

### 3. Approximation uniforme des fonctions continues. Courbes de Bernstein, Bézier et B-Splines

#### 3.1 Position du problème

Etant donnée une fonction continue,  $f : [a,b] \rightarrow \mathbb{R}$ , dont on connaît les valeurs en un nombre fini de points  $x_0, x_1, \dots, x_n$ , on voudrait calculer, pour tout  $x$  dans  $[a,b]$ , une valeur approchée de  $f(x)$ . De plus on voudrait que la précision de cette approximation augmente avec  $n$ .

Le *théorème de Weirstrass* (1855), qui nous dit que toute fonction continue sur un intervalle fermé borné peut être approchée uniformément par des polynômes, nous permet de répondre à la question. Une démonstration de ce théorème peut se faire de façon constructive en utilisant les *polynômes de Bernstein* (Cf § 3.2).

Les restrictions de cette construction sont, d'une part que le modèle de courbe est de type cartésien, i. e. de la forme  $y = f(x)$ , ce qui exclut les courbes fermées, et d'autre part qu'il est nécessaire de connaître  $f$  en des points équidistants.

Si on s'intéresse au cas, plus général, des courbes définies par des équations paramétriques dans  $\mathbb{R}^2$  ou  $\mathbb{R}^3$ , on aboutit aux procédés d'approximations de Bézier et aux approximants B-Splines.

L'idée de base de ces méthodes est, étant donné un nuage de points :

$S = \{M_i = (x_i, y_i) ; 0 \leq i \leq n\}$ , de chercher des approximations polynomiales d'une paramétrisation  $(x(t), y(t))$  en prenant  $[0,1]$  comme espace des paramètres, avec :

$$\begin{cases} x\left(\frac{i}{n}\right) = x_i \\ y\left(\frac{i}{n}\right) = y_i \end{cases} \quad (0 \leq i \leq n)$$

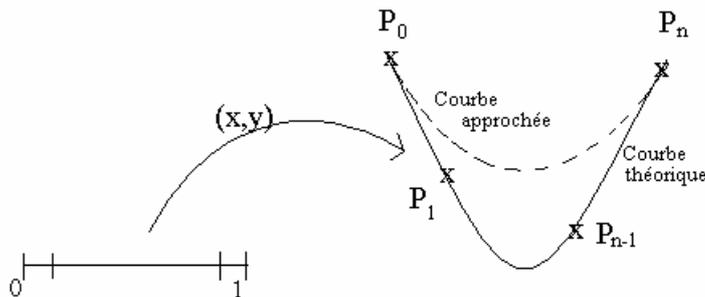


Figure 5.3

Ces procédés vont aussi s'appliquer à la définition de surfaces à partir d'une série de points de  $\mathbb{R}^3$  donnés par l'utilisateur en fonction d'expérimentations ou toute autre considération.

Ces idées sont utilisées en *conception assistée par ordinateur (CAO)* pour définir des carrosseries de voiture ou des fuselages d'avions.

En France, c'est *P. Bezier* qui est à l'origine de ce genre d'idées avec le logiciel *UNISURF* qu'il développa pour le compte de la régie Renault à partir de 1962 (Cf. Math et C.A.O., vol. 4, Chap. 4).

La technique des courbes B-Splines est une généralisation des courbes de Bézier qui ont le désavantage d'avoir une nature globale, c'est-à-dire que la modification d'un point de contrôle  $M_i$  va modifier toute la courbe, contrairement aux B-Splines qui ont un caractère local.

### 3.2 Les bases de Bernstein. Polynômes de Bernstein

#### 3.2.1 Les bases de Bernstein

Pour tout  $n \geq 1$ , on note  $R_n[t]$  l'espace vectoriel des polynômes de degré au plus  $n$ .

*Définition* : On appelle *base de Bernstein d'ordre  $n$* , la base de  $R_n[t]$  formée des polynômes  $B_{n,k}$  définis par :

$$B_{n,k}(t) = C_n^k \cdot t^k \cdot (1-t)^{n-k} \quad (0 \leq k \leq n)$$

*Remarque* — En écrivant que, pour tout  $k = 0, 1, \dots, n$  :

$$t^k = (t + (1-t))^{n-k} \cdot t^k = \sum_{j=0}^{n-k} C_{n-k}^j \cdot t^j \cdot (1-t)^{n-k-j} \cdot t^k = \sum_{i=k}^n \lambda_{i,k} \cdot B_{n,i}(t)$$

on déduit que  $\{ B_{n,k} ; 0 \leq k \leq n \}$  engendrent  $R_n[t]$  et c'est donc bien une base.

Il est facile de vérifier les :

*Propriétés* :

(i)  $\forall t \in ]0,1[, B_{n,k}(t) > 0$  ;

(ii)  $B_{n,k}(0) = \begin{cases} 0 & \text{si } 1 \leq k \leq n \\ 1 & \text{si } k = 0 \text{ et } n \geq 1 \end{cases}$

(iii)  $B_{n,k}(t) = B_{n,n-k}(1-t)$  ( $0 \leq k \leq n$ )

(iv)  $\sum_{k=0}^n B_{n,k}(t) = 1$

(v) On a la récurrence qui permet de calculer les  $B_{n,k}(t)$  :

$$B_{n,n}(t) = t \cdot B_{n-1,n-1}(t)$$

$$B_{n,0}(t) = (1-t) \cdot B_{n-1,0}(t)$$

$$B_{n,k}(t) = (1-t) \cdot B_{n-1,k}(t) + t \cdot B_{n-1,k-1}(t) \quad (1 \leq k \leq n-1)$$

*Remarque* — Les fonctions  $B_{n,k}(t)$  sont aussi appelées *fonctions mélanges* ou *fonctions pondérantes*.

Sur la figure 5.4, on trouvera les graphes des fonctions  $B_{n,k}$  pour  $0 \leq k \leq n = 5$ .

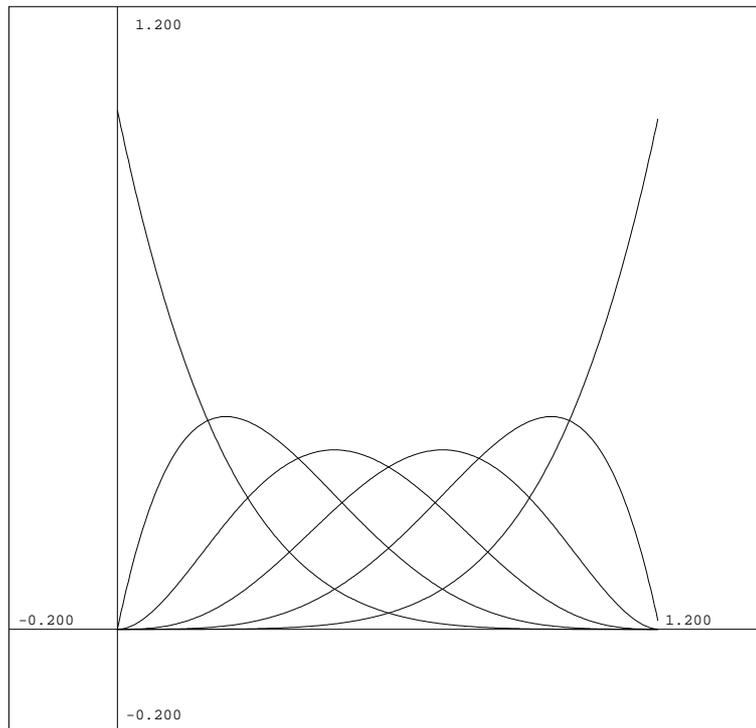


Figure 5.4

Les  $B_{n,k}(t)$  peuvent être calculées de façon récursive avec la procédure suivante.

*FONCTION BaseBernstein\_R(Entrée n, k : Entier ; t : Réel) : Réel ;*

*Début*

*Si n = 0*

*Alors Début*

*BaseBernstein\_R = 1 ;*

*Sortir ;*

*Fin ;*

*Si k = 0*

*Alors BaseBernstein\_R = (1 - t)<sup>n</sup>*

*Sinon Début*

*Aux1 = BaseBernstein\_R(n - 1, k, t) ;*

*Aux2 = BaseBernstein(n - 1, k - 1, t) ;*

*BaseBernstein\_R = (1 - t)·Aux1 + t·Aux2 ;*

*Fin ;*

*Fin ;*

Une autre façon de calculer est d'utiliser la récurrence (V) ci-dessus ce qui donnera, pour n grand, une procédure de calcul beaucoup plus rapide.

L'algorithme de calcul est le suivant :

$$\text{Etape } 0 \text{ — } B_{n,i}^{(0)} = \begin{cases} 1 & \text{si } i = k \\ 0 & \text{sinon} \end{cases} \quad (i = 0, 1, \dots, n)$$

$$\text{Etape } j \text{ — } B_{n,i}^{(j)} = (1 - t) \cdot B_{n,i}^{(j-1)} + t \cdot B_{n,i+1}^{(j-1)} \quad (i = 0, 1, \dots, n - j) \text{ pour } j = 1, 2, \dots, n.$$

Et alors :

$$B_{n,k}(t) = B_{n,0}^{(n)}(t)$$

Cet algorithme est connu sous le nom d'algorithme de *De Casteljeau*. On le retrouvera pour calculer les polynômes de Bernstein et de Bézier.

### 3.2.2 Polynômes de Bernstein associés à une fonction continue

Dans ce paragraphe, on se donne une fonction :  $f : [0,1] \rightarrow \mathbb{R}$ .

*Définition* : Pour tout  $n \leq 1$ , le  $n^{\text{ème}}$  polynôme de Bernstein associé à  $f$  est défini par :

$$B_n(f, t) = \sum_{k=0}^n f\left(\frac{k}{n}\right) \cdot B_{n,k}(t)$$

L'intérêt de cette suite de polynômes est lié au :

*Théorème* : La suite  $(B_n)_{n \in \mathbb{N}}$  converge uniformément vers  $f$  sur  $[0,1]$ .

*Démonstration* — Cf. Davis p. 109.

*Remarque* — Malheureusement la convergence n'est pas très rapide, on peut montrer que l'erreur est un  $O(1/n)$ .

De plus, du point de vue numérique il ne sera pas très pratique de manipuler des polynômes de degré trop élevé.

Un algorithme de calcul est celui de De Casteljeau qui est décrit dans le paragraphe suivant dans le cas des courbes de Bézier.

## 3.3 Les courbes de Bézier

### 3.3.1 Données du problème

On suppose que l'on dispose d'un nuage de points expérimentaux (ou définis par l'utilisateur) :

$$S = \{ P_i = (x_i, y_i) ; 0 \leq i \leq n \}$$

et on voudrait trouver une courbe d'équations paramétriques :

$$\begin{cases} x = x(t) \\ y = y(t) \end{cases} \quad t \in [0,1]$$

qui vérifie :

$$\begin{cases} x\left(\frac{i}{n}\right) = x_i \\ y\left(\frac{i}{n}\right) = y_i \end{cases} \quad (0 \leq i \leq n)$$

En approximant les fonctions  $x$  et  $y$  par leurs approximations de Bernstein respectifs, on aboutit à la définition des courbes de Bézier.

*Remarque* — Les points  $P_i$  sont appelés *points de contrôles* ou *pôles* et le polygone de  $\mathbb{R}^2$  qu'ils définissent est le *polygone descripteur*.

Les courbes de Bézier font partie d'une famille de courbes appelées *courbes à pôles*.

### 3.3.2 Les courbes de Bézier

*Définition* : La *courbe de Bézier* associée aux  $n$  pôles  $P_i$  est la courbe polynomiale de degré  $n$  définie par les équations paramétriques :

$$\begin{cases} P_x(t) = B_n(x, t) \\ P_y(t) = B_n(y, t) \end{cases} \quad t \in [0, 1]$$

ou encore par :

$$\forall t \in [0, 1], P(t) = \sum_{k=0}^n B_{n,k}(t) \cdot P_k$$

Les propriétés de ces courbes sont résumées dans le :

*Théorème* : (i)  $P(0) = P_0$  ;  $P(1) = P_n$  ( $P_0$  et  $P_n$  sont donc les extrémités de la courbe) ;

(ii) la courbe de Bézier est contenue dans l'enveloppe convexe des points de contrôles ;

(iii) la tangente en  $P_0$  à la courbe de Bézier est dirigée par  $\vec{P_0P_1}$  et la tangente en  $P_n$  est dirigée par  $\vec{P_{n-1}P_n}$  ;

(iv) chaque point de contrôle exerce sur la courbe une attraction qui est proportionnelle à  $B_{n,k}(t)$ .

*Démonstration* — Dony p. 111.

On a donc l'allure de courbe suivante :

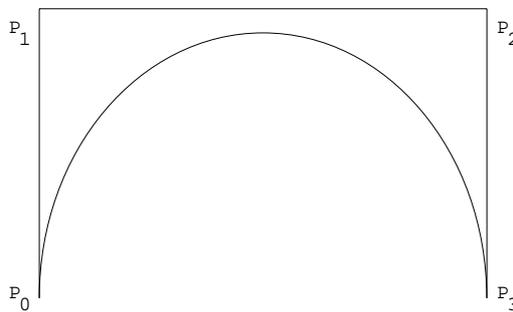


Figure 5.5

*Remarque* — Avec la récurrence (v) du paragraphe (3.2.1) sur les polynômes de Bernstein, en notant  $P^{(u,v)}$  la courbe de Bézier associée au polygone  $\{P_i; u \leq i \leq v\}$ , on déduit que :

$$P^{(0,n)}(t) = (1-t) \cdot P^{(0,n-1)}(t) + t \cdot P^{(1,n)}(t)$$

c'est-à-dire que la courbe de Bézier associée au polygone  $\{P_i; 0 \leq i \leq n\}$  se déduit des courbes associées aux polygones  $\{P_i; 0 \leq i \leq n-1\}$  et  $\{P_i; 1 \leq i \leq n\}$ , ce qui permet de donner un algorithme de construction de telles courbes.

### 3.3.3 Algorithme de De Casteljeau

De ce qui précède, on déduit que si on pose, pour  $t \in [0,1]$  :

$$P_k^{(0)}(t) = P_k \quad (0 \leq k \leq n)$$

$$P_k^{(j)}(t) = (1-t) \cdot P_k^{(j-1)}(t) + t \cdot P_{k+1}^{(j-1)}(t) \quad (0 \leq k \leq n-j; 1 \leq j \leq n)$$

on a alors :

$$P^{(0,n)}(t) = \sum_{k=0}^{n-j} B_{n-j,k}(t) \cdot P_k^{(j)}(t) \quad (j = 0, 1, \dots, n)$$

Ce qui se démontre par récurrence sur  $j$  en utilisant la remarque précédente.

Et pour  $j = n$ , on a alors :

$$P^{(0,n)}(t) = P_0^{(n)}$$

Ce qui nous donne la programmation structurée :

*PROCEDURE Bézier*(Entrée  $n$  : Entier ;  $P$  : TableauDePoints ;  $t$  : Réel ; Sortie  $B$  : Point) ;

*Début*

$Q = P ; \{ Q_k = P_k^{(0)} \}$

*Pour*  $j$  allant de 1 à  $n$  faire

*Début*

*Pour*  $k$  allant de 0 à  $n-j$  faire

*Début*

$P_k = (1-t) \cdot Q_k + t \cdot Q_{k+1} ; \{ P_k = P_k^{(j)}, Q_k = P_k^{(j-1)} \}$

*Fin* ;

*Pour*  $k$  allant de 0 à  $n-j+1$  faire  $Q_k = P_k$  ;

*Fin* ;

$B = P_0$  ;

*Fin* ;

*Remarque 1* — Cet algorithme peut aussi être utilisé pour calculer l'approximant de Bernstein, les calculs se faisant seulement sur la deuxième composante  $y$ .

*Remarque 2* — Cet algorithme est très performant car il évite le calcul récursif des fonctions de base, qui peut être très lent pour  $n$  grand.

*Remarque 3* — Si on désire que la courbe de Bézier passe plus près de l'un des points de contrôle, il suffit de le compter plusieurs fois, on dit que l'on augmente sa multiplicité.

*Remarque 4* — Dans le cas des courbes fermées, on aura en général la situation de la figure 5.6 :

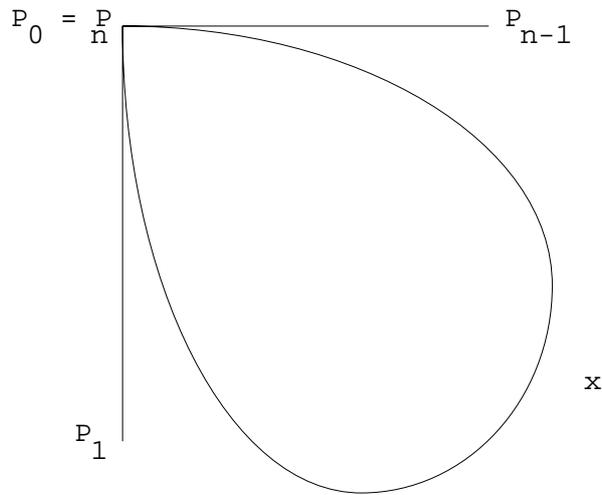


Figure 5.6

Des exemples de tracés sont donnés sur la figure 5.7 où on travaille avec le même fichier de points, en affectant successivement à tous les points les multiplicités 1, 2, 3 et 4.

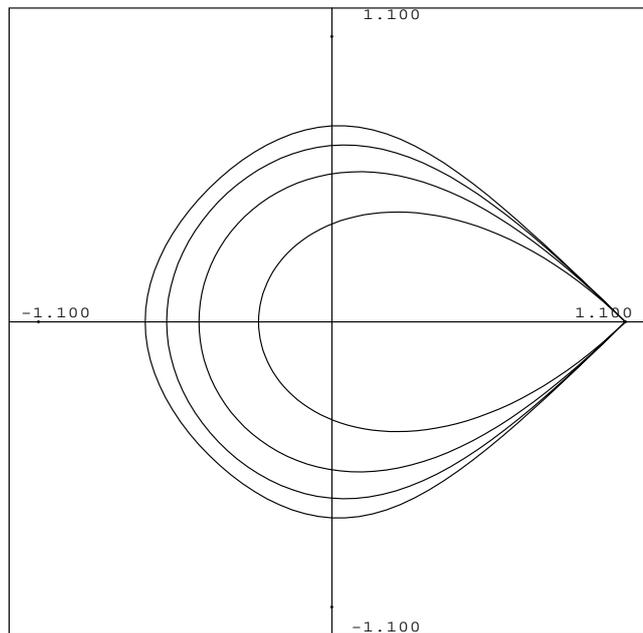


Figure 5.7

### 3.4 Les surfaces de Bézier

#### 3.4.1 Définition

Les définitions précédentes s'étendent au cas des surfaces de  $\mathbb{R}^3$ .

Les données sont un nuage de points de contrôle de  $\mathbb{R}^3$ , formant le *polyèdre descripteur* :

$$S = \{ P_{i,j} ; 0 \leq i \leq n ; 0 \leq j \leq m \}$$

et le *carreau de Bézier* correspondant est la surface d'équations paramétriques :

$$\begin{aligned} B: [0,1]^2 &\rightarrow \mathbb{R}^3 \\ (u,v) &\mapsto \sum_{\substack{0 \leq i \leq n \\ 0 \leq j \leq m}} B_{n,i}(u) \cdot B_{m,j}(v) \cdot P_{i,j} \end{aligned}$$

Une carrosserie automobile peut être définie avec une vingtaine de carreaux de Bézier correctement raccordés.

*Remarque* — On peut écrire que  $B(u,v) = \sum_{i=0}^n B_{n,i}(u) \cdot P_i(v)$ , où on a noté

$$P_i(v) = \sum B_{m,j}(v) \cdot P_{i,j}. \text{ C'est-à-dire que pour tout } v \text{ fixé, } u \rightarrow B(u,v) \text{ est la courbe}$$

de Bézier associée au polygone descripteur  $\{ P_i(v) ; 1 \leq i \leq n \}$ . On a un résultat analogue à  $u$  fixé.

### 3.4.2 Propriétés

Les propriétés de ces surfaces sont résumées dans le théorème ci-dessous.

*Théorème* : (i) Le bord de la surface de Bézier définie par le polyèdre descripteur  $S$  est formé des courbes de Bézier définies par les polygones  $\Gamma_0 = \{P_{0,0}, P_{0,1}, \dots, P_{0,m}\}$ ,  $\Gamma_n = \{P_{n,0}, P_{n,1}, \dots, P_{n,m}\}$ ,  $\Gamma'_0 = \{P_{0,0}, P_{1,0}, \dots, P_{n,0}\}$  et  $\Gamma'_m = \{P_{0,m}, P_{1,m}, \dots, P_{n,m}\}$ . Soit :

$$B(0, v) = \sum_{j=0}^m B_{m,j}(v) \cdot P_{0,j}$$

$$B(1, v) = \sum_{j=0}^m B_{m,j}(v) \cdot P_{n,j}$$

$$B(u, 0) = \sum_{i=0}^n B_{n,i}(u) \cdot P_{i,0}$$

$$B(u, 1) = \sum_{i=0}^n B_{n,i}(u) \cdot P_{i,m}$$

(ii) la surface de Bézier est contenue dans l'enveloppe convexe des points de contrôle ;

(iii) le plan tangent à la surface de Bézier en  $P_{0,0}$  est défini par  $\left( \begin{array}{c} \overrightarrow{P_{0,0}P_{0,1}} \\ \overrightarrow{P_{0,0}P_{1,0}} \end{array} \right)$ ; en  $P_{n,0}$  par  $\left( \begin{array}{c} \overrightarrow{P_{n,0}P_{n,1}} \\ \overrightarrow{P_{n,0}P_{n-1,0}} \end{array} \right)$ ; en  $P_{n,m}$  par  $\left( \begin{array}{c} \overrightarrow{P_{n,m}P_{n,m-1}} \\ \overrightarrow{P_{n,m}P_{n-1,m}} \end{array} \right)$ ; et en  $P_{0,m}$  par  $\left( \begin{array}{c} \overrightarrow{P_{0,m}P_{0,m-1}} \\ \overrightarrow{P_{0,m}P_{1,m}} \end{array} \right)$ ;

### 3.4.3 Algorithme de De-Casteljeau

En utilisant la remarque du paragraphe (3.4.1), il suffit d'utiliser l'algorithme de De-Casteljeau  $n + 1$  fois pour calculer les  $P_i(v)$ , puis de nouveau cet algorithme pour les  $B(u,v)$ .

## 3.5 Les courbes B-Splines

### 3.5.1 Introduction

Les courbes de Bézier présentent les inconvénients suivants :

- ce sont des courbes polynomiales de degré  $n$  pour un polygone à  $n + 1$  points, ce qui entraînera une forte instabilité numérique pour  $n$  grand ;
- la modification d'un seul point de contrôle va se répercuter sur toute la courbe (caractère global des courbes de Bézier) ;
- comme on peut le voir sur les exemples de tracés du paragraphe précédent, la courbe peut passer assez loin des points de contrôle ; pour s'en rapprocher on devra augmenter la multiplicité des points ce qui entraînera une augmentation du degré des fonctions polynomiales à traiter.

C'est en relation avec ces problèmes que dans les années 1970, *M.G. Cox* et *C. De Boor* ont introduit la notion de fonction B-Spline comme généralisation des courbes de Bézier.

Les courbes B-Splines sont définies localement par des expressions polynomiales de degré  $j$  assez faible (usuellement 3 ou 4), ce dernier étant indépendant du nombre de points et la modification d'un point de contrôle ne se fait ressentir qu'au voisinage de ce dernier. De plus elles ont les mêmes propriétés que celles de Bézier.

### 3.5.2 Fonctions de base B-Splines

*Définition* : Soient  $n \geq 1$  et  $j$  dans  $\mathbb{N}$ , avec  $2 \leq j \leq n + 1$ . On appelle alors *vecteur nodal* associé à  $(n,j)$ , le vecteur  $v$  de  $\mathbb{N}^{n+j+1}$  défini par :

$$\begin{cases} v_i = 0; & 0 \leq i \leq j - 1 \\ v_i = i - j + 1; & j \leq i \leq n \\ v_i = n - j + 2; & n + 1 \leq i \leq n + j \end{cases}$$

*Définition* : Soient  $n \geq 1$  et  $j$  dans  $\mathbb{N}$ , avec  $2 \leq j \leq n + 1$ . En notant  $v$  le vecteur nodal associé à  $(n,j)$ , on appelle *base B-Spline d'ordre  $j$* , la suite de fonctions  $\{ N_{i,j} ; 0 \leq i \leq n \}$ , définie sur  $[0, t_{\max}]$ , avec  $t_{\max} = n - j + 2$ , par la récurrence :

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

$$N_{i,k}(t) = \alpha_{i,k} \cdot (t - v_i) \cdot N_{i,k-1}(t) + \beta_{i,k} \cdot (v_{i+k} - t) \cdot N_{i+1,k-1}(t)$$

pour  $2 \leq k \leq j$ ,  $0 \leq i \leq n$ , où :

$$\alpha_{i,k} = \begin{cases} \frac{1}{v_{i+k-1} - v_i} & \text{si } v_{i+k-1} \neq v_i \\ 0 & \text{sinon} \end{cases}$$

$$\beta_{i,k} = \begin{cases} \frac{1}{v_{i+k} - v_{i+1}} & \text{si } v_{i+k} \neq v_{i+1} \\ 0 & \text{sinon} \end{cases}$$

*Remarque 1* — L'algorithme de construction des  $N_{i,j}$  défini ci-dessus est dû à Cox et De Boor.

*Remarque 2* — Les fonctions  $N_{i,j}$  sont aussi appelées *fonctions mélanges* ou *fonctions pondérantes*.

*Exemple* — Pour  $n = 3$ ,  $j = 2$ , le vecteur nodal est  $v = (0, 0, 1, 2, 3, 3)$  et :

$$N_{0,2}(t) = \begin{cases} 1 - t & \text{si } t \in [0, 1] \\ 0 & \text{sinon} \end{cases}$$

$$N_{1,2}(t) = \begin{cases} t & \text{si } t \in [0, 1] \\ 2 - t & \text{si } t \in [1, 2] \\ 0 & \text{si } t \in [2, 3] \end{cases}$$

$$N_{2,2}(t) = \begin{cases} 0 & \text{si } t \in [0, 1] \\ t - 1 & \text{si } t \in [1, 2] \\ 3 - t & \text{si } t \in [2, 3] \end{cases}$$

$$N_{3,2}(t) = \begin{cases} t - 2 & \text{si } t \in [2, 3] \\ 0 & \text{sinon} \end{cases}$$

Les propriétés de ces fonctions sont résumées dans le :

*Théorème* : (i)  $\forall t \in [0, t_{\max}]$ ,  $N_{i,j}(t) \geq 0$ , avec, pour  $j \geq 2$  :

$$\begin{cases} N_{i,j}(t) > 0 & \text{si } t \in ]v_i, v_{i+j}[ \\ 0 & \text{sinon} \end{cases}$$

et sur  $]v_i, v_{i+j}[$ ,  $N_{i,j}$  est polynomiale de degré  $j - 1$  ;

(ii)  $\forall t \in ]0, t_{\max}[$ ,  $\sum_{i=0}^n N_{i,j}(t) = 1$  ;

(iii)

$$\begin{cases} N_{i,j}(0) = 0 & (1 \leq i \leq n) \\ N_{0,j}(0) = 1 \\ N_{i,j}(t_{\max}) = 0 & (0 \leq i \leq n - 1) \\ N_{n,j}(t_{\max}) = 1 \end{cases}$$

(iv)  $\{ N_{i,n+1} ; 0 \leq i \leq n \}$  est la base de Bernstein de degré  $n$  sur  $[0,1]$  ( $t_{\max} = 1$ ).

*Tracé des fonctions de base B-Spline* — Tout d'abord on définit une procédure qui calcule les composantes du vecteur nodal  $v$ , soit :

*PROCEDURE VecteurNodal(Entrée  $n, j$  : Entier ; Sortie  $v$  : VecteurEntier) ;*

*Début*

*Pour  $i$  allant de 0 à  $j - 1$  faire  $v_i = 0$  ;*

*Pour  $i$  allant de  $j$  à  $n$  faire  $v_i = i - j + 1$  ;*

*Pour  $i$  allant de  $n + 1$  à  $n + j$  faire  $v_i = n - j + 2$  ;*

*Fin ;*

Ensuite, on peut définir les fonctions de base de manière récursive de la manière suivante :

*FONCTION BaseBSpline(Entrée  $n, i, j$  : Entier ;  $v$  : VecteurEntier ;  $t$  : Réel) : Réel ;*

*Début*

*Si ( $t < v_j$ ) ou ( $t \geq v_{i+j}$ )*

*Alors Début*

*BaseBSpline = 0 ;*

*Sortir ;*

*Fin*

*Sinon Début*

*Si  $j = 1$*

*Alors Début*

*BaseBSpline = 1 ; { puisque  $t \in [v_i, v_{i+1}]$  }*

*Sortir ;*

*Fin*

*Sinon Début*

*Aux1 = BaseBSpline( $n, i, j - 1, v, t$ ) ;*

*Si ( $Aux1 = 0$ ) ou ( $v_{i+j-1} = v_j$ )*

*Alors  $S1 = 0$*

*Sinon  $S1 = \frac{t - v_i}{v_{i+j-1} - v_i} \cdot Aux1$  ;*

*Aux2 = BaseBSpline( $n, i+1, j - 1, v, t$ ) ;*

*Si ( $Aux2 = 0$ ) ou ( $v_{i+j} = v_{i+1}$ )*

*Alors  $S2 = 0$*

*Sinon  $S2 = \frac{v_{i+j} - t}{v_{i+j} - v_{i+1}} \cdot Aux2$  ;*

*BaseBSpline =  $S1 + S2$  ;*

*Fin ;*

*Fin ;*

*Fin ;*

En fait, comme on le verra dans le paragraphe suivant, on peut se passer de la récursivité qui peut donner un calcul très long pour de grandes valeurs de  $j$ .

Sur la figure 5.8, on donne les graphes des fonctions de base pour  $n = 5$  et  $j = 3$ .

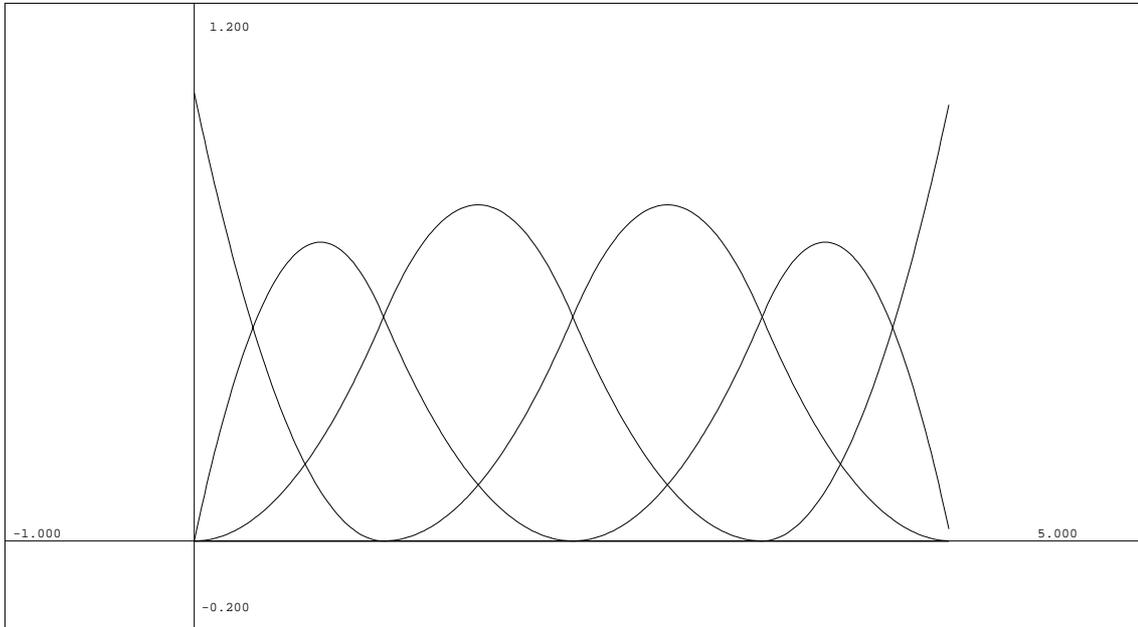


Figure 5.8

### 3.5.3 Courbes B-Splines

On reprend les notations du paragraphe (3.3.1) et on se donne  $n \geq 1$  et  $2 \leq j \leq n + 1$  dans  $\mathbb{N}$ .

*Définition* : On appelle *courbe B-Spline d'ordre  $j$*  (ou de degré  $j - 1$ ) associée au polygone descripteur  $S$ , la courbe d'équation paramétrique :

$$B(t) = \sum_{i=0}^n N_{i,j}(t) \cdot P_i \quad \text{si } t \in [0, t_{\max}] = [0, n - j + 2]$$

où  $\{N_{i,j}(t) ; 0 \leq i \leq n\}$  est la base B-Spline associée à  $(n, j)$ .

*Remarque* — On a  $[0, t_{\max}] = [v_{j-1}, v_{n+1}]$  et pour  $k = j - 1, \dots, n$ , si  $t$  est dans  $[v_k, v_{k+1}]$ , on aura  $t \notin ]v_i, v_{i+j}[$  pour  $i \leq k - j$  ou  $i \geq k + 1$ , ce qui donnera  $N_{i,j}(t) = 0$  pour de tels  $i$ . En définitive on a donc :

$$B(t) = \sum_{i=k-j+1}^k N_{i,j}(t) \cdot P_i \quad (k = j - 1, \dots, n; t \in [v_k, v_{k+1}])$$

On a donc seulement  $j$  termes à calculer dans la somme et non pas  $n + 1$  comme dans le cas des courbes de Bézier. De plus les polynômes que l'on manipule sont de degré au plus  $j$ . Tout cela étant indépendant du nombre de points  $n + 1$ .

*Exemple* — Pour  $n = 3, j = 2$  :  $B(t) = \sum_{i=0}^3 N_{i,3}(t) \cdot P_i \quad t \in [0,3]$ .

Sur  $[0,1] = [v_1, v_2]$  ( $k = j - 1 = 1$ ), on a :

$$B(t) = N_{0,2}(t) \cdot P_0 + N_{1,2}(t) \cdot P_1 = (1 - t) \cdot P_0 + t \cdot P_1 ;$$

sur  $[1,2] = [v_2, v_3]$  ( $k = j = 2$ ), on a :

$$B(t) = N_{1,2}(t) \cdot P_1 + N_{2,2}(t) \cdot P_2 = (2 - t) \cdot P_1 + (t - 1) \cdot P_2 ;$$

Sur  $[2,3] = [v_3, v_4]$  ( $k = n$ ), on a :

$$B(t) = N_{2,2}(t) \cdot P_2 + N_{3,2}(t) \cdot P_3 = (3 - t) \cdot P_2 + (t - 2) \cdot P_3 ;$$

En utilisant la procédure de calcul des fonctions de base B-Spline, on en déduit immédiatement un algorithme de calcul des courbes B-Splines.

L'indice  $k \in \{j - 1, \dots, n\}$  tel que  $t \in [v_k, v_{k+1}[$ , pour  $t$  dans  $[0, t_{\max}[$ , est simplement donné par :  $k = \text{PartieEntière}(t) + j - 1$

De plus, pour  $t = 0$ , on a :

$$B(0) = \sum_{i=0}^n N_{i,j}(0) \cdot P_i = 1 \cdot P_0 = P_0$$

et pour  $t = t_{\max}$ , on a :

$$B(t_{\max}) = \sum_{i=0}^n N_{i,j}(t_{\max}) \cdot P_i = 1 \cdot P_n = P_n$$

Si on ne veut pas utiliser la récursivité (qui donnera un calcul assez lent pour de grandes valeurs de  $j$  et même très lent pour  $j = n + 1$ ), on peut s'inspirer de l'algorithme de De Casteljeau. L'algorithme obtenu est connu sous le nom d'algorithme d'Oslo. (Cf. Math. et C. A. O., Vol. 6, p. 316 et 317).

La procédure de calcul est alors la suivante :

*PROCEDURE B-Spline*(Entrée  $n, j$  : Entier ;  $v$  : VecteurEntier ;  $P$  : TableauDePoints ;  $t$  : Réel ;  
Sortie  $B$  : Point) ;

*Début*

Si  $t \geq n - j + 2$

Alors Début

$B = P_n$  ;

Sortir ;

Fin ;

$k = \text{PartieEntière}(t) + j - 1$  ;

Pour  $i$  allant de  $k - j + 1$  à  $k$  faire  $Q_i = P_i$  ;

Pour  $p$  allant de  $j$  à 2 faire

Début

Pour  $i$  allant de  $k - p + 1$  à  $k$  faire

Début

Si  $v_{i+p-1} = v_i$

Alors Début

$S1 = 0$  ;  $S2 = 0$  ;

Fin

Sinon Début

$$S1 = \frac{t - v_i}{v_{i+p-1} - v_i} \cdot Q_i ;$$

$$S2 = \frac{v_{i+p-1} - t}{v_{i+p-1} - v_i} \cdot Q_{i-1} ;$$

```
    Fin ;  
    P = S1 + S2 ;  
  Fin ;  
  Pour i Allant de k - p + 1 à k faire Qi = Pi ;  
  Fin ;  
  B = Pk ;  
  Fin ;
```