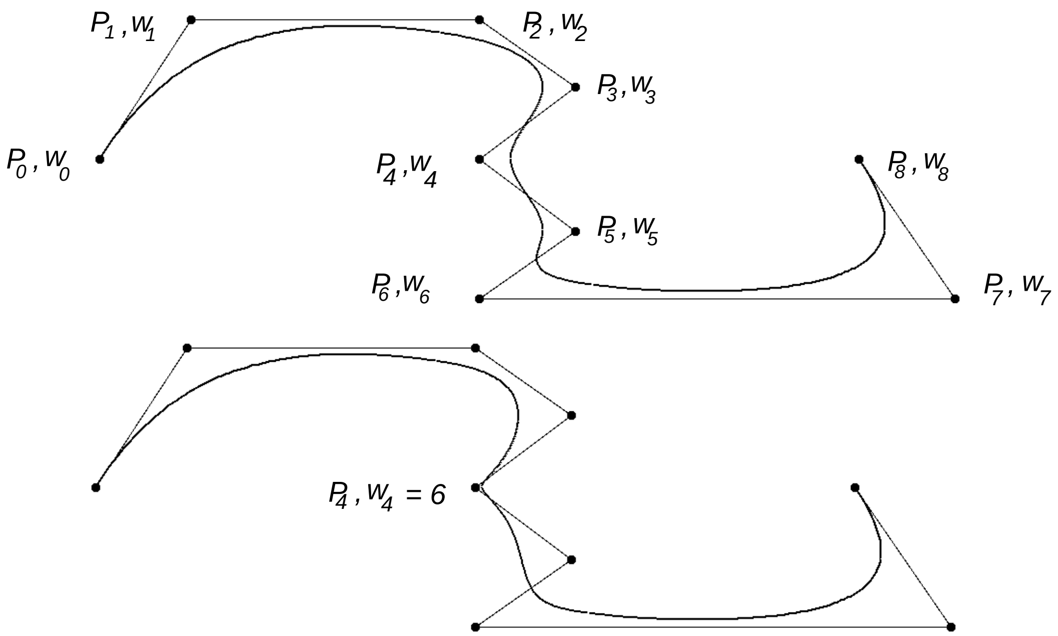


Figure 3.8: *Modification d'un poids*
(a) $\forall i \quad w_i = 1$, *(b)* $w_4 = 6$ *et* $w_i = 1$ *pour* $i \neq 4$

En particulier cela signifie qu'en modifiant de manière uniforme le poids de tous les points de contrôle d'une portion du réseau de contrôle, la courbe n'est attirée (respectivement repoussée) que par les points de contrôle qui se trouvent sur la frontière de la portion du réseau modifiée.

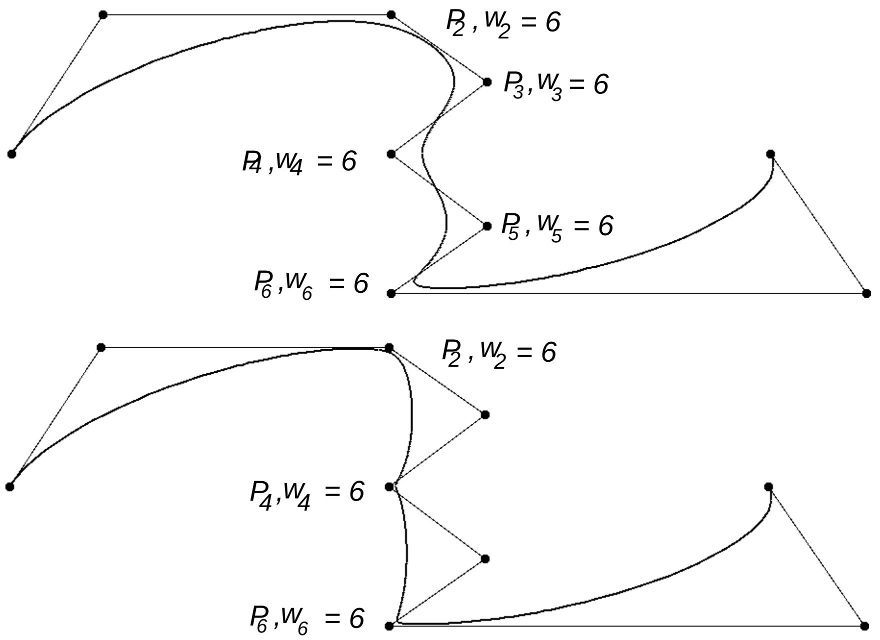


Figure 3.9: *Modification de plusieurs poids*
(a) $w_i = 6$ *pour* $i = 2, 3, 4, 5, 6$, *(b)* $w_i = 6$ *pour* $i = 2, 4, 6$

Par exemple, sur la courbe supérieure de la Figure 3.9, on a positionné à 6 les poids des points P_2, P_3, P_4, P_5 et P_6. Par comparaison avec la courbe supérieure de la Figure 3.8, on constate que les modifications ne se produisent qu'à la frontière de la zone modifiée (*i.e.* P_1, P_2 et P_6, P_7).

Concrètement, cette interdépendance de l'influence des poids signifie qu'on ne peut rapprocher la courbe que d'un nombre limité de points de contrôle : au mieux un point sur $1 + p/2$ pour une courbe de degré p. La partie inférieure de la Figure 3.9 confirme qu'avec une NURBS cubique on peut rapprocher la courbe d'un point sur deux. Dans le cas des surfaces, la restriction est plus forte, puisqu'elle passe à un point sur $(1 + p/2)(1 + q/2)$.

Cette limitation sur l'utilisation des poids pour la modification de la forme des courbes et des surfaces peut apparaître comme étant peu contraignante. En fait, la limitation est surtout d'ordre sémantique : un utilisateur non mathématicien qui sait que le rôle du poids est d'attirer ou de repousser la courbe par rapport à un point de contrôle, aura du mal à comprendre que lorsqu'il augmente le poids d'un groupe de points successifs, la courbe ne s'approche pas de chacun de ces points. Le résultat obtenu est tellement troublant par rapport au résultat attendu que beaucoup d'utilisateurs décident de se passer de la manipulation des poids pour se contenter de celle des points de contrôle.

B2 - Influence de la multiplicité des noeuds

Nous avons vu parmi les propriétés des courbes et surfaces NURBS, que l'augmentation de la multiplicité d'un noeud permet de diminuer localement la continuité au point correspondant. Dans ce paragraphe, nous allons étudier ce phénomène de manière plus précise.

Dans le cas général, une courbe NURBS est définie par des segments de courbe C_k qui se raccordent entre eux pour les valeurs de t données par les noeuds t_k avec $(k = p, ..., n+1)$ de T[1]. Comme la courbe est de localité L^{p+1} deux segments de courbe successifs C_k et C_{k+1} ont p points de contrôle en commun dans leur définition. Si on se place dans le cas où la multiplicité μ_k d'un noeud t_k est supérieure à 1, alors sa valeur sera répétée μ_k fois dans T. Cela signifie qu'il y aura, dans la courbe finale, $\mu_k - 1$ morceaux de courbe qui commencent et finissent sur la même valeur de t. Ce sont donc des segments de courbes restreints à un seul point ; on dira qu'ils sont de longueur nulle.

Par exemple, lorsque $\mu_k = 2$ le segment C_k est de longueur nulle. Pour construire la courbe globale il faudra donc raccorder C_{k-1} et C_{k+1} qui n'ont que $p-1$ points de contrôle en commun dans leur définition. Pour respecter la propriété d'enveloppe convexe stricte, le raccordement devra se faire impérativement dans la portion de l'espace délimitée par ces points de contrôle communs aux deux segments de courbe. De même, si $\mu_k = 3$, le raccordement devra se faire entre C_{k-1} et C_{k+2} qui n'ont que $p-2$ points de contrôle en commun et, dans le cas général, il faudra raccorder C_{k-1} et $\mathrm{C}_{k+\mu_k-1}$ qui n'ont que $p - \mu_k + 1$ points de contrôle en commun. C'est cette diminution du nombre de points de contrôle communs aux segments à raccorder lors de l'augmentation de la multiplicité d'un noeud, qui entraîne la diminution du degré de continuité maximum que l'on peut atteindre pour réaliser le raccordement en ce noeud.

Dans le cas précis où $\mu_k = p$, les deux segments de courbe à raccorder n'ont qu'un seul point de contrôle en commun, le raccordement se fait précisément sur ce point qui devient donc un point d'interpolation pour la courbe. Dans le cas où $\mu_k > p$, les deux segments de courbe n'ont plus aucun point de contrôle en commun et par conséquent la courbe n'est plus continue.

A titre d'illustration, la Figure 3.10 représente une courbe NURBS cubique pour laquelle on a progressivement augmenté la multiplicité d'un noeud de 1 à 4.

[1]Les p premiers et les p derniers noeuds de T ne sont pas, quant à eux, des valeurs de raccordement, mais ils sont nécessaires au calcul récursif des fonctions de bases des B-splines par l'Equation 3.2

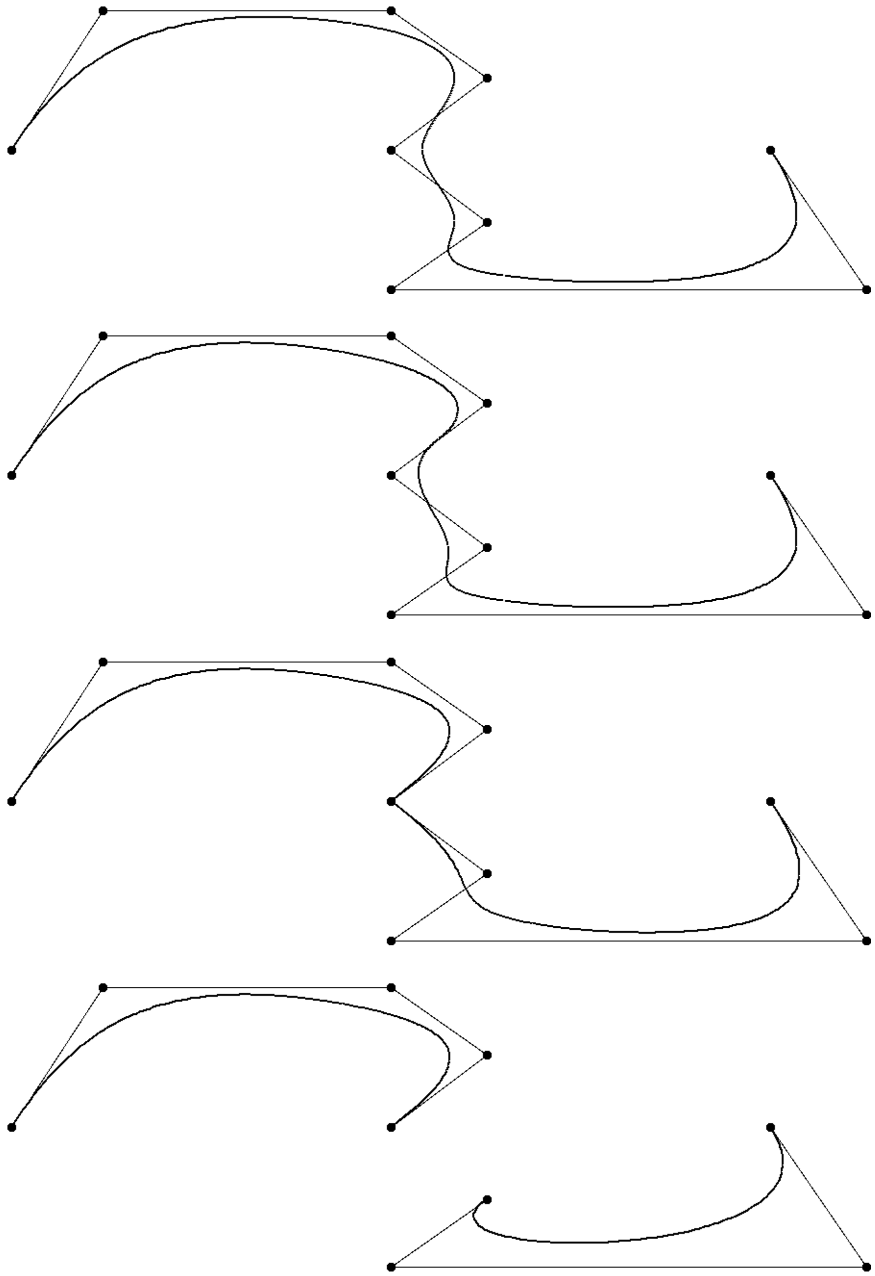


Figure 3.10: *Augmentation progressive de 1 à 4 de la multiplicité d'un noeud sur une courbe NURBS cubique*

Dans le cas d'une surface, on peut modifier la multiplicité d'un noeud u_i de U et/ou la multiplicité d'un noeud v_j de V. Nous allons détailler les effets produits dans le cas d'un noeud u_i, sachant que les effets sont similaires avec v_j.

Dans un premier temps, considérons un noeud u_i de U tel que $\mu_i = 1$. Ce noeud représente la valeur de u pour laquelle on raccorde les deux carreaux $S_{i-1,j}$ et $S_{i,j}$ pour $j = q..m+1$, c'est à dire $m+1-q$ paires de carreaux. Dans un deuxième temps, augmentons la multiplicité μ_i de u_i. Les carreaux qui doivent être raccordés en u_i seront, de manière analogue au cas des courbes, séparés par $\mu_i - 1$ carreaux de longueur nulle en u mais *non nulle* en v. De ce fait, le nombre de points de contrôle $P_{i,j}$ commun aux carreaux à raccorder ne diminue que dans la "direction" des i au fur et à mesure que μ_i augmente, ce qui est illustré sur la Figure 3.11 qui montre la frontière des carreaux pour $\mu_6 = 1$ et $\mu_6 = 3$.

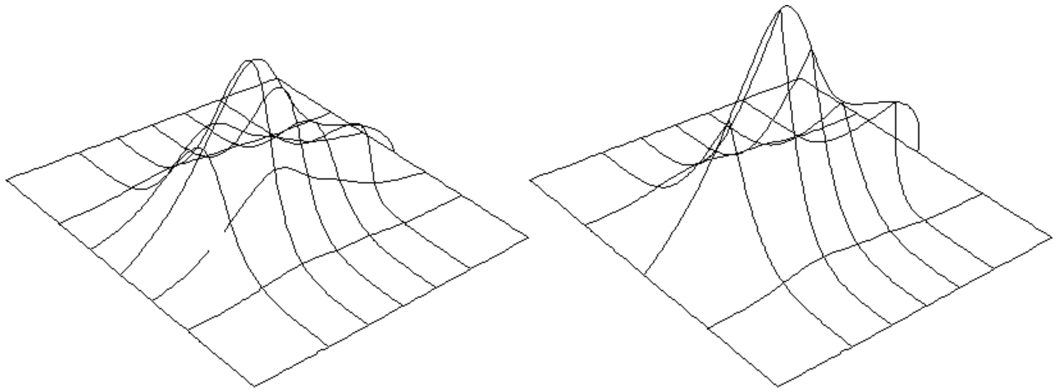


Figure 3.11: *Diminution du nombre de carreaux par augmentation de la multiplicité*

En particulier, lorsque $\mu_i = p$, cela signifie que les points de contrôle communs aux carreaux ne sont pas interpolés, contrairement au cas des courbes. Par contre, la diminution de la continuité se produit effectivement pour le paramètre u : le raccordement des carreaux est de continuité C^0, ce qui se traduit par une arête vive tout le long de l'isoparamétrique $u = u_i$ (partie gauche de la Figure 3.12). Si l'on désire interpoler un point de contrôle, il faut également imposer $\mu_j = q$, ce qui va se traduire par une arête vive supplémentaire tout le long de l'isoparamétrique $v = v_j$ (partie droite de la Figure 3.12).

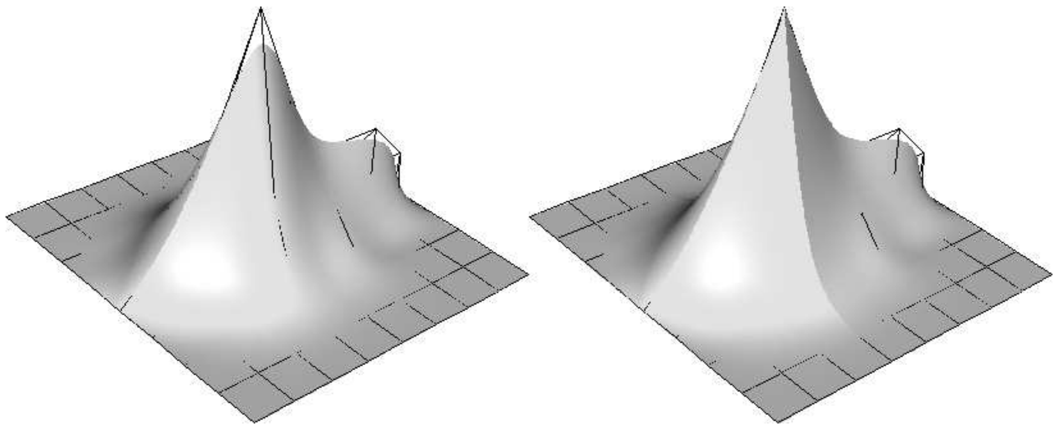


Figure 3.12: *Arêtes vives par augmentation de la multiplicité*

La création de telles arêtes vives est quelquefois intéressante dans un processus de modélisation géométrique. Cependant un inconvénient majeur de ce procédé est qu'une arête ainsi créée traverse la surface de bord à bord et qu'il est impossible de la localiser plus précisément. De plus, lorsque la multiplicité d'un noeud du vecteur nodal a été modifiée pour produire cet effet sur une surface, l'augmentation de la multiplicité d'un autre noeud dans ce vecteur, peut selon les cas, déplacer l'effet précédent sur la surface (*cf.* Figure 3.13). Ceci est dû au fait que l'augmentation (ou la diminution) de la multiplicité d'un noeud modifie la position des points de raccordement entre les carreaux et donc déplace également les modifications de forme rattachées à ces points. Par conséquent, les modifications produites sur une surface lors de la manipulation de la multiplicité des noeuds ne sont pas stables, ce qui est extrêmement perturbant lors de la création d'un objet.

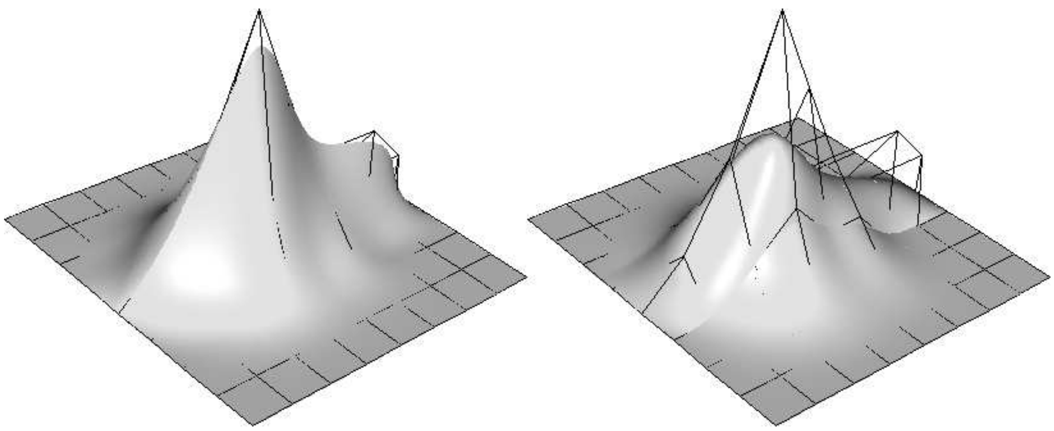


Figure 3.13: *Instabilité des effets produits lors des modifications de multiplicités*

B3 - Influence de la multiplicité des points

Un dernier moyen pour attirer une courbe ou une surface vers un point de contrôle P_k consiste à augmenter la multiplicité ν_k de ce point, c'est à dire à superposer plusieurs points voisins de P_k sur la position occupée par P_k.

Supposons que pour une courbe NURBS de degré p, nous ayons $\nu_k = 2$ (*i.e.* $P_k = P_{k+1}$). Dans ce cas, les p segments de courbes qui utilisent l'un et l'autre de ces deux points de contrôle dans leur définition ont une enveloppe convexe composée seulement de p points de contrôle différents (au lieu de $p+1$ si ν_k valait 1). De façon plus générale, les $p - \nu_k + 2$ segments de courbes qui utilisent simultanément ces ν_k points dans leur définition ont chacun une enveloppe convexe restreinte à $p - \nu_k + 2$ points de contrôle différents.

Donc, dans le cas précis où $\nu_k = p$, les deux seuls segments de courbes qui utilisent ces p points de contrôle confondus ont une enveloppe convexe restreinte à un segment de droite : $[P_{k-1}, P_k]$ pour le premier, $[P_k, P_{k+1}]$ pour le second. En raison de la propriété d'enveloppe convexe stricte, les morceaux de courbes correspondants seront donc inclus dans ces segments de droite. De plus ils se raccorderont sur P_k qui est leur seul point commun. Dans le cas où $\nu_k > p$, un raisonnement analogue permet de montrer que l'effet produit est le même que celui obtenu dans le cas précédent. En effet, augmenter la multiplicité de P_k au delà de p revient simplement à ajouter dans la courbe NURBS résultante, des segments de longueur nulle sur P_k, ce qui ne change pas la forme de la courbe.

Pour les surfaces, on peut de manière similaire augmenter la multiplicité de toute une ligne (ou de toute une colonne) de points de contrôle dans la grille. Visuellement certains des effets ainsi produits sont proches de ceux obtenus en augmentant la multiplicité d'un noeud. Par exemple la Figure 3.14 représente l'obtention d'une arête vive en augmentant soit la multiplicité d'un noeud (image de gauche), soit la multiplicité d'une ligne de points de contrôle (image de droite).

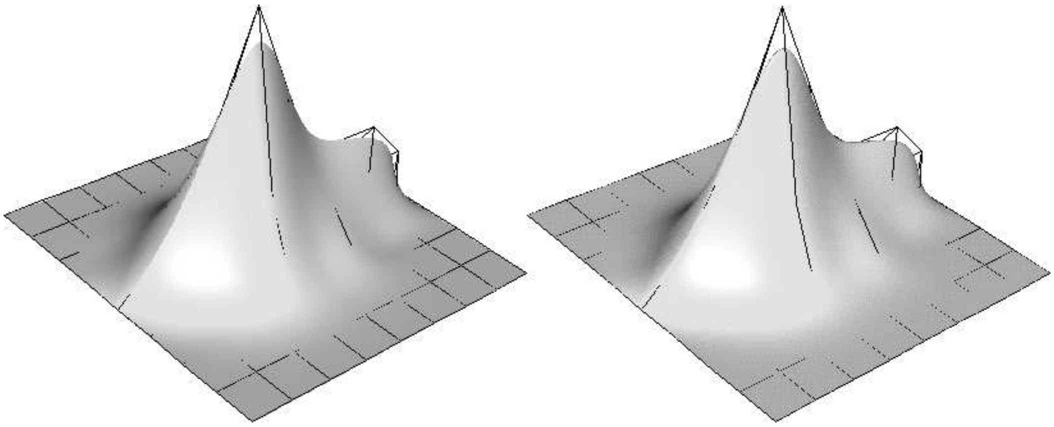


Figure 3.14: *Arêtes vives obtenus par deux processus différents*

Mais, contrairement à ceux obtenus lors des modifications de la multiplicité des noeuds, les effets produits par les manipulations de la multiplicité des points de contrôle sont localisables sur la surface. Par exemple, pour créer une arête qui ne se propage pas sur toute une longueur de la surface, il suffit d'augmenter uniquement la multiplicité de quelques points consécutifs sur une ligne (ou une colonne) du réseau de contrôle (*cf.* Figure 3.15).

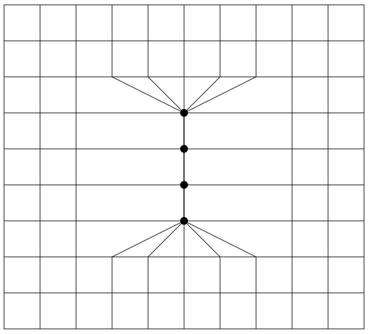


Figure 3.15: *Schéma de grille pour la localisation des arêtes vives*

La partie gauche de la Figure 3.16 représente une surface sur laquelle on peut voir une portion d'arête vive sur l'avant de la bosse principale ; la partie droite montre la même surface selon un autre point de vue, et permet de confirmer que l'arête ne s'est pas propagée sur l'arrière de cette bosse.

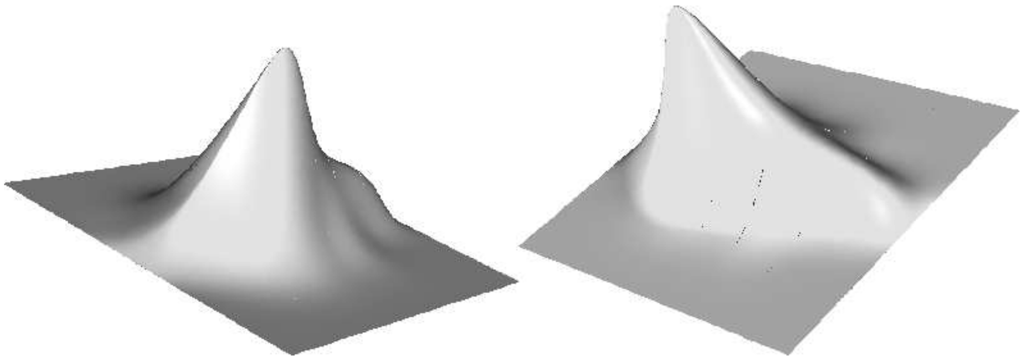


Figure 3.16: *Arêtes vives localisées*

Ainsi l'augmentation de la multiplicité des points permet d'obtenir des effets géométriques plus variés que ceux produits par l'augmentation de la multiplicité des noeuds[2]. Cependant, pour éviter de déformer excessivement la surface lors du déplacement des points que l'on souhaite superposer, il est préférable d'augmenter au préalable la définition du réseau de contrôle. On utilise pour cela des algorithmes spécifiques qui sont détaillés dans le paragraphe suivant.

C - Algorithmes de raffinement

Le rôle d'un algorithme de raffinement est d'augmenter le nombre de points de contrôle, et par conséquent de noeud[3], utilisés pour calculer l'équation d'une courbe ou d'une surface NURBS.

Ces algorithmes peuvent servir à atteindre plusieurs buts :

- On peut, par exemple, les utiliser pour diminuer la taille de la zone d'influence d'un point de contrôle sur une surface. En effet, plus le nombre de points de contrôle est élevé et plus la taille des carreaux qui composent la surface diminue. Ainsi pour une surface dont la définition a été raffinée, les $(p+1)(q+1)$ carreaux influencés par la position d'un point de contrôle $P_{i,j}$, représentent une portion plus petite de la surface totale que pour une surface définie plus grossièrement (*cf.* Figure 3.17)

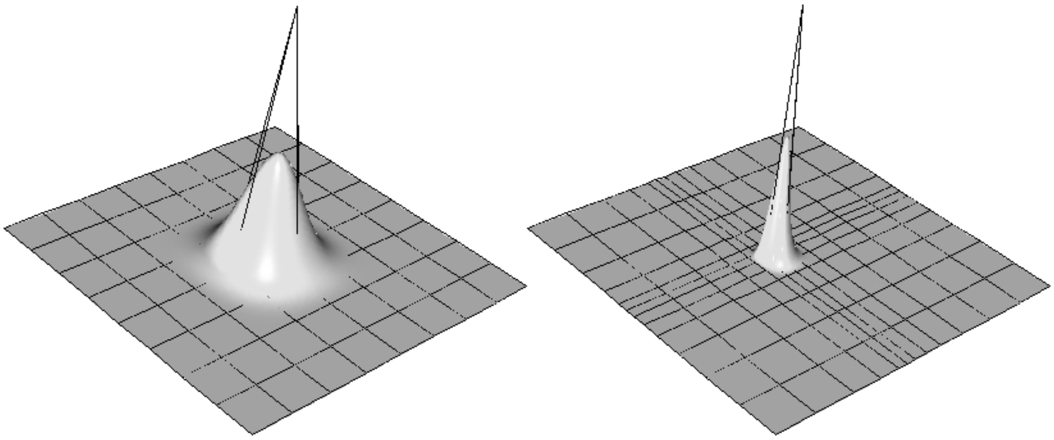


Figure 3.17: *Diminution de la zone d'influence d'un point de contrôle*

- On peut aussi raffiner la définition d'une surface dans le but d'obtenir plus de degré de liberté pour représenter une forme très accidentée ou très détaillée, ou encore pour représenter des arêtes vives selon le procédé décrit précédemment.

- Enfin on peut utiliser un algorithme de raffinement pour isoler une portion de la surface et permettre par exemple de déformer cette portion sans altérer le reste de la surface (*cf.* Figure 3.18)

[2] Avec néanmoins un coût de stockage légèrement plus important puisqu'il faut ajouter à la surface, une rangée complète de points de contrôle pour chaque ordre de multiplicité.

[3] Rappelons que dans le cas de courbes NURBS de degré p, le nombre de points de contrôle et le nombre de noeuds sont liés : pour n points de contrôle, la courbe est composée de $n+p+1$ noeuds.

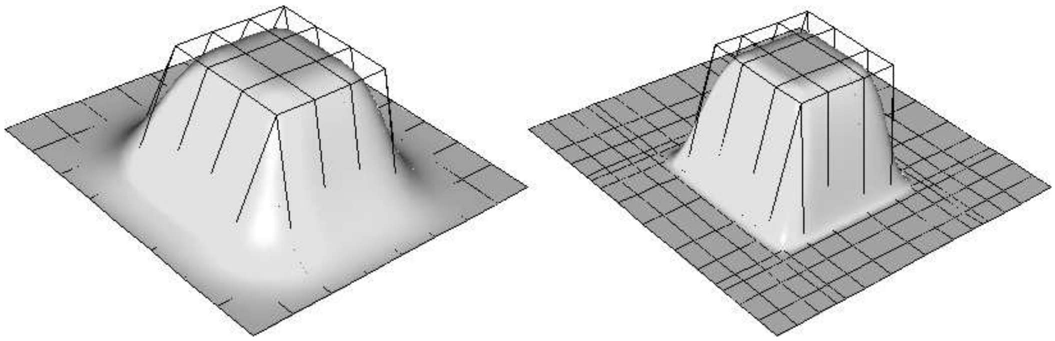


Figure 3.18: *Isolement d'une portion de surface*

Pour être utilisable dans le cadre d'un modeleur, le raffinement doit être réalisé sans modifier la forme de la courbe ou de la surface sur laquelle on l'applique. En effet, cela permet d'augmenter la définition d'un objet en cours de conception sans altérer les formes déjà produites, simplement pour favoriser, selon le but du raffinement, la suite de l'élaboration de l'objet.

Pour vérifier cette propriété, les algorithmes de raffinement doivent générer de nouveaux points de contrôle dont la position est liée à la valeur des nouveaux noeuds insérés dans les vecteurs nodaux. Concrètement, un algorithme de raffinement peut être conçu selon deux approches duales :

- *choisir la valeur des nouveaux noeuds* à insérer dans le vecteur nodal et s'en servir pour *calculer la position des nouveaux points de contrôle.*
- *choisir la position des nouveaux points de contrôle* sur le réseau et *calculer la valeur des nouveaux noeuds* en fonction de ces positions.

Au début des années 80, deux algorithmes de raffinement ont été développés de manière indépendante, l'algorithme de Boehm [BOEH80] et l'algorithme de Cohen-Lynch-Riesenfeld (dit "algorithme d'Oslo") [COHE80]. La différence principale entre ces deux techniques est que la première permet d'insérer un noeud unique, alors que la seconde consiste à transformer un vecteur nodal en un autre et donc permet d'insérer plusieurs noeuds simultanément. A priori, l'algorithme d'Oslo semble donc plus intéressant puisqu'il permet, par exemple, de réaliser en une seule application, un raffinement global. Néanmoins on peut montrer [BOEH85] que du fait de sa complexité, une application de l'algorithme d'Oslo est généralement plus coûteuse que plusieurs itérations successives de l'algorithme de Boehm.

Pour cette raison nous décrirons dans la suite de ce chapitre l'algorithme de Boehm qui a été initialement conçu pour insérer un nouveau noeud dans le vecteur nodal d'une courbe NUBS. Nous étendrons ensuite cette méthode au cas des courbes NURBS. Puis nous décrirons un algorithme proposé par Piegl [PIEG89] qui permet, dans le cas des courbes NURBS, d'insérer un point de contrôle dont les coordonnées sont choisies par l'utilisateur. Enfin, nous présenterons ces différents algorithmes dans le cas des surfaces.

C1 - Algorithme d'insertion de Boehm

Soit $T = (t_0, ..., t_{n+p+1})$ le vecteur nodal d'une courbe NUBS de degré p définie par l'équation :

$$\forall t \in [0,1] \quad C(t) = \sum_{k=0}^{n} N_{k,p}(t)\ P_k \tag{3.17}$$

On souhaite insérer dans T un nouveau noeud t' tel que $t' \in\ ]t_k, t_{k+1}]$. Ainsi le nouveau vecteur nodal sera $T = (t_0, ..., t_k, t', t_{k+1}, ..., t_{n+p+1})$.

On cherche à obtenir les nouveaux points de contrôle P'_k (avec $k = 0..n+1$) tels que :

$$\forall t \in [0,1] \quad C(t) = \sum_{k=0}^{n+1} N_{k,p}(t)\ P'_k \tag{3.18}$$

L'algorithme de Boehm fournit ces nouveaux points de contrôle de la façon suivante :

$$P'_0 = P_0 \qquad P'_{n+1} = P_{n+1} \qquad P'_i = (1-\alpha_i)\ P_{i-1} + \alpha_i\ P_i \tag{3.19}$$

$$\text{avec} \qquad \alpha_i = \begin{cases} 1 & \text{si } i = 1 \ .. \ k-p \\ \dfrac{t' - t_i}{t_{i+p} - t_i} & \text{si } i = k-p+1 \ .. \ k \\ 0 & \text{si } i = k+1 \ .. \ n \end{cases}$$

Si on compare la position des points P_k à celle des points P'_k obtenus grâce à l'algorithme de Boehm, on remarque que seule la position des points P'_i tels que $i = k-p+1 \ .. \ k$ est modifiée. Parmi ces p points, un seul est nouveau, les autres sont issus du déplacement de $p-1$ points P_k qui existaient avant l'insertion du nouveau noeud.

C2 - Extension de l'algorithme d'insertion de Boehm aux courbes NURBS

Si on conserve les mêmes notations et les mêmes conditions que précédemment, alors les nouveaux points de contrôle P'_k $(k = 0..n+1)$ et les nouveaux poids w'_k $(k = 0..n+1)$ tels que la courbe NURBS définie par l'Equation 3.1 s'écrive aussi :

$$\forall t \in [0,1] \quad C(t) = \frac{\displaystyle\sum_{k=0}^{n} w'_k\ N_{k,p}(t)\ P'_k}{\displaystyle\sum_{k=0}^{n} w'_k\ N_{k,p}(t)} \tag{3.20}$$

sont calculés à l'aide du α_i précédent, de la manière suivante :

$$P'_0 = P_0 \qquad P'_{n+1} = P_{n+1} \qquad P'_i = \frac{(1-\alpha_i)\ w_{i-1}\ P_{i-1} + \alpha_i\ w_i\ P_i}{(1-\alpha_i)\ w_{i-1} + \alpha_i\ w_i} \tag{3.21}$$

$$w'_0 = w_0 \qquad w'_{n+1} = w_{n+1} \qquad w'_i = (1-\alpha_i)\ w_{i-1}\ +\ \alpha_i\ w_i$$

Démonstration

On sait qu'une courbe NURBS définie par l'Equation 3.1 est la projection d'une courbe NUBS définie par :

$$\forall t \in [0,1] \quad \widehat{C}(t) = \sum_{k=0}^{n} N_{k,p}(t)\ \widehat{P}_k \tag{3.22}$$

avec $\widehat{P}_k = (w_k\ x_k, w_k\ y_k, w_k\ z_k, w_k)$. Donc si on choisit t' tel que : $t' \in\ ]t_k, t_{k+1}]$, on peut appliquer l'algorithme de Boehm à cette courbe pour insérer t' dans son vecteur nodal T.

On a alors

$$\forall t \in [0,1] \quad \widehat{C}(t) = \sum_{k=0}^{n+1} N_{k,p}(t)\ \widehat{P}'_k \tag{3.23}$$

où $\widehat{P}'_k$ sont des points définis par rapport à $\widehat{P}_k$ selon l'Equation 3.19.

De manière plus explicite, $\forall i = k-p+1 \;..\; k$:

$$\widehat{P}'_i = (x'_i, y'_i, z'_i, w'_i) \quad \text{avec} \quad \begin{cases} x'_i = (1-\alpha_i)\ w_{i-1}\ x_{i-1} + \alpha_i\ w_i\ x_i \\ y'_i = (1-\alpha_i)\ w_{i-1}\ y_{i-1} + \alpha_i\ w_i\ y_i \\ z'_i = (1-\alpha_i)\ w_{i-1}\ z_{i-1} + \alpha_i\ w_i\ z_i \\ w'_i = (1-\alpha_i)\ w_{i-1}\ + \alpha_i\ w_i \end{cases}$$

La projection du point $\widehat{P}'_i$ dans l'espace euclidien est $P'_i = (x'_i/w'_i, y'_i/w'_i, z'_i/w'_i,)$ et son poids est $w'_i = (1-\alpha_i)\ w_{i-1}\ + \alpha_i\ w_i$. Ainsi si on projette la courbe NUBS définie dans l'espace homogène par l'Equation 3.23, on obtient la courbe NURBS définie dans l'espace euclidien par l'Equation 3.20 □

C3 - Algorithme d'insertion de Piegl

Supposons que l'on souhaite insérer dans le réseau de contrôle d'une courbe NURBS définie par l'Equation 3.1, un point P' tel que $P' \in\ ]P_k, P_{k+1}]$. Ce point P' peut alors s'écrire de la façon suivante :

$$P' = \frac{(1-s)\ w_k\ P_k + s\ w_{k+1}\ P_{k+1}}{(1-s)\ w_k + s\ w_{k+1}} \tag{3.24}$$

$$\text{avec} \quad s = \frac{w_k\ \|P' - P_k\|}{w_k\ \|P' - P_k\| + w_{k+1}\ \|P' - P_{k+1}\|}$$

Dans ce cas, le noeud $t' = t_{k+1} + s\ (t_{k+p+1} - t_{k+1})$ est le nouveau noeud qu'il faut insérer dans T pour que P' soit un nouveau point de contrôle de la courbe. L'insertion de t' se fait simplement par l'algorithme de Boehm dans sa version étendue aux courbe NURBS, en déterminant l'entier j tel que $t' \in\ ]t_j, t_{j+1}]$.

Piegl a proposé cet algorithme [PIEG89] en tenant compte du fait qu'il est plus intuitif pour l'utilisateur d'un modeleur, qui n'est généralement pas familiarisé avec la théorie des courbes NURBS, de choisir la position d'un nouveau point de contrôle plutôt que la valeur d'un nouveau noeud. En effet, l'insertion d'un nouveau noeud engendre de manière peu prévisible, la création d'un nouveau point ainsi que le déplacement de $p-1$ points existants, ce qui rend le procédé quelquefois peu intuitif. Cependant il faut noter que l'algorithme de Piegl ne permet de choisir que la position du nouveau point et donc la nouvelle position des $p-1$ autres points peut également être déroutante.

La Figure 3.19 représente une courbe NURBS cubique définie par 8 points de contrôle (image en haut à gauche) sur laquelle on a inséré un noeud (et donc un point de contrôle) en utilisant différentes techniques. Sur l'image en haut à droite, l'insertion est obtenue en choisissant à la fois la position du nouveau point de contôle et la valeur du nouveau noeud. On constate, dans ce cas, que la courbe est déformée par rapport à la courbe initiale.

En bas à gauche, l'insertion est obtenue en utilisant l'algorithme de Boehm (dans sa version étendue aux NURBS) et donc en choisissant la valeur du noeud à insérer. En bas à droite, l'insertion est réalisée en utilisant l'algorithme de Piegl et donc en choisissant la position du nouveau point de contrôle. Par comparaison avec la courbe initiale, on constate que la courbe NURBS raffinée n'est pas déformée pour ces deux cas, seul son réseau de contrôle a été modifié. Pour l'application de l'algorithme de Piegl, nous avons choisi d'insérer un point de contrôle au milieu du segment de droite $[P_3, P_4]$ sur la courbe d'origine. Le nouveau point est effectivement sur cette position mais, dans la courbe finale, cette position ne marque plus le milieu de ses points voisins.

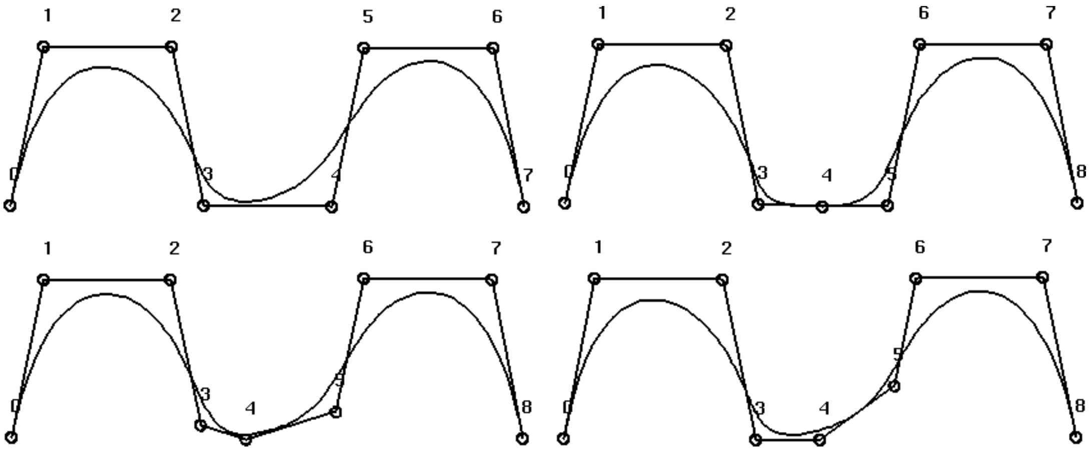


Figure 3.19: *Insertion d'un point de contrôle et d'un noeud*
(a) *Courbe initiale* **(b)** *Insertion par choix d'un point et d'un noeud*
(c) *Insertion par choix d'un noeud* **(d)** *Insertion par choix d'un point*

Pour étendre les algorithmes de raffinement aux surfaces NURBS, il faut tenir compte, d'une part, de la structure de grille de son réseau de contrôle et, d'autre part, du fait que toutes les lignes (respectivement les colonnes) de points de contrôle $P_{i,j}$, sont liées au même vecteur nodal U (respectivement V). Dans ce cas, il est clair que l'insertion d'un nouveau noeud dans un des deux vecteurs nodaux de la surface, engendrera une ligne (ou une colonne) entière de nouveaux points de contrôle sur la grille.

Pour réaliser une telle insertion avec l'algorithme de Boehm, il faut commencer par choisir la valeur u' du noeud que l'on souhaite insérer dans U. Puis on applique l'algorithme de Boehm, ligne par ligne, pour calculer sur chacune d'elles, la nouvelle position des p points de contrôle correspondants. Globalement cela revient donc à calculer la nouvelle position de p colonnes de points de contrôle (*cf.* Figure 3.20).

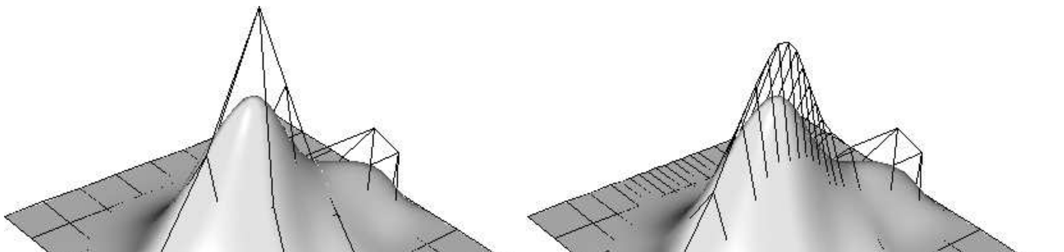


Figure 3.20: *Résultat de l'insertion successive de plusieurs noeuds*

Pour raffiner une surface en appliquant l'algorithme de Piegl, il faut d'abord choisir la position d'un nouveau point de contrôle P' que l'on souhaite insérer sur la grille de contrôle. Puis on calcule, selon le procédé décrit pour les courbes (*cf.* Equation 3.24), la valeur du noeud dont l'insertion générera P' sur le réseau. Enfin, on insère ce noeud en utilisant pour cela l'algorithme de Boehm tel que nous venons de le décrire pour une surface. Là encore, il y a p colonnes de points de contrôle dont la position est nouvelle. Et parmi tous ces points, un seul se trouve sur une position choisie par l'utilisateur.

Il faut noter que dans certains cas, les points calculés pour insérer le noeud correspondant à la création de P' ont des positions très variables d'une ligne à l'autre (*cf.* Figure 3.21). Ceci vient de l'utilisation des poids des points de contrôle dans le calcul de ces nouvelles positions. En effet, chaque point de contrôle possédant son propre poids le résultat de l'Equation 3.21 pour l'insertion du même noeud, peut être différent pour chaque ligne. Cette variation, parfois importante, de la position des nouveaux points ou du déplacement des points existants d'une ligne à l'autre, donne un aspect aléatoire au processus de raffinement, ce qui, bien entendu, est propice à troubler un utilisateur profane.

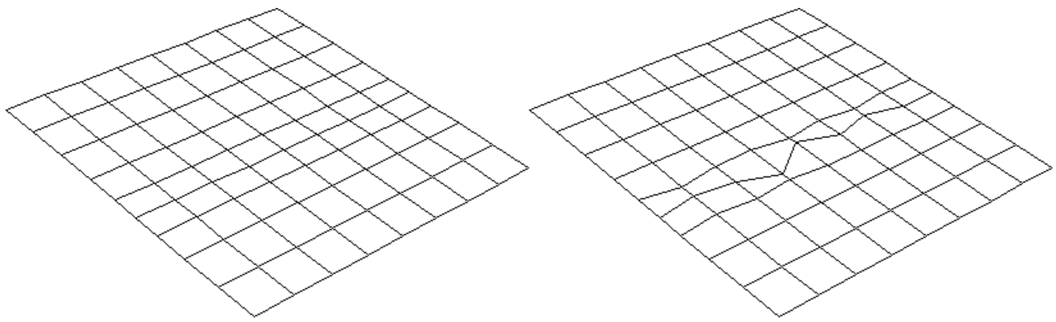


Figure 3.21: *Raffinement d'une surface dans le cas d'une NUBS (image de gauche) et dans le cas d'une NURBS avec des poids variables (image de droite)*

Bilan

Parmi les nombreux types de splines proposés dans la littérature, la famille des NURBS est devenu le standard de fait dans le domaine. En particulier, les NURBS constituent le seul standard défini par l'ISO pour l'échange de données géométriques destinées à la production industrielle.

Par rapport aux autres splines, les courbes et surfaces NURBS regroupent plusieurs avantages. D'une part, les NURBS disposent de nombreux degrés de liberté (coordonnées des points de contrôle, espacements et multiplicité des noeuds, valeurs des poids) qui permettent d'obtenir une grande variété de formes. D'autre part, les NURBS permettent de représenter de manière exacte des courbes et surfaces régulières (cercle, cylindre, sphère ou tore), comme nous allons le voir dans le chapitre suivant, mais également d'autres modèles de courbes et surfaces splines (B-splines uniformes ou extrémales, Bézier ou Bézier par morceaux). Enfin, les NURBS possèdent des algorithmes de raffinement qui permettent d'ajouter de nouveaux noeuds et de nouveaux points de contrôle sans modifier la forme initiale.

Néanmoins, nous avons montré dans ce chapitre que la manipulation d'un certain nombre des degrés de liberté figurant dans la définition des NURBS (en l'occurrence, les noeuds et les poids) est loin d'être simple. Ainsi, de nombreuses manipulations fournissent des résultats troublants, voire incompréhensibles, pour un utilisateur profane. En clair, cela signifie qu'une bonne connaissance du modèle mathématique des NURBS est indispensable pour tirer pleinement parti de toutes leurs capacités. Ceci est en contradiction avec notre but et c'est cette constatation qui nous a amené à développer un nouveau modèle de splines que nous détaillerons dans le Chapitre 5.

Chapitre IV

Cercles, sphères et splines

L'un des avantages généralement mis en avant pour expliquer la popularité des NURBS est le fait qu'elles permettent à la fois de définir des courbes et surfaces de formes libres (comme la plupart des modèles de splines) mais également de représenter, de manière exacte, les arcs de coniques et donc par extension, tout un ensemble de courbes et surfaces utilisées de manière intensive en CAO : arc de cercle, cercle, cylindre, cône, sphère, tore, surface de révolution, etc.

Pourtant, la littérature consacrée à la manière de représenter ces courbes et surfaces par des NURBS est extrêmement peu fournie. Sur le plan pratique, la plupart des implémentations se contentent d'utiliser la représentation du cercle et de la sphère décrites par Tiller [TILL83]. Sur le plan théorique, aucune étude exhaustive des propriétés mathématiques spécifiques liées à la représentation du cercle et de la sphère par des NURBS n'est disponible. A notre connaissance, les travaux les plus complets sur ce sujet ont été effectués par Farin [FARI89].

Ce chapitre ne prétend pas combler cette lacune théorique, il a simplement pour but de s'attaquer aux deux problèmes suivants :

- Toutes les représentations connues du cercle et de la sphère à l'aide de NURBS n'offrent qu'une continuité C^1. De plus, il n'existe aucune méthode qui permettrait de trouver une éventuelle représentation C^2, C^3 ou autre.
- Il est bien connu que la paramétrisation d'une courbe est d'autant meilleure qu'elle est proche de l'abscisse curviligne. De cette manière, on peut donc effectuer une analyse qualitative des différentes représentations du cercle et de la sphère. Mais là encore, il n'existe aucune méthode qui permettrait de trouver une éventuelle paramétrisation optimale.

La solution que nous proposons ici pour ces deux problèmes, consiste en une nouvelle méthode de reparamétrisation des NURBS, à l'aide d'une famille particulière de polynômes rationnels, que nous avons appelée la *reparamétrisation en zigzag*. Ce type de reparamétrisation permet aussi bien d'obtenir des continuités d'ordres plus élevés, que des paramétrisations plus proches de l'abscisse curviligne, au prix d'une certaine augmentation du degré des courbes rationnelles résultantes.

Avant de détailler le fonctionnement de cette technique de reparamétrisation, nous allons d'abord rappeler les principaux résultats connus concernant la représentation des coniques, cercles et sphères à l'aide de courbes et surfaces NURBS.

A - Représentation par splines du cercle et de la sphère

A1 - Arc de conique

L'interprétation géométrique classique, qui définit les coniques comme des sections planes d'un cône, est connue depuis les travaux d'Archimède dans l'Antiquité grecque. Toutefois, il existe une autre interprétation géométrique plus récente, dérivée des travaux de Moebius au début du siècle dernier, qui permet d'interpréter les coniques comme des projections dans $\mathbb{R}^2$ d'une parabole définie dans $\mathbb{R}^3$.

Partant de cette interprétation géométrique, on peut effectuer le raisonnement suivant, initialement proposé par Versprille [VERS75]. Comme les polynômes de Bernstein constituent une base de l'espace des polynômes, un arc de parabole dans $\mathbb{R}^3$ est également une courbe de Bézier de degré 2. Ainsi, comme nous l'avons vu dans le Chapitre II, cette courbe de Bézier devient rationnelle par projection, ce qui signifie qu'un *arc de conique est une courbe de Bézier rationnelle de degré 2*, et donc également une courbe NURBS de degré 2.

Une telle courbe est définie par trois points P_0, P_1, P_2 et trois coefficients w_0, w_1, w_2 selon l'équation :

$$\forall t \in [0,1] \quad C(t) = \frac{w_0\ B_0^2(t)\ P_0 + w_1\ B_1^2(t)\ P_1 + w_2\ B_2^2(t)\ P_2}{w_0\ B_0^2(t) + w_1\ B_1^2(t) + w_2\ B_2^2(t)} \tag{4.1}$$

ou encore, en explicitant les polynômes de Bernstein :

$$\forall t \in [0,1] \quad C(t) = \frac{w_0\ (1-t)^2\ P_0 + 2w_1\ (1-t)t\ P_1 + w_2\ t^2\ P_2}{w_0 - 2t(w_0 - w_1) + t^2(w_0 - 2w_1 + w)} \tag{4.2}$$

Lorsque la position des points P_0, P_1 et P_2 est fixée, les coefficients w_0, w_1 et w_2 procurent, a priori, trois degrés de liberté. Mais en fait, ces paramètres sont liés par l'*invariant de conique* $w_0 w_2 / w_1^2$ [VERS75]. Cela signifie que tout triplet (w_0, w_1, w_2) fournissant la même valeur pour l'invariant de conique, définit le même segment de courbe $C(t)$. D'autre part, comme P_0 et P_2 jouent un rôle symétrique, il est préférable d'avoir $w_0 = w_2$, et par conséquent, il n'y a plus qu'un véritable degré de liberté. Enfin, grâce à l'existence de l'invariant de conique, on peut poser sans perte de généralité $w_0 = w_2 = 1$ et $w_1 = w$, ce qui nous donne une reformulation de l'Equation 4.2 :

$$\forall t \in [0,1] \quad C(t) = \frac{(1-t)^2\ P_0 + 2w\ (1-t)t\ P_1 + t^2\ P_2}{1 - 2t(1-w) + 2t^2(1-w)} \tag{4.3}$$

L'avantage de cette reformulation est qu'elle permet de déterminer facilement le type de la conique résultante [VERS75]. En effet, on obtient (*cf.* Figure 4.1) :

- $w < 1$: $C(t)$ est un arc d'ellipse
- $w = 1$: $C(t)$ est un arc de parabole
- $w > 1$: $C(t)$ est un arc d'hyperbole

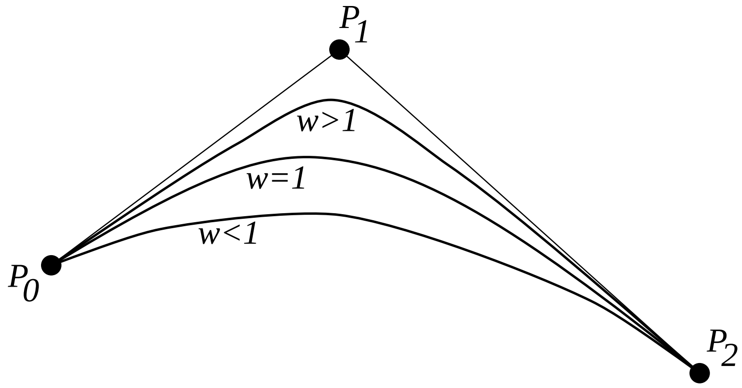


Figure 4.1: *Arcs de conique*

A2 - Arc de cercle

Parmi les arcs de coniques, l'arc de cercle est très certainement le cas particulier le plus utile pour la modélisation géométrique. Supposons plus précisément que l'on désire représenter un arc du cercle unité démarrant de l'origine trigonométrique. Comment faut-il positionner les points P_0, P_1 et P_2 pour représenter cet arc de cercle avec une courbe NURBS ?

Pour des raisons de symétrie, il est évident que le triangle $P_0P_1P_2$ définissant l'arc de cercle doit être isocèle. De plus, si on appelle ϕ l'angle $(\overline{P_0P_1}, \overline{P_0P_2})$ (*cf.* Figure 4.2), on montre qu'il faut avoir $w = \cos\phi$ [VERS75]. Par quelques équivalences trigonométriques, on obtient alors :

$$P_0\ (1,0) \qquad P_1\ (1,\tan\phi) \qquad P_2\ (\cos 2\phi, \sin 2\phi) \tag{4.4}$$

ou encore, en posant $T = \tan\phi$:

$$P_0\ (1,0) \qquad P_1\ (1,T) \qquad P_2\ (\frac{1-T^2}{1+T^2}, \frac{2T}{1+T^2}) \tag{4.5}$$

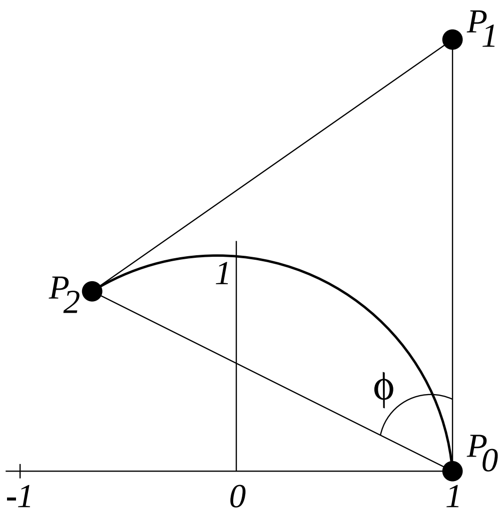


Figure 4.2: *Arc de cercle*

Comme les points P_0 et P_2 sont interpolés, cela signifie que l'arc correspondant varie entre les valeurs angulaires 0 et 2ϕ. Par conséquent, le triangle $P_0P_1P_2$ ainsi positionné permet de représenter un arc de longueur 2ϕ. En combinant l'Equation 4.3 et l'Equation 4.5, on obtient une paramétrisation de cet arc de cercle par un polynôme rationel :

$$\forall t \in [0,1] \qquad C(t) = \left(\frac{1 - 2(1-w)t - 2(1-w)t^2}{1 - 2(1-w)t + 2(1-w)t^2} \, , \, \frac{2Twt - 2Tw(1-w)t^2}{1 - 2(1-w)t + 2(1-w)t^2} \right) \tag{4.6}$$

A3 - Evaluation de la paramétrisation

Il existe une autre paramétrisation de l'arc de cercle par un polynôme rationnel, généralement plus connue, qui utilise la tangente de l'angle moitié. La forme habituelle consiste à représenter l'arc de cercle de longueur $\pi/2$ situé dans le premier quadrant du cercle trigonométrique par :

$$\forall s \in [0,1] \qquad C(s) = \left(\frac{1-s^2}{1+s^2} \, , \, \frac{2s}{1+s^2} \right) \tag{4.7}$$

Mais cette expression peut se généraliser à tout arc de cercle de longueur 2ϕ (avec $\phi \in \,]0, \pi/2[$) démarrant de l'origine trigonométrique :

$$\forall s \in [0,1] \qquad C(s) = \left(\frac{1-T^2s^2}{1+T^2s^2} \, , \, \frac{2Ts}{1+T^2s^2} \right) \tag{4.8}$$

L'avantage de cette représentation de l'arc de cercle est qu'elle fournit une relation entre son paramètre $s \in [0,1]$ et son abscisse curviligne $\theta \in [0, 2\phi]$:

$$Ts = \tan\frac{\theta}{2} \qquad \text{d'où} \qquad \theta = 2 \text{ atan } Ts \tag{4.9}$$

L'abscisse curviligne représente la paramétrisation *canonique* (également appelée *parfaite*) d'une courbe, dans le sens où les variations de l'abscisse curviligne entre deux points représentent la distance exacte parcourue sur la courbe entre ces deux points. Parmi toutes les paramétrisations non-canoniques d'une courbe, il existe une famille particulière de paramétrisations dites *idéales*, pour lesquelles l'abscisse curviligne est directement proportionnelle à la valeur du paramètre, en tout point de la courbe. Plus précisément, on considère habituellement que plus on s'éloigne de cette relation de proportionnalité, plus la paramétrisation est mauvaise — nous verrons plus loin que la qualité d'une paramétrisation peut être quantifiée précisément en évaluant l'écart cordal entre le paramètre et l'abscisse curviligne.

Dans le cas de l'arc de cercle, notre paramètre s réalise une paramétrisation idéale lorsque $\forall s \in [0,1]$ $\theta = 2\phi s$. Au vu de l'Equation 4.9, il est évident que cet idéal n'est jamais obtenu[1] quelque soit la valeur de ϕ. L'étude de la fonction `atan` permet même de préciser que plus ϕ est grand, moins la paramétrisation de l'arc de cercle par s est correcte.

Ce résultat est confirmé par la partie gauche de la Figure 4.3 qui montre que la fonction $\theta(s)$ devient de moins en moins "rectiligne" lorsque l'angle ϕ augmente. Pour illustrer plus précisément cette non-linéarité, la partie droite de la figure montre les variations de la dérivée de $\theta(s)$. Dans le cas idéal, les courbes résultantes forment des droites horizontales placées à divers niveaux, puisque la pente doit être constante pour un angle ϕ donné. On constate qu'avec la paramétrisation par s, on s'écarte très vite de cet idéal.

[1]En fait, la paramétrisation idéale d'un arc de cercle est constituée par la paramétrisation trigonométrique $(\cos\theta, \sin\theta)$ qui, bien évidemment, ne peut pas être représentée de manière exacte par un polynôme rationnel.

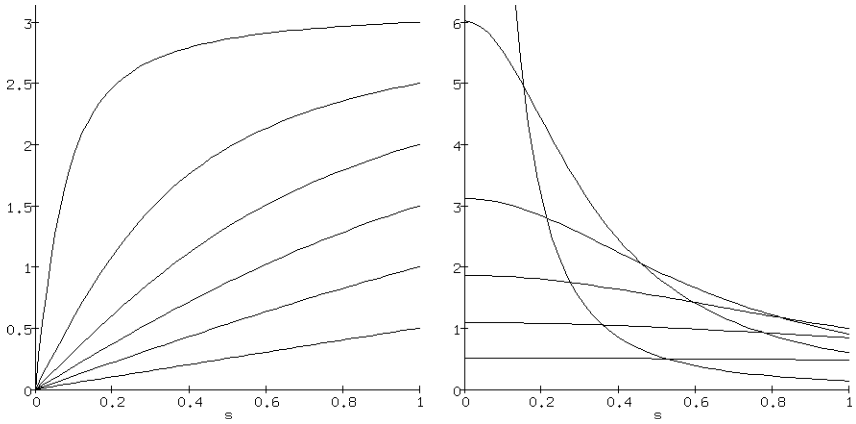


Figure 4.3: *$\theta(s)$ et $\theta'(s)$ pour différentes valeurs de ϕ*

Pour évaluer, de manière similaire, la paramétrisation de l'arc de cercle définie par l'Equation 4.6, il faut arriver à exprimer θ en fonction de t, ou plus simplement s en fonction de t (puisque l'on connait déjà θ en fonction de s). En fait, l'Equation 4.6 et l'Equation 4.8 constituent deux paramétrisations différentes du même arc de cercle. Or, un résultat fondamental, dû à Moebius, permet d'affirmer que deux paramétrisations d'une même courbe par des polynômes rationnels de même degré sont liés par une fonction homographique, c'est-à-dire :

$$\exists (a,b,c,d) \in \mathbb{R}^4 \quad / \quad s = \frac{a+bt}{c+dt} \tag{4.10}$$

Dans notre cas, comme les deux paramétrisations coincident pour les valeurs 0 et 1, on peut affirmer plus précisément que :

$$\exists p \in \mathbb{R} \quad / \quad s = \frac{(1+p)t}{1+pt} \tag{4.11}$$

Pour trouver la valeur de p, il suffit d'égaler l'Equation 4.6 et l'Equation 4.8 terme à terme. On obtient $p = w - 1$, c'est-à-dire :

$$s = \frac{wt}{1-t+wt} \tag{4.12}$$

et donc finalement

$$\theta = 2 \text{ atan } \frac{t\sqrt{1-w^2}}{1-t+wt} \tag{4.13}$$

Le tracé de cette fonction (*cf.* Figure 4.4) permet d'établir le fait que la courbe se met à osciller autour de la droite idéale lorsque l'angle ϕ augmente. Là encore, le tracé de la pente de la fonction permet de mieux évaluer les caractéristiques de la paramétrisation par t. Néanmoins, les variations par rapport à cette droite sont beaucoup moins importantes que celles engendrées par la paramétrisation précédente. En fait, on peut montrer [TILL83] que parmi toutes les paramétrisations de l'arc de cercle par un polynôme rationnel de degré 2, la paramétrisation par t définie ici (obtenue en posant $w_0 = w_2 = 1$ dans l'Equation 4.2) est celle qui se rapproche le plus de la paramétrisation trigonométrique.

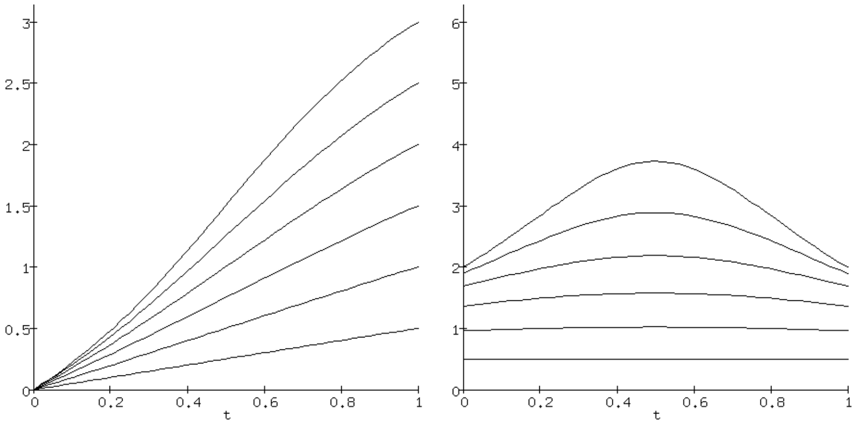


Figure 4.4: *$\theta(t)$ et $\theta'(t)$ pour différentes valeurs de ϕ*

A4 - Cercle et sphère

L'Equation 4.6 permet d'obtenir un arc de cercle d'une longueur 2ϕ strictement inférieure à π. Par conséquent, pour obtenir un cercle complet, il faut concaténer au minimum trois arcs de cercle. Par cette opération de concaténation, le point final de chaque arc devient le point initial de l'arc suivant, et donc, un cercle constitué de n arcs sera défini par $2n+1$ points de contrôle. La courbe obtenu par ce processus de construction peut ainsi être considérée soit comme une courbe de Bézier par morceaux rationnelle, soit comme une courbe NURBS. Il faut noter qu'a priori rien n'interdit de concaténer des arcs de longueurs différentes, mais seule la concaténation d'arcs identiques permet d'obtenir une paramétrisation homogène pour le cercle résultant. Parmi toutes les constructions possibles, les trois modèles suivants nous semblent particulièrement intéressants.

$n = 3$ (Cercle de Versprille)

Il s'agit de la première représentation exacte d'un cercle par une courbe NURBS proposée dans la littérature [VERS75]. Dans ce modèle, le cercle est composé de 3 arcs de longueur $2\pi/3$ et défini par 7 points de contrôle répartis en *triangle équilatéral* (*cf.* partie gauche de la Figure 4.5) :

$$P_0(1,0) \quad P_1(1,\sqrt{3}) \quad P_2(\frac{-1}{2},\frac{\sqrt{3}}{2}) \quad P_3(-2,0) \quad P_4(\frac{-1}{2},\frac{-\sqrt{3}}{2}) \quad P_5(1,-\sqrt{3}) \quad P_6(1,0)$$

où les points d'indices pairs (respectivement impairs) ont pour poids 1 (respectivement 1/2).

$n = 4$ (Cercle de Tiller)

L'inconvénient principal du cercle de Verprille est le fait que l'enveloppe convexe formé par le polygone de contrôle est très étendue par rapport à l'espace réellement occupée par la courbe. Une amélioration, proposée dans [TILL83], consiste à définir le cercle par 4 arcs de longueur $\pi/2$ en utilisant 9 points de contrôle répartis en *carré* (*cf.* partie centrale de la Figure 4.5) :

$$P_0(1,0) \quad P_1(1,1) \quad P_2(0,1) \quad P_3(-1,1) \quad P_4(-1,0) \quad P_5(-1,-1) \quad P_6(0,-1) \quad P_7(1,-1) \quad P_8(1,0)$$

où les points d'indices pairs (respectivement impairs) ont pour poids 1 (respectivement $\sqrt{2}/2$).

$n = 6$ (Cercle hexagonal)

Le cercle de Tiller est de loin le plus utilisé dans les logiciel de modélisation géométrique, principalement à cause de la simplicité du positionnement des points de contrôle. Néanmoins, il existe un autre modèle simple (qui n'a jamais été utilisé, à notre connaissance) qui procure une bien meilleure enveloppe convexe ainsi qu'une meilleure paramétrisation : le cercle composé de 6 arcs de longueur $\pi/3$ et défini par 13 points de contrôle répartis en *hexagone* (*cf.* partie droite de la Figure 4.5) :

$$P_0(1,0) \quad P_1(1,\frac{\sqrt{3}}{3}) \quad P_2(\frac{1}{2},\frac{\sqrt{3}}{2}) \quad P_3(0,\frac{2\sqrt{3}}{2}) \quad P_4(\frac{-1}{2},\frac{\sqrt{3}}{2}) \quad P_5(-1,\frac{\sqrt{3}}{3}) \quad \cdots \quad P_{12}(1,0)$$

où les points d'indices pairs (respectivement impairs) ont pour poids 1 (respectivement $\sqrt{3}/2$).

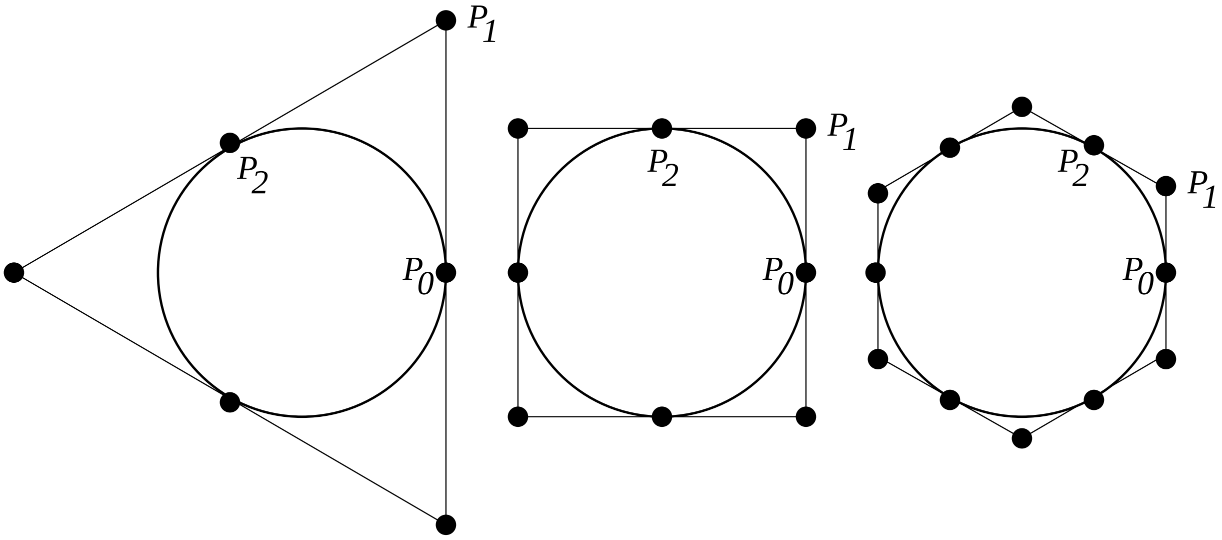


Figure 4.5: *Trois modèles de cercles définis par une courbe NURBS*

Sphère

Représenter une sphère par une surface NURBS, lorsque l'on dispose d'un modèle NURBS pour le cercle, se fait de manière naturelle en effectuant le produit tensoriel d'un cercle (qui va définir les parallèles) par un demi-cercle (qui va définir les méridiens).

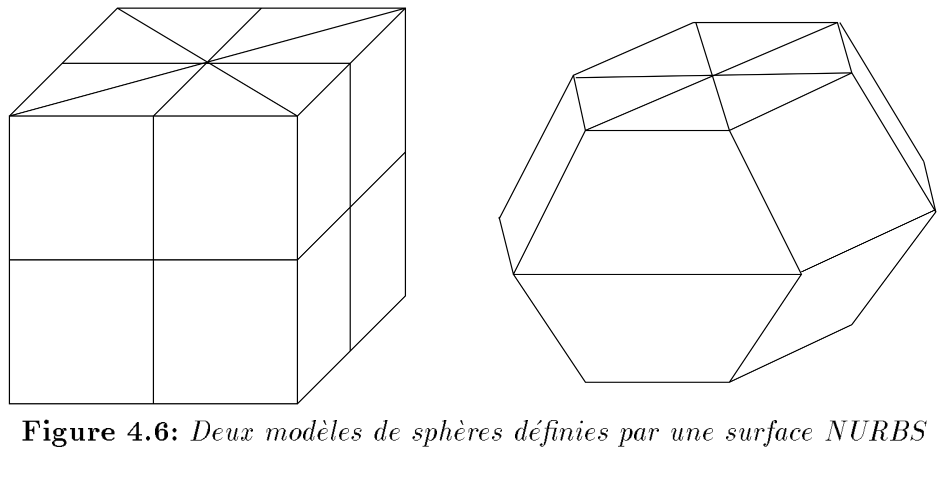


Figure 4.6: *Deux modèles de sphères définies par une surface NURBS*

Il faut noter que, si le cercle peut être représenté par n'importe lequel des trois modèles précédents, le demi-cercle ne peut être défini que par dichotomie du cercle de Tiller ou du cercle hexagonal. En combinant tous ces modèles, on obtient en fait six possibilités différentes pour représenter une sphère par une surface NURBS. La plus connue de ces représentations, proposée dans [TILL83], est définie par un réseau de forme cubique (composé de 45 points de contrôle), mais le cercle hexagonal peut également être utilisé pour obtenir un réseau de forme hexaèdrique (composé de 91 points de contrôle) plus proche de la surface (*cf.* Figure 4.6).

A5 - Etude de la continuité

Comme le cercle est obtenu par concaténation de segments de courbes distincts (chaque segment représentant un arc de cercle) le problème de la continuité aux points de jonction se pose naturellement. Pour étudier cette continuité, plaçons-nous dans le cas où l'on concatène deux segments définis respectivement par les triangles $P_0P_1P_2$ et $P_2P_3P_4$ (*cf.* Figure 4.7). Si les arcs sont de même longueur, on a nécessairement $w_0 = w_2 = w_4 = 1$ et $w_1 = w_3 = w$. On suppose, d'autre part, que la variation du paramètre t est l'intervalle $[0, 1]$ pour le premier segment et $[1, 2]$ pour le second. En utilisant la notation 1^- et 1^+ pour représenter les limites à gauche et à droite (*cf.* Chapitre II), on trouve :

$$\begin{array}{lll} C(1^-) = P_2 & C'(1^-) = 2w\ (P_2 - P_1) & C''(1^-) = 2\ (P_0 - P_2) + 4w\ (1-w)\ (P_1 - P_2) \\ C(1^+) = P_2 & C'(1^+) = 2w\ (P_3 - P_2) & C''(1^+) = 2\ (P_4 - P_2) + 4w\ (1-w)\ (P_3 - P_2) \end{array} \tag{4.14}$$

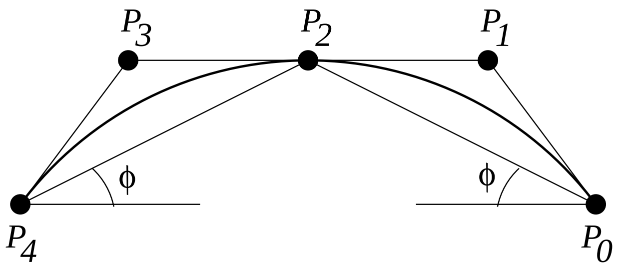


Figure 4.7: *Raccord de deux arcs de cercle*

Comme les deux arcs sont jointifs, on a, de manière évidente, une continuité C^0. De plus, comme les deux arcs ont la même longueur, les triangles $P_0P_1P_2$ et $P_2P_3P_4$ sont semblables, ce qui signifie en particulier que :

$$P_2 - P_1 = P_3 - P_2 \tag{4.15}$$

Par conséquent, la jonction possède également une continuité C^1. D'autre part, du fait de la similitude des triangles $P_0P_1P_2$ et $P_2P_3P_4$, on a également (*cf.* Figure 4.7) :

$$P_4 - P_2 = P_0 - P_2\ +\ 2(1 + \cos 2\phi)\ (P_2 - P_1) = P_0 - P_2\ +\ 4w^2\ (P_2 - P_1) \tag{4.16}$$

En substituant l'Equation 4.16 dans l'expression de $C''(1^+)$, on constate que les seules possibilités pour avoir $C''(1^-) = C''(1^+)$ sont $w = 0$ (c'est-à-dire un arc de longueur π, ce qui est interdit par définition) et $w = 1$ (c'est-à-dire un arc de longueur nulle, ce qui est sans intérêt). Une conséquence directe de ce résultat est que *l'utilisation de l'Equation 4.1 pour représenter un arc de cercle ne permet pas de définir un cercle possédant une paramétrisation C^2.*

B - Reparamétrisation en zigzag

La partie A de ce chapitre a permis d'établir que la représentation habituelle des cercles et sphères par des courbes et surfaces NURBS souffre de deux défauts principaux : d'une part, la paramétrisation peut s'éloigner de manière non négligeable de l'abscisse curviligne, et d'autre part, la continuité de cette paramétrisation est, au mieux, d'ordre C^1.

Ces deux problèmes ont en fait la même origine : comme les courbes utilisées sont de degré 2, il n'y a pas suffisamment de degrés de liberté pour arriver à corriger l'un ou l'autre (ou les deux) défauts. L'idée qui se présente donc naturellement consiste à essayer d'élever le degré des courbes afin de disposer d'un plus grand nombre de paramètres à manipuler. Malheureusement, le procédé classique [BART87, FARI90] pour élever le degré d'une courbe de Bézier ou d'une courbe B-spline sans en modifier la forme ne convient pas du tout dans ce cas précis. En effet, non seulement l'ordre de la continuité ne change pas par cette opération [FARI90], mais en plus, les oscillations de la paramétrisation autour de la droite idéale, deviennent plus importantes à chaque élévation de degré.

C'est pourquoi nous proposons ici une nouvelle méthode (spécifique aux courbes rationnelles) pour élever le degré d'une courbe sans en modifier la forme, qui permet de contrôler précisément la paramétrisation résultante.

B1 - Principe

Le principe général de cette méthode est inspiré de la reparamétrisation homographique de Moebius. En effet, comme nous l'avons vu précédemment (*cf.* Equation 4.10), la reparamétrisation d'une courbe rationnelle[2] par une fonction homographique ne change ni sa forme, ni son degré. Un corollaire de ce résultat est qu'une reparamétrisation par un polynôme rationnel de degré 2 double le degré de la courbe initiale, un polynôme rationnel de degré 3 triple le degré, et ainsi de suite. Ainsi, partant de la paramétrisation de l'arc de cercle définie par l'Equation 4.6, on peut représenter le même arc par une courbe rationnelle de degré 4 en remplaçant t par $u = S(t)$ où S est un polynôme rationnel de degré 2.

A priori, il y a 6 degrés de liberté pour définir u :

$$u = \frac{a + bt + ct^2}{d + et + ft^2}$$

mais en fait, un certain nombre de contraintes spécifiques apparaissent souhaitables pour cette reparamétrisation. Par exemple, il est primordial que les deux paramètres aient les mêmes domaines de variation et donc qu'ils coincident en 0 et en 1 (*i.e.* $S(0) = 0$ et $S(1) = 1$). D'autre part, du fait de l'isocélisme du triangle de contrôle, la symétrie par rapport à $1/2$ que présente la paramétrisation par t doit être conservée (*i.e.* $S(1-t) = 1 - S(t)$, ce qui peut s'exprimer également par $S'(1-t) = S'(t)$). Toutes ces contraintes font qu'il n'y a finalement plus qu'un seul degré de liberté. Si l'on appelle ce degré p et qu'on le définit comme étant la dérivée aux extrémités (*i.e.* $S'(0) = S'(1) = p$), on obtient :

$$u = \frac{pt + (1-p)t^2}{1 - 2(1-p)t + 2(1-p)t2} \tag{4.17}$$

La forme de cette fonction pour différentes valeurs de p (*cf.* Figure 4.8) nous a inspiré le nom de notre nouvelle technique *reparamétrisation en zigzag*. Il faut noter que le paramètre p doit impérativement être positif ou nul, pour que le domaine de variation de u soit le même que celui de t (en l'occurrence $[0, 1]$).

[2]Le théorème de Moebius n'a été étendu aux courbes rationnelles par morceaux que très récemment [LEE91], c'est-à-dire après un délai de plus de 170 ans !

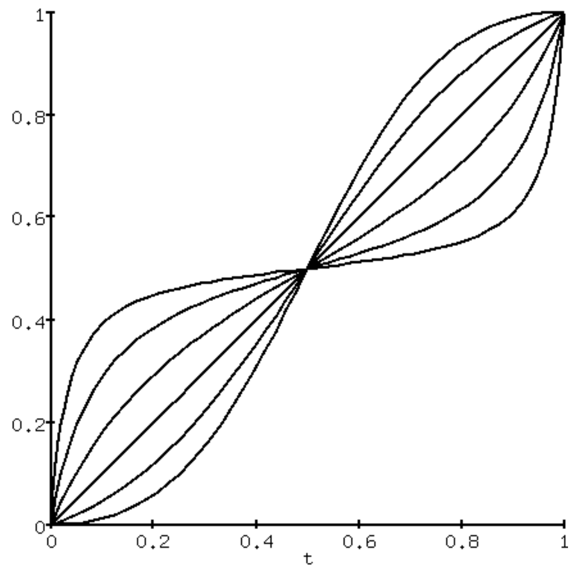


Figure 4.8: *Fonction de reparamétrisation en zigzag : $u = S(t)$*

B2 - Reparamétrisation en zigzag d'un arc de conique

Si l'on part de la courbe de Bézier rationnelle définie par l'Equation 4.18 et que l'on effectue une reparamétrisation par l'Equation 4.17, on obtient :

$$C(t) = \frac{a^2\ P_0 + 2wab\ P_1 + b^2\ P_2}{a^2 + 2wab + b^2} \tag{4.18}$$

$$\text{avec} \qquad a = 1 - (2-p)t + (1-p)t^2 \qquad \text{et} \qquad b = pt + (1-p)t^2$$

Le résultat est une courbe rationnelle de degré 4, mais qui *n'est plus une courbe de Bézier rationnelle* puisqu'elle n'est plus exprimée à l'aide des polynômes de Bernstein. Mais comme ces polynômes constituent justement une base de l'espace des polynômes, on est assuré qu'il existe une courbe de Bézier rationnelle de degré 4 qui est équivalente à l'Equation 4.18. Trouver cette courbe de Bézier consiste en fait à trouver les points $(Q_0, Q_1, Q_2, Q_3, Q_4)$ et les poids $(w_0, w_1, w_2, w_3, w_4)$ tels que :

$$C(t) = \frac{w_0(1-t)^4\ Q_0 + 4w_1t(1-t)^3\ Q_1 + 6w_2t^2(1-t)^2\ Q_2 + 4w_3t^3(1-t)\ Q_3 + w_4t^4\ Q_4}{w_0(1-t)^4 + 4w_1t(1-t)^3 + 6w_2t^2(1-t)^2 + 4w_3t^3(1-t) + w_4t^4} \tag{4.19}$$

Comme les points extrémités Q_0 et Q_4 sont interpolés, on en déduit immédiatement $Q_0 = P_0$ et $Q_4 = P_2$. D'autre part, on peut poser $w_0 = 1$ et $w_4 = 1$, sans perte de généralité. Les poids w_1, w_2 et w_3 s'obtiennent ensuite en comparant, terme à terme, le dénominateur de l'Equation 4.19 avec celui de l'Equation 4.18. Enfin, les points Q_1, Q_2 et Q_3 s'obtiennent de manière similaire, en comparant les deux numérateurs.

Finalement, partant d'une courbe de Bézier rationnelle de degré 2 définie par l'Equation 4.3, la reparamétrisation en zigzag fournit une courbe de Bézier rationnelle de degré 4 définie par :

$$Q_0 = P_0 \quad Q_1 = \frac{P_0 + wP_1}{1+w} \quad Q_2 = \frac{p^2P_0 + 2w(1+p^2)P_1 + p^2P_2}{2(p^2 + p^2w + w)} \quad Q_3 = \frac{P_2 + wP_1}{1+w} \quad Q_4 = P_2$$

$$w_0 = 1 \quad w_1 = \frac{1+w}{2}p \quad w_2 = \frac{p^2 + p^2w + w}{3} \quad w_3 = \frac{1+w}{2}p \quad w_4 = 1 \tag{4.20}$$

Trois remarques importantes peuvent être faites concernant ce résultat :

- La position des points Q_0, Q_1, Q_3, Q_4 est indépendante de valeur du paramètre p. Plus précisément, cela signifie que la reparamétrisation en zigzag possède une interprétation géométrique très simple : changer la valeur de p consiste à déplacer le point Q_2 le long d'une droite (en fait, une demi-droite puisqu'on impose $p \geq 0$). C'est la modification simultanée de w_1, w_2 et w_3 qui permet d'avoir une courbe inchangée (exceptée sa paramétrisation) malgré le déplacement de Q_2.

- Lorsque $p = 1$, on retrouve les points et les poids obtenus par le procédé classique d'élévation du degré d'une courbe de Bézier, déjà évoqué plus haut. Concrètement, cela signifie que la reparamétrisation en zigzag constitue une généralisation de ce procédé, pour laquelle on dispose d'un degré de liberté supplémentaire.

- Le principe de cette reparamétrisation peut se généraliser au delà du degré 2. Très brièvement, cela revient à définir une famille de polynômes rationnels pour laquelle chaque représentant $S_k(t)$ de degré k vérifie :

$$S_k(0) = 0 \qquad S_k(1) = 1 \qquad \forall i = 0..k \quad \exists p_i \in \mathbb{R}^+ \ / \ \frac{d^i S}{dt^i}(0) = p_i \quad \frac{d^i S}{dt^i}(1) = (-1)^{i-1}\ p_i$$

  Ainsi, les représentants consécutifs de cette famille possèdent chacun un degré de liberté supplémentaire qui permet d'agir de plus en plus précisément sur la paramétrisation résultante.

B3 - Obtention d'un cercle de continuité C^2

Une première application de la reparamétrisation en zigzag va consister à obtenir une paramétrisation de continuité C^2 pour le cercle. Partant de la configuration des points de contrôle représentée sur la Figure 4.7, on peut étudier la continuité de l'Equation 4.18 au point P_2 :

$$\begin{aligned} C'(1^-) &= 2wp\ (P_2 - P_1) \qquad C''(1^-) = 2p^2\ (P_0 - P_2) + 4w\ (1+p-p^2-2wp^2)\ (P_1 - P_2) \\ C'(1^+) &= 2wp\ (P_3 - P_2) \qquad C''(1^+) = 2p^2\ (P_4 - P_2) + 4w\ (1+p-p^2-2wp^2)\ (P_3 - P_2) \end{aligned} \tag{4.21}$$

Comme précédemment, la continuité C^1 est vérifiée en vertu de l'Equation 4.15. Pour étudier la continuité C^2, il faut substituer l'Equation 4.15 et l'Equation 4.16 dans l'expression de $C''(1^+)$:

$$C''(1^+) = 2p^2\ (P_0 - P_2) + 4w\ (p^2 - p - 1)\ (P_1 - P_2)$$

Par conséquent, pour avoir une continuité C^2, il faut :

$$p^2\ (1 + w) - p - 1 = 0 \tag{4.22}$$

Cette équation du second degré possède toujours deux solutions ($\Delta = 4w + 5 > 0$) mais, comme précisé plus haut, seule la solution positive est utile pour notre reparamétrisation :

$$p = \frac{1 + \sqrt{5 + 4w}}{2\ (1 + w)} \tag{4.23}$$

Ainsi, l'Equation 4.23 combinée à l'Equation 4.20 permet de raccorder, avec une continuité C^2, deux arcs de cercle[3] de même longueur (représentés par des courbes de Bézier rationnelles ou des

[3] et plus généralement deux arcs de coniques définis avec le même poids w.

courbes NURBS). A titre d'exemple, la Figure 4.9 représente le cercle de continuité C^2, obtenu par reparamétrisation en zigzag du cercle de Versprille. Pour conclure ce paragraphe, notons que la technique de reparamétrisation s'étend de manière naturelle aux surfaces. Par conséquent, on peut obtenir sans difficulté une sphére de continuité C^2.

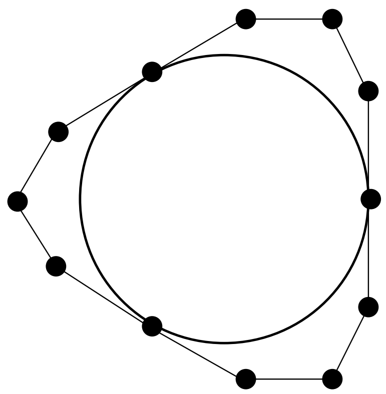


Figure 4.9: *Reparamétrisation de continuité C^2 du cercle de Versprille*

B4 - Obtention d'un cercle de paramétrisation quasi-idéale

Une deuxième application de la reparamétrisation en zigzag va consister à obtenir une paramétrisation *quasi-idéale* du cercle, c'est-à-dire une paramétrisation qui s'approche le plus possible de la paramétrisation trigonométrique.

Comme nous l'avons vu plus haut (*cf.* Figure 4.4), si l'on représente un arc de cercle par l'Equation 4.3, la fonction qui exprime l'abscisse curviligne $\theta(t)$ (*cf.* Equation 4.13) se met à osciller autour de la droite idéale $\theta = 2\phi t$ lorsque la longueur de l'arc augmente. Ce phénomène peut s'évaluer quantitativement en calculant, pour chaque valeur du paramètre t, la distance de la courbe $\theta(t)$ par rapport à la droite idéale (*i.e.* l'écart cordal de la courbe). En utilisant une norme euclidienne, l'écart cordal $\Delta(t)$ est défini par l'Equation 4.24 dont le graphe est représenté sur la Figure 4.10 :

$$\Delta(t) = \sqrt{\frac{(\theta(t) - 2\phi t)^2}{1 + 4\phi^2}} \tag{4.24}$$

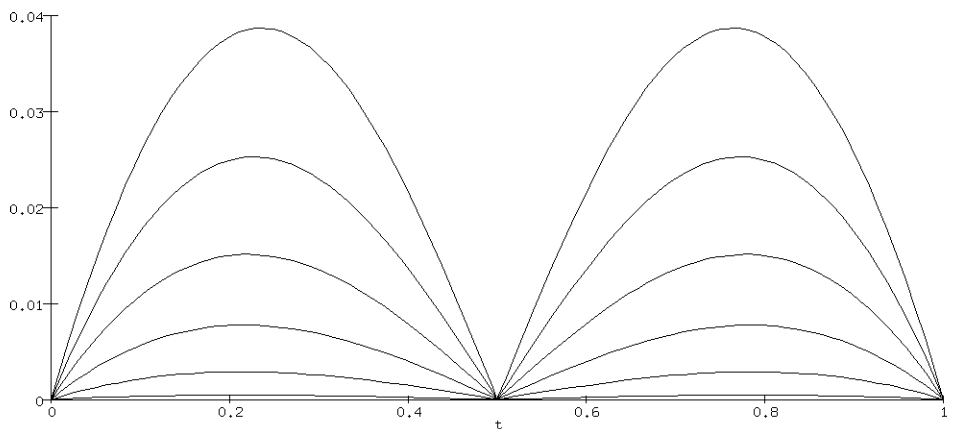


Figure 4.10: *Ecart cordal $\Delta(t)$ pour différentes valeurs de ϕ*

En substituant l'Equation 4.17 dans l'Equation 4.24, on obtient l'expression de l'écart cordal pour un arc de cercle auquel on a fait subir une reparamétrisation en zigzag. La formulation (relativement complexe) de ce nouvel écart cordal est fonction du paramètre p qui détermine la reparamétrisation. Par conséquent, comme pour la recherche d'une continuité C^2, on va pouvoir jouer sur la valeur de ce paramètre p afin de minimiser l'écart cordal de la courbe.

Nous sommes donc en présence d'un problème d'optimisation classique dans lequel il s'agit de trouver la valeur optimale d'un paramètre afin de minimiser une condition donnée. La solution la plus classique pour résoudre un problème de ce type consiste à effectuer cette minimisation au sens des moindres carrés. L'inconvénient de cette méthode (et des autres méthodes de la même famille) est de fournir une solution numérique et non pas analytique. Ce qui signifie que pour chaque nouvel angle ϕ voulu par l'utilisateur, il faut relancer tout le processus complexe des moindres carrés pour trouver la valeur de p.

Une autre possibilité consiste à minimiser simplement la valeur du maximum pour cet écart cordal. Ce maximum possède une expression analytique qu'un logiciel de calcul symbolique comme `Maple` ou `Mathématica` est capable de trouver, mais cette expression est tellement complexe[4] qu'elle est inutilisable en pratique.

C'est pourquoi nous proposons ici une troisième solution, certes heuristique, mais en fait, très proche de l'optimum. Lorsque l'on étudie la variation du maximum de l'écart cordal pour différentes valeurs de ϕ, on constate que ce maximum est toujours obtenu pour une valeur de t dans $[0.17, 0.23]$. Si l'on considère le milieu de cet intervalle, c'est-à-dire $t = 1/5$, on est donc assuré que l'écart cordal obtenu en ce point $\Delta(1/5)$ n'est pas trop éloigné de l'écart cordal maximum. Par conséquent, il suffit d'imposer $\Delta(1/5) = 0$, c'est-à-dire :

$$\frac{(1+4p)\ \sin\phi}{18+12p-(1+4p)\cos\phi} - \tan(\phi/5) = 0 \tag{4.25}$$

pour obtenir une valeur de p quasi-optimale pour la reparamétrisation :

$$p = \frac{4-2\cos^3(\phi/5)+\cos(\phi/5)}{-1+8\cos^3(\phi/5)-4\cos(\phi/5)} \tag{4.26}$$

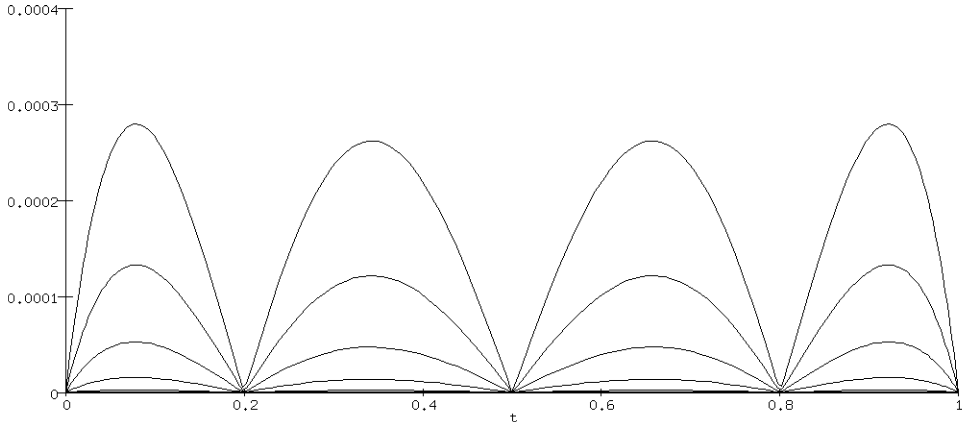


Figure 4.11: *Nouvel ecart cordal $\Delta(t)$ pour différentes valeurs de ϕ*

[4] La formule complète trouvée par Maple représente trois pages imprimées.

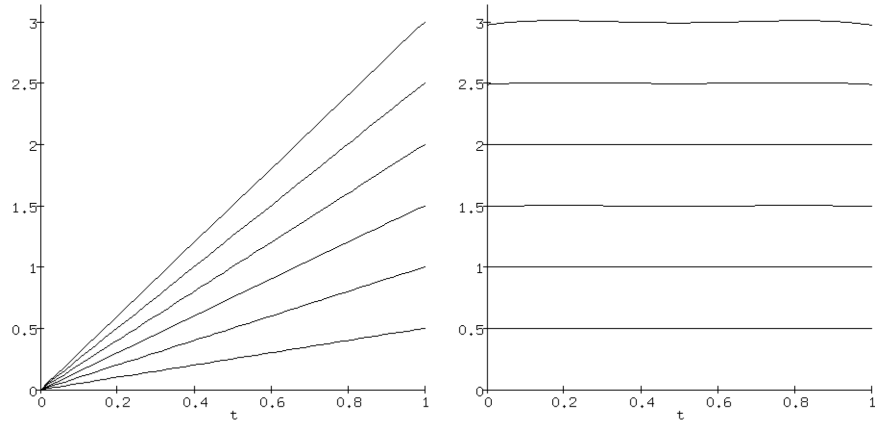


Figure 4.12: *Nouveau $\theta(t)$ et $\theta'(t)$ pour différentes valeurs de ϕ*

La Figure 4.11 représente le graphe de l'écart cordal obtenu en utilisant la valeur quasi-optimale de p. Il faut noter qu'il y a un facteur d'échelle de 100 entre la Figure 4.10 et la Figure 4.11, ce qui signifie que l'erreur de paramétrisation (par rapport à la paramétrisation idéale) a été réduite de plus d'un facteur 100, sur ces exemples. Ce résultat est confirmé sur la Figure 4.12 qui représente l'abscisse curviligne $\theta(t)$ pour l'arc reparamétrisé : les pentes des courbes sont presque constantes, plus aucune oscillation n'est visible.

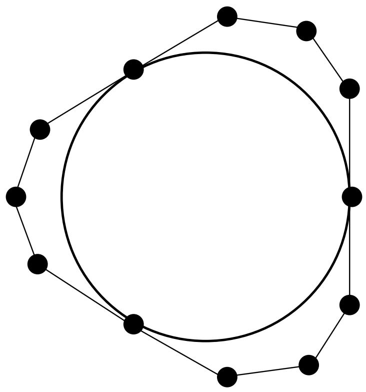


Figure 4.13: *Reparamétrisation quasi-idéale du cercle de Versprille*

A titre d'exemple, la Figure 4.13 représente le cercle de paramétrisation quasi-idéale, obtenu par la reparamétrisation en zigzag du cercle de Versprille, en utilisant la valeur de p fourni par l'Equation 4.26. Sur cet exemple précis, l'écart cordal (et donc l'erreur de paramétrisation) a été réduit d'un facteur 260. En employant la minimisation algébrique (au lieu de notre heuristique) le facteur de réduction obtenu est de 275, soit à peine 5% de précision supplémentaire, pour un coût nettement plus élevé.

Pour conclure, notons que le cercle résultant ressemble beaucoup au cercle de continuité C^2 défini plus haut. Néanmoins ces deux cercles ne sont pas tout à fait identiques puisqu'ils ont été obtenus par une reparamétrisation légèrement différente : $p \simeq 1.2153$ pour le premier et $p \simeq 1.2067$ pour le second.

Bilan

L'un des avantages généralement mis en avant pour expliquer la popularité des NURBS est le fait que ce modèle permet à la fois de définir des courbes et des surfaces de formes libres (comme la plupart des modèles de splines) mais également de représenter, de manière exacte, les arcs de coniques et donc par extension, tout un ensemble de courbes et surfaces utilisées de manière intensive en CAO : arc de cercle, cercle, cylindre, cône, sphère, tore, surfaces de révolution, etc.

La manière usuelle de représenter un arc de conique par une spline consiste à utiliser une courbe NURBS de degré 2 avec une configuration spécifique des points de contrôle et des valeurs particulières pour les poids associés. Partant de là, on obtient des courbes et des surfaces plus complètes, telles que le cercle et la sphère, en raccordant plusieurs segments de coniques.

Néanmoins, dans toutes les représentations connues du cercle et de la sphère à l'aide de courbes et surfaces NURBS, deux défauts majeurs persistent. D'une part, ces diverses représentations n'offrent qu'une continuité C^1 et, à notre connaissance, il n'existe aucune méthode qui permettrait de trouver une éventuelle représentation C^2, C^3 ou plus. D'autre part, la paramétrisation utilisée dans ces modèles s'éloigne de manière non négligeable de la paramétrisation idéale que constitue l'abscisse curviligne.

Dans ce chapitre, nous avons proposé une solution pour résoudre ces deux problèmes, qui consiste en une nouvelle méthode de reparamétrisation des NURBS, que nous avons appelé la *reparamétrisation en zigzag*. Cette reparamétrisation utilise une famille particulière de polynômes rationnels et peut être vue comme une généralisation de la reparamétrisation homographique de Moebius. Appliquée à une courbe NURBS (et plus généralement à toute courbe rationnelle obtenue par projection), cette technique permet aussi bien d'obtenir des continuités d'ordres plus élevés, que des paramétrisations plus proches de l'abscisse curviligne, au prix d'une certaine augmentation du degré des courbes rationnelles résultantes.

Chapitre V

Nouvelle famille de splines

Les nombreuses familles de courbes et surfaces splines que nous avons détaillées depuis le début de ce document possèdent toutes leurs avantages et leurs inconvénients propres. La famille la plus intéressante, principalement pour son universalité, est très certainement celle des courbes et surfaces NURBS, ce qui justifie la place importante que nous lui avons attribuée ici.

Depuis une dizaine d'années, la popularisation et la standardisation des NURBS a engendré un grand nombre de logiciels de modélisation géométrique qui utilisent ce modèle comme base pour la représentation des courbes et surfaces. Ces logiciels ont une origine aussi bien universitaire (Alpha 1 de l'université de Utah, NURBS de l'université de Manchester) que commerciale (Explore de TDI, Softimage de Softimage, Amapi de Yonowat). La plupart de ces modeleurs sont implémentés sous la forme d'un *outil interactif* qui permet la modification en temps-réel des paramètres qui figurent dans la définition des NURBS (position des points de contrôle, valeurs des poids, espacements et multiplicités des noeuds).

Néanmoins, comme nous l'avons montré dans le Chapitre III, la manipulation d'un certain nombre de ces degrés de liberté est loin d'être simple et, en particulier, peut fournir des résultats troublants, voire incompréhensibles, pour un utilisateur profane. En fait, une bonne connaissance du modèle mathématique des NURBS est indispensable pour tirer pleinement parti de toutes leurs capacités. Par conséquent, la plupart des utilisateurs de ces logiciels de modélisation se contentent souvent du déplacement des points de contrôle qui est la seule manipulation à fournir des résultats immédiatement prévisibles.

Ainsi, pour la quasi-totalité des modeleurs existants, le but d'un *outil intuitif* pour l'utilisateur profane n'est pas atteint, tout au moins par rapport à l'ensemble des possibilités offertes par les courbes et surfaces NURBS. Mais en fait, la cause de ce demi-échec n'est pas à imputer aux logiciels mais plutôt au modèle utilisé qui inclut des paramètres dont le rôle n'est pas compréhensible par un utilisateur non-spécialiste.

C'est pour cette raison que nous proposons dans ce chapitre, un nouveau modèle de courbes et surfaces splines, appelées les *A-splines*. Tout le développement de la formulation mathématique de ce modèle a été effectué en suivant une seule idée : le modèle ne doit inclure que des degrés de liberté dont le rôle est immédiatement compréhensible, grâce notamment à une interprétation visuelle à l'écran pour chacun d'entre eux.

A - Courbes A-splines basiques

A1 - Cahier des charges

La création d'un nouveau modèle de splines passe par la construction d'une nouvelle famille de fonctions $F_k(t)$ qui serviront de fonctions d'influence pour les points de contrôle P_k (*cf.* Equation 2.1). Nous avons vu dans le Chapitre II que les caractéristiques des courbes splines se déduisent des propriétés de leurs fonctions d'influence. Une première partie de ce travail de création a donc consisté à recenser (aussi bien dans la littérature qu'auprès d'utilisateurs rompus à la manipulation des splines) une liste de caractéristiques qui apparaissent indispensables pour un modèle performant de splines.

Parmi les propriétés génériques des splines définies dans le Chapitre II, nous avons vu que les propriétés d'invariance affine, d'enveloppe convexe, de régularisation des oscillations et de contrôle local semblent indispensables pour la modélisation géométrique au vu des caractéristique qu'elles offrent pour les courbes splines correspondantes. Cela signifie qu'il nous faut trouver des fonctions d'influence qui soient *normales, positives, régulières* et *locales.*

Si les trois dernières propriétés sont assez simples à obtenir, la première est nettement plus complexe. En effet, trouver une famille de fonctions $F_k(t)$ dont la somme est égale à 1, quelque soit la valeur de t, n'est pas une tâche triviale. De plus, imposer une telle condition restreint nettement les choix possibles pour les fonctions $F_k(t)$. C'est pourquoi nous avons choisi de créer nos nouvelles fonctions d'influence indépendamment du critère de normalité, puis d'appliquer le processus de normalisation (*cf.* Equation 2.16) aux fonctions obtenues. Ainsi, en choisissant d'utiliser des polynômes pour les fonctions $F_k(t)$, les fonctions normalisées $R_k(t)$ seront des polynômes rationnels normalisés ce qui permet d'ajouter la propriété d'invariance par projection parallèle ou perspective à nos splines.

Une autre caractéristique très importante est de pouvoir générer d'une part des courbes de *continuité* C^2 (pour représenter des objets de forme douce), et d'autre part des courbes de *continuité* C^0 (pour modéliser des objets de forme anguleuse)[1]. L'idéal étant de pouvoir combiner des zones C^2 et C^0, dans une même courbe ou surface.

Une caractéristique supplémentaire consiste à intégrer le concept de *tension* dans la définition du modèle. Ce concept, que l'on trouve dans certains modèles de splines existants, est généralement très apprécié des utilisateurs car il correspond à une notion visuelle très claire : la distance de la courbe par rapport au réseau de contrôle. Là encore, l'idéal est de pouvoir combiner des zones plus ou moins tendues, dans une même courbe ou surface.

Enfin, une dernière caractéristique, qu'on ne trouve dans aucun modèle existant, mais qui nous semble particulièrement intéressante, est la possibilité pour un utilisateur de pouvoir manipuler le modèle soit comme une *spline d'approximation*, soit comme une *spline d'interpolation*. Et bien évidemment, l'idéal, une fois de plus, est de pouvoir combiner des zones qui approximent le réseau de contrôle avec des zones qui l'interpolent pour une même courbe ou surface.

A2 - Principe

Nous avons vu dans le Chapitre II que pour une spline locale, la partie non-nulle des fonctions d'influence $F_k(t)$ forme une courbe en cloche (*cf.* Figure 2.4) qui démarre sur un premier noeud T_k^-, possède un maximum unique pour un second noeud T_k et redevient nulle sur un troisième noeud

[1] Nous suivons ici le raisonnement de certains auteurs [FARI90, SARF92] qui ont montré que la continuité C^1 et les continuités C^3, C^4 et plus, présentent moins d'intérêt pour la modélisation géométrique que les continuités C^0 et C^2.

T_k^+. Dans les modèles de splines classiques, la fonction $F_k(t)$ est définie par plusieurs segments entre T_k^- et T_k, puis plusieurs segments entre T_k et T_k^+.

L'idée directrice du nouveau modèle que nous proposons, est la suivante : la partie non-nulle des fonctions d'influence $F_k(t)$ doit être composée uniquement de deux segments, le premier, appelé $F_k^-(t)$, est défini entre T_k^- et T_k, le second, appelé $F_k^+(t)$, est défini entre T_k et T_k^+.

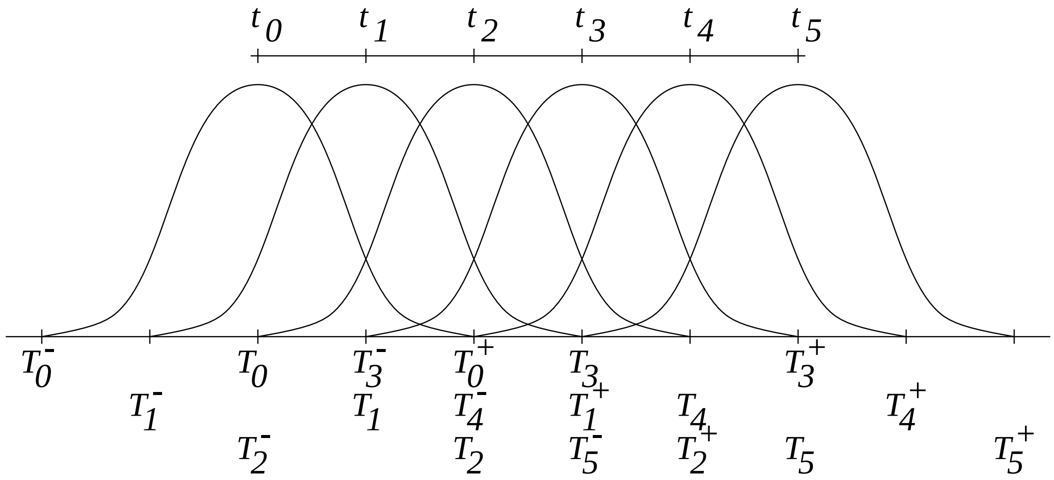


Figure 5.1: *Alternance de la prise en compte des noeuds* t_k

Pour préciser cette idée, supposons que nous soyons dans le cas d'une spline de localité L^4 (comme pour une B-spline cubique, par exemple). Par définition, un point P_k influence quatre segments de courbe et donc sa fonction d'influence $F_k(t)$ est non-nulle sur quatre intervalles successifs. Par conséquent, si l'on indice les noeuds t_k par ordre croissant, cela signifie que $F_k(t)$ devient non-nulle en t_{k-2}, maximale en t_k et à nouveau nulle en t_{k+2} (*cf.* Figure 5.1). Comme $F_k(t)$ est composée uniquement de deux segments, elle ne dépend que des noeuds t_{k-2}, t_k et t_{k+2}. On constate donc qu'il y a une sorte d'*alternance* dans l'ordre de prise en compte des noeuds (les points d'indice pairs n'utilisent que des noeuds d'indice pairs, et réciproquement). C'est pour cette raison que nous avons appelé ce nouveau modèle, les *splines alternées* ou encore les *A-splines*.

A3 - Formulation

En fait, une fois que le principe général, énoncé ci-dessus a été établi, la formulation de base du modèle, ainsi que celles de ses possibles extensions, en découlent assez naturellement. Dans un premier temps, plaçons-nous dans le cas d'un vecteur de noeud uniforme :

$$\forall k = 1..n \qquad t_k - t_{k-1} = \Delta$$

Supposons que nous ayons reparamétrisé notre courbe par la fonction :

$$u(t) = \frac{t - t_{k-2}}{t_k - t_{k-2}} = \frac{t - t_{k-2}}{2\Delta}$$

Cela signifie que $u = 0$ à l'endroit où $F_k(t)$ commence à être non-nulle et $u = 1$ à l'endroit où $F_k(t)$ devient maximale. Par conséquent, on cherche une fonction $f(u)$ définie sur $[0, 1]$ qui peut être reliée à la partie gauche de $F_k(t)$ par :

$$F_k^-(t) = f\left(\frac{t - t_{k-2}}{2\Delta}\right) \tag{5.1}$$

Puisqu'on désire une continuité C^2 pour la courbe, il faut qu'au moment où la fonction d'influence d'un point de contrôle devient non-nulle, sa dérivée première et sa dérivée seconde soient nulles, c'est-à-dire :

$$f(0) = 0 \qquad f'(0) = 0 \qquad f''(0) = 0 \tag{5.2}$$

Comme le maximum de la fonction d'influence est atteint pour $u = 1$, la dérivée première y est nécessairement nulle. D'autre part, on peut imposer $f(1) = 1$ puisque, de toutes manières, on va appliquer un processus de normalisation aux fonctions d'influence, qui va réduire le maximum à sa valeur correcte. Enfin, on peut également imposer que la valeur de la dérivée seconde soit égale à un constante pour $u = 1$ (nous appelons cette constante $-2p$, ce qui va simplifier la suite des calculs). Toutes ces conditions nous donnent :

$$f(1) = 1 \qquad f'(1) = 0 \qquad f''(1) = -2p \tag{5.3}$$

Nous sommes donc en présence d'un système à six contraintes. Comme nous cherchons à vérifier ces contraintes par un polynôme, celui-ci sera nécessairement de degré cinq, afin de posséder six degrés de liberté. En égalant les contraintes et les coefficients du polynômes, on trouve :

$$f(u) = u^3 \left(10 - p + (2p - 15)\ u + (6 - p)\ u^2\right) \tag{5.4}$$

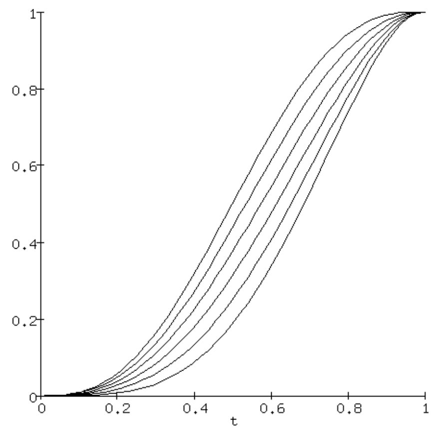


Figure 5.2: *Allure de la fonction $f(u)$ pour $p = 0, 2, 4, 6, 8, 10$*

Pour que la fonction $f(u)$ soit strictement croissante sur $[0, 1]$, il faut avoir :

$$0 \leq p \leq 10$$

La fonction $f(u)$ permet donc de représenter la partie gauche de $F_k(t)$, après la reparamétrisation décrite par l'Equation 5.1. En inversant le sens et l'origine de la reparamétrisation, on obtient la partie droite de $F_k(t)$ par :

$$F_k^+(t) = f\left(\frac{t_{k+2} - t}{2\Delta}\right) \tag{5.5}$$

Les fonctions F_k^- et F_k^+ se raccordent en t_k selon une continuité C^2 (*cf.* Equation 5.6) ce qui implique que la fonction d'influence globale F_k est de continuité C^2.

$$F_k'(t_k^-) = F_k'(t_k^+) = 0 \qquad F_k''(t_k^-) = F_k''(t_k^+) = \frac{-p}{2\Delta^2} \tag{5.6}$$

Ainsi, nous pouvons obtenir l'équation complète d'un segment de courbe $C(t)$ sur l'intervalle $[t_{k+1}, t_{k+2}]$, défini par les points P_k, P_{k+1}, P_{k+2}, et P_{k+3} :

$$C(t) = \frac{A_0(t)\ P_k + A_1(t)\ P_{k+1} + A_2(t)\ P_{k+2} + A_3(t)\ P_{k+3}}{A_0(t) + A_1(t) + A_2(t) + A_3(t)} \tag{5.7}$$

$$A_0(t) = f\left(\frac{t_{k+2}-t}{2\Delta}\right) \qquad A_1(t) = f\left(\frac{t_{k+3}-t}{2\Delta}\right) \qquad A_2(t) = f\left(\frac{t-t_k}{2\Delta}\right) \qquad A_3(t) = f\left(\frac{t-t_{k+1}}{2\Delta}\right)$$

Le mécanisme de construction défini ci-dessus permet donc de créer une courbe *spline rationnelle de degré 5* qui inclut les propriétés de normalité, de positivité, de régularité, de localité L^4 et de continuité C^2. De plus, les courbes disposent d'un degré de liberté $p \in [0, 10]$ qui permet de faire varier (très légèrement) leur forme.

Il faut noter que l'on obtient un cas particulier très intéressant lorsque $p = 8$. En effet, après l'étape de normalisation, les fonctions d'influence résultantes sont très proches des fonctions de base des B-splines cubiques uniformes, comme on peut le voir sur la Figure 5.4. Cela signifie que la courbe obtenue, que nous appelerons A-spline basique (Figure 5.3) est quasiment identique à une B-spline cubique uniforme (Figure 2.7).

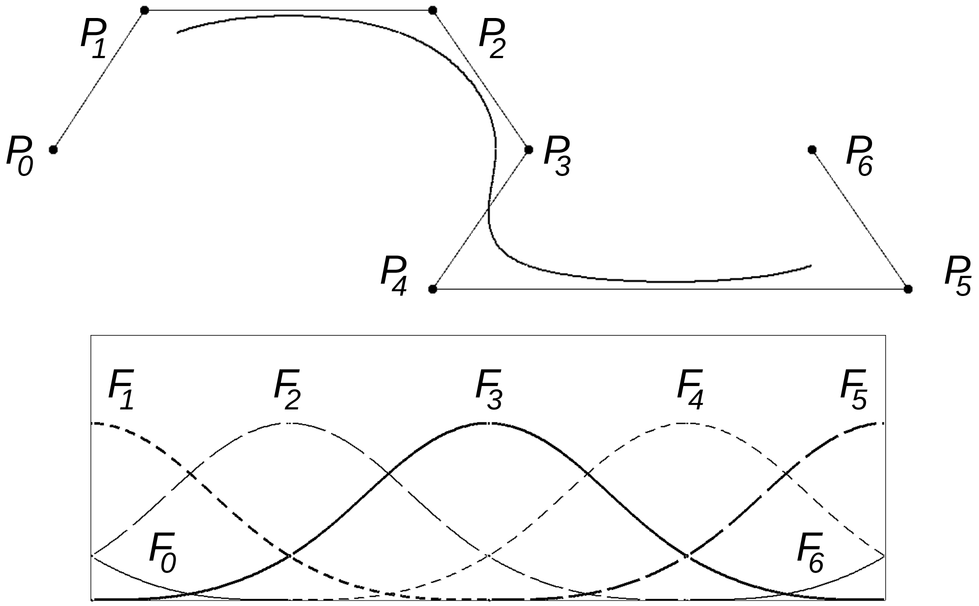


Figure 5.3: *Courbe A-spline basique*

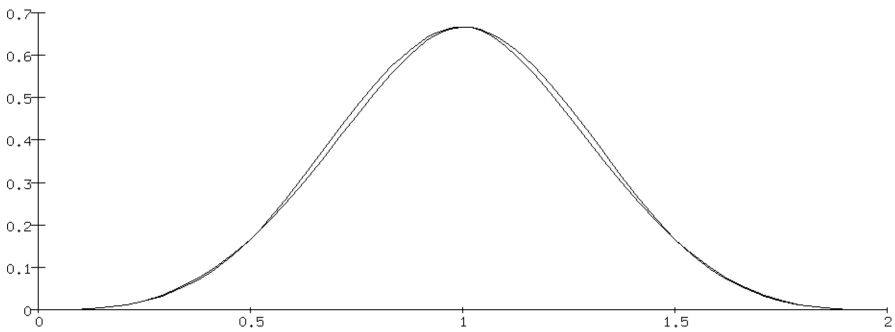


Figure 5.4: *Similarité des B-splines cubiques uniformes et des A-splines basiques*

B - Courbes A-splines étendues

B1 - Formulation

Le degré de liberté p qui existe dans l'Equation 5.4 ne permet pas de faire varier suffisamment la forme des fonctions d'influence (*cf.* Figure 5.2) pour produire des effets intéressants sur les courbes. Par conséquent, il ne présente guère d'intérêt pour l'utilisateur d'autant plus que son rôle est difficilement interprétable de manière intuitive. En fait, l'existence de ce degré de liberté va même être soigneusement cachée à l'utilisateur. Comme nous allons le voir, ce paramètre variable p est nécessaire à la bonne gestion d'un autre paramètre s, que nous allons introduire maintenant et qui va représenter le véritable degré de liberté accessible à l'utilisateur.

Parmi les caractéristiques que nous avons mises en exergue dans notre cahier des charges, le concept de tension et celui de forme anguleuse (autrement dit, obtention d'une continuité C^0) relève de la même approche. En effet comme son nom l'indique, l'idée de tension est née à partir d'un désir de pouvoir *tendre* la courbe (ou la surface), comme si celle-ci était symboliquement reliée par des cordes aux points de contrôle. En raccourcissant une ou plusieurs de ces cordes virtuelles, la courbe va s'approcher d'un ou plusieurs points de contrôle tout en se tendant, par effet de résistance. Poussé à sa limite, ce processus permet de forcer la courbe à passer par un ou plusieurs points de contrôle en créant des points anguleux ou des arêtes vives.

Dans ce concept de tension, figure donc l'idée de rapprocher la courbe d'une partie du réseau de contrôle, c'est à dire d'augmenter l'influence des points de contrôle correspondants. Une solution possible pour réaliser cette action, est de donner des coefficients de pondération variables aux différents points de contrôle. Mais comme nous l'avons vu au Chapitre III, cette solution (qui est à la base de la notion de poids pour les courbes rationnelles) ne fonctionne pas correctement car ces coefficients agissent de manière relative, les uns par rapport aux autres.

Nous proposons ici une solution originale pour intégrer la notion de tension qui ne possède pas les inconvénients qui existent dans les modèles existants. Considérons les fonctions d'influence F_2, F_3 et F_4 sur la Figure 5.1. Nous savons que F_3 atteint son maximum en t_3. Mais comme les fonctions F_2 et F_4 ne sont pas nulles en t_3, le processus de normalisation a positionné la valeur de $F_3(t_3)$ à $1 - F_2(t_3) - F_4(t_3)$. Cela signifie que si l'on veut augmenter $F_3(t_3)$ afin de rapprocher la courbe de P_3, il faut diminuer $F_2(t_3)$ et $F_4(t_3)$.

Or, nous savons que dans la zone qui nous intéresse, F_2 décroît de manière régulière depuis t_2 jusqu'à t_4, et inversement, F_4 croît depuis t_2 jusqu'à t_4. Ainsi, pour que ces deux fonctions aient des valeurs plus petites en t_3, il suffit d'accélérer la décroissance de l'une et de retarder la croissance de l'autre. Si ces deux actions se font de manière symétrique, les fonctions résultantes vont garder toutes leurs propriétés (en particulier, le maximum de F_3 restera en t_3).

Concrètement, pour réaliser cette opération, nous allons ajouter un degré de liberté $s_3 \in [0, 1]$ au point de contrôle P_3. Ce paramètre va servir, d'une part à calculer la valeur T_2^+ (pour laquelle F_2 devient nulle) par interpolation entre t_4 et t_3 :

$$T_2^+ = t_3 + s_3\ (t_4 - t_3)$$

et d'autre part à calculer la valeur T_4^- (pour laquelle F_4 devient non-nulle) par interpolation entre t_2 et t_3 :

$$T_4^- = t_3 + s_3\ (t_2 - t_3)$$

En particulier, cela signifie qu'entre T_2^+ et t_4 (respectivement t_2 et T_4^-) la fonction F_2 (respectivement F_4) sera nulle.

Ce raisonnement est bien évidemment valable quelque soit la valeur de k. Les couples de valeur (T_k^-, T_k^+) obtenus pour chaque point doivent être substitués dans les phases de reparamétrisation (Equation 5.1 et Equation 5.5) de la manière suivante :

$$F_k^-(t) = f\left(\frac{t - T_k^-}{t_k - T_k^-}\right) \qquad F_k^+(t) = f\left(\frac{t - T_k^+}{t_k - T_k^+}\right) \tag{5.8}$$

Les parties gauche et droite de $F_k(t)$ se raccordent toujours en t_k, leurs dérivées premières sont toujours nulles, par contre, leurs dérivées secondes sont différentes :

$$F_k''(t_k^-) = \frac{-2p}{(t_k - T_k^-)^2} \qquad F_k''(t_k^+) = \frac{-2p}{(t_k - T_k^+)^2} \tag{5.9}$$

C'est ici que notre paramètre p va enfin être utilisé, et va nous permettre d'égaler les deux termes afin d'obtenir une continuité C^2. En effet, il suffit d'utiliser une valeur de p (notée p_{k-1}) dans le calcul de F_k^- et une autre (notée p_{k+1}) dans le calcul de F_k^+. En prenant :

$$p_{k-1} = \frac{2\ (t_k - T_k^-)^2}{\Delta^2} \quad \text{et} \quad p_{k+1} = \frac{2\ (t_k - T_k^+)^2}{\Delta^2} \tag{5.10}$$

on obtient

$$F_k''(t_k^-) = F_k''(t_k^+) = -\frac{4}{\Delta^2}$$

ce qui non seulement procure une continuité C^2, mais en plus, fournit la même courbure que la formulation de base vue dans la partie A.

Nous obtenons ainsi une nouvelle équation[2] d'un segment de courbe $C(t)$ sur l'intervalle $[t_{k+1}, t_{k+2}]$ défini par les points P_k, P_{k+1}, P_{k+2} et P_{k+3} :

$$C(t) = \frac{A_0(t)\ P_k + A_1(t)\ P_{k+1} + A_2(t)\ P_{k+2} + A_3(t)\ P_{k+3}}{A_0(t) + A_1(t) + A_2(t) + A_3(t)} \tag{5.11}$$

[2] Nous utilisons ici l'opérateur (`test ? a : b`) du langage C (qui renvoie a lorsque `test` est vrai, et b sinon) pour représenter les expression multiples de manière compacte.

$$
\begin{array}{llll}
A_0(t) = t < T_k^+ & ? \quad f\left(\dfrac{t - T_k^+}{t_k - T_k^+}\right) & : \ 0 & \text{avec} \quad p = \dfrac{2}{\Delta^2}\,(t_k - T_k^+)^2 \\
A_1(t) = t < T_{k+1}^+ & ? \ f\left(\dfrac{t - T_{k+1}^+}{t_{k+1} - T_{k+1}^+}\right) & : \ 0 & \text{avec} \quad p = \dfrac{2}{\Delta^2}\,(t_{k+1} - T_{k+1}^+)^2 \\
A_2(t) = t > T_{k+2}^- & ? \ f\left(\dfrac{t - T_{k+2}^-}{t_{k+2} - T_{k+2}^-}\right) & : \ 0 & \text{avec} \quad p = \dfrac{2}{\Delta^2}\,(t_{k+2} - T_{k+2}^-)^2 \\
A_3(t) = t > T_{k+3}^- & ? \ f\left(\dfrac{t - T_{k+3}^-}{t_{k+3} - T_{k+3}^-}\right) & : \ 0 & \text{avec} \quad p = \dfrac{2}{\Delta^2}\,(t_{k+3} - T_{k+3}^-)^2
\end{array}
$$

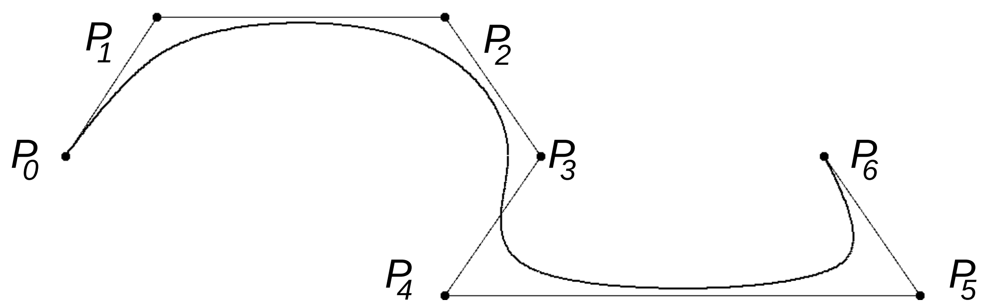
L'expression de $C(t)$ semble complexe mais en fait elle s'implémente de manière très compacte et très efficace (12 lignes de code en langage C).

Une A-spline, dans sa formulation étendue, est donc entièrement définie par un ensemble de quadruplets (x_k, y_k, z_k, s_k) avec $k = 0..n$. Chacun de ces $4n+4$ degrés de liberté possède une interprétation très simple. Les triplets $(x_k, y_k, z_k) \in \mathbb{R}^3$ représentent les coordonnées des points P_k. Le paramètre $s_k \in [0,1]$ quant à lui, symbolise la *distance entre la courbe et le réseau de contrôle* au voisinage de P_k : lorsque $s_k = 1$, la courbe passe au large de P_k, lorsque s_k diminue, la courbe se rapproche de plus en plus de P_k, enfin, lorsque $s_k = 0$, la courbe passe par P_k.

Il faut noter qu'avec le mécanisme de construction utilisé, la courbe est toujours de continuité C^2, même lorsqu'elle passe par un point P_k. Par contre, comme les dérivées première et seconde s'annulent en t_k, dans le cas de l'interpolation, la courbe est généralement de continuité G^0 en ce point, ce qui signifie que l'on obtient bien des points anguleux et des arêtes vives.

B2 - Exemples

Dans ce paragraphe, nous allons étudier plus en détail, le rôle du paramètre s ainsi que la manière de configurer les différents s_k pour obtenir des effets spécifiques quant à la forme de la courbe. La formulation de base définie dans la partie A correspond au cas où tous les paramètres s_k sont égaux à un. Comme nous l'avons vu, les courbes A-splines basiques résultantes sont quasiment identiques aux courbes B-spline cubique uniforme. Une première variante intéressante consiste à positionner s_0 et s_n à zéro afin d'interpoler les extrémités du réseau et ainsi mieux contrôler la position du début et de la fin de la courbe. On obtient ainsi (*cf.* Figure 5.5) des courbes très proches des B-splines extrémales définies dans le chapitre II (*cf.* Figure 2.8).



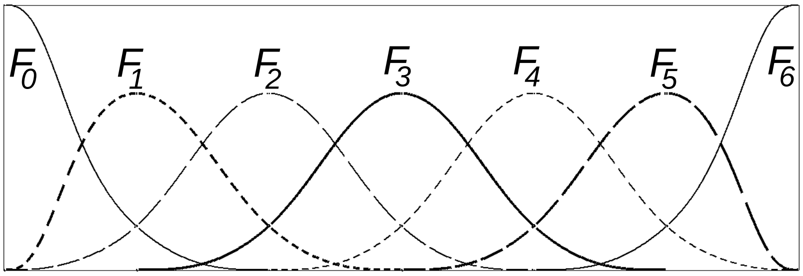


Figure 5.5: *Courbe A-spline extrémale*
$s_0 = 0, s_1 = 1, s_2 = 1, s_3 = 1, s_4 = 1, s_5 = 1, s_6 = 0$

Partant de cette configuration, essayons de diminuer la valeur de l'un des coefficients s_k (s_3 en l'occurence). Comme on peut le voir sur la Figure 5.6 la partie droite de la fonction F_2 (représenté en trait tireté fin) devient nulle plus tôt et la partie gauche de la fonction F_4 (en pointillé fin) devient non-nulle plus tard. Ainsi, par le procédé de normalisation, le maximum de F_3 (en trait continu épais) va augmenter, ce qui rapproche la courbe de P_3. De plus il faut noter que ni le maximum de F_2, ni celui de F_4 ne change, ce qui signifie que la courbe ne bouge ni au voisinage de P_2 ni à celui de P_4 : toutes les modifications sont localisées très précisément autour de P_3.

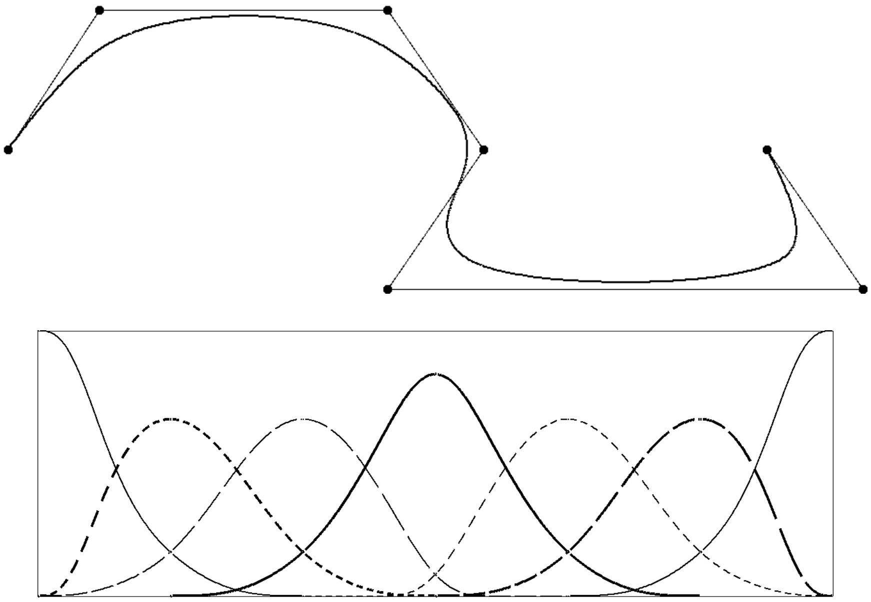


Figure 5.6: *Augmentation de l'influence du point* P_3
$s_0 = 0, s_1 = 1, s_2 = 1, s_3 = 0.5, s_4 = 1, s_5 = 1, s_6 = 0$

En continuant la diminution de s_3, le maximum de F_3 va continuer à augmenter. Finalement, pour $s_3 = 0$, ce maximum passe à un (*cf.* Figure 5.7) ce qui provoque l'interpolation "anguleuse" (*i.e.* de continuité G^0) du point P_3.

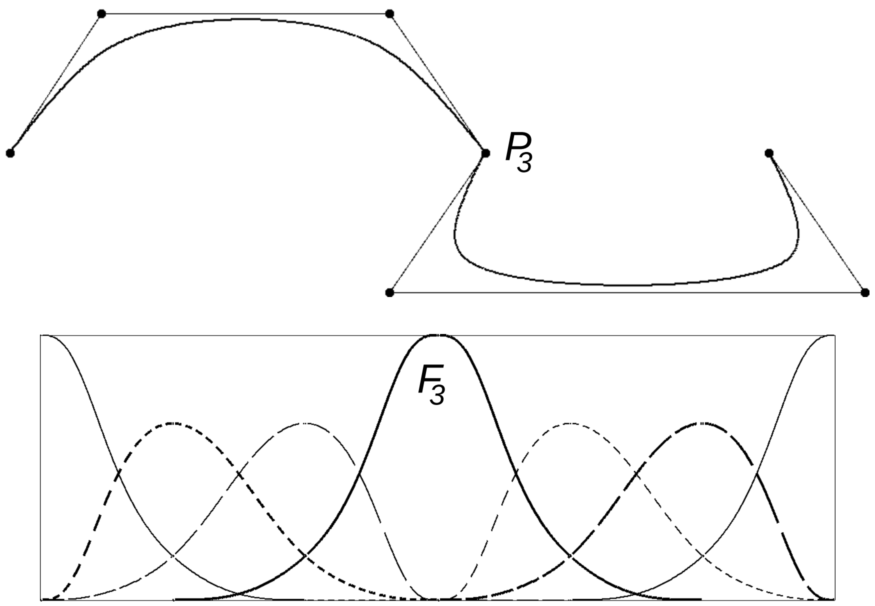


Figure 5.7: *Interpolation anguleuse du point P_3*
$s_0 = 0, s_1 = 1, s_2 = 1, s_3 = 0, s_4 = 1, s_5 = 1, s_6 = 0$

L'extrême localité de l'influence des paramètres s_k va permettre de réaliser le même type d'opérations sur plusieurs points de contrôle adjacents. Par exemple, en diminuant s_2, s_3 et s_4 on rapproche la courbe simultanément des points P_2, P_3 et P_4 (*cf.* Figure 5.8).

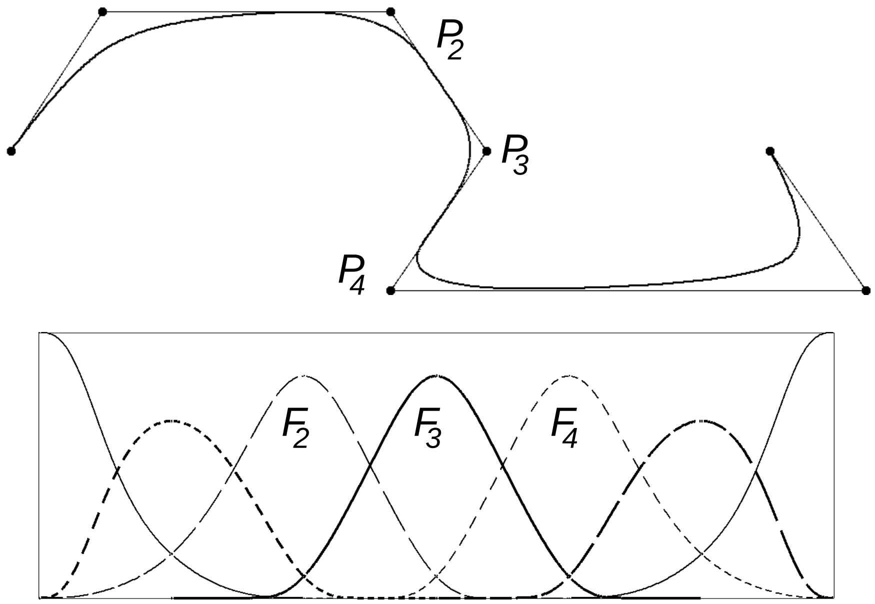


Figure 5.8: *Augmentation de l'influence des points P_2, P_3 et P_4*
$s_0 = 0, s_1 = 1, s_2 = 0.5, s_3 = 0.5, s_4 = 0.5, s_5 = 1, s_6 = 0$

et en positionnant les trois paramètres à zéro, on interpole ces trois points successifs de manière anguleuse (*cf.* Figure 5.9).

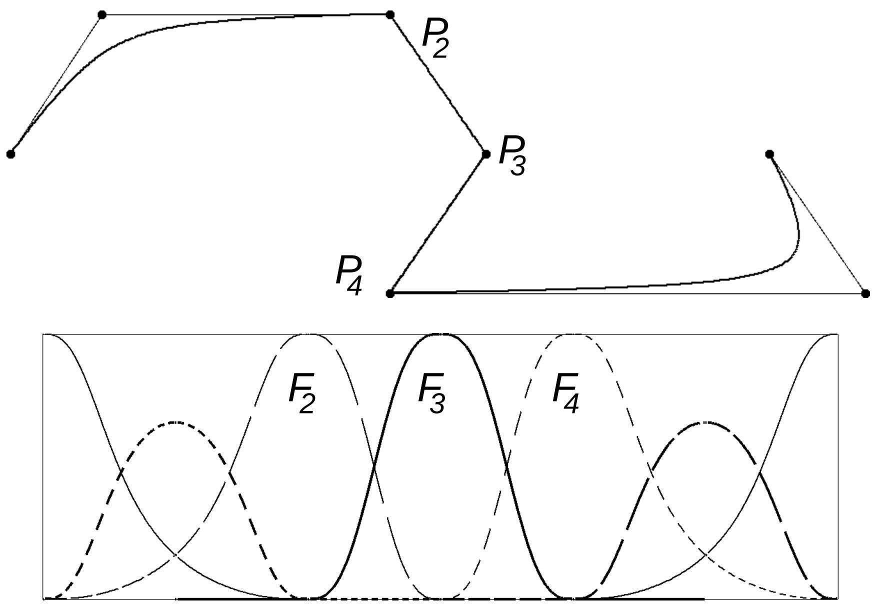


Figure 5.9: *Interpolation anguleuse des points P_2, P_3 et P_4*
$s_0 = 0, s_1 = 1, s_2 = 0, s_3 = 0, s_4 = 0, s_5 = 1, s_6 = 0$

Enfin, le cas limite (qui est le cas dual de la formulation de base) est obtenu lorsque tous les paramètres s_k sont égaux à zéro. La courbe résultante est confondue avec le réseau de contrôle (*cf.* Figure 5.10). Néanmoins, il ne s'agit pas d'une spline linéaire (*cf.* Figure 2.11) puisque la paramétrisation est ici de continuité C^2 alors qu'elle est C^0 pour les splines linéaires.

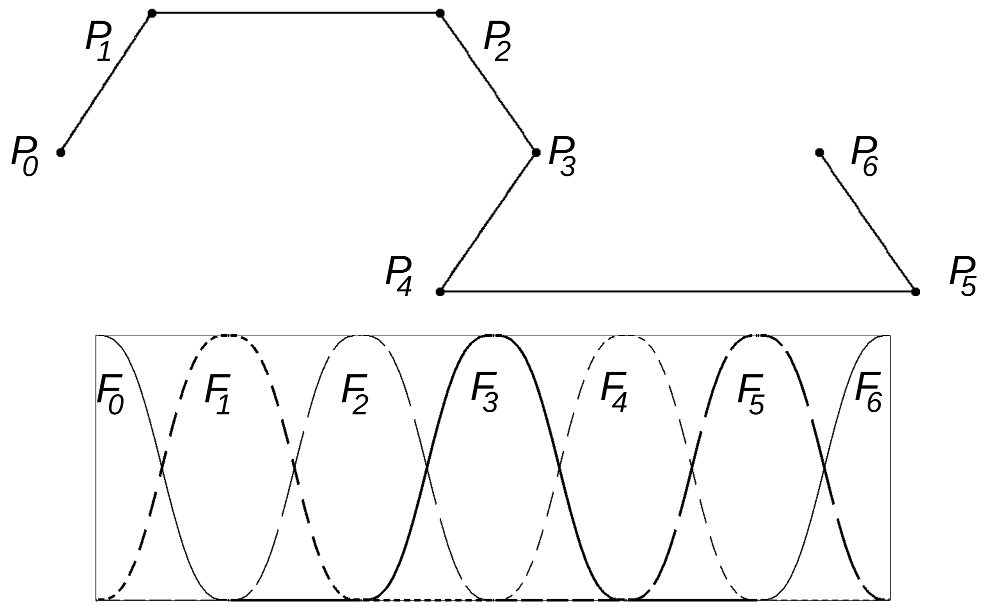


Figure 5.10: *Interpolation anguleuse de tous les points*
$s_0 = 0, s_1 = 0, s_2 = 0, s_3 = 0, s_4 = 0, s_5 = 0, s_6 = 0$

C - Courbes A-splines complètes

C1 - Formulation

Telles qu'elles ont été formulées dans la partie B, les A-splines étendues remplissent déjà la plupart des contraintes de notre cahier des charges. Pourtant, même si elles permettent d'interpoler un ou plusieurs points de contrôle, les A-splines sont avant tout des splines d'approximation puisque seules des interpolations anguleuses peuvent être obtenues. La dernière caractéristique incluse dans notre cahier des charges consistait à pouvoir manipuler un même modèle soit comme une spline d'approximation, soit comme une spline d'interpolation. Le but de cette partie C est de montrer comment on peut obtenir cette caractéristique.

Néanmoins, comme nous l'avons vu dans le Chapitre II, utiliser des splines d'interpolation nécessite de renoncer à la propriéte de positivité et donc à la notion d'enveloppe convexe. Pour certaines applications (et pour certains utilisateurs) ce renoncement est inconcevable. C'est pourquoi nous avons *volontairement* séparé cette extension de la partie précédente. Ainsi, le lecteur peut choisir librement entre la formulation qui vérifie la propriété d'enveloppe convexe et la formulation qui utilise la dualité approximatiom/interpolation.

Nous avons vu dans la partie B qu'en diminuant la valeur du paramètre s_k, se créait une zone entre t_{k-1} et T^-_{k+1} (respectivement T^+_{k-1} et t_{k+1}) dans laquelle F^-_{k+1} (respectivement F^+_{k-1}) était égale à zéro. Dans le cas limite où $s_k = 0$, c'est tout l'intervalle $[t_{k-1}, t_k]$ (respectivement $[t_k, t_{k+1}]$) qui annule la fonction F^-_{k+1} (respectivement F^+_{k-1}). Pour obtenir une spline qui réalise une interpolation "arrondie" (*i.e.* de continuité G^2), du point P_k, il faut permettre à F^-_{k+1} et F^+_{k-1} de devenir négative sur ces intervalles. De la même manière que dans la formulation précédente, nous avons cherché à nous rapprocher le plus possible des B-splines cubiques, nous allons chercher ici à nous rapprocher des splines cardinales cubiques.

Supposons que nous ayons représenté notre courbe par la fonction :

$$u(t) = \frac{t - t_k}{t_{k+1} - t_k} \tag{5.12}$$

Cela signifie que $u = -1$ à l'endroit ou F^-_{k+1} commence à être négative, $u = 0$ à l'endroit où F^-_{k+1} commence à être positive et $u = 1$ à l'endroit où F^-_{k+1} devient maximum. Par conséquent, on cherche deux fonctions : $g(u)$ définie sur $[0, 1]$ qui va représenter la partie positive de F^-_{k+1} et $h(u)$ définie sur $[-1, 0]$ qui va représenter la partie négative de F^-_{k+1}. Ces deux fonctions $g(u)$ et $h(u)$ doivent se raccorder en $u = 0$ avec une continuité C^2.

En imposant les contraintes existant dans la définition des splines cardinales (*cf.* Equation 2.32) et en rajoutant la contrainte de continuité C^2, on trouve un ensemble de douze contraintes :

$$g(0) = 0 \quad g'(0) = q \quad g''(0) = 4q \quad g(1) = 1 \quad g'(1) = 0 \quad g''(1) = -2p \tag{5.13}$$

$$h(0) = 0 \quad h'(0) = q \quad h''(0) = 4q \quad h(-1) = 0 \quad h'(-1) = 0 \quad h''(-1) = 0 \tag{5.14}$$

où q est un degré de liberté qui joue le même rôle que le paramètre r dans les splines cardinales. Toutes ces contraintes peuvent être vérifiée par deux polynômes de degré cinq :

$$g(u) = q\ u + 2q\ u^2 + (10-15q-p)\ u^3 + (-15+17q+2p)\ u^4 + (6-6q-p)\ u^5 \tag{5.15}$$

$$h(u) = q\ u + 2q\ u^2 + 2q\ u^3 + 2q\ u^4 + q\ u^5 \tag{5.16}$$

A partir de ces équations, le même mécanisme de construction que dans la partie B permet d'obtenir un modèle de spline d'interpolation de continuité C^2 qui est une spline rationnelle de degré cinq. De plus, les courbes disposent d'un degré de liberté $q \in [0, 1]$ qui permet de faire varier leur forme. En particulier, avec la valeur $q = 1/2$, on obtient des fonctions d'influence qui sont très proches des splines cardinales basiques (*cf.* Figure 2.13), comme on peut le voir sur la Figure 5.11 ; la différence principale est que les nouvelles courbes d'interpolation possèdent une continuité C^2 et non C^1.

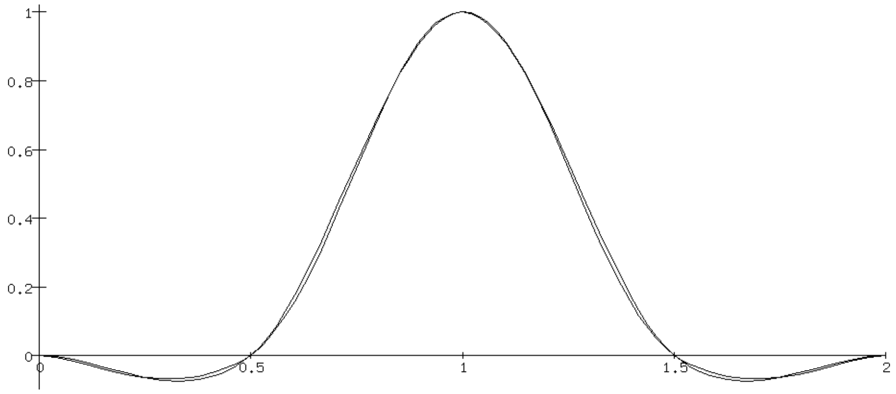


Figure 5.11: *Similarité des splines cardinales basiques et des A-splines (avec $q = 1/2$)*

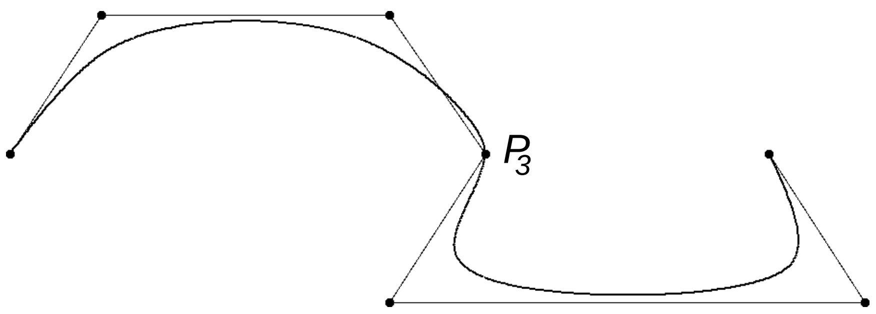
L'étape suivante consiste à mixer le paramètre s du modèle d'approximation avec le paramètre q du modèle d'interpolation. Là encore le but cherché est de simplifier au maximum les degrés de liberté manipulés par l'utilisateur. Concrètement, un seul paramètre s_k par point de contrôle (en plus des coordonnées (x_k, y_k, z_k) bien évidemment) est utilisé, avec la convention suivante :

- Lorsque $s_k \in [0, 1]$ cela signifie que l'utilisateur veut manipuler des splines d'approximation et dans ce cas s_k est le paramètre de distance courbe-réseau défini dans la partie B (ce qui permet d'obtenir une approximation des B-splines cubiques pour $s_k = 1$).
- Lorsque $s_k \in [-1, 0]$ cela signifie que l'utilisateur veut manipuler des splines d'interpolation et dans ce cas le paramètre q_k est obtenu à partir de s_k par $q_k = -s_k/2$ (ce qui permet d'obtenir une approximation des splines cardinales basiques pour $s_k = -1$).

La distinction positif/négatif pour s_k permet de bien indiquer à l'utilisateur qu'il y a une rupture profonde : pour les s_k positifs, la propriété d'enveloppe convexe est vérifiée, pour les s_k négatif, elle ne l'est plus. Par contre, la notion intuitive de distance courbe-réseau est conservée pour les s_k négatifs. En effet, comme nous allons le voir ci-dessous, plus les s_k s'éloignent de zéro, plus la courbe s'écarte du réseau.

C2 - Exemples

Plaçons-nous dans le cas de la configuration d'une A-spline extrémale (*cf.* Figure 5.5). Si l'on désire réaliser une interpolation "arrondie" (*i.e.* de continuité G^2) pour l'un des points de contrôle P_k, il suffit de positionner son paramètre s_k à une valeur négative. Par exemple, la Figure 5.12 montre la courbe obtenue avec $s_2 = -1$. Comme prévu, les fonctions F_1 et F_3 deviennent négatives par endroit, et donc la propriété d'enveloppe convexe est perdue.



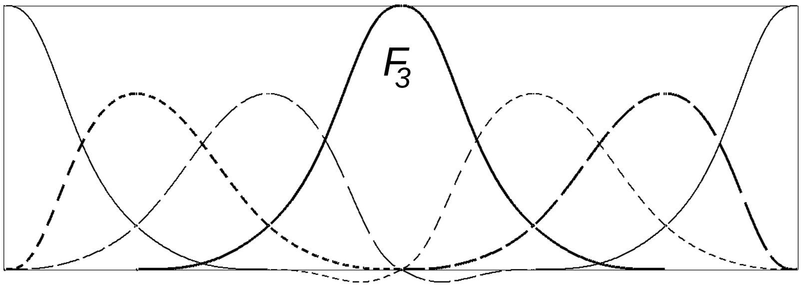


Figure 5.12: *Interpolation arrondie du point P_3*
$s_0 = 0, s_1 = 1, s_2 = 1, s_3 = -1, s_4 = 1, s_5 = 1, s_6 = 0$

En positionnant tous les paramètres s_k à -1, on obtient une courbe d'interpolation quasiment identique à une spline cardinale basique (*cf.* Figure 2.13).

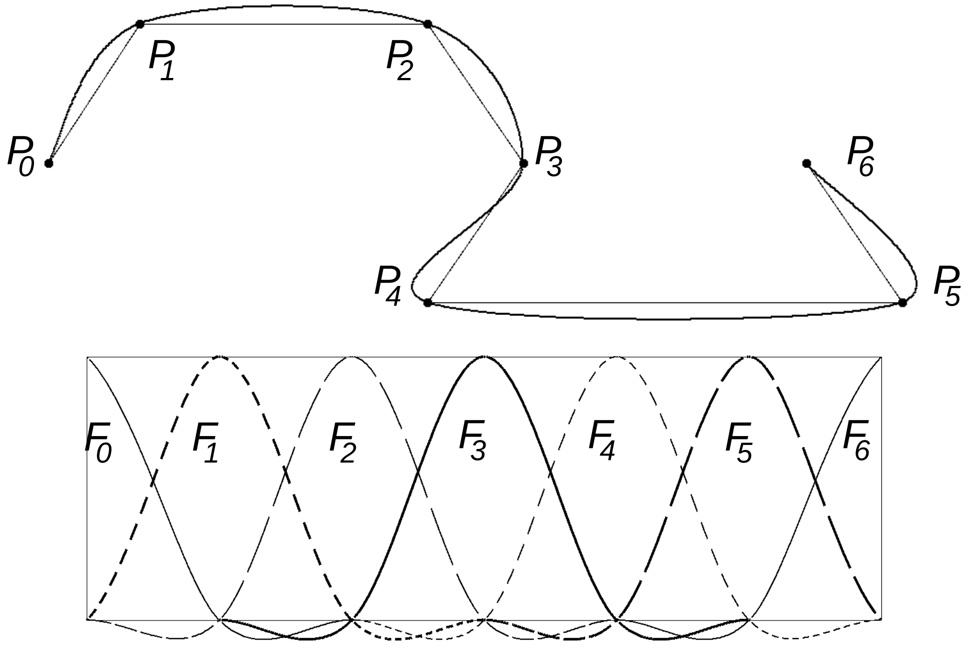
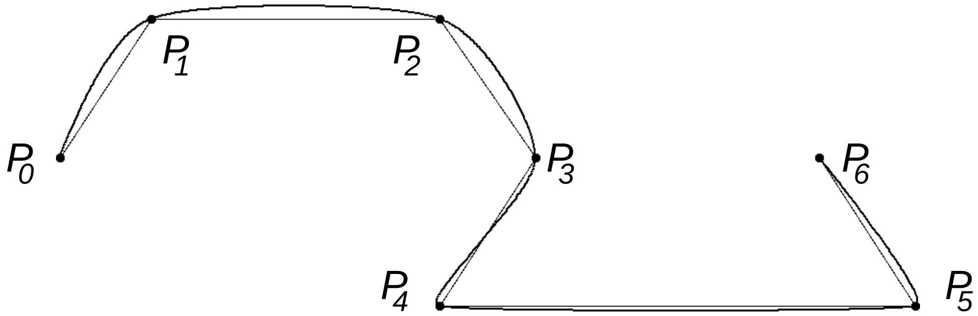


Figure 5.13: *Interpolation arrondie de tous les points*
$s_0 = -1, s_1 = -1, s_2 = -1, s_3 = -1, s_4 = -1, s_5 = -1, s_6 = -1$

En changeant la valeur des s_k, on peut contrôler précisément la forme de la courbe d'interpolation. En particulier, on peut autoriser une interpolation très lâche pour certaines zones du réseau et une interpolation plus serrée pour d'autres zones (*cf.* Figure 5.14).



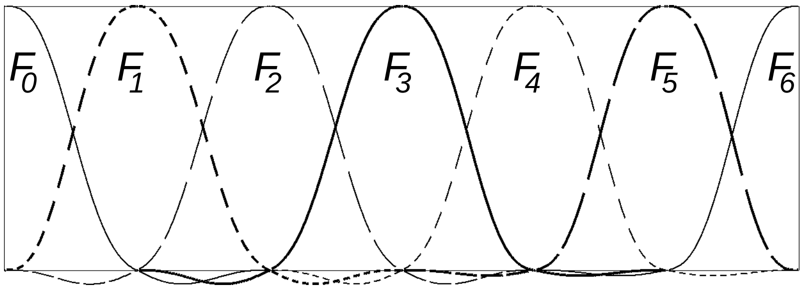


Figure 5.14: *Contrôle de la forme de l'interpolation*
$s_0 = 0, s_1 = -2, s_2 = -2, s_3 = -1, s_4 = -0.5, s_5 = -0.5, s_6 = 0$

Enfin, ce qui est peut-être la caractéristique la plus intéressante des A-splines, on peut combiner sans aucune restriction, des paramètres s_k positifs et négatifs pour créer, dans une même courbe, des zones d'approximation et des zones d'interpolation (*cf.* Figure 5.15).

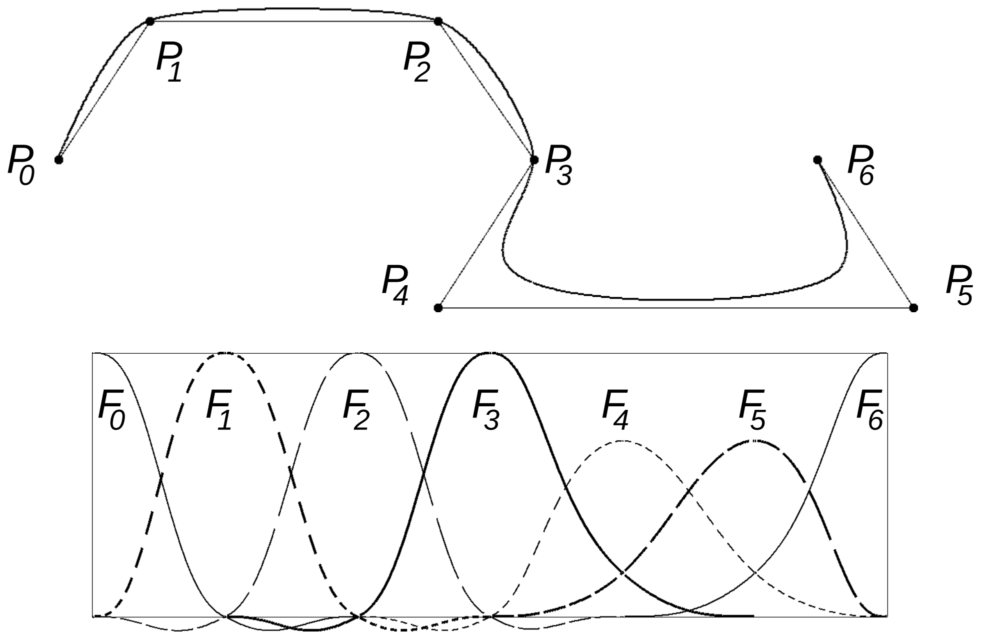


Figure 5.15: *Combinaison de zones approximées et de zones interpolées*
$s_0 = 0, s_1 = -2, s_2 = -2, s_3 = -1, s_4 = 1, s_5 = 1, s_6 = 0$

C3 - Extension à un vecteur nodal non-uniforme

Toute la discussion précédente à été menée dans le cas d'un vecteur nodal uniforme. En fait, le paramètre s permet de réaliser la plupart des effets géométriques qu'engendre la manipulation du vecteur nodal dans le cas des NURBS. Cela signifie que l'utilisateur des courbes A-splines n'éprouvera pas un sentiment de limitation en se restreignant à un vecteur nodal uniforme, comme c'est le cas pour les B-splines uniformes. Ceci est d'autant plus important que, comme nous l'avons vu dans le Chapitre III, les manipulations du vecteur nodal sont, d'une part, loin d'être intuitives, et d'autre part, engendrent des effets instables et non-localisés.

Néanmoins, il existe un domaine où la non-uniformité représente un avantage indéniable : le domaine de l'animation. En effet, la technique d'animation la plus classique consiste à donner la position des objets pour différents instants t (généralement non-uniformes) puis de calculer les positions intermédiaires par des splines d'interpolation. Par conséquent, comme le besoin de splines non-uniformes existe, nous allons très rapidement montrer comment les A-splines peuvent être étendues pour remplir cette condition.

En fait, dans la discussion précédente, nous avons presque tout le temps manipulé les noeuds t_k sans utiliser véritablement la propriété d'uniformité. La seule exception s'est produite lorsque nous avons raccordé les parties gauche et droite de $F_k(t)$ pour obtenir une continuité C^2. En effet, pour effectuer ce raccordement, nous avons calculé deux valeurs différentes p_{k-1} et p_{k+1} qui utilisait la valeur Δ de l'écart uniforme entre les noeuds (*cf.* Equation 5.10).

Pour étendre ce processus de raccordement aux splines non-uniformes, il suffit de trouver une autre valeur Δ qui soit constante pour tous les noeuds. La solution la plus simple consiste à définir Δ de la manière suivante :

$$\Delta = \max_{k=1..n} t_k - t_{k-1} \tag{5.17}$$

Ainsi, par rapport aux A-splines uniformes, il suffit d'ajouter un parcours initial sur le vecteur de noeud pour trouver l'écartement maximal entre deux noeuds consécutifs. Ensuite, la formulation et le reste du calcul est parfaitement identique.

C4 - Extension aux surfaces

L'extension aux surfaces est encore plus simple que l'extension aux vecteurs nodaux non-uniformes : la notion de produit tensoriel (*cf.* Equation 2.4) est tout à fait applicable aux courbes A-splines pour définir des surfaces. La caractéristique des A-splines de pouvoir réaliser de nombreux effets géométriques différents en ne travaillant qu'avec des vecteurs nodaux uniformes se révèle primordiale ici. En effet, comme nous l'avons vu dans le Chapitre III, tous les effets obtenus par modification de l'espacement et/ou de la multiplicité des noeuds ne peuvent pas être localisés précisément sur les surfaces, et sont susceptibles de se déplacer sur la surface, suite à des modifications ultérieures. Par contre, les degrés de libertés des A-splines sont tous directement reliés à un point de contrôle, ce qui permet une localisation précise et robuste.

Pour conclure, notons que lors de l'extension des courbes A-splines aux surfaces A-splines, deux choix différents peuvent être faits :

- Soit on utilise un seul paramètre s_k pour chaque point de contrôle P_k, qui va symboliser, d'une manière globale, la distance du point à la surface. On manipule ainsi des points de contrôle à quatre dimensions. Le comportement du paramètre s_k est exactement le même que dans le cas des courbes :
  - ⋆ $s_k < 0$: P_k est un point d'interpolation générant une continuité (C^2, G^2)
  - ⋆ $s_k = 0$: P_k est un point d'interpolation générant une continuité (C^2, G^0)
  - ⋆ $s_k > 0$: P_k est un point d'approximation générant une continuité (C^2, G^2)
- Soit on utilise deux paramètres r_k et s_k pour chaque point de contrôle P_k tels que r_k va agir dans le sens du paramètre u et s_k dans celui du paramètre v. On manipule donc ainsi des points de contrôle à cinq dimensions. Par contre, le comportement des paramètres r_k et s_k est ici plus subtil :
  - ⋆ $r_k > 0$ ou $s_k > 0$: P_k est un point d'approximation générant une continuité (C^2, G^2)
  - ⋆ $r_k < 0$ ou $s_k < 0$: P_k est un point d'interpolation générant une continuité (C^2, G^2)
  - ⋆ $r_k = 0$ et $s_k = 0$: P_k est un point d'interpolation générant une continuité (C^2, G^0)
  - ⋆ $r_k = 0$ et $s_k > 0$: P_k est un point d'approximation générant une continuité (C^2, G^0) en u et (C^2, G^2) en v
  - ⋆ $r_k = 0$ et $s_k < 0$: P_k est un point d'interpolation générant une continuité (C^2, G^0) en u et (C^2, G^2) en v

Même si elle est légèrement plus coûteuse en mémoire, la seconde solution nous semble préférable car elle permet de traiter les deux directions de manière différente (par exemple, créer une zone anguleuse dans le sens des u et arrondie dans celui des v).

Bilan

En examinant les nombreux modèles de splines proposés dans la littérature, on peut constater qu'aucun d'eux ne vérifie l'ensemble des caractéristiques d'un éventuel *modèle idéal de splines* par rapport au but de manipulation interactive et d'utilisation intuitive que nous nous sommes fixé. Dans ce chapitre, nous avons proposé un nouveau moèle de splines — les *A-splines* — qui tente de s'approcher de cet idéal.

Pour établir l'expression de ce nouveau modèle, nous avons, dans un premier temps, défini une sorte de cahier des charges regroupant les différentes propriétés "ergonomiques" que doit impérativement vérifier un modèle de splines pour permettre une utilisation intuitive. La traduction mathématique de ces caractéristiques nous a fourni, dans un deuxième temps, un ensemble de contraintes pour les fonctions d'influence. Enfin, nous avons déterminé une famille de fonctions rationnelles de degré cinq, qui remplit la quasi-totalité[3] de ces contraintes.

La caractéristique principale de ce nouveau modèle est le fait que tous les paramètres de définition, sur lesquels peut influer l'utilisateur lors de la création d'un objet, possèdent une représentation visuelle à l'écran ainsi qu'une signification intuitive. De plus, le champ d'action de chaque paramètre est localisé très précisément sur la courbe ou la surface. D'autre part, les A-splines permettent de représenter de manière approchée de nombreux modèles de splines classiques (B-splines uniformes, Bézier par morceaux, splines linéaires, splines cardinales) mais autorise également des manipulations impossibles avec l'ensemble des modèles existants. En contre-partie, les A-splines nécessitent des polynômes rationnels de degré cinq, ce qui est plus coûteux que le degré trois habituellement utilisés dans les modèles de splines courants.

[3]En fait, toutes les contraintes ne peuvent pas être vérifiées simultanément car certaines d'entre elles (par exemple, interpolation G^2 et enveloppe convexe) sont mutuellement exclusives.

Modélisation de bas-niveau, Modélisation de haut-niveau

Tout au long de ce document, nous avons implicitement admis jusqu'ici que la création d'un objet au moyen d'une surface spline, s'effectue en spécifiant les valeurs des différents paramètres de définition du modèle géométrique (position des points de contrôle, espacement ou multiplicité des noeuds, valeurs des poids ou des tensions, etc). Ce principe correspond à une première approche du processus de modélisation géométrique, qui est généralement appelée *modélisation de bas-niveau* dans le sens où l'utilisateur manipule directement les caractéristiques mathématiques du modèle choisi.

Ainsi, compte tenu de leurs propriétés géométriques particulièrement intéressantes, les modèles de splines les plus complets que nous avons étudiés (notamment les NURBS et les A-splines) permettent, dans l'absolu, de concevoir des objets de formes quelconques. Néanmoins, lorsque la complexité de l'objet que l'on souhaite créer augmente, le processus de modélisation de bas-niveau devient rapidement fastidieux pour l'utilisateur, compte tenu du nombre de paramètres à ajuster indépendamment. On atteint donc une *limitation pratique* de la variété et la complexité des formes modélisables, qui se trouve bien en-deçà de la *limitation théorique* due au modèle mathématique employé. Par conséquent, même si un outil qui permet la manipulation individuelle des paramètres de la surface est extrêmement utile (voire indispensable pour effectuer certaines modifications très précises sur l'objet), il ne peut prétendre être à lui seul, un outil complet de modélisation géométrique.

Pour faciliter la création de formes complexes, une autre approche du processus de modélisation géométrique, appelée *modélisation de haut-niveau*, a été développée. Son idée directrice est de fournir des outils qui simulent la manière dont se déroule la création d'un objet dans la réalité.

Deux familles de techniques correspondant à ce principe ont été proposées dans la littérature. D'une part, on trouve les méthodes de *combinaisons de surfaces* qui recréent le processus de construction mécanique d'un objet par une machine-outil, en simulant les opérations de découpage, d'assemblage, de fraisage ou encore d'arasage. D'autre part, on trouve les méthodes de *déformations de surfaces* qui reproduisent le processus de création manuelle d'un objet par un artisan, en simulant les outils et les gestes d'un sculpteur, d'un potier, d'un ferronnier ou encore d'un souffleur de verre.

Dans la suite de document, nous allons étudier plus précisemment ces deux familles de méthodes en particulier, pour évaluer leur degré d'adéquation avec le but que nous nous sommes fixés : réaliser un outil intuitif et interactif pour la modélisation géométrique.

Chapitre VI

Combinaisons de surfaces

Une première famille de méthodes de modélisation de haut-niveau, permettant d'augmenter la variété et la complexité des objets modélisables, est constituée par les méthodes de *combinaison de surfaces* dont le rôle est de simuler le processus de création mécanique d'un objet par une machine-outil. Le principe des techniques de cette famille consiste à combiner plusieurs objets de formes simples en utilisant des opérations booléennes telles que l'union, l'intersection ou la différence. Comme nous l'avons vu dans le Chapitre 1, les surfaces implicites se prêtent de manière simple et efficace à ce genre de manipulation, en particulier parce que les fonctions d'intérieur/extérieur se combinent naturellement (*cf.* Equation 1.3). Donc, sachant que toute surface paramétrique (et par conséquent, toute surface spline) peut-être représentée par une équation implicite — *processus d'implicitisation* — un moyen logique pour combiner les surfaces splines pourrait être de les transformer en surfaces implicites afin d'utiliser ces techniques performantes.

Néanmoins cette solution présente plusieurs inconvénients. D'une part, le processus d'implicitisation est loin d'être trivial, même pour des surfaces polynômiales ou rationnelles comme les surfaces splines. D'autre part, un grand nombre d'utilisateurs ne désirent tout simplement pas travailler avec une représentation implicite pour les surfaces qu'ils manipulent. Cette préférence est due, comme nous l'avons déjà montré, au fait que la représentation paramétrique cumule plusieurs avantages importants. D'une part, trouver un point sur une surface paramétrique est une opération peu coûteuse, ce qui facilite grandement tout un ensemble d'algorithmes utilisés en CAO/FAO et en SI. D'autre part, avec les stations de travail disponibles à l'heure actuelle, les surfaces explicites (paramétriques ou facettisées) sont les seules à permettre une visualisation en temps réel. Enfin, et c'est peut-être le plus important, les surfaces paramétriques (les surfaces NURBS, en particulier) sont les seules à disposer de standards internationaux de description, ce qui est indispensable pour pouvoir échanger des données géométriques entre plusieurs logiciels.

C'est pour ces différentes raisons que la solution d'implicitisation a rarement été retenue pour les combinaisons booléennes de surfaces paramétriques[1] et qu'ont été développées de nombreuses méthodes alternatives permettant de conserver une représentation explicite des objets. Dans ce chapitre, notre but est d'effectuer un tour d'horizon relativement complet, suivi d'une analyse critique, des méthodes de ce type proposées dans la littérature.

[1] Par contre, l'implicitisation a été utilisée à diverses reprises pour des combinaisons plus générales (*i.e.* non booléennes), telles que les techniques de "fondu" (en anglais, *blending*) entre surfaces.

A - Classification des méthodes de combinaisons

A1 - Méthodes exactes

Si l'on souhaite représenter de manière explicite, le résultat de la combinaison de deux surfaces par des opérations booléennes, il faut avant tout être capable de représenter la courbe (ou plus généralement les courbes) d'intersection entre ces surfaces. Or quel que soit le modèle géométrique, trouver une représentation de cette courbe n'est jamais simple. Plus précisément, lorsque l'intersection concerne des surfaces paramétrique de forme libre, telles que les surfaces splines, ce problème est encore aujourd'hui l'obstacle le plus important à l'utilisation efficace des opérations booléennes, et ce malgré les nombreux travaux effectués dans le domaine.

Parmi les techniques proposées dans la littérature pour calculer les courbes d'intersection entre deux surfaces paramétriques, une première famille de méthodes vise à obtenir une représentation exacte des courbes cherchées. Toute la difficulté du problème provient de la vitesse à laquelle la complexité des courbes d'intersection augmente par rapport à la complexité des surfaces initiales. En effet, si on considère deux surfaces paramétriques S et S' définies par produit tensoriel de polynômes, ayant pour degré $p{\times}q$ et $p'{\times}q'$ respectivement, elles s'intersectent selon des courbes dont l'équation implicite est de degré $4pp'qq'$ [KATZ88]. Ainsi, dans le cas pourtant simple de deux surfaces bicubiques, on obtient des courbes algébriques de degré 324.

Pour trouver l'équation exacte d'une telle courbe, la technique classique en mathématiques consiste à convertir les équations paramétriques des deux surfaces en équations algébriques (équations implicites polynômiales) puis à calculer l'équation de la courbe d'intersection en appliquant une méthode d'élimination algébrique aux deux surfaces.

Cependant, connaître l'équation exacte d'une telle courbe n'est pas suffisant dans un environnement de modélisation géométrique. Ce que l'on désire, c'est pouvoir visualiser la courbe afin de la comparer avec des contraintes esthétiques ou techniques. Nous avons vu (*cf.* Chapitre I) que l'affichage d'une courbe ou une surface définie par une équation implicite nécessite d'échantillonner l'espace plus ou moins finement selon la précision souhaitée, puis d'évaluer le polynôme pour chaque point de l'échantillonnage afin de déterminer si le point se trouve à l'intérieur ou à l'extérieur. Lorsque le polynôme est de degré 324 et que l'on désire un nombre suffisant de points, les temps de calcul deviennent rapidement prohibitifs. De plus, l'évaluation de polynômes de degré élevé pose toujours des problèmes en informatique du fait de la précision numérique limitée pour stocker les coefficients. Par conséquent, non seulement la visualisation de la courbe va être extrêmement lente, mais en plus, elle risque d'être complètement fausse.

Une solution pour visualiser rapidement une telle courbe serait de la paramétriser. Or un résultat fondamental de la géométrie analytique montre qu'il n'existe pas de paramétrisation pour les courbes dont le genre (en anglais, *genus*) est strictement supérieur à 2. Et malheureusement, le genre de l'équation algébrique résultant de l'intersection de deux surfaces bicubiques est égal à 433 dans le cas général [KATZ88]. Une telle courbe est donc loin d'être paramétrisable et, par conséquent, le problème de la visualisation des courbes d'intersection reste entier.

Malgré toutes ses limitations, la méthode d'élimination algébrique s'est tout de même révélée utile et efficace pour les cas où les surfaces sont de très faible degré (surfaces quadriques, par exemple [LEVI79]). Mais en dehors de ces cas particuliers, elle n'est généralement pas utilisée dans le cadre de la modélisation géométrique.

A2 - Méthodes approchées

Compte tenu de la complexité des courbes d'intersection entre deux surfaces, une deuxième famille de méthodes a été développé qui se contente de n'en chercher qu'une représentation approchée. Selon ce principe, les méthodes les plus anciennes commençaient par transformer les surfaces paramétriques en surfaces facettisées (obtenue en échantillonnant la surface de manière plus ou moins performante) puis à calculer directement les intersections entre ces facettes. Les méthodes proposées qui utilisent ce principe [CARL82, YAMA84, LAID86, TORI91] diffèrent en général par la technique d'échantillonnage ou par le moyen utilisés pour calculer l'intersection entre deux facettes. Mais toutes ces méthodes ne sont applicables que dans un contexte où l'utilisateur peut se satisfaire d'une représentation discrète des surfaces. En effet, en utilisant ces méthodes, la représentation continue initiale des surfaces est perdue dès que les combinaisons booléennes sont effectuées et l'on bascule vers une description moins riche et moins précise.

Pour éviter ceci, une troisième famille de méthodes, plus récentes, proposent d'évaluer une approximation des courbes d'intersection entre deux surfaces paramétriques en conservant leur description d'origine. En général, ces méthodes utilisent, pour chaque surface, une représentation paramétrique *et* une représentation facettisée. Le plus souvent, la seconde est utilisée pour calculer l'approximation des courbes d'intersection, tandis que la première permet d'affiner cette approximation au fur et à mesure du déroulement des calculs. On dispose donc à la fin de ces calculs, d'une représentation approchée des courbes d'intersection et d'une représentation exacte des surfaces de départ. Comme ces méthodes représentent un bon compromis entre les deux premières familles de méthodes, elles ont engendré un grand nombre de travaux qui vont être détaillés dans les parties B et C.

B - Méthodes d'intersection par subdivisions

B1 - Principe

Comme nous l'avons vu dans la partie précédente, calculer les courbes d'intersection entre deux surfaces paramétriques quelconques est un problème dont la résolution est complexe. Un problème beaucoup plus simple à résoudre, par contre, est celui qui consiste à calculer l'intersection entre deux surfaces planes. En tenant compte de cette constatation élémentaire, les *méthodes d'intersection par subdivisions* proposent de transformer le problème complexe en plusieurs (*i.e.* un grand nombre de) problèmes simples.

La démarche à suivre consiste à subdiviser chacune des deux surfaces dont on cherche l'intersection, jusqu'à ce qu'elles ne soient composées que de morceaux plans. Il suffit ensuite de calculer l'intersection des morceaux qui décrivent la première surface avec ceux qui décrivent la seconde pour obtenir une succession de segments de droite. Mis bout à bout, ces segments donneront une approximation des courbes recherchées et donc une solution au problème. Bien entendu, cet exemple est volontairement schématique, il n'a pour seul but que de mettre en évidence le principe de base commun des méthodes décrites dans cette partie : *diviser pour règner.*

B2 - Algorithme général

Pour calculer les courbes d'intersection entre deux surfaces, toutes les méthodes par subdivisions sont composées des cinq étapes suivantes :

- **Etape 1 :** Au cours de cette phase, on vérifie si les deux surfaces étudiées S et S' sont susceptibles de s'intersecter. Si tel est le cas, on passe à l'étape 2 ; sinon, le processus de calcul des courbes d'intersection s'arrête pour les deux surfaces traitées.

- **Etape 2 :** On met ici en oeuvre un critère d'arrêt pour le processus de subdivision des surfaces. Le rôle de ce critère est de signaler que l'intersection des deux surfaces en cours de traitement est suffisamment simple pour être calculée. Si le critère est vérifié *pour les deux surfaces*, on passe à l'étape 4. Dans le cas contraire, il est nécessaire d'effectuer (ou de continuer) la subdivision par l'étape 3.

- **Etape 3 :** Dans cette étape, on réalise la subdivision de chacune des deux surfaces en n sous-surfaces. On notera S_i (avec $i = 1...n$) et S'_j (avec $j = 1...n$), les n sous-surfaces issues de S et S' respectivement. On associe ensuite les différentes sous-surfaces pour former l'ensemble des n^2 paires possibles $\{S_i, S'_j\}$. Enfin, on recommence le processus à partir de l'étape 1 pour chacune des paires de sous-surfaces ainsi formées ; c'est donc ici que s'établit la récursion. Il faut noter que pour les besoins de l'étape 5, on stocke l'historique des subdivisions effectuées pour chaque surface dans un arbre d'arité n.

- **Etape 4 :** Lorsqu'on se trouve dans l'étape 4, cela signifie que les deux sous-surfaces en cours sont susceptibles de s'intersecter et que leur intersection est simple à calculer. On effectue donc ce calcul et on stocke, dans les arbres de subdivisions, le segment de courbe obtenu en vue de l'étape 5.

- **Etape 5 :** Cette dernière étape est effectuée lorsque toutes les paires de sous-surfaces générées lors des subdivisions successives ont, selon le cas, soit été éliminées lors de l'étape 1, soit été effectivement intersectées lors de l'étape 4. Il ne reste donc plus qu'à reconstruire les courbes d'intersection finales en raccordant les différents segments de courbes obtenus à l'étape 4. Afin d'effectuer une reconstruction cohérente, on utilise les arbres de subdivisions construits à l'étape 3.

B3 - Détail des méthodes

Les méthodes d'intersection par subdivisions proposées dans la littérature se composent toutes des cinq étapes de l'algorithme général. Les différences entre les méthodes tiennent aux techniques spécifiques employées par les différents auteurs pour implémenter ces étapes.

a - Détection d'intersections potentielles

Pour détecter si deux surfaces sont susceptibles de s'intersecter, la plupart des méthodes se servent de volumes englobant chacune des deux surfaces. La forme de ces volumes doit être assez simple pour permettre d'évaluer facilement s'ils s'intersectent ou s'ils sont disjoints. Si l'intersection des volumes englobants est vide, celle des surfaces l'est *a fortiori* ; dans le cas contraire, l'intersection des surfaces n'est que potentielle. Les méthodes qui appliquent ce principe diffèrent par leur technique de construction des volumes englobants. On peut citer trois constructions différentes :

Dans [NATA90] les volumes englobants sont des parallélépipèdes rectangles dont les faces sont alignés par rapport aux axes de coordonnées du repère global dans lequel sont définies les surfaces. Les coordonnées des sommets de ce parallélépipède sont calculées simplement en fonctions des valeurs maximales et minimales des coordonnées des quatres coins de la surface englobée. Dans [HOUG85] les volumes sont définies de manière plus fine par des parallélépipèdes rectangles alignés par rapport aux axes d'un repère local à la surface englobée. Ce repère local est obtenu par produits vectoriels, à partir de trois des quatre coins de la surface étudiée. Enfin, quand le modèle géométrique le permet (surfaces de Bézier, surfaces NU(R)BS, etc) on peut, comme dans [PENG84, LASS86], définir le volume englobant en utilisant l'enveloppe convexe de la surface formée par le réseau polygonal des points de contrôle.

b - Arrêt de la subdivision

Pour décider d'arrêter la subdivision d'une surface, le critère retenu est généralement le même pour toutes les méthodes existantes : on ne subdivise pas une surface que l'on considère “suffisamment” plane. Ceci nécessite donc de pouvoir définir et évaluer un *critère de pseudo-planéité* pour une surface. Pour ce faire, il existe là-encore plusieurs techniques :

Dans le cas de surfaces définies à l'aide de points de contrôle, on peut se servir de ces derniers pour évaluer cette condition de planéité. Par exemple dans [PENG84], l'auteur propose de former un carreau plan avec trois des quatre points de contrôle qui se trouvent aux coins du réseau polygonal. Puis il évalue la distance de chacun des autres points de contrôle à ce plan. Si pour tous les points, cette distance est inférieure à une tolérance choisie, il considère que la surface est plane.

Lorsque l'on désire rester indépendant d'un modèle géométrique précis, une solution consiste à utiliser des caractéristiques mathématiques générales de la surface, telles que la variation des vecteurs tangents et/ou normaux. Par exemple, un critère d'arrêt raisonnable pour le processus de subdivision [HOUG85] peut être de calculer les variations de ces vecteurs aux quatre coins de la surface puis de les comparer avec une tolérance choisie. De manière moins empirique, on peut montrer [NATA90] que l'erreur effectuée lorsque l'on approxime une surface par les deux triangles formés par ses quatres coins, est bornée par la norme de la dérivée seconde de la surface. Ainsi, on vérifiant que cette norme reste inférieure à une tolérance donnée, on obtient un critère de planéité fiable.

c - Subdivision des surfaces

Deux solutions principales existent dans la littérature pour subdiviser une surfaces paramétrique en plusieurs sous-surfaces. La première technique consiste à générer les sous-surfaces en partageant l'espace des paramètres (u, v) de la surface initiale [HOUG85]. Par exemple, si $S(u, v)$ est une surface paramétrique définie sur l'intervalle $[u_1, u_2]\times[v_1, v_2]$, on peut la subdiviser en quatre sous-surfaces définies respectivement sur les intervalles $[u_1, u_m]\times[v_1, v_m]$, $[u_1, u_m]\times[v_m, v_2]$, $[u_m, u_2]\times[v_1, v_m]$ et $[u_m, u_2]\times[v_m, v_2]$ où $u_m \in\ ]u_1, u_2[$ et $v_m \in\ ]v_1, v_2[$. Cette technique a le double avantage d'être très simple à mettre en oeuvre et d'être applicable à n'importe quel modèle de surface paramétrique.

Une autre solution, plus complexe à mettre en oeuvre mais souvent plus performante, consiste à utiliser des algorithmes de subdivision spécifiques au modèle de la surface en cours de traitement [PENG84, LASS86]. On peut citer l'insertion d'un nouveau noeud dans le cas de surfaces NU(R)BS ou la subdivision de De Casteljau dans le cas de surfaces de Bézier.

d - Calcul d'intersection

Le rôle de la quatrième étape est de réaliser le calcul effectif des segments de courbes d'intersection obtenus par intersection des sous-surfaces. En fait, ce calcul est assez simple puisqu'il ne débute que lorsque toutes les paires de sous-surfaces susceptibles de s'intersecter sont quasiment planes. Il existe ainsi deux manières d'aborder le problème :

Soit on considère que les deux sous-surfaces forment des quadrangles *parfaitement* plans [PENG84, AZIZ90], et l'on calcule directement l'intersection entre ces quadrangles (en utilisant les plans définis par trois des quatre coins des sous-surfaces). Soit on considère que les deux sous-surfaces sont *imparfaitement* planes, et l'on commence par découper chacune d'elles en deux triangles pour assurer la planéité [HOUG85, LASS86, NATA90]. On forme ensuite les quatre paires de triangles possibles et on calcule les segments d'intersection (qui sont donc au nombre de quatre au maximum). Dans les deux cas, on stocke le résultat dans les arbres de subdivision, au niveau des noeuds correspondants aux sous-surfaces en cours de traitement.

e - Construction des courbes d'intersection

On doit maintenant reconstruire les courbes d'intersection finales. Pour cela, la manière la plus simple consiste à recoller entre eux tous les segments de droite stockés dans l'historique des subdivisions [HOUG85, NATA90]. Cette reconstruction nécessite par exemple d'évaluer la distance entre les extrémités de deux segments et de les recoller si cette distance est suffisamment faible. Avec un même critère de distance, on peut également raccorder un segment sur la frontière d'une surface. Enfin, il faut noter que ce mécanisme fournit sans difficulté toutes les courbes-solutions.

Une autre possibilité [PENG84] consiste regrouper les quatrième et cinquième étapes en une seule. En effet, à chaque calcul d'un segment d'intersection, on peut utiliser les arbres de subdivisions pour détecter la paire de sous-surfaces qui permet de calculer la suite de la courbe d'intersection. L'avantage est que les courbes sont construites progressivement, dans un ordre cohérent ; l'inconvénient est qu'une même courbe peut être représentée de manière non-connexe par plusieurs sous-courbes si l'itération n'a pas débuté par l'une de ses deux extrémités.

Il faut noter que pour l'une ou l'autre des deux techniques, cette dernière étape est loin d'être triviale, principalement dans les deux cas où les courbes d'intersection ne sont pas simples et contiennent des embranchements ou des boucles.

f - Remarques

Les méthodes présentées ici proposent donc des réponses différentes pour réaliser les cinq étapes nécessaires à l'élaboration de l'algorithme général. Il faut cependant remarquer que souvent le choix d'une technique spécifique pour une étape limite les choix possibles pour une autre.

Par exemple, dans le cas des surfaces de Bézier, si on choisit d'utiliser l'enveloppe convexe formée des points de contrôle pour créer le volume englobant à l'étape 2, il faudra ensuite utiliser impérativement la méthode de subdivision de De Casteljau lors de l'étape 3, afin que chaque sous-surface dispose aussi de sa propre enveloppe convexe. Si on choisissait de subdiviser la surface en partageant simplement l'espace des paramètres, on ne créerait pas de nouveaux points de contrôle et chaque sous-surface disposerait de la même enveloppe convexe que la surface initiale et par conséquent du même volume englobant. Par conséquent, le fait de subdiviser deux surfaces susceptibles de s'intersecter ne donnerait que des paires de sous-surfaces susceptibles de s'intersecter également ; on ne pourrait donc plus éliminer les paires de sous-surfaces disjointes.

Néanmoins, si l'on tient compte des exclusions mutuelles entre les solutions proposées, il est tout à fait possible de créer de nouvelles méthodes en utilisant des combinaisons différentes pour les choix proposés à chacune des étapes.

Pour conclure, remarquons que la notion de "diviser pour règner" est connue pour s'adapter particulièrement bien à des implémentation sur des architectures parallèles. De fait, différents auteurs ont proposé des implémentations parallèles des techniques de calculs d'intersection de surfaces par subdivisions (*cf.* [KAUF90] pour un survol des méthodes proposées). Malheureusement, si la répartition des calculs se fait de manière simple et naturelle, la gestion des liens entre les différentes sous-surfaces, nécessaires à la reconstruction correcte des courbes d'intersection, est difficile et engendre souvent de forts goulots d'étranglement.

C - Méthodes d'intersection par progressions

C1 - Principe

Lorsque l'on souhaite calculer une courbe d'intersection entre deux surfaces S et S', le principe des *méthodes d'intersection par progressions* est d'utiliser un point P de cette courbe ainsi que les caractéristiques des surfaces S et S' au voisinage de P pour trouver un point P' proche de P sur la même courbe d'intersection. En réitérant le procédé à partir de P', on peut ainsi calculer la courbe au fur et à mesure, par progressions successives d'un point à l'autre. Bien entendu, un requis primordial pour un telle technique est de connaître au moins un point de la courbe d'intersection pour débuter la progression.

C2 - Algorithme général

Pour calculer les courbes d'intersection entre deux surfaces, toutes les méthodes par progressions sont composées des trois étapes suivantes :

- **Etape 1 :** L'étape initiale consiste à trouver sur la courbe d'intersection, un point P_0 (appelé *point de départ*). Il est important de rappeler que l'intersection entre deux surfaces peut être formée de plusieurs courbes différentes. Par conséquent, il faut que les techniques utilisées dans cette étape soient capables de donner au moins un point de départ par courbe-solution.

- **Etape 2 :** Dans cette étape, on cherche un point P_1 (appelé *point suivant*) qui doit se trouver sur la courbe d'intersection et dans le voisinage du point de départ. Pour cela, il faut définir la direction, le sens et la norme du pas de progression à utiliser. Puis, on applique ce pas à partir de P_0 pour trouver P_1 et on réitère le procédé pour trouver les points suivants (P_2, P_3...). A chaque itération, on calcule un nouveau pas de progression, en fonction des caractéristiques des surfaces au voisinage du point courant.

  La progression s'arrête lorsque la courbe en cours de calcul vérifie une condition d'arrêt spécifique. Si la courbe calculée n'est pas fermée, il faut alors revenir à son point de départ et calculer la courbe dans l'autre sens. Quand une courbe est entièrement calculée, on construit les autres courbes éventuelles de la même manière, à partir des autres points de départ.

- **Etape 3 :** Enfin, la dernière étape consiste à raccorder, si nécessaire, les différents morceaux obtenus pour former les courbes finales.

Avec un tel algorithme, on génère donc une série de points (P_0, P_1, P_2...) qui forment une approximation, souvent assez grossière, de la courbe d'intersection. En particulier, certains de ces points peuvent n'être sur aucune des deux surfaces étudiées. Par conséquent, afin d'obtenir une approximation plus fine, on applique à chaque point P_i ainsi calculé, une méthode de raffinement itératif jusqu'à ce que l'on considère que la position du point obtenu est "suffisamment" proche de la courbe réelle. On partira ensuite de ces nouvelles coordonnées pour trouver le point suivant selon le principe général décrit ci-dessus.

C3 - Détail des méthodes

Parmi les méthodes d'intersection par progressions proposées dans la littérature, on peut recenser plusieurs techniques pour effectuer, de manière plus ou moins performante, chacune des trois étapes de l'algorithme général.

a - Point de départ

En ce qui concerne la première étape, le choix d'un point de départ doit être judicieux. En effet, comme on sera amené à appliquer aux coordonnées de ce point, une méthode de raffinement itératif, il faut être certain que ce raffinement va converger, et si possible rapidement, vers un point de la courbe d'intersection. En tenant compte de cette remarque, les différentes techniques proposent de choisir un point déjà suffisamment proche de cette courbe pour converger de manière certaine.

Une possibilité, présentée dans [BARN87], consiste à trianguler adaptativement les surfaces S et S' et représenter chacune de leurs isoparamétriques par des suites de segments de droite. Il suffit alors de chercher l'intersection entre les isoparamétriques approximées de S et les triangles de S' puis de recommencer l'opération en échangeant les rôles de S et S'. Tous les points ainsi calculés sont des points de départ d'une éventuelle courbe d'intersection. Toutefois, il est possible que plusieurs de ces points de départ appartiennent à la même courbe d'intersection. Par conséquent, pour éviter de parcourir plusieurs fois une même courbe, les auteurs proposent l'algorithme suivant : dès qu'un point de départ est trouvé, la courbe correspondante est calculée en utilisant les étapes 2 et 3 de l'algorithme, ensuite, sur chaque isoparamétrique approximée, on supprime toutes les intersections avec cette courbe avant de chercher un autre point de départ.

Dans [BARN90] la technique utilisée pour calculer les points de départ est issue des méthodes d'intersection par subdivisions (*cf.* Partie B). Plusieurs critères différents sont proposés pour arrêter la subdivision des surfaces. On peut, par exemple, générer une subdivision uniforme ou une subdivision adaptative basée sur la géométrie des surfaces. Dans tous les cas, les volumes englobants utilisés sont issus de ceux proposés dans [HOUG85] et l'évaluation de leur intersection est faite par des tests successifs d'appartenance ou d'intersection entre leurs sommets, arêtes ou faces. Lorsque le critère d'arrêt est vérifié, on calcule un point de départ pour chaque paire de sous-surfaces susceptibles de s'intersecter, en faisant la moyenne des huit coins des sous-surfaces concernées.

Enfin, dans [KOPA91], le calcul du point de départ utilise également la subdivision récursive des surfaces. Cependant, le critère d'arrêt de la subdivision, à la fois original et très puissant, utilise une théorie mathématique établie par l'auteur, basée sur le théorème du point fixe. En effet, lorsque deux sous-surfaces vérifient ce critère d'arrêt, on est assuré, d'une part, que la courbe d'intersection entre ces deux sous-surfaces est unique et non fermée, et d'autre part, que n'importe quel point pris sur ces deux sous-surfaces, convergera vers cette courbe d'intersection par une méthode de raffinement itératif. De plus, si on considère alors l'hypercube H des paramètres[2], on peut affirmer que la courbe d'intersection cherchée débute et finit sur deux des faces de H. Il suffit donc de calculer l'un de ces deux points-extrémité et de s'en servir comme point de départ.

b - Point suivant

La deuxième étape débute lorsque le point de départ P_0 a été trouvé et que sa position a été raffinée itérativement. Il s'agit maintenant de calculer la direction et la norme du pas de progression à appliquer à P_0 pour trouver le point suivant P_1.

Dans [BARN87] et [BARN90], la direction du pas est celle du vecteur directeur de la droite d'intersection des deux plans tangents aux surfaces S et S' en P_0. La norme du pas est donnée dans [BARN87] par une tolérance fixée par l'utilisateur, alors que dans [BARN90], elle varie en fonction d'une approximation du rayon de courbure au point P_0. Le pas ainsi calculé est appliqué à partir de P_0 et son extrémité définit donc le point suivant P_1 auquel on applique le même procédé après

[2] Si $S(u,v)$ est définie sur $[u_1,u_2]\times[v_1,v_2]$ et $S'(s,t)$ est définie sur $[s_1,s_2]\times[t_1,t_2]$, l'hypercube H des paramètres est défini par $H=[u_1,u_2]\times[v_1,v_2]\times[s_1,s_2]\times[t_1,t_2]$

avoir raffiné itérativement sa position. La courbe est ainsi construite pas à pas jusqu'à ce qu'un critère d'arrêt soit vérifié. Dans les deux méthodes décrites ici, les critères d'arrêt sont, par exemple, l'intersection avec une courbe précédemment calculée, l'intersection avec les frontières des surfaces ou encore la fermeture de la courbe. Lorsque toutes les courbes ont été calculées, ces critères servent à faciliter la reconstruction dans l'étape finale.

Dans [KOPA91], la progression se fait dans l'hypercube des paramètres H. En effet, lors du calcul du point de départ, le critère d'arrêt de la subdivision a permis de placer les deux extremités de la courbe d'intersection sur deux faces de l'hypercube. Par conséquent, il suffit de tracer, dans H, la droite qui relie ces deux points et de l'échantillonner avec un pas constant pour obtenir les coordonnées paramétriques (u_i, v_i, s_i, t_i) des différents points de la progression. Ces points servent ensuite de base à un raffinement itératif dans H jusqu'à ce que les points $P_i = S(u_i, v_i)$ et $P'_i = S'(s_i, t_i)$ soient suffisamment proches. Dans cette méthode, il n'y a pas besoin de critère d'arrêt de la progression puisque, d'une part, on connait le point d'arrivée du segment de courbe d'intersection entre les deux sous-surfaces, et que d'autre part, la subdivision initiale assure l'unicité et la non-fermeture de ce segment de courbe.

c - Reconstruction des courbes d'intersection

Du fait du critère qui arrête la construction des courbes lorsque deux d'entre-elles se coupent, les méthodes de [BARN87] et [BARN90] fournissent, à la fin de la deuxième étape, un ensemble de segments de courbes disjoints. Au cours de la troisième étape, il faut donc reconstruire toutes les courbes représentant l'intersection des surfaces initales. Dans [BARN90], le morcellement est également issu de la subdivision récursive utilisée pour trouver les points de départ. Dans les deux méthodes, on raccorde donc les courbes qui s'intersectent en calculant le point de branchement. Dans [BARN90], on doit également raccorder les segments issus de sous-surfaces différentes, en s'aidant des arbres de subdivisions (*cf.* Partie B).

Dans [KOPA91], le principe est différent car on regroupe les deuxième et troisième étapes en une seule. Chaque fois qu'une paire de sous-surfaces susceptibles de s'intersecter vérifie le critère d'arrêt de la subdivision, on la stocke dans une liste. Lorsque toutes les paires de sous-surfaces ont été soit éliminées, soit stockées dans cette liste, le calcul du premier segment de courbe peut démarrer. Pour cela, on extrait une paire de sous-surfaces de la liste et on effectue les calculs décrits dans la deuxième étape. A la fin de ces calculs, du fait du critère mathématique utilisé, le segment de courbe obtenu se termine nécessairement sur la frontière d'une des sous-surfaces, il est donc facile de trouver dans la liste, la paire de sous-surfaces qui va prolonger le segment courant, ce qui permet de calculer les courbes de manière continue. Enfin, le calcul s'arrête naturellement lorsque la liste est vide.

d - Raffinement itératif

Les trois méthodes décrites ci-dessus permettent donc de calculer l'intersection de deux surfaces paramétriques S et S'. Cependant, la série de points P_i qu'elles calculent dans un premier temps ne représente pas une approximation suffisamment correcte de la véritable courbe d'intersection. Afin d'améliorer cette approximation, on utilise une méthode de raffinement itératif. Dans les trois méthodes, le processus utilisé est très similaire et peut être résumé par les quatre phases suivantes (*cf.* Figure 6.1) :

- Projeter le point P que l'on souhaite raffiner, en A sur S et en A' sur S'.
- Tracer la droite D, intersection entre le plan tangent à S en A et le plan tangent à S' en A'.
- Projeter A et A' orthogonalement sur D pour obtenir les points B et B'.

- Si la distance entre A et A' est supérieure à une tolérance prédéfinie, réitérer tout le processus à partir du point Q, milieu de B et B'.

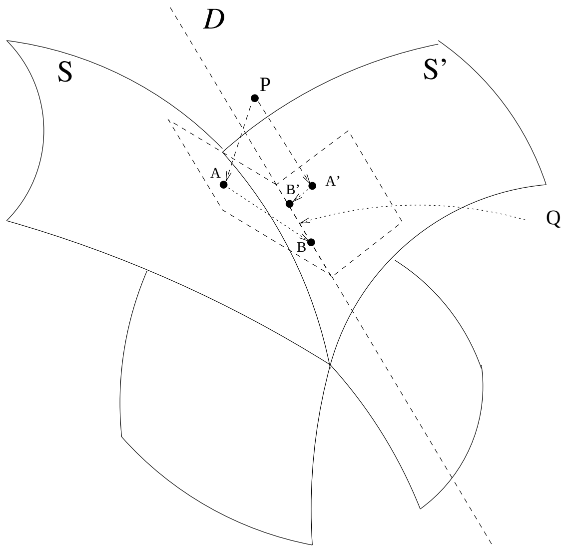


Figure 6.1: *Principe géométrique du raffinement itératif*

D - Discussion

Les méthodes par subdivisions et les méthodes par progressions, apportent différentes réponses au problème complexe qui consiste à calculer les courbes d'intersection entre deux surfaces paramétriques. Néanmoins ces méthodes ne sont pas pleinement satisfaisantes, et ce pour plusieurs raisons.

a - Comparaison entre les deux types de méthodes

D'un côté, les méthodes par subdivisions sont robustes au sens où elles permettent de détecter sans oubli toutes les courbes d'intersection entre deux surfaces données. Cependant, à cause même du principe de subdivisions récursives, le nombre de paires de surfaces à étudier pour résoudre le problème croît de manière exponentielle, ce qui engendre un coût de calcul important lorsque la condition de planéité tarde à être vérifiée (*i.e.* pour les surfaces relativement "tourmentées"). Une comparaison entre les méthodes par subdivisions et les méthodes par progressions, effectuée dans [AZIZ90], montre que les secondes sont bien plus performantes en temps de calcul. De plus, la reconstruction des courbes finales après subdivisions nécessite de stocker l'intégralité des arbres de subdivision pour les deux surfaces (auxquels il faut rajouter des liens supplémentaires entre les noeuds des deux arbres) ce qui implique que le coût de stockage est également exponentiel.

De leur côté, les premières méthodes par progressions qui ont été proposées, n'utilisent pas de subdivisions récursives et sont donc peu encombrantes en mémoire. Par contre, elles ne sont pas robustes car rien ne peut garantir que l'on va obtenir toutes les courbes d'intersection entre les surfaces initiales. Afin d'essayer de remédier à ce défaut, les méthodes par progressions plus récentes intègrent une phase de subdivisions récursives pour arriver à détecter au moins un point de départ pour chaque courbe-solution.

Or la qualité de telles méthodes dépend directement du critère utilisé pour basculer de la phase de subdivision à la phase de progression. En effet, si l'on bascule trop tôt, il se peut qu'on n'ait pas encore détecté tous les points de départ nécessaires ; par contre, si on bascule trop tard, le nombre de subdivisions effectuées rendra la méthode aussi coûteuses que les méthodes de subdivisions complètes. A ce titre, la méthode de [BARN90] est très insatisfaisante car le basculement entre les deux phases se fait selon des critères empiriques qui ne garantissement ni la robustesse (on peut manquer des courbes-solutions) ni l'économie du coût de stockage (on peut parfois subdiviser jusqu'à vérifier une condition de planéité pure et simple). La méthode de [KOPA91] par contre, garantit que le basculement se fait exactement au moment adéquat, du fait du critère mathématique proposé par l'auteur. En contre-partie, la technique de raffinement itératif destinée à ramener les points sur la courbe d'intersection (technique spécifique qui est imposée par l'utilisation de ce critère mathématique) possède une convergence beaucoup plus lente que le raffinement géométrique utilisé dans [BARN87, BARN90].

b - Obtention d'une courbe d'intersection continue

Aussi bien les méthodes par subdivisions que celles par progressions, ne fournissent qu'un ensemble de points comme solution du problème d'intersection. Et malgré l'attention apportée au calcul de ces points, ils demeurent une approximation des véritables points de la courbe d'intersection. Pour obtenir une courbe, il faut ensuite relier ces points entre eux ; deux possibilités s'offrent alors à nous.

La première solution consiste à utiliser les coordonnées euclidiennes (x_i, y_i, z_i) des points P_i ainsi calculés pour les joindre par des segments de droite ou des splines d'interpolation. Or le degré de la courbe exacte étant très élevé (*cf.* Partie A) on comprend aisément que quel que soit le moyen utilisé, la courbe finalement obtenue ne pourra être qu'une approximation de la courbe exacte. Il est même tout à fait possible qu'elle n'appartienne pas entièrement à l'une ou l'autre — voire aux deux — surfaces initiales.

La deuxième solution consiste à interpoler les points P_i dans l'espace des paramètres, c'est-à-dire employer une courbe interpolant les (u_i, v_i) pour S et une autre interpolant les (s_i, t_i) pour S'. De ce fait, on est assuré que la première courbe se trouve entièrement sur S et la seconde entièrement sur S'. Par contre, avec un tel procédé, rien ne permet d'affirmer que les deux courbes vont être confondues, et de fait, il pourra exister des fentes ou des chevauchements entre les deux surfaces après combinaison.

Parmi ces deux solutions, la seconde est devenue très populaire depuis son introduction [CASA87, CROC87] dans le cadre plus vaste des "surfaces découpées" (en anglais, *trimming surfaces*). Cette popularité s'explique par le fait que des algorithmes de visualisation spécifiques ont été développés [SHAN88, ROCK89] qui permettent de visualiser, quasiment en temps-réel, ce type de surfaces sur des stations de travail spécialisées. L'existence de tels algorithmes permet à l'utilisateur de ne manipuler à tout moment que des représentations explicites continues (aussi bien pour les surfaces que pour les courbes) ; ce n'est qu'au moment de la visualisation que la surface est discrétisée en facettes, de manière totalement automatique par l'algorithme.

c - Manipulations interactives

En règle générale, toutes les méthodes précédentes se prêtent assez mal à la création d'un outil intuitif et interactif. En effet, quelle que soit la technique employée, lorsqu'on combine des objets avec des opérations booléennes, on obtient des courbes d'intersection (sous forme de liste de points-échantillons ou de splines d'interpolation) qui sont *figées*. En effet, comme ces courbes sont le résultat d'un calcul d'intersection, l'utilisateur ne peut pas modifier leur forme (en déplaçant les points de contrôle, par exemple) sans briser leur propriété intrinsèque qui est d'appartenir aux deux surfaces. Par conséquent, la seule manière pour un utilisateur de changer le résultat d'une opération de combinaison qui ne lui convient pas, est de modifier les objets initiaux (position, orientation, taille ou forme) puis de relancer un calcul complet d'intersection. Comme un calcul de ce type peut prendre usuellement plusieurs dizaines de secondes, voire plusieurs minutes, l'utilisateur se retrouve enfermé dans un *processus de rétroaction lente* qui est justement le cas à éviter, selon les spécialistes des interfaces homme/machine.

Il faut remarquer qu'il existe tout de même une exception notable (et à notre connaissance, unique) à cette interdiction de modification directe, représentée par la technique de *raccordement perpendiculaire* proposée par M-P. Gascuel [GAS90]. Avec cette méthode, l'utilisateur peut directement "dessiner" la courbe d'intersection désirée, puis les deux surfaces vont se déformer localement pour s'adapter à cette courbe d'intersection. De plus, comme les calculs résultants sont généralement moins complexes que pour les méthodes d'intersection génériques, cette méthode permet d'obtenir des temps de réponse plus agréables. Même si en l'état, la méthode est limitée à une famille de surfaces particuliéres (les M-splines uniformes) et à des types d'intersection précis (la courbe d'intersection doit nécessairement devenir une isoparamétrique pour l'une des deux surfaces), elle constitue une approche nouvelle pour les combinaisons de surfaces qui doit ne demande qu'à être approfondie.