Computing 29, 365—371 (1982)

Computing

A Simplified B-Spline Computation Routine

E. T. Y. Lee, Seattle

Received March 10, 1982

Abstract — Zusammenfassung

A Simplified B-Spline Computation Routine. A simple algorithm is presented for computing the value together with all derivatives up to a specified order of a B-spline series.

Ein vereinfachtes Verfahren zur B-Spline Berechnung. Ein einfacher Algorithmus zur gleichzeitigen Berechnung der Werte und der Ableitungen höherer Ordnung einer B-Spline-Reihe wird untersucht.

1. Introduction

The purpose of this note is to present a new algorithm for computing the value and all derivatives (up to a specified order) of a B-spline series. When no derivative is desired, the algorithm is identical with the well-known de Boor-Cox evaluation algorithm (see [5], [6]). On the other hand, when derivatives are to be computed, the algorithm is more efficient than using, for example, the existing routines in de Boor's package [3]. The derivation of this algorithm is based on a paper of Boehm [1] on the insertion of new knots. It is felt that the simplicity and compactness of the algorithm should prove useful for those workers in the field who have frequent dealings with B-spline computations.* See more discussions in Remark (ii).

Let us first set down some notations. Let $\boldsymbol{t}=(t_i)$ be a non-decreasing sequence of reals, the knot sequence. The B-splines of order k (degree $m=k-1$) subordinate to $\boldsymbol{t}$ are

$$B_{i,k}(t)=B(t\,|\,t_i,\ldots t_{i+k})=(t_{i+k}-t_i)\,[t_i,\ldots t_{i+k}](\cdot-t)_+^{k-1}.$$

See de Boor [2] for basic properties of these functions. A B-spline series (of order k, subordinate to $\boldsymbol{t}$) is a locally finite sum

$$F(t)=\sum_i A_i B_{i,k}(t). \tag{1}$$

* Subroutines written based on this algorithm have been in use for well over a year at the Boeing Company by the group to which the author belongs. The author wishes to thank B. Tiger for incentives in this work.

0010-485X/82/0029/0365/$ 01.40

A nontrivial knot interval $[t_r, t_{r+1}]$, with at least m t_i's in $\boldsymbol{t}$ preceding t_r and m such after t_{r+1}, will be hereby called a span. On this span, the sum in (1) extends from $i=r-m$ to $i=r$. Values at the end points of the span will be understood to mean limiting values from the interior of the span. To avoid carrying an extra index (here r) in the descriptions, we will, without loss of generality, concentrate attention on the particular span $[t_m, t_{m+1}]$, on which

$$F(t)=\sum_{i=0}^{m} A_i B_{i,k}(t), \tag{2}$$

the relevant knots being $\{t_0 \dots t_{2m+1}\}$. See Remark (i) in the general situation.

2. Knot Insertion

We now recall Boehm's result on the insertion of knots (see also Remark (iii)) and consider one of its implications. Let x be in the span $[t_m, t_{m+1}]$. We wish to insert a knot at x. The new knot sequence $\boldsymbol{t}'$ consists of

$$t'_i=t_i \text{ for } i\leq m, \quad t'_{m+1}=x, \quad t'_i=t_{i-1} \text{ for } i>m+1.$$

Let the B-splines subordinate to $\boldsymbol{t}'$ be $B'_{i,k}(t)=B(t \mid t'_i, \dots t'_{i+k})$. Then according to Boehm,

$$F(t)=\sum A_i B_{i,k}(t)=\sum A'_i B'_{i,k}(t)$$

with $A'_i=A_i$ for $i\leq 0$, $A'_i=A_{i-1}$ for $i\geq m+1$, and

$$A'_i=\frac{(x-t_i)A_i+(t_{i+m}-x)A_{i-1}}{t_{i+m}-t_i}, \quad 1\leq i\leq m \tag{3}$$

Thus the addition of a knot at x gives the new coefficient sequence $\dots A_0, A'_1, \dots A'_m, A_m \dots$. As noted by Boehm, $A'_1, \dots A'_m$ agree exactly with the second column in de Boor's triangle (p. 147 of [2], with $i=m$ here). This furnishes another point of view of the de Boor-Cox algorithm: adding m knots at x generates the triangle

$$\begin{array}{ccccccc}
A_{0,0} & & & & & & \\
 & A_{1,1} & & & & & \\
A_{1,0} & & A_{2,2} & & & & \\
 & A_{2,1} & & \ddots & & & \\
A_{2,0} & & \ddots & & & & \\
\vdots & & & \ddots & & A_{m,m} & \\
 & \ddots & & & \ddots & & \\
 & & \ddots & & & & \\
A_{m-1,0} & & A_{m,2} & & & & \\
 & A_{m,1} & & & & & \\
A_{m,0} & & & & & &
\end{array} \tag{4}$$

(where we now use $A_{i,0}=A_i$, $A_{i,1}=A'_i$), in which

$$A_{i,j}=\frac{(x-t_i)A_{i,j-1}+(t_{i-j+k}-x)A_{i-1,j-1}}{t_{i-j+k}-t_i}, \quad i=j, \dots m. \tag{5}$$

After j insertions at x, the new B-spline coefficients are

$$\dots A_{0,0}, \dots A_{j-1,j-1}, A_{j,j}, A_{j+1,j}, \dots A_{m,j}, A_{m,j-1}, \dots A_{m,0}, \dots$$

In particular, after m insertions

$$\begin{aligned} F(t) = &\dots + A_{0,0} B(t \mid t_0, \dots t_m, x) + \dots + A_{m-1,m-1} B(t \mid t_{m-1}, t_m, x, \dots x) \\ &+ A_{m,m} B(t \mid t_m, \underbrace{x, \dots x}_{m}, t_{m+1}) \\ &+ A_{m,m-1} B(t \mid x, \dots x, t_{m+1}, t_{m+2}) + \dots + A_{m,0} B(t \mid x, t_{m+1}, \dots t_{2m+1}) + \dots \end{aligned}$$

Suppose $t_m < x < t_{m+1}$ for a moment. Since clearly

$$B(t \mid t_m, \underbrace{x, \dots x}_{m}, t_{m+1}) = B(t \mid t_m, \underbrace{x, \dots x}_{m+1}) + B(t \mid \underbrace{x, \dots x}_{m+1}, t_{m+1}),$$

we conclude that the spline $F(t)$ has been split into two B-spline series, abutting at a full $(=k)$ multiple knot x. The left series has as coefficients the upper diagonal of table (4)

$$\dots A_{0,0}, A_{1,1}, \dots A_{m,m} \tag{6}$$

and the right series has as coefficients the lower diagonal of table (4)

$$A_{m,m}, A_{m,m-1}, \dots A_{m,0} \dots . \tag{7}$$

This fact has been indicated in [7], p. 107 (comment by J. Lane), though without detail.

It is now clear why $F(x) = A_{m,m}$. For if

$$V(t) = \sum_{i \geq 0} V_i B(t \mid s_i, \dots s_{i+k}),$$

with knots $s_0 = \dots = s_m = x < s_{m+1}$, then $V(x) = V_0$ since only $B(x \mid x, \dots x, s_{m+1}) = 1$ is non-zero among all $B(x \mid s_i, \dots s_{i+k})$, $i \geq 0$. Our point is that the derivatives $D^j F(x)$ can be read off from (6) or (7) in the same simple manner, as will be seen in the next section.

What we said about splitting remains true if x is one of the end points of the span $[t_m, t_{m+1}]$. If, for example, $x = t_{m+1}$, then the left part of spline F will still have the coefficients (6), (which is all we care about below), while the accounting for the right part depends on the multiplicity of t_{m+1} in $\boldsymbol{t}$. (Thus, if t_{m+1} had multiplicity p, then after m insertions, the coefficients of the right series are $A_{m,m-p}$, $A_{m,m-p-1}, \dots A_{m,-p}, \dots$, where $A_{m,-j}$ is understood to mean A_{m+j} in the original expansion (1), and the right part has knot sequence $x, \dots x$ (k times), $t_{m+p+1}, \dots$. We will not go into any details since this does not concern us here.)

3. Derivatives

Let $t_m \leq x < t_{m+1}$ and consider computing $D^j F(x)$ from the right part of the split B-spline series. The relevant B-coefficients are $A_{m,m}, \dots A_{m,0}$, the lower diagonal of (4). Change the ordering by writing

$$A_{i,j} = Q_{i-j,j} \qquad j \leq i \leq m, \; j = 1, \dots m$$

(so that the B-coefficients are now $Q_{0,m}, \ldots Q_{m,0}$). Also write

$$a_i = x - t_i, \quad b_i = t_{m+i} - x, \qquad i = 1, \ldots m. \tag{8}$$

Then, from (5) and after an obvious re-indexing, we have

$$Q_{i,j} = \frac{a_{i+j} Q_{i+1,j-1} + b_{i+1} Q_{i,j-1}}{a_{i+j} + b_{i+1}}, \qquad \begin{matrix} i = 0, \ldots m-j \\ j = 1, \ldots m; \end{matrix} \tag{9}$$

and by overwriting locations we can use a one-dimensional array. Thus:

$$\begin{aligned} &\text{For } j = 1, \ldots m \\ &\text{For } i = 0, \ldots m-j \\ &\qquad Q_i \leftarrow \frac{a_{i+j} Q_{i+1} + b_{i+1} Q_i}{a_{i+j} + b_{i+1}}; \end{aligned} \tag{10}$$

and the right part of the spline is now

$$F(t) = \sum_{i \geq 0} Q_i B(t \mid s_i, \ldots s_{i+k}), \tag{11}$$

where $s_0 = \ldots = s_m = x$ and $s_i = t_i, i \geq m+1$. Differentiating (11) gives (see [2, p. 139])

$$D^j F(t) = \sum_{i \geq j} R_{i,j} B(t \mid s_i, \ldots s_{i+k-j}), \tag{12}$$

where $R_{i,0} = Q_i$, $0 \leq i \leq m$, and

$$R_{i,j} = \frac{R_{i,j-1} - R_{i-1,j-1}}{(s_{i+k-j} - s_i)/(k-j)}, \qquad \begin{matrix} i = j, \ldots m \\ j = 1, \ldots m \end{matrix} \tag{13}$$

See table (14). (Note that $s_{i+k-j} \leq s_{k-1} = x$ for $i = 0, \ldots j-1$, so that

$$B(t \mid s_i, \ldots s_{i+k-j}) = B(t \mid x, \ldots x) = 0;$$

hence the sum in (12) starts with $i = j$.)

$$\begin{matrix} R_{0,0} & & & \\ R_{1,0} & R_{1,1} & & \\ \vdots & \vdots & \ddots & \\ \vdots & \vdots & & R_{m,m} \\ R_{m,0} & R_{m,1} & & \end{matrix} \tag{14}$$

By the well known continuity, support, and partition of unity properties of B-splines, it is clear at once that for the B-splines of order $k-j$ involved in (12),

$$\begin{aligned} &B(x \mid s_j, \ldots s_k) = B(x \mid x, \ldots x, s_k) = 1, \\ &B(x \mid s_{j+1}, \ldots s_{k+1}) = 0, \quad B(x \mid s_{j+2}, \ldots s_{k+2}) = 0, \ldots \end{aligned}$$

so that (12) yields

$$D^j F(x) = R_{j,j}, \qquad j = 0, \ldots m.$$

Again we can use a one-dimensional array by overwriting; however, we have to start each column of (14) from the bottom up. Note since $R_{i,0}=Q_i$, we can use the same array (Q_i) obtained from (10). Thus

$$\begin{aligned}&\text{Do } * \ j=1,\dots m\\&\text{Do } * \ i=m,\dots j\\&*\quad Q_i \leftarrow \frac{Q_i-Q_{i-1}}{b_{i-j+1}/(m-j+1)}\\&\text{Then } D^j F(x)=Q_j,\quad j=1,\dots m.\end{aligned}\tag{15}$$

$(s_{i+k-j}-s_i=s_{i+k-j}-x=b_{i-j+1})$. Derivatives of order higher than m are of course zero. Note if one needs only derivatives up to order $d\leq m$, only the upper triangle of size d in (14) is needed.

Having obtained the algorithm for computing from the right, one can obtain the corresponding algorithm for computing from the left simply by a reflection. Let now $t_m<x\leq t_{m+1}$. Clearly

$$B(t\mid t_i,\dots t_{i+k})=B(-t\mid -t_{i+k},\dots -t_i).$$

On writing $s=-t$ and

$$\begin{aligned}t'_i&=-t_{2m+1-i} \qquad i=0,\dots 2m+1\\C_i&=A_{m-i} \qquad i=0,\dots m,\end{aligned}$$

we have

$$F(t)=G(s)=\sum_{i\geq 0} C_i B(s\mid t'_i,\dots t'_{i+k}).$$

Hence computing F from the left at $t=x$ is the same as computing G from the right at $s=-x$, except that with each differentiation, a minus sign is introduced, since $D_t F=-D_s G$. Thus all one needs to do is to change, in the previous algorithm, A_i to A_{m-i}, a_{i+j} to $a'_{i+j}\equiv(-x)-t'_{i+j}=t_{2m+1-i-j}-x=b_{m-j+1-i}$, and b_{i+1} to $b'_{i+1}\equiv t'_{m+i+1}-(-x)=a_{m-i}$, b_{i-j+1} to $b'_{i-j+1}=a_{m-i+j}$; and to each statement * in (15), we attach a minus sign.

The full algorithm is described below after one remark: If x is, say, the right end point of the span, one can only compute from the left. Hence for an interior point x, we naturally choose to compute from the left if x is closer to the right end point. This avoids possible numerical inaccuracies in the derivatives which might otherwise occur when x is close to one of the end points. For instance, consider a spline with a single span (i.e., t_m and t_{m+1} are full multiple knots and the spline is a Bernstein polynomial). Suppose we compute from the right. The right series is supported on $[x, t_{m+1}]$; thus for x sufficiently close to t_{m+1}, its B-coefficients (Q_i), obtained from (10), must necessarily be very close (by continuity and the fact that B-coefficients model the function they represent, see [2] p. 159). Thus in the subsequent derivative computations (15), one may encounter loss of significance in computing the differences Q_i-Q_{i-1}, in addition to such possible loss in forming the divisors b_{i-j+1}.

4. The Algorithm

Given knots $t_0, \ldots t_{2m+1}$, with $t_m < t_{m+1}$, and B-coefficients $A_0, \ldots A_m$ for the B-spline series (2), and given also $x \in [t_m, t_{m+1}]$, $d \leq m$, the following routine computes

$$Q_j = D^j F(x), \quad j = 0, \ldots d.$$

If $m=0$ then $Q_0 \leftarrow A_0$ and exit.
$a_i \leftarrow x - t_i$, $b_i \leftarrow t_{m+i} - x$, $i = 1, \ldots m$
Fromright $\leftarrow (b_1 > a_m)$

If fromright then
$Q_i \leftarrow A_i$, $i = 0, \ldots m$
For $j = 1, \ldots m$ do
For $i = 0, \ldots m - j$ do

$$Q_i \leftarrow \frac{a_{i+j} Q_{i+1} + b_{i+1} Q_i}{a_{i+j} + b_{i+1}}$$

For $j = 1, \ldots d$ do
For $i = d, \ldots j$ do

$$Q_i \leftarrow \frac{Q_i - Q_{i-1}}{b_{i-j+1}/(m-j+1)}$$

Else
$Q_i \leftarrow A_{m-i}$, $i = 0, \ldots m$
For $j = 1, \ldots m$ do
For $i = 0, \ldots m - j$ do

$$Q_i \leftarrow \frac{b_{m-j+1-i} Q_{i+1} + a_{m-i} Q_i}{a_{m-i} + b_{m-j+1-i}}$$

For $j = 1, \ldots d$ do
For $i = d, \ldots j$ do

$$Q_i \leftarrow -\frac{Q_i - Q_{i-1}}{a_{m-i+j}/(m-j+1)}$$

Note: It is easily seen that no division by zero can occur above. One may also observe that the extreme knots t_0 and t_{2m+1} are not needed; this is because varying t_i (keeping $t_i \leq t_{i+1}$) does not change the value of $B(t \mid t_i, \ldots t_{i+k})$ on $[t_{i+k-1}, t_{i+k}]$.

5. Remarks

(i) Given a spline $F(t) = \sum_{i=1}^{n} C_i B_{i,k}(t)$, with knots $(s_i)_{i=1}^{n+k}$ (in the notation of [2]), and $x \in [s_k, s_{n+1}]$, to compute $D^j F(x)$, one first locates the span $[s_r, s_{r+1}]$ containing x, then applies the above algorithm to

$$(t_i)_{i=0}^{2m+1} \leftarrow (s_i)_{i=r-m}^{r+m+1} \quad \text{and} \quad (A_i)_{i=0}^{m} \leftarrow (C_i)_{i=r-m}^{r}.$$

If x is a left (or right) endpoint, apply the algorithm to the previous (or next) span (if existing) to obtain limiting values from left (or right). Thus it is easy to calculate the left and right values $D^i F(x^-)$ and $D^i F(x^+)$ at a knot.

(ii) In applications, there are frequent occasions calling for derivative computations. In one area, computer-aided geometric modelling and design, for instance, curves and surfaces are often described by (vector valued parametric) *B*-spline series and tensor products of such. Geometric characteristics such as tangent, normal, curvature, etc., then involve derivative evaluations.

The usual method for computing a derivative of a *B*-spline is to difference the coefficients and then apply the basic de Boor-Cox algorithm to the resulting spline of one lower order. Note that in the present algorithm, the work involved in the derivative part (the loop "for $j=1$ to d") is generally less than that in the difference operations for coefficients of various derivatives (see [2, p. 139]). Thus the savings amounts to at least all subsequent applications of de Boor-Cox to splines of lower orders. Even if one ignores the work involved in the differencings, on the assumption that they have been preprocessed (for instance, when one computes at a large number of points in a fixed span), those subsequent applications of de Boor-Cox still cost somewhat more than the work in the derivative part here.

We have tested (VAX 11/780, double precision) a FORTRAN version of the present algorithm against BSPLEV of [3]. Results agree completely. However, even for $m=d=3$ (cubic splines with all derivatives computed), there is already a savings of about one third in CPU time over the use of BSPLEV.

(iii) Generally, results for knot insertions can be obtained through a formula, already given in de Boor ([4], p. 15), expressing a divided difference as a linear combination of divided differences (of the same order) over a finer partition; the details of which have been carried out and expanded into the "Oslo algorithm" by Cohen, Lyche and Riesenfeld [7]. Boehm considered a single insertion at a time, and his proof is much simpler and more direct.

References

[1] Boehm, W.: Inserting new knots into *B*-spline curves. Computer Aided Design *12*, 199–201 (1980).
[2] de Boor, C.: A practical guide to splines. Berlin-Heidelberg-New York: Springer 1978.
[3] de Boor, C.: Package for calculating with *B*-splines. SIAM J. Numer. Anal *14*, 441–472 (1977).
[4] de Boor, C.: Splines as linear combinations of *B*-splines: A survey. In: Approximation Theory II (Lorentz, G. G., Chui, C. K., Schumaker, L., eds.), pp. 1–47. Academic Press 1976.
[5] de Boor, C.: On calculating with *B*-splines. J. Approx. Theory *6*, 50–62 (1972).
[6] Cox, M. G.: The numerical evaluation of *B*-splines. J. Inst. Math. Appl. *10*, 134–149 (1972).
[7] Cohen, E., Lyche, T., Riesenfeld, R.: Discrete *B*-splines and subdivision techniques in computer-aided geometric design and computer graphics. Computer Graphics and Image Processing *14*, 87–111 (1980).

Dr. E. T. Y. Lee
Boeing Commercial Airplane Company
P. O. Box 3707 M/S 6E-30
Seattle, WA 98124
U.S.A.