HUNGARIAN JOURNAL
OF INDUSTRIAL CHEMISTRY
VESZPRÉM
Vol. 29. pp. 105 – 111 (2001)

B-SPLINES CONTRA BÉZIERSPLINES

N. HERRMANN

(Institute für Angewandte Mathematik, Universität Hannover, Hannover, GERMANY)

Received: April 9, 2001

P. E. Bézier, a mechanical and electrical engineer, developed his idea of interpolating given data during his work at Renault. Instead of using ordinary polynomials he introduced Bernstein polynomials in his algorithm. With the help of the Bernstein operator and a piecewise forgoing he invented a method to interpolate given data with a smooth spline curve.
I. J. Schoenberg is often entitled as the father of B-Splines, although Laplace probably worked with some kind of B-Splines in 1820. They are introduced as basis for the spline functions, and this is the reason for their name.
We present the basic properties of Bézier- and B-Splines including their shape preserving behaviour. In many examples we illustrate the main differences in their interpolating properties.

Keywords: interpolation, algorithms of de Boor and de Casteljau, comonotone behaviour

History

(a) B-Splines

1820 P.S. Laplace worked with some kind of B-Splines
1850 N.I. Lobatschewski gave a short explanation
1946 I.J. Schoenberg started with the investigation of "basic spline curves"
1967 I.J. Schoenberg introduced the name "B-splines"
1972 M.Cox, C.de Boor, L.Mansfield invented the "recurrence relation"

(b) Béziersplines

1959 P. de Casteljau worked the first time with these splines
1962 P. Bézier developed the same idea a second time
1970 R. Forrest found the connection to Bernstein polynomials

B-Splines: Alternative definitions

We start proposing four different kinds of definitions for B-Splines.

(a) Recursive definition

Given a knot sequence of real numbers in $\boldsymbol{R}$:

$$T = (x_0, x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_{n+l+1})$$

Then we define

$$l = 0: \quad N_{0,i}(x) = \begin{cases} 1 & \text{if} \quad x_i \le x \le x_{i+1} \\ 0 & \text{otherwise} \end{cases}, \quad i = 0,\ldots,n$$

$$l > 0: \quad N_{l,i}(x) := \frac{x - x_i}{x_{i+l} - x_{i+1}} N_{l-1,i}(x) + \frac{x_{i+l+1} - x}{x_{i+l+1} - x_{i+1}} N_{l-1,i+1}(x), \quad i = 0,\ldots,n$$

(b) Definition by divided differences

We use the well-known divided differences to define

$$N_{l,i}(x) := (x_{i+l+1} - x_i)\,[x_i, \ldots, x_{i+l+1}](\cdot - x)_+^l, \quad i = 0,\ldots,n$$

with

$$(t-x)_+^l := \begin{cases} (t-x)^l & \text{if} \quad t \ge x \\ 0 & \text{if} \quad t < x \end{cases}$$

Contact information: E-mail: herrmann@ifam.uni-hannover.de

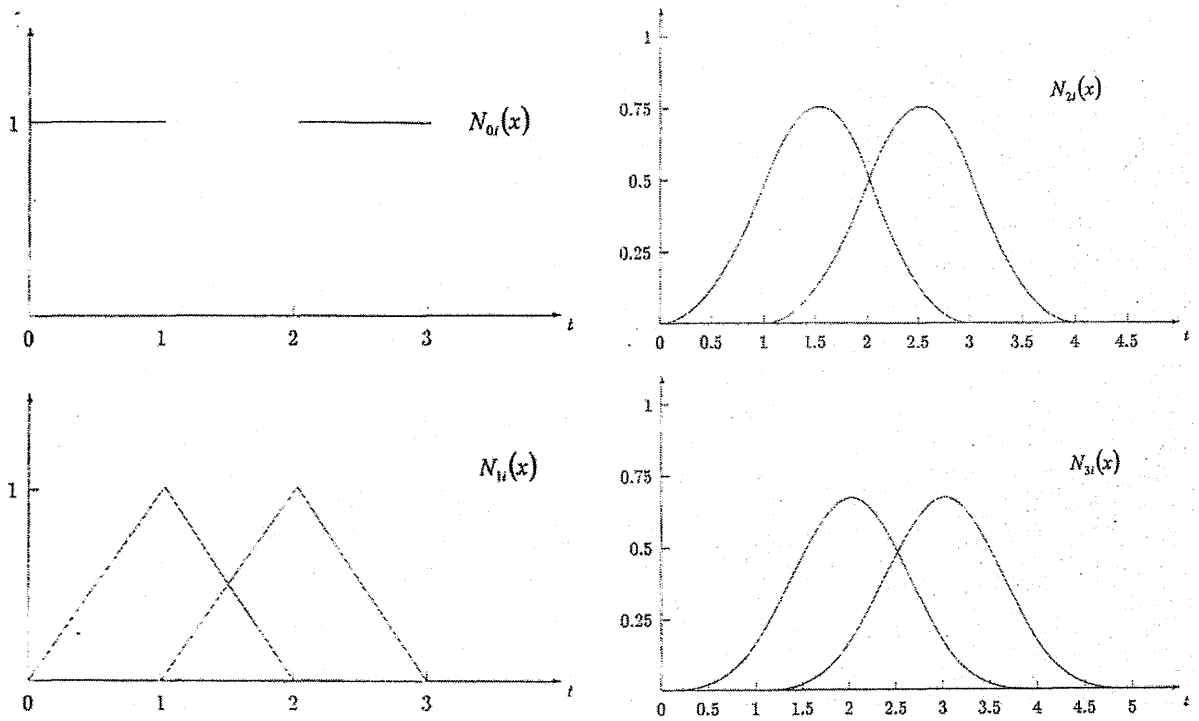

Fig.1 In each figure we see two B-Splines of order 0,1,2 and 3.

(c) Definition by the +-Functions

Without the help of the divided differences we can use the +-functions alone to define the B-Splines:

$$N_{l,i}(x) := (x_{i+l+1} - x_i) \cdot \sum_{k=i}^{i+l+1} \frac{(x_k - x)_+^l}{\prod_{r=i, r\neq k}^{i+l+1} (x_k - x_l)}, \quad i = 0,\ldots,n$$

(d) Definition by construction

Let $[a,b] \subseteq R$, $\Delta := \{x_0,\ldots,x_{n+1}\}$, $x_i \in R$,
$a = x_0 < x_1 < \cdots < x_n < x_{n+1} = b$.

Then we define:

$$S_l(\Delta) := \{s \in C^{l-1}[a,b] : s|_{[x_i,x_{i+1}]} \in IP^l, i = 0,\ldots,n\}$$

For this vector space we have a basis built with the +-functions:

$$S_l(\Delta) = span < 1, x, x^2, \ldots, x^l, (x-x_1)^l_+, \ldots, (x-x_n)^l_+ >$$

and we can prove the following theorem:

Theorem: Let $\Omega_\infty := \{x_\nu\}_{\nu\in Z}, x_\nu < x_{\nu+1}$. For each $i \in Z$ there exists a unique spline function $s \in S_l(\Delta)$ with

$$s(x) = 0 \quad in\ (-\infty, x_i) \cup (x_{i+l+1}, \infty) \quad and \quad \int_{-\infty}^{\infty} s(x)\,dx = 1.$$

It is not very surprising that we can prove it easily that the four definitions lead exactly to the same functions.

Theorem: All definitions above are equivalent.

Definition: We will call the functions developed in the four definitions B-Splines.

Basic properties

We summarize some of the important properties of B-splines in the following theorem:

Theorem:

1. *We have* $N_{l,i}(x) \equiv 0$ *if* $x \notin (x_i, x_{i+l+1})$,
2. *We have* $N_{l,i}(x) \in C^{l-1}$ *if* $x \in [x_i, x_{i+l+1}]$,
3. *We have* $N_{l,i}(x) > 0$ *if* $x \in (x_i, x_{i+l+1})$,
4. *We have the property of partition of unity:*

$$\sum_{i=-l}^{n-1} N_{l,i}(x) = 1 \quad \text{if} \quad x \in [x_i, x_{i+l+1}]$$

Remarks:

1. This means that the B-splines have a small support. Only in an interval with $l+2$ knots we have a chance to find values unlike zero.
2. Here we are told, that the B-splines have a defect of 1, therefore the B-splines of degree l are $l-1$-times continuously differentiable.
3. The B-splines are positive functions.
4. We can conclude on the bound 1 for all B-splines.

See *Fig.1*.

$T = (t_0, t_1, t_2, t_3, t_4) = (0, 1, 2, 3, 4)$

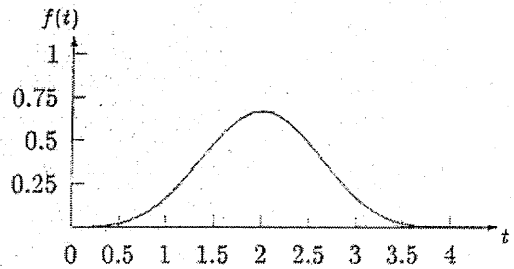

Fig.2a 4 intervals

$T = (0, 1, 1, 2, 3)$

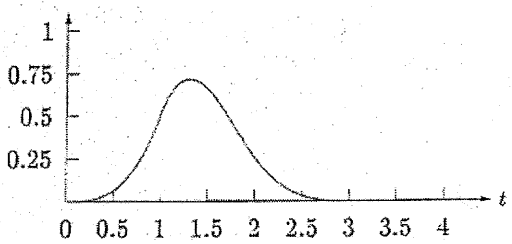

Fig.2b 3 intervals

Fig.2 Support-Reduction of a B-Spline of order 3 from 4 intervals to 2 intervals, if 1 is a knot of multiplicity 3.

$T = (0, 1, 1, 1, 2)$

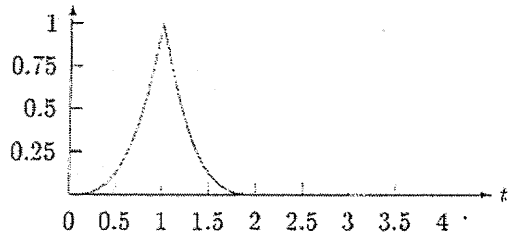

Fig.2c 2 intervals

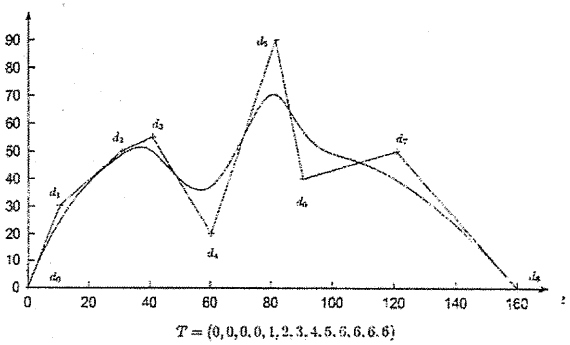

Fig.3 Example of a B-Spline curve with knot sequence T=(0,0,0,0,1,2,3,4,5,6,6,6,6).

Multiple knots

Given a knot sequence $x_0, x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_{n+l+1}$, we did not demand that all points should be different. What happens if some neighbor points fall together?

Theorem: *If* $x_i = x_{i+1} = \cdots = x_{i+l+1}$, *then we have*

$$N_{l,i}(x_i) = 0, i = 0,\ldots,n$$

From this we conclude immediately that $N_{l,i} \equiv 0$, which is not a very interesting result. So we ask for the case when less than $l+2$ knots coincide.

Theorem: *If* x_i *is a knot with multiplicity* $m \le l$, *it follows*

$$N_{l,i}(x_i) \in C^{l-m}, i = 0,\ldots,n$$

The smoothness of a B-Spline decreases therefore with multiple knots. The decrease of the smoothness comes together with a decrease of the support of the B-spline:

Theorem: *If* x_i *is a knot of multiplicity* $m \le l$, *then the support of* $N_{l,i}(x)$ *is reduced from* $l+1$ *intervals to* $l+1-(m-1)$ *intervals.*

We show this behaviour in *Fig.2*.

B-spline curves of degree *l*

Now we come to curves built with B-splines. At first the definition:

Definition: *Given* $l \in IN$ *and points* $\vec{d}_0, \vec{d}_1, \ldots, \vec{d}_n \in IR^2$ *or* IR^3 *and a knot sequence*

$$T = (x_0, x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+l+1}),$$

then we call the following linear combination

$$B_l(x) = \sum_{i=0}^{n} \vec{d}_i N_{l,i}(x), \qquad n \ge l, x \in [x_l, x_{n+1}]$$

a B-spline curve of degree l for the knot sequence T. $\vec{d}_0, \vec{d}_1, \ldots, \vec{d}_n$ *are the de Boor points, they build the de Boor polygon.*

One of the most important properties for such a B-spline curve is its local behaviour. If one of the points $\vec{d}_i$ is changed, the curve will be changed in a small vicinity of $\vec{d}_i$ only. This is what designers whish. So the front of a car can be changed whereas the back remains unchanged. This is the content of the following theorem:

Theorem: *Given a B-spline curve*

$$B_l(x) = \sum_{i=0}^{n} \vec{d}_i N_{l,i}(x), \qquad n \geq l,\ x \in [x_l, x_{n+1}]$$

with knot sequence T

$$T = (x_0, x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+l+1})$$

and $x^* \in (x_j, x_{j+1})$. *The point on the curve belonging to* x^* *is only influenced by the de Boor points* $\vec{d}_{j-l}, \ldots, \vec{d}_j$. *Hereby* $\vec{d}_j$ *acts in* (x_j, x_{j+l+1}) *only. Figure 3* gives an example of a B-Spline curve.

Bézier curves

The main tool in constructing Bézier curves are the Bernstein polynomials.

Definition: $B_i^n(x) = \binom{n}{i} x^i (1-x)^{n-i},\ i = 0,\ldots,n,\ x \in [0,1]$

We summarize some important properties of the Bernstein polynomials in the following theorem.

Theorem: $B_i^n(x) > 0$ *for* $x \in (0,1)$ a ba

$\{B_i^n\}_{i=0,\ldots,n}$ *is a basis for the vector space of all polynomials of degree* $\leq n$.

$\sum_{i=0}^{n} B_i^n(x) \equiv 1$ *for* $x \in (0,1)$

Definition: *A Bézier curve is defined as follows:*

$$X^n(x) := \sum_{i=0}^{n} \vec{b}_i B_i^n(x),\ x \in (0,1)$$

The points $\vec{b}_i$ *are called the controlpoints of the Bezier curve.*

There are some relations between B-Splines and Béziersplines. The next theorem shows that each spline curve can be expressed in terms of B-Splines.

Theorem: *Let s be a spline function of degree l in [a,b]. Then we have a unique representation*

$$s(x) = \sum_{i=-l}^{n} \vec{d}_i N_{l,i}(x),\ \vec{d}_i \in R^3,\ x \in [a,b]$$

In a very special case we have a direct coincidence between B-Splines and Béziersplines.

Theorem: *The B-Splines* $N_{l,i}(x)$ *of degree l are exactly the Bernstein polynomials* $B_i^l(x)$, *if the knot sequence T contains 2(l+1) elements where both 0 and 1 are of multiplicity l+1:*

$$T = (\underbrace{0,\ldots,0}_{(l+1)\text{times}},\ \underbrace{1,\ldots,1}_{(l+1)\text{times}})$$

The algorithm of de Boor and de Casteljau

The formula for a B-spline curve is not easy to handle. For polynomials we have the formula of Horner for a quick evaluation of single values. For B-spline curves this part is played by the algorithm of de Boor.

Algorithm (de Boor) for B-Splines

Given $n+1$ *de Boor points* $\vec{d}_0, \vec{d}_1, \ldots, \vec{d}_n, n \geq l$, *and a knot sequence*

$$T = (x_0 = x_1 = \cdots = x_l, x_{l+1}, \ldots, x_n, x_{n+1} = x_{n+2} = \cdots = x_{n+l+1})$$

Find $B(x^*)$:

1. *Find* r *with* $x_r \leq x^* < x_{r+1}$
2. *Calculate* $\alpha_i^l = \dfrac{x^* - x_i}{x_{i+l} - x_i},\quad i = r+1-l,\ldots,r$

$$\vec{d}_i^{\,l} = (1-\alpha^1{}_i)\,\vec{d}_{i-1} + \alpha_i^l \vec{d}_i$$

3. *For* $j = 2,3,\ldots,l,\ i = r-l+j,\ldots,r$

$$\alpha_i^j = \frac{x^* - x_i}{x_{i+l-(j-1)} - x_i}$$

$$\vec{d}_i^{\,j} = (1-\alpha_i^j)\,\vec{d}^{\,j-1}{}_{i-1} + \alpha_i^j \vec{d}_i^{\,j-1}$$

4. *Then we have* $B(x^*) = \vec{d}_r^{\,l}$

The following table gives an impression how this algorithm works. On the horizontal lines we multiply by α_i^j and on the diagonal lines by $1-\alpha_i^j$. The sum of both values gives the new de Boor point on the right:

$$\begin{array}{ccccccccc}
\vec{d}_{r-l} & & & & & & & & \\
 & \ddots & & & & & & & \\
\vec{d}_{r-l+1} & \cdots & \vec{d}^{\,1}{}_{r-l+1} & & & & & & \\
 & \ddots & & \ddots & & & & & \\
\vec{d}_{r-l+2} & \cdots & \vec{d}^{\,1}{}_{r-l+2} & \cdots & \vec{d}^{\,2}{}_{r-l+2} & & & & \\
 & \ddots & & \ddots & & \ddots & & & \\
\vec{d}_{r-l+3} & \cdots & \vec{d}^{\,1}{}_{r-l+3} & \cdots & \vec{d}^{\,2}{}_{r-l+3} & \cdots & & & \\
\vdots & \ddots & & & & \ddots & \vec{d}^{\,l-1}{}_{r-r} & & \\
 & & & & & & & \ddots & \\
\vec{d}_r & & \vec{d}^{\,1}{}_r & \cdots & \vec{d}^{\,2}{}_r & \cdots & \vec{d}^{\,l-1}{}_r & \cdots & \vec{d}^{\,l}{}_r = B(x^*)
\end{array}$$

The most important advantage of this algorithm is how fast it works. It is possible to evaluate thousands of points in less than a second. So this algorithm makes it possible to change a curve and in the same way a surface nearly in real time.

A similar algorithm for the Bézier splines was developed by de Casteljau. It is based on the recurrence relation:

$$\vec{b}_0^{\,n}(x) = (1-x)\,\vec{b}_0^{\,n-1}(x) + \vec{b}_1^{\,n-1}(x)$$

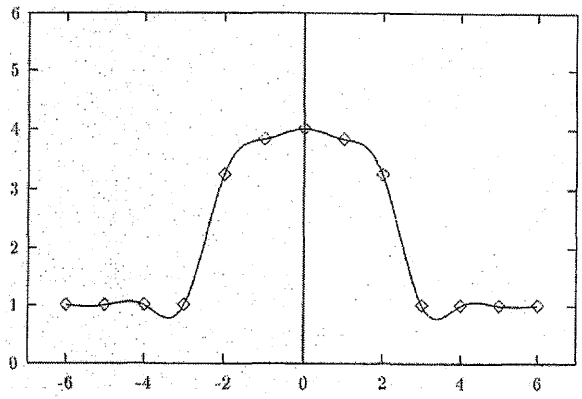

Fig.4 13 given points which build a tunnel are interpolated with B-Splines.

Algorithm (de Casteljau) for Béziersplines

Given $n+1$ *Bézier points or control points* $\vec{b}_0,\ldots,\vec{b}_n$.
Find $X^n(x^*)$:

1. *Start with* $\vec{b}_0^0 := \vec{b}_0, \vec{b}_1^0 := \vec{b}_1, \ldots, \vec{b}_n^0 := \vec{b}_n$
2. *Calculate* $\vec{b}_0^1, \vec{b}_1^1, \ldots, \vec{b}_{n-1}^1$ *using the recurrence relation above.*
3. *Continue calculating* $\vec{b}_0^i, \ldots, \vec{b}_{n-i}^i$, $i = 2,\ldots,n$
4. *Then we have* $X^n(x^*) = \vec{b}_0^n(x^*)$.

This calculation can be filled in a table quite similar to the table for the de Boor algorithm above. We have only to replace the de Boor points a by the Bézier points b and then to multiply in the horizontal line with x^*, in the diagonal line with $1-x^*$, and afterwards to sum up.

Interpolation using B-Splines

Now we will try to compare both methods in their behaviour regarding interpolation problems. Consider the points $P_0, P_1, \ldots, P_{q-1}$ in IR^2 or IR^3. We want to find an interpolating spline curve.

For the use of B-splines we could try to take each point in the knot sequence $(l+1)$-times. Then the curve runs through each of the points, but it is really not a smooth curve. In each point there is perhaps a sharp cusp. This is therefore no answer to the problem.

In order to combine the given points with the points in the knot sequence we should note, that in the interval $[x_0, x_l]$ less than $l+1$ B-splines are $\neq 0$. So we try the following combination:

$$x_l \to P_0, x_{l+1} \to P_1, \ldots, x_{l+q-1} \to P_{q-1}$$

This leads to the following linear system:

$$\sum_{i=0}^{n} \vec{d}_i N_{l,i}(x_{l+u}) = P_u, \quad u = 0,1,\ldots,q-1$$

This is a system of q linear equations for the $n+1$ unknown $\vec{d}_0, \vec{d}_1, \ldots, \vec{d}_n$. For $q = n-l+2$ there are $n+1-(n-l+2)$ equations missing. For cubic splines $(l=3)$ we have to find 2 equations more. A typical choice is to consider spline functions which are linear outside the interval $[x_0, x_n]$. They are called natural and the following two equations are based on the fact that their second derivative is zero:

$$\vec{d}_0 N''_{3,0}(x_3) + \vec{d}_1 N''_{3,1}(x_4) + \vec{d}_2 N''_{3,2}(x_5) = 0$$

$$\vec{d}_{n-2} N''_{3,n-2}(x_{n+1}) + \vec{d}_{n-1} N''_{3,n-}(x_{n+2}) + \vec{d}_n N''_{3,n}(x_{n+3}) = 0$$

Altogether we come to the following pentadiagonal system for natural cubic splines:

$$\frac{1}{6}\begin{pmatrix} \frac{6}{h^2} & -\frac{12}{h^2} & \frac{6}{h^2} & & & & & \\ 1 & 4 & 1 & 0 & & & & \\ 0 & \ddots & \ddots & \ddots & \ddots & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \ddots & 0 & \\ & & & 0 & 1 & 4 & 1 & \\ & & & & \frac{6}{h^2} & -\frac{12}{h^2} & \frac{6}{h^2} & \end{pmatrix} \begin{pmatrix} \vec{d}_0 \\ \vec{d}_1 \\ \vec{d}_2 \\ \vdots \\ \vdots \\ \vec{d}_{n-1} \\ \vec{d}_n \end{pmatrix} = \begin{pmatrix} 0 \\ P_0 \\ P_1 \\ \vdots \\ \vdots \\ P_{q-1} \\ 0 \end{pmatrix}$$

In *Fig.4* we have interpolated 13 points, which could be understood to build a tunnel, using B-Splines. But the oscilating behaviour of the curve is not acceptable.

Interpolation using Béziersplines – Comonotone Interpoation

After the bad result with our tunnel example using B-Splines we are interested to develop a better method for interpolating given data. One reason for the misbehaved shape can be seen in *Fig.4*. From the left hand side up to the midpoint the datas are monotone increasing, whereas the spline curve oscilates through the first four points.

At first let us define what we want to understand by a comonotone interpolating procedure:

Definition: *Consider the given data* $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ *with* $x_0 < x_1 < \cdots < x_n$. *Let* $s \in C^1[x_0, x_n]$ *be a function which interpolates these data. Then we will call s comonotone with the data, iff*

$$s'(x) \cdot m_i > 0 \quad \text{if} \quad m_i \neq 0, x \in (x_{i-1}, x_i)$$
$$s'(x) = 0 \quad \text{if} \quad m_i = 0$$

This is easy to explain. The slope of the spline function s should always be the same sign as the slope of the straight line between two neighbour points. Then s will be monotone increasing or decreasing in the same intervals where the data are monotone increasing or decreasing, respectively.

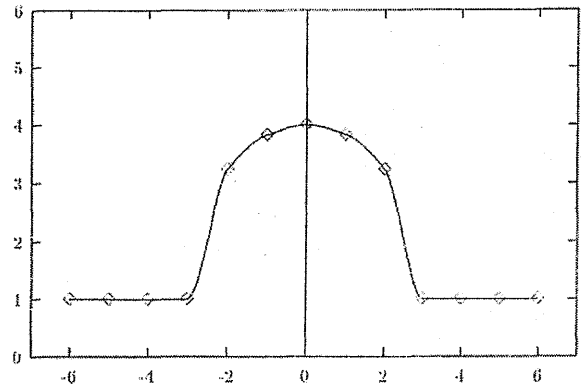

Fig.5 The same 13 given points as in *Fig.4* are now interpolated with the comonotone algorithm using Bézier-Splines.

The Bernstein operator and its shape preserving behaviour

The main point in developing our new algorithm is to use the Bernstein operator. It works with the help of the Bernstein polynomials.

Definition: *Let f be a vector valued continuous function on the interval [0,1]. Then the Bernstein operator is defined as*

$$B_n : \begin{cases} C[0,1] & \rightarrow & IR^n \\ B_n f(x) & = & \sum_{i=0}^{n} f\left(\frac{i}{n}\right)\binom{n}{i} x^i (1-x)^{n-i} \end{cases}$$

In the following theorem we summarize some of the most important properties regarding the shape preserving of the Bernstein operator:

Theorem: *For the Bernstein operator we have:*

1. *f bounded* $\Rightarrow$ $B_n f$ *is bounded (with the same bounds)*
2. $f \geq 0$ *in* $[0,1] \Rightarrow B_n f \geq 0$ *in* $[0,1]$
3. $B_n f(0) = f(0),\ B_n f(1) = f(1)$
4. *f monotone* $\Rightarrow$ $B_n f$ *monotone*
5. *f convex* $\Rightarrow$ $B_n f$ *convex*

Now we are able to explain the algorithm:

Comonotone algorithm:

Given

$$D := \{(x_i, y_i) \in IR^2, i = 0,\ldots,N, x_0 < x_1 < \cdots < x_N\}$$

1. *Construct the subknots* $t_0, t_1, \ldots, t_{2N+1}$

   with $r_i = \frac{x_i - x_{i-1}}{3}$ *by*

   $$t_0 = x_0, t_{2i-1} = x_{i-1} + r_i, t_{2i} = x_i - r_i, i = 1,\ldots,N, t_{2N+1} = x_N$$

2. *Calculate the difference quotients:*

$$m_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}}, \quad i = 1,\ldots,N$$

3. *Determine a piecewise linear comonotone function l by*

   $l(x_i) := y_i, l'(x_i) := d_i$ *with*
   $d_0 := 2m_1 - d_1, d_N := 2m_N - d_{N-1}$ *and for*
   $i = 1,\ldots,N-1$

$$d_i := \begin{cases} 0 & \text{if} \quad m_i \cdot m_{i+1} = 0 \\ \frac{m_i}{m_{i+1}} \frac{3m_{i+1} - m_i}{2} & \text{if} \quad 0 < |m_i| \leq |m_{i+1}| \\ \frac{m_{i+1}}{m_i} \frac{3m_i - m_{i+1}}{2} & \text{if} \quad |m_i| > |m_{i+1}| \end{cases}$$

4. *Calculate* $l(t_1), l(t_{2i}), l(t_{2i+1}), l(t_{2N})$
5. *Apply in each subinterval the Bernstein operator. This leads to the desired function:*

$$s(x) = B_3(l(x)) \in C^1[x_0, x_N]$$

Remark: The formula in 3. is the heart of the algorithm. The derivative d_i at the knots x_i is evaluated in a way that the resulting linear spline function is comonotone with the data. Because of the shape preserving behaviour of the Bernstein operator we get therefore the desired comonotone cubic spline function.

In *Fig.5* we show the tunnel data from above, but now comonotone interpolated. The result looks much better.

Conclusion

In this paper we compared B-Splines and Béziersplines. After a short introduction for both spline cancidates we explained some of the most important properties. The similar algorithms of de Boor and de Casteljau lead to very fast evaluation processes. Several figures show the different behaviour regarding the interpolation procedure. In contrast to the disadvantage of the monotony preserving of the B-Splines we propose a comonotone interpolation algorithm using Bézier splines.

SYMBOLS

R	real numbers
$x_0,\ldots,x_{n+l+1}$	knots in R
$T = (x_0,\ldots,x_{n+l+1})$	knot sequence
N_{li}	B-Splines
$(t-x)_+^l$	+-function
$\Delta = (x_0,\ldots,x_{n+l})$	given knots
$C^{l-1}[a,b]$	*(l-1)*-times continuously differentiable functions
IP^l	vector space of polynomials of degree less or equal l

$S_l(\Delta)$	vector space of spline functions
Ω_∞	infinite knot sequence
$B_l(x)$	B-Spline curve
$\vec{d}_0,...,\vec{d}_n$	de Boor points
$B_i^n(x)$	Bernstein polynomials
$X^n(x)$	Bézier curve
$s(x)$	spline function
α_i^j	coefficients in the de Boor algorithm
$P_0,...,P_q$	interpolation points
$l(x)$	linear spline function
m_i	difference quotients

REFERENCES

1. COSTATINI, P., MORANDI, R.: Monotone and convex cubic spline interpolation, Calcolo, *1984*, 21, 281-294
2. COSTATINI, P., MORANDI, R.: An algrithm for computing shape-preserving cubic spline interpolation to data, Calcolo, *1984*, 21, 295-305
3. DEPPE, U.: Anwendung der BEM auf eine Integralgleichung 1. Art zur Bestimmung der Porenradienverteilung eines SLP-Katalysator-Pellets Diploma Thesis, Universität Hannover, 1994
4. FARIN, G.: Curves and Surfaces for Computer Aided Geometric Design, Academic Press, Boston et al., 1990
5. GRONEWOLD, G.: Co-Monotone Spline-Interpolation in Chemical Engineering, Hung. J. Ind. Chem., *1997*, 25, 59-62
6. HERRMANN, N.: Höhere Mathematik für Ingenieure I, II, Oldenbourg Verlag, München, 1995
7. HERRMANN, N.: Spline-Funktionen und ihre Anwendung, Preprint Institut für Angewandte Mathematik, Uni. Hannover, 1996
8. HERRMANN, N.: Surface Fitting in CAGD via Finite Elements, Hung. J. Ind. Chem., *1997*, 25, 63-69
9. HOSCHEK, J., LASSER, D.: Grundlagen der geometrischen Datenverarbeitung, B. G. Teubner, Stuttgart, 2. ed. 1992