

REFERENCES

1. A. Rosenfeld, Isotonic grammars, parallel grammars and picture grammars, in *Machine Intelligence* (D. Michie and B. Meltzer, Eds.), Vol. 6, pp. 281-294, Univ. of Edinburgh Press, Edinburgh, Scotland, 1971.
2. C. R. Cook and P. S. Wang, A Chomsky hierarchy of isotonic array grammars and languages, *Computer Graphics and Image Processing* 8, 1978, 144-152.
3. D. L. Milgram and A. Rosenfeld, Array automata and array grammars, Proc. IFIP Congress, 71, booklet TA-2, pp. 166-173, North-Holland, Amsterdam, 1971.
4. A. Rosenfeld, *Picture Languages*, Academic Press, New York, 1979.

*See: Computer Graphics and Image Processing, Vol. 14, No. 2, 1980
pp. 87-111*

Discrete B-Splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics

ELAINE COHEN*

Central Institute for Industrial Research, Oslo, Norway

TOM LYCHE

University of Oslo, Oslo, Norway

AND

RICHARD RIESENFELD*

Central Institute for Industrial Research, Oslo, Norway

Received December 17, 1979; accepted January 14, 1980

The relevant theory of discrete B-splines with associated new algorithms is extended to provide a framework for understanding and implementing general subdivision schemes for nonuniform B-splines. The new derived polygon corresponding to an arbitrary refinement of the knot vector for an existing B-spline curve, including multiplicities, is shown to be formed by successive evaluations of the discrete B-spline defined by the original vertices, the original knot vector, and the new refined knot vector. Existing subdivision algorithms can be seen as proper special cases. General subdivision has widespread applications in computer-aided geometric design, computer graphics, and numerical analysis. The new algorithms resulting from the new theory lead to a unification of the display model, the analysis model, and other needed models into a single geometric model from which other necessary models are easily derived. New sample algorithms for interference calculation, contouring, surface rendering, and other important calculations are presented.

1. INTRODUCTION

During the last 15 years there has been increasing attention given to the problem of geometric modeling, that is, defining and representing freeform curves and surfaces in an interactive computing system. Two of the most widely used methods for interactively designing shapes are the Bézier method [2] and the B-spline method [34]. Even when B-splines or Bézier curves are not used in the initial specification of a curve or surface, these bases can be very valuable for later properties of splines, because these special bases have some extraordinary representations of geometrical significance not shared by other, more traditional representations. Since Bézier curves are a special case of B-splines, it is meant to be understood that remarks made about B-splines apply to Bézier curves and surfaces, a fortiori [1, 10].

Although splines have gained increasing acceptance in applications to computer-aided geometric design (CAD) and computer graphics, there have been some obstacles to their convenient use, namely, computing intersections of spline surfaces with spline surfaces, and accurately rendering either line drawings or smooth

*On leave from University of Utah.

shaded pictures of spline surface models. The latter problem, in fact, has inhibited the more extensive use of splines in computer graphics applications because most surface-rendering algorithms require a polygonal database, a piecewise linear C^0 representation, of the model. This requirement entailed either tedious hand generation of such a database, or reliance on very fine automatically generated piecewise approximations resulting in impractically large representations necessary to cover all possible situations. Some heuristic approaches [32, 33] and hierarchical approaches [5, 8] have been suggested to cope with this problem, but heuristic methods, like the method proposed by Blinn and the method proposed by Whitted in a joint paper [3], do not always work. Some relatively efficient hierarchical algorithms like the Lane and Carpenter algorithm [19] and the Lane and Riesenfeld algorithms [23] are not defined for splines with nonuniform knot spacing.

In this paper we will develop the necessary theory of discrete B -splines, including a new extension to nonuniform refinements including multiple knots, so that it can be used as a general framework for understanding the underlying structure of recursive subdivision algorithms like Lane and Riesenfeld's [23], and for generating new algorithms for nonuniform B -splines. The recursively generated new vertices will be shown to lie on a discrete B -spline defined by the previous vertices. The general utility of this new framework will be demonstrated by its application to many standard problems which occur in CAGD, computer graphics, and elsewhere [31].

2. BACKGROUND

A. B -Splines

We recall from [34] the definition of a B -spline curve.

DEFINITION. The space curve $\gamma(s) = \sum_{i=0}^M P_i N_{i,k}(s)$ is a B -spline curve for the polygon

$$P = P_0 P_1 \dots P_n, \quad (2.1)$$

where

P_i are points in \mathbf{R}^3 forming an open (nonperiodic) or closed (periodic) polygon P ,

$\tau = (\tau_0, \tau_1, \dots, \tau_q)$ is a knot vector over which $N_{i,k}(s)$ are defined, and

$N_{i,k}(s)$ are the normalized local support B -spline basis functions of order k (degree $k-1$).

There are three common methods of dealing with the end conditions. The open case, resembling the behavior of Bézier curves most closely, is defined by a knot vector having the smallest and largest knots occurring with multiplicity k . If no interior knots appear in the knot vector, the open B -spline curve specializes properly to a Bézier curve. The closed (or periodic) case is defined by considering the polygon array as a ring structure in which vertex P_0 is defined as the successor of P_n . The knot vector τ must also be treated in a periodic way so that τ is defined as a repeating sequence modulo τ_q . This periodicity in τ can also be implemented by making a ring structure of the relative knot spacings $\Delta\tau_i = \tau_{i+1} - \tau_i$.

A third treatment of the end conditions leads to the so-called *floating case* in which the k vertices necessary to define each span are used, just as though the span

were part of a periodic curve. Since the summation in (2.1) is local, it is well defined as long as there are k vertices to use for the generation of each span. Whether the global topology of a ring data structure is used is not relevant for the local calculations of each span. The procedure is tantamount to having no special treatment of the end vertices at all; simply let them "float" without interpolation requirements at all. Sometimes the extra vertices at each are made coincident in the floating case to produce an interpolation effect which mimics the open case, although it is not identical to it. All variants of the floating end conditions share the advantage that the basis functions can be made standard and tabled for uniformly spaced knots, but there can be some undesirable geometrical properties associated with the procedure of assigning multiplicity to the end vertices [38].

The deBoor-Cox algorithm (2.2) can be used for evaluating a point $P^{(k)}(s)$ on the curve in (2.1). First the parameter value s is used to determine j such that $\tau_j \leq s < \tau_{j+1}$; then (2.2) is recursively applied:

$$P^{(i)}(s) = P_j \quad \text{for } i = 0 \\ = \lambda_j P^{(i-1)}(s) + (1 - \lambda_j) P^{(i-1)}(s) \quad \text{else,} \quad (2.2)$$

where $\lambda_j = (s - \tau_j) / (\tau_{j+1} - \tau_j)$.

In the evaluations of periodic B -splines, the knot vector must be shifted by $k/2$ and the knots and vertices used in a periodic manner. In practice, the deBoor-Cox algorithm is usually used only once in each polynomial span, after which a Taylor expansion or incremental method is used.

B. Subdivision Methods

The procedure of describing a given curve $\gamma(s) = \sum_{i=0}^M P_i \theta_i(s)$, with corresponding control points P_i , in terms of a larger set of basis functions and net control points P'_i so that $\gamma(s) = \sum_{i=0}^M P'_i \phi_i(s)$ for $N > M$, is often referred to, together with the surface analogs, as a subdivision technique. These techniques in CAGD, which are also called "curve-splitting" or "surface-splitting" methods, have been suggested as a method of top-down design and have actually appeared quite early in the literature, for example, in Forrest [16], MacCallum [26], and more recently Knapp [17]. All of these approaches have been motivated by the common notion that the gross overall shape of a curve ought to be defined first, after which a designer might desire additional flexibility in terms of control points for portions of the design still requiring some finer detail. The extra control points should be made available to the designer without perturbing the previously defined shape. In this vein subdivision has emerged out of efforts to provide more satisfactory hierarchical schemes for interactive shape specification.

Subdivision has also arisen as a technique in computer graphics for obtaining a satisfactory piecewise linear approximation to smooth polynomial and spline surfaces. Methods such as those of Catmull [5] and Lane and Riesenfeld [23] have relied on the convergence properties of the new control points for Bézier and B -spline curves, respectively. After enough levels of recursive subdivision, the piecewise linear polygon determined by the newly defined control points can be used in place of the original curve $\gamma(s)$. Important calculations such as surface rendering with hidden surface removal, shading functions, and object interference can then be carried out effectively with the piecewise linear representations.

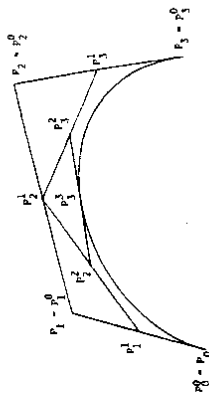


FIG. 1. Subdivision of cubic Bézier curve.

As examples for specific "halving" subdivision algorithms, we refer the reader to Figs. 1 and 2, which describe subdivision procedures for Bézier curves and for uniform B -splines. The original cubic Bézier curve defined by P_0, P_1, P_2, P_3 has been subdivided at the parameter value $s = \frac{1}{2}$ so that the same curve is now described as two articulated Bézier curves defined by P_0, P_1, P_2, P_3 and by P_1, P_2, P_3, P_4 . At each level the Bézier subdivision algorithm consists of connecting midpoints of the previous polygon. The "halving" algorithm for uniform cubic B -splines curves illustrated in Fig. 2 is performed similarly after the original polygon P_0, P_1, P_2, P_3 is augmented with extra midpoint vertices P_1^1, P_2^1, P_3^1 .

Both of the above constructions generate a convergent, shape-preserving, sequence of piecewise linear approximations in which each new vertex is given recursively as a convex combination of two previous vertices. This leads to simple and fast algorithms for generating subdivisions. Since the convex hull property of B -spline curves is not altered by subdivision, this characteristic of the B -spline basis has been exploited to solve hierarchically for points of intersection between two B -spline curves, or between two B -spline surfaces, or between a curve and a surface. The general rule is simple: Recursively subdivide and test only when relevant convex hulls overlap. No intersection is possible if the convex hulls do not intersect. Since the parametric value of the curve is always known at the point of subdivision, the parametric value of an intersection found by subdivision can easily

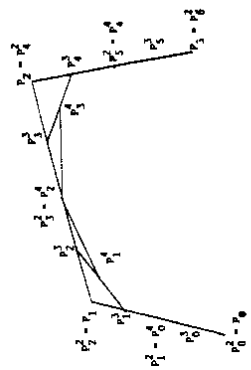


FIG. 2. Subdivision of cubic B -spline curve segment.

be known to within the tolerance of the testing criterion. That is, the interval containing the intersection is always known. A fuller discussion of these methods with more generality and rigor appears in [3, 19, 20, 21, 23].

The convergent properties of subdivision methods can be used as a method of curve generation itself, for the constructions lead to a close piecewise linear approximation to a smooth curve or surface, which is exactly what is necessary for driving a display or other numerical control machines. Several years ago Chaikin [7] proposed such a scheme, which he viewed as successively "cutting" off the corners of a polygon. Riesenfeld later proved that his "midpoint" construction actually yielded quadratic B -splines with a certain intrinsic parametrization [35]. Recently Deo and Sabin [15] and Caimull and Clark [6] studied very intriguing local subdivision schemes for surfaces derived from meshes involving nonrectilinear topologies. All of the uses of subdivision for curve and surface generation substitute a recursively applied procedure for the more traditional use of a closed-form mathematical function.

3. DISCRETE B -SPINES

In this section we extend and present the necessary theory of a discrete B -spline so that it can be applied to the general area of subdivision in the remaining sections.

Discrete splines have been studied extensively. They were introduced by Mangasarian and Schumaker [28] as solutions to certain minimization problems involving differences instead of derivatives. They are connected to best summation formulas [29], and have been used by Malcolm [27] to compute nonlinear splines iteratively. Approximation properties of discrete splines have been studied by Lyche [24, 25]. Discrete B -splines on a uniform partition were introduced by Schumaker [37]. Discrete B -splines on a nonuniform partition were defined by deBoor [13, p. 15].

Consider a piecewise polynomial written in terms of B -splines B_k of order k :

$$f(x) = \sum_{i=1}^n P_i B_k(x). \tag{3.1}$$

The knots $\tau = \{\tau_1, \dots, \tau_{m+1}\}$ can be nonuniform and multiple. There are certain situations where it is useful to increase the degrees of freedom of f by adding additional knots $\tau^0 = \{\tau_1^0, \dots, \tau_m^0\}$ to the already existing ones. Suppose we let $m = n + l$ and $t = \{t_1, \dots, t_{m+l}\} = \tau \cup \tau^0$ be the new knot sequence in nondecreasing order. With N_k the B -splines on t , f can also be written as a linear combination of $N_{1,k}, \dots, N_{m+l,k}$ with certain (unknown) coefficients d_j :

$$f(x) = \sum_{j=1}^{m+l} d_j N_{j,k}(x). \tag{3.2}$$

We consider now the following

PROBLEM. Given k, n, m, τ, t , and $\{P_1, \dots, P_n\}$ as above, compute d_1, \dots, d_{m+l} . There are several ways to compute the d_j 's. For instance, we can choose m points

ρ_1, \dots, ρ_m and solve the interpolation problem,

$$\sum_{j=1}^m d_j N_{jk}(\rho_i) = f(\rho_i), \quad i = 1, 2, \dots, m. \quad (3.3)$$

If $t_j < \rho_j < t_{j+k}$, $j = 1, 2, \dots, m$, then this set of linear equations has a unique solution d_1, \dots, d_m . The coefficient matrix is totally positive and banded, and it can be inverted by Gaussian elimination without pivoting in $\mathcal{O}(mk^2)$ operations (see deBoor and Pinkus [12]).

Alternatively one can use the quasi-interpolant of deBoor and Fix [11] to compute the d_j 's. Thus if

$$\lambda_i f = \frac{1}{(k-1)!} \sum_{r=0}^{k-1} (-1)^{k-1-r} \psi_j^{(r)}(a_j) f^{(k-1-r)}(a_j), \quad (3.4)$$

where a_j is any point in (t_j, t_{j+k}) and

$$\Psi_j(y) = \prod_{r=1}^{k-1} (y - t_{j+r}), \quad (3.5)$$

then

$$\lambda_j N_{jk} = \delta_{ij} = 1, \quad i = j, \\ = 0, \quad i \neq j.$$

Therefore applying λ_j on both sides of (3.2) we have

$$d_j = \lambda_j f, \quad j = 1, 2, \dots, m. \quad (3.6)$$

This method of computing d_j could be advantageous if f is given in its piecewise polynomial representation.

We will, not, however discuss these methods any further. Let us instead consider a third method. Here d_j will be computed recursively. The method is similar to the subdivision scheme considered by Lane and Riesenfeld [23] for the special cases of Bézier curves (k -tuple knots) and for uniform knots. In fact, our method, which is valid for any knot configuration, reduces to theirs in these special situations. To start we note by linearity that

$$d_j = \sum_{i=1}^n \alpha_{ik}(j) P_i \quad (3.7)$$

for some numbers $\alpha_{ik}(j)$.

To gain some insight we start by studying the cases $k = 1$ (step-functions) and $k = 2$ (piecewise linear functions).

Suppose first that $k = 1$. Then

$$f(x) = \sum_{i=1}^n P_i B_{i1}(x), \quad (3.8)$$

where

$$B_{i1}(x) = 1, \quad \tau_i \leq x < \tau_{i+1}, \\ = 0, \quad \text{otherwise.} \quad (3.9)$$

If

$$f(x) = \sum_{j=1}^m d_j N_{j1}(x), \quad (3.10)$$

where

$$N_{j1}(x) = 1, \quad t_j \leq x < t_{j+1}, \\ = 0, \quad \text{otherwise.} \quad (3.11)$$

then

$$d_j = P_i, \quad \tau_i \leq t_j < \tau_{i+1}. \quad (3.12)$$

Hence in (3.7) we have

$$\alpha_{i1}(j) = 1, \quad \tau_i \leq t_j < \tau_{i+1}, \\ = 0, \quad \text{otherwise.} \quad (3.13)$$

We note that $\alpha_{i1}(j) = B_{i1}(t_j)$. Consider next the case $k = 2$. Then

$$f(x) = \sum_{i=1}^n P_i B_{i2}(x), \quad (3.14)$$

where

$$B_{i2}(x) = (x - \tau_i) / (\tau_{i+1} - \tau_i), \quad \tau_i \leq x < \tau_{i+1}, \\ = (\tau_{i+2} - x) / (\tau_{i+2} - \tau_{i+1}), \quad \tau_{i+1} \leq x < \tau_{i+2}, \\ = 0, \quad \text{otherwise.} \quad (3.15)$$

Suppose

$$f(x) = \sum_{j=1}^m d_j N_{j2}(x), \quad (3.16)$$

where

$$N_{j2}(x) = (x - t_j) / (t_{j+1} - t_j), \quad t_j \leq x < t_{j+1}, \\ = (t_{j+2} - x) / (t_{j+2} - t_{j+1}), \quad t_{j+1} \leq x < t_{j+2}, \\ = 0, \quad \text{otherwise.} \quad (3.17)$$

Now if ν and j are such that

$$\tau_\nu \leq t_{j+1} < \tau_{\nu+1}$$

then

$$f(t_{j+1}) = d_j = [(t_{\nu+1} - t_{j+1})P_{\nu-1} + (t_{j+1} - t_\nu)P_\nu] / (t_{\nu+1} - t_\nu). \quad (3.18)$$

Thus (3.7) is valid with

$$\begin{aligned} \alpha_2(j) &= (t_{j+1} - \tau_1) / (t_{\nu+1} - \tau_1), & \tau_1 \leq t_{j+1} < \tau_{\nu+1}, \\ &= (t_{j+2} - t_{j+1}) / (t_{\nu+2} - \tau_{\nu+1}), & \tau_{\nu+1} \leq t_{j+1} < \tau_{\nu+2}, \\ &= 0, & \text{otherwise.} \end{aligned} \quad (3.19)$$

We also note that $\alpha_2(j) = B_2(t_{j+1})$. We observe the interesting fact that the numbers $\alpha_k(j)$ are related to the B -spline B_k for $k = 1, 2$. As we shall see the relation between α_k and B_k for $k > 2$ is not as simple as that for $k = 1$ and 2. In fact, $\alpha_k(j)$ is a discrete B -spline. The relation between B -splines N_k on a partition $\{t_j\}$ and B -splines B_k on a (coarser) subpartition $\{\tau_i\}$ is given in the following theorem.

THEOREM 1. For all x we have

$$B_k(x) = \sum_{j=1}^m \alpha_k(j) N_k(x), \quad i = 1, 2, \dots, m, \quad (3.20)$$

where

$$\begin{aligned} \alpha_i(j) &= (t_{j+k} - \tau_i) [\tau_1, \dots, \tau_{i+k}] \phi_k, & (3.21) \\ \phi_k(y) &= (y - a_i)_+^0 \Psi_k(y), & (3.22a) \end{aligned}$$

and where $\Psi_k(y)$ is given by (3.5). Here

$$(y - a_i)_+^0 = \begin{cases} 1, & y > a_i, \\ 0, & \text{otherwise,} \end{cases}$$

a_i can be chosen anywhere in $[t_i, t_{i+1})$, and $[\tau_1, \dots, \tau_{i+k}] \phi_k$ denotes a divided difference.

Remarks.

- $\alpha_k(j)$ is called a discrete B -spline.
- The number $\alpha_k(j)$ in (3.7) are the discrete B -splines given by (3.21).

Proof. From (3.1) and (3.20) we have

$$\begin{aligned} f(x) &= \sum_{i=1}^n P_i B_k(x) = \sum_{i=1}^n \sum_{j=1}^m P_i \alpha_k(j) N_k(x) \\ &= \sum_{j=1}^m \left[\sum_{i=1}^n P_i \alpha_k(j) \right] N_k(x). \end{aligned}$$

where $\alpha_k(j)$ is given by (3.21). Comparing this with (3.7) the statement follows.

3. For $k = 1$ we have from (3.21)

$$\alpha_1(j) = (t_{j+1} - a_1)_+^0 - (t_j - a_1)_+^0, \quad (3.22b)$$

which agrees with $\alpha_1(j)$ given by (3.13) for any $a_j \in [t_j, t_{j+1})$. Similarly for $k = 2$ we find

$$\alpha_2(j) = [\tau_{j+1}, \tau_{j+2}] \phi_2 - [\tau_j, \tau_{j+1}] \phi_2$$

with

$$\phi_2(y) = (y - a_j)_+^0 (y - t_{j+1}).$$

This can be seen to agree with $\alpha_2(j)$ given by (3.19) for any $a_j \in [t_j, t_{j+2})$.

4. Identity (3.20) can also be found in deBoor [13, p. 15]. However, deBoor used

$$\phi_k(y) = \prod_{r=1}^{k-1} (y - t_{j+r}).$$

to define $\alpha_k(j)$ in (3.21). With this ϕ_k it was not clear to us how to interpret (3.21) for multiple τ 's.

To prove Theorem 1 we need two lemmas. The first lemma is due to Marsden [30].

LEMMA 1. For any $y \in \mathbb{R}$ and any $x \in [t_k, t_{m+1})$ we have

$$(y - x)^{k-1} = \sum_{j=1}^m \Psi_k(y) N_k(x), \quad (3.23)$$

where $\Psi_k(y)$ is given by (3.5).

Proof (deBoor [14]). For $k = 1$, (3.23) takes the form $1 = \sum_{j=1}^m N_j(x)$, which follows from (3.1). For $k \geq 2$ we use the recurrence relation (deBoor [14], Cox [9])

$$N_k(x) = (x - t_j) Q_{j,k-1}(x) + (t_{j+k} - x) Q_{j+1,k-1}(x), \quad (3.24)$$

where

$$\begin{aligned} Q_k(x) &= N_k(x) / (t_{j+k} - t_j), & t_{j+k} > t_j, \\ &= 0, & \text{otherwise.} \end{aligned} \quad (3.25)$$

Denoting the right-hand side of (3.23) by ξ_k we find

$$\xi_k = \sum_{j=1}^m \Psi_k(y) [(x - t_j) Q_{j,k-1}(x) + (t_{j+k} - x) Q_{j+1,k-1}(x)].$$

Since $x \in [t_k, t_{m+1})$ we have $Q_{i,k-1}(x) = Q_{m+1,k-1}(x) = 0$. Hence rearranging the

terms in the sum we have

$$\xi_k = \sum_{j=2}^m \gamma_{j\mu}(x, y) \mathcal{Q}_{j, k-1}(x), \quad (3.26)$$

where

$$\gamma_{j\mu}(x, y) = \Psi_{j\mu}(y)(x - t_j) + \Psi_{j-1, k}(y)(t_{j+k-1} - x).$$

But straightforward calculation gives

$$\gamma_{j\mu}(x, y) = (y - x)(t_{j+k-1} - t_j) \Psi_{j, k-1}(y).$$

Hence, (3.26) can be written

$$\xi_k = (y - x) \sum_{j=2}^m \Psi_{j, k-1}(y)(t_{j+k-1} - t_j) \mathcal{Q}_{j, k-1}(x).$$

Since $(t_{j+k-1} - t_j) \mathcal{Q}_{j, k-1}(x) = N_{j, k-1}(x)$ and $N_{j, k-1}(x) = 0$ we see that $\xi_k = (y - x) \sum_{j=2}^m \xi_k$. (Recall that ξ_k is the right-hand side of (3.23).) But then $\xi_k = (y - x)^{k-1} \xi_1$. Since $\xi_1 = 1$, (3.23) follows.

LEMMA 2. Let ϕ_k and a_j be as in Theorem 1. For any $y \in I$ and any $x \in [t_k, t_{m+1})$ we have

$$(y - x)^{k-1} = \sum_{j=1}^m \phi_{j\mu}(y) N_{j\mu}(x). \quad (3.27a)$$

Proof. Fix x and let μ be such that $t_\mu \leq x < t_{\mu+1}$. Since $N_{j\mu}(x) = 0$ for $x \notin [t_j, t_{j+k})$ we have to show that

$$(y - x)^{k-1} = a_k = \sum_{j=\mu-k+1}^{\mu} \phi_{j\mu}(y) N_{j\mu}(x). \quad (3.27b)$$

Suppose first $y = t_\mu$. Since $\phi_{j\mu}(t_\mu)$ contains a factor $t_\mu - t_\mu$ for $j = \mu - k + 1, \dots, \mu - 1$, we have $a_k = \phi_{\mu\mu}(t_\mu) N_{\mu\mu}(x)$. But $\phi_{\mu\mu}(t_\mu) = 0$ since $a_\mu \in [t_\mu, t_{\mu+k})$. Hence $a_k = 0 = (t_\mu - x)^{k-1}$ and (3.27b) follows in this case. Similarly if $y = t_{\mu-1}$ then $a_k = \phi_{\mu-1, \mu}(t_{\mu-1}) N_{\mu-1, \mu}(x) + \phi_{\mu-1, \mu-1}(t_{\mu-1}) N_{\mu-1, \mu-1}(x) = 0 = (t_{\mu-1} - x)^{k-1}$. Continuing in this way we see that (3.27b) holds for $y = t_s$ and $s \leq \mu$. Suppose next $y = t_{\mu+1}$. Then $a_k = \phi_{\mu+1, \mu+1}(t_{\mu+1}) N_{\mu+1, \mu+1}(x)$. But $\phi_{\mu+1, \mu+1}(t_{\mu+1}) = \Psi_{\mu-k+1, k}(t_{\mu+1})$ and (3.27b) follows from (3.23). Similarly (3.27b) follows (3.23) for $y = t_s$ and $s \geq \mu + 1$.

Proof of Theorem 1. Suppose first $a_j \notin \{t_{j+1}, \dots, t_{j+k-1}\}$. Then we can apply the divided difference (3.27a) $[\tau_0, \dots, \tau_{i+k}]$ on both sides of (3.27a). Multiplying also by $\tau_{i+k} - \tau_i$, (3.20) follows. Since the right-hand side of (3.27a) is constant as a function of a_j in $[t_j, t_{j+k})$, $a_{j\mu}(f)$ will also be independent of a_j . We can then let $a_j \in \{t_{j+1}, \dots, t_{j+k-1}\}$ and take limits either from left or right.

We next prove a recurrence relation for the $\alpha_{ik}(j)$. A similar recurrence relation for uniform t 's has been given by Schumaker [37] and Lyche [24].

THEOREM 2. Suppose $\tau_{i+k} > \tau_i$ and that $\alpha_{ik}(j)$ is given by (3.21). Then

$$\begin{aligned} \alpha_{i1}(f) &= 1, & \tau_i \leq t_j < \tau_{i+j}, \\ &= 0, & \text{otherwise.} \end{aligned} \quad (3.28)$$

Moreover for $k \geq 2$ and all i, j ,

$$\alpha_{ik}(j) = (t_{j+k-1} - \tau_i) \beta_{i, k-1}(j) + (\tau_{i+k} - t_{j+k-1}) \beta_{i+1, k-1}(j), \quad (3.29)$$

where

$$\begin{aligned} \beta_{ik}(j) &= \alpha_{ik}(j) / (\tau_{i+k} - \tau_i), & \tau_{i+k} > \tau_i, \\ &= 0, & \text{otherwise.} \end{aligned} \quad (3.30)$$

Proof. Equation (3.28) follows from (3.22b). To prove (3.29) we proceed as in [14], applying the Leibnitz' rule to the product $\phi_{ik}(y) = (y - t_{j+k-1}) \phi_{j, k-1}(y) = g(y) \cdot h(y)$. Then

$$[\tau_i, \dots, \tau_{i+k}] \phi_{j\mu} = \sum_{r=i}^{i+k} [\tau_i, \dots, \tau_r] g[\tau_i, \dots, \tau_{i+k}] h$$

Since $[\tau_i] g = g(\tau_i) = (\tau_i - t_{j+k-1}) [\tau_i, \tau_{i+1}] g = 1$ and $[\tau_i, \dots, \tau_i] g = 0$ for $r > i + 1$, we obtain

$$[\tau_i, \dots, \tau_{i+k}] \phi_{j\mu} = g(\tau_i) [\tau_i, \dots, \tau_{i+k}] h + [\tau_{i+1}, \dots, \tau_{i+k}] h.$$

By the definition of divided differences

$$(\tau_{i+k} - \tau_i) [\tau_i, \dots, \tau_{i+k}] h = [\tau_{i+1}, \dots, \tau_{i+k}] h - [\tau_i, \dots, \tau_{i+k-1}] h.$$

Hence

$$\begin{aligned} \alpha_{ik}(j) &= (\tau_{i+k} - \tau_i) [\tau_i, \dots, \tau_{i+k}] \phi_{j\mu} \\ &= (\tau_i - t_{j+k-1}) ([\tau_{i+1}, \dots, \tau_{i+k}] h - [\tau_i, \dots, \tau_{i+k-1}] h) \\ &\quad + (\tau_{i+k} - \tau_i) [\tau_{i+1}, \dots, \tau_{i+k}] h \\ &= (\tau_{j+k-1} - \tau_i) [\tau_i, \dots, \tau_{i+k-1}] h + (\tau_{i+k} - \tau_i) [\tau_{i+1}, \dots, \tau_{i+k}] h. \end{aligned}$$

But this is precisely identity (3.29). Note the similarity between (3.29) and the recurrence relation for B_{ik} .

$$B_{ik}(x) = (x - \tau_i) C_{i, k-1}(x) + (\tau_{i+k} - x) C_{i+1, k-1}(x). \quad (3.31)$$

where

$$C_{\mu}(x) = B_{\mu}(x) / (\tau_{\mu+k} - \tau_{\mu}), \quad \tau_{\mu+k} > \tau_{\mu} \tag{3.32}$$

= 0, otherwise.

Thus the discrete splines $\alpha_{\mu k}(j)$ have properties very similar to those for B_{μ} . We collect some properties of $\alpha_{\mu k}(j)$ in a corollary to Theorem 2.

COROLLARY. Let $\alpha_{\mu k}(j)$ be as in Theorem 2. Then

- (A) For $1 \leq j \leq m$ let μ be such that $\tau_{\mu} \leq t_j < \tau_{\mu+1}$. Then $\alpha_{\mu k}(j) = 0 \quad i \notin \{\mu - k + 1, \dots, \mu\}$;
- (B) $\alpha_{\mu k}(j) \geq 0$, all i, j ;
- (C) $\sum_{i=\mu-k}^{\mu} \alpha_{\mu k}(j) = 1, \tau_k \leq t_j < \tau_{k+1}$.

Part (A) says that for each j there are at most k discrete B -splines $\alpha_{\mu-k+i,k}(j), \dots, \alpha_{\mu,k}(j)$ with a (possible) nonzero value.

Proof. By (3.28), (A) follows immediately for $k = 1$. Suppose (A) is true for $k - 1$; i.e., only $\alpha_{\mu-k+2,k-1}(j), \dots, \alpha_{\mu,k-1}(j)$ can be nonzero. By the recurrence relation (3.29) it is clear that only $\alpha_{\mu-k+1,k}(j), \dots, \alpha_{\mu,k}(j)$ can be nonzero. Hence (A) follows by induction.

To prove (B) we need a slightly stronger version of (A).

(A'). If $t_j < \tau_i$ or $t_{j+k-1} \geq \tau_{i+k}$ then $\alpha_{\mu k}(j) = 0$. The fact that $\alpha_{\mu k}(j) = 0$ for $t_j < \tau_i$ is clear from (A). Suppose $t_{j+k-1} \geq \tau_{i+k}$. If $k = 1$ then $\alpha_{\mu k}(j) = 0$. We proceed by induction. Since $t_{j+k-1} \geq \tau_{i+k}$ we have $t_{j+k-2} \geq \tau_{i+k-1}$. Hence $\beta_{i,k-1}(j) = 0$. Also, if $t_{j+k-1} > \tau_{i+k}$ then $t_{j+k-2} \geq \tau_{i+k}$ and $\beta_{i+1,k-1}(j) = 0$. Hence by (3.29) we have $\alpha_{\mu k}(j) = 0$ for $t_{j+k-1} \geq \tau_{i+k}$. This completes the proof of (A). To prove (B) we note that by (A'), we only have to prove that $\alpha_{\mu k}(j) \geq 0$ for $t_j \geq \tau_i$ and $t_{j+k-1} < \tau_{i+k}$. But then all factors in (3.29) are nonnegative.

Finally, to prove (C) we note that

$$1 = \sum_{j=1}^m B_{\mu k}(x) = \sum_{j=1}^m N_{\mu k}(x).$$

Thus by taking d_j and all P_i 's equal to 1 in (3.7), (C) follows.

We return now to the problem of computing d_j in (3.7) once the P_i 's are known. Writing (3.7) in the form

$$d(j) = \sum_{i=1}^n P_i \alpha_{\mu k}(j)$$

we see that $d(j)$ is a discrete spline, i.e., a linear combination of discrete B -splines. The fact that discrete splines have local support is proved in Corollary A and used in Algorithms 1 and 2. The similarity between the recurrence relations (3.29) for $\alpha_{\mu k}(j)$ and (3.31) for B_{μ} therefore makes the computation of $d(j)$ very similar to

the computation of $f(x)$ for some x where

$$f(x) = \sum_{i=1}^n P_i B_{\mu k}(x).$$

This leads to the following algorithms:

ALGORITHM 1. For integers $k \geq 2$ and j, μ let $\tau_{\mu+2-k}, \dots, \tau_{\mu+k-1}$ and $t_{j+1}, \dots, t_{j+k-1}$ be given such that

$$\tau_{\mu+2-k} \leq \dots \leq \tau_{\mu+1} \leq \dots \leq \tau_{\mu+k-1}, \tag{3.33}$$

$$\tau_{\mu} \leq t_j < \tau_{\mu+1}. \tag{3.34}$$

This algorithm computes $\alpha_{\mu k}(j)$ given by (3.21) or (3.29) $r = 1, \dots, k; i = \mu + 1 - r, \dots, \mu$. These are all the discrete B -splines of order $\leq k$ that can be nonzero for the given j .

- (1) $\alpha(\mu, 1) = 1, \mu 2 = \mu$
- (2) For $r = 1, 2, \dots, k - 1$
 - (i) $\text{beta}_1 = 0, y = r(j + r)$
 - (ii) For $i = \mu 2, \mu 2 + 1, \dots, \mu$
 - (a) $d1 = y - \tau(i), d2 = \tau(i + r) - y$
 - (b) $\text{beta} = \alpha(i, r) / (d1 + d2)$
 - (c) $\alpha(i - 1, r + 1) = d2 * \text{beta} + \text{beta}_1$
 - (d) $\text{beta}_1 = 1 = d1 * \text{beta}$
 - (iii) $\alpha(\mu, r + 1) = \text{beta}_1$
 - (iv) $\mu 2 = \mu 2 - 1$.

Algorithm 1 can be used to compute

$$d(j) = \sum_{i=1}^n \alpha_{\mu k}(j) P_i,$$

for if $\tau_{\mu} \leq t_j < \tau_{\mu+1}$

$$d(j) = \sum_{i=\mu-k+1}^{\mu} \alpha_{\mu k}(j) P_i \tag{3.35}$$

and the values $\alpha_{\mu k}(j)$ are available from Algorithm 1.

There is an alternative way to compute $d(j)$ given by (3.35) which parallels Algorithm 1 in [23]. To derive it we have by (3.29)

$$d(j) = \sum_{i=\mu-k+1}^{\mu} P_i \alpha_{\mu k}(j) = \sum_{i=\mu-k+1}^{\mu} P_i [(t_{j+k-1} - \tau_i) \beta_{i,k-1}(j) + (\tau_{i+k} - t_{j+k-1}) \beta_{i+1,k-1}(j)].$$

Since $\beta_{\mu-k+1,k-1}(j) = \beta_{\mu+1,k-1}(j) = 0$ by (A) in the corollary, we have

$$d(j) = \sum_{i=\mu-k+2}^{\mu} P_i \beta_{i,k-1}(j).$$

where

$$P_j^{(2)} = [(j_{j+k-1} - \tau_j)P_j + (\tau_{j+k-1} - j_{j+k-1})P_{j+1}] / (\tau_{j+k-1} - \tau_j).$$

In general for $r = 1, 2, \dots, k$,

$$d(j) = \sum_{i=\mu-k+r}^{\mu} P_{i,j}^{(r)} \alpha_{i,k-r+i}(j),$$

where

$$P_{i,j}^{(1)} = P_i,$$

$$P_{i,j}^{(r+1)} = [(j_{j+k-r} - \tau_j)P_{i,j}^{(r)} + (\tau_{j+k-r} - j_{j+k-r})P_{i,j}^{(r-1)}] / (\tau_{j+k-r} - \tau_j).$$

Thus taking $r = k$ we have

$$d(j) = P_{\mu,j}^{(k)} \alpha_{\mu,1}(j) = P_{\mu,j}^{(k)}.$$

The next algorithm computes $d(j)$ in this way.

ALGORITHM 2. Let k, j, μ, τ , and t_i be as in Algorithm 1 such that (3.33) and (3.34) hold. This algorithm computes $d(j)$ given by (3.35)

- (1) $\mu_2 = \mu - k + 1$
- (2) For $i = \mu_2, \mu_2 + 1, \dots, \mu$
 - (i) $P_i^{(1)} = P_i$
- (3) For $r = 1, 2, \dots, k - 1$
 - (i) $\mu_2 = \mu_2 - 1, kr = k - r, j = i(j + kr)$
 - (ii) For $i = \mu, \mu - 1, \dots, \mu_2$
 - (a) $d_1 = j - \tau(i), d_2 = \tau(i + kr) - j$
 - (b) $P_i^{(r+1)} = (d_1 * P_i^{(r)} + d_2 * P_i^{(r-1)}) / (d_1 + d_2)$
- (4) $d(j) = P_j^{(k)}$

Note that division by zero cannot occur in Algorithms 1 and 2; i.e., since $\tau_\mu < \tau_{\mu+1}$ we always have $d_1 + d_2 > 0$.

4. THE OSLO ALGORITHM

In order to facilitate use of the subdivision methods discussed, we have created both iterative and recursive procedures and have listed input and output. For ease we have used Algorithm 2 as the prototype, since its continuous version is commonly used to calculate B -spline curves.

INPUT

$N =$ where the original polygon has $N + 1$ vertices

$P = (P(0), \dots, P(N))$ the vertices of the defining polygon in either planar or spatial coordinates

K the order of the B -spline curve

$TAU = (TAU(0), \dots, TAU(N + K))$ the knot vector used with the original defining polygon P

$T = (T(0), \dots, T(Q))$ the refinement knot vector for the particular application, $Q \geq N + K$

OUTPUT

$D = (D(0), \dots, D(Q - K))$ the vertices of the subdivided polygon which define the same curve as P .

```

procedure loop (K, N, Q, P, TAU, T, D)
begin for j ← 0 to (Q - K) do
  call find (K + N, TAU, T, j, MU)
  call SUBDIV (P, K, TAU, T, K, MU, j, D(j)) /* line
end
end

```

where

```

procedure find (KN, TAU, T, j, MU)
/* this routine finds the unique MU */
/* so that TAU(MU) ≤ T(j) < TAU(MU + 1) */
begin for i ← 0 to (KN - 1)
  if (T(j) ≥ TAU(i)) then MU ← i
end

```

and

```

recursive procedure SUBDIV (P, K, TAU, T, RPI, I, J, PP)
/* PP is output and equals D[RPI] */
begin
  r ← RPI - 1
  if (r > 0) then
    begin
      PP2 ← 0
      PPI ← 0
      P1 ← T(J + K - r) - TAU(I)
      P2 ← TAU(I + k - r) - T(J + K - r)
      if (P1 < > 0) call SUBDIV (P, K, TAU, T, r, I, J, PPI)
      if (P2 < > 0) call SUBDIV (P, K, TAU, T, r, J - 1, J, PP2)
      PP ← (P1 * PPI + P2 * PP2) / (P1 + P2)
    end
  else PP ← P(I)
end

```

If an iterative form is desired, we can replace /* line with

```
call SUBDIV (P, K, TAU, T, MU, J, D(J))
```

where

```

procedure SUBDIV (P, K, TAU, T, MU, J, PP)
/* PP is output and equals P_{\mu,j}^{(k)} */

```

```

begin
  for I ← (MU ← K + 1) to MU TEMP(1, I) ← P(I);
    for r ← 1 to K - 1 do
      begin
        for i ← (MU - k + 1 + r) to MU
          begin
            T1 ← T(J + K - r) - TAU(i)
            T2 ← TAU(i + K - r) - T(J + K - r)
            TEMP(r + 1, i) ← (T1 * TEMP(r, i) + T2 * TEMP(r, i - 1))
              / (T1 + T2)
          end
        end
      end
    PP ← TEMP(K, MU)
  end

```

5. APPLICATIONS OF THE OSLO SUBDIVISION ALGORITHM

The Oslo algorithm described in the previous section has many applications in CAGD, computer graphics, and numerical analysis. In this section we will give some sample extensions and generalizations that are made possible by the power to perform subdivision on arbitrary refinements. It is evident to the authors that the optimal choice of the refinement vector i in many applications is a freedom that can be exploited considerably beyond what is proposed here. The refinement choice may also be optimized for a particular application in which additional a priori knowledge is available. So the following algorithms might be viewed as starting points for the construction of more specialized or sophisticated (intelligent) algorithms based on the same theory. This section is meant to suggest a collection of algorithms whose optimal forms cannot be adequately developed within the scope of this paper.

APPLICATION 1. Add a new (pseudo)knot at a prescribed point and calculate the new control polygon.

This application is an essential step in Knapp's approach to designing curves and surfaces by introducing more flexibility, i.e., control points, in regions where finer specification is required [17]. The calculations for the new vertices in (a) are carried out in detail for this application, so that it can serve as an example in computing with the Oslo algorithm.

(a) The original curve is an open cubic B-spline curve with $\tau = (0, 0, 0, 0, 1, 2, 3, 3, 3, 3)$. A pseudoknot is added at 1.3, so $t = (0, 0, 0, 0, 1, 1, 3, 2, 3, 3, 3, 3)$.

Solution. Since it is cubic $K = 4$.

$$\tau = (0, 0, 0, 0, 1, 2, 3, 3, 3, 3)$$

$N + K = 9$, so $N = 5$ and the original polygon is $P = (P_0, P_1, \dots, P_5)$. Also, $t = (0, 0, 0, 0, 1, 1, 3, 2, 3, 3, 3, 3)$ so $Q = 10$ and $D = (D_0, \dots, D_8)$.

Now in procedure loop, $\mu = 3$ for $j = 0, 1, 2, 3$ since

$$\tau_j = 0 \leq t_j < \tau_4 = 1.$$

so $D_j = P_{3,j}^{(4)}, j = 0, 1, 2, 3$.

Now let $j = 0$.

$$\begin{aligned}
 P_{3,0}^{(4)} &= [(t_1 - \tau_3)P_{3,0}^{(3)} + (\tau_3 - t_1)P_{2,0}^{(3)}] / (\tau_4 - \tau_3) \\
 &= [(0 - 0)P_{3,0}^{(3)} + (1 - 0)P_{2,0}^{(3)}] / 1 \\
 P_{2,0}^{(3)} &= [(t_2 - \tau_2)P_{2,0}^{(2)} + (\tau_2 - t_2)P_{1,0}^{(2)}] / (\tau_3 - \tau_2) \\
 &= [(0 - 0)P_{2,0}^{(2)} + (1 - 0)P_{1,0}^{(2)}] / 1 \\
 P_{1,0}^{(2)} &= [(t_3 - \tau_1)P_{1,0}^{(1)} + (\tau_3 - t_3)P_{0,0}^{(1)}] / (\tau_4 - \tau_1) \\
 &= [(0 - 0)P_{1,0}^{(1)} + (1 - 0)P_{0,0}^{(1)}] / 1 \\
 &= P_{0,0}^{(1)} = P_0.
 \end{aligned}$$

Therefore $D_0 = P_0$.

Let $j = 1$.

$$\begin{aligned}
 P_{3,1}^{(4)} &= [(t_2 - \tau_3)P_{3,1}^{(3)} + (\tau_4 - t_2)P_{2,1}^{(3)}] / (\tau_4 - \tau_3) \\
 &= [(0 - 0)P_{3,1}^{(3)} + (1 - 0)P_{2,1}^{(3)}] / 1 \\
 P_{2,1}^{(3)} &= [(t_3 - \tau_2)P_{2,1}^{(2)} + (\tau_4 - t_3)P_{1,1}^{(2)}] / (\tau_4 - \tau_2) \\
 &= [(0, 0)P_{2,1}^{(2)} + (1 - 0)P_{1,1}^{(2)}] / 1 \\
 P_{1,1}^{(2)} &= [(t_4 - \tau_1)P_{1,1}^{(1)} + (\tau_4 - t_4)P_{0,1}^{(1)}] / (\tau_4 - \tau_1) \\
 &= [(1 - 0)P_{1,1}^{(1)} + (1 - 1)P_{0,1}^{(1)}] / (1 - 0) \\
 &= P_{1,1}^{(1)} = P_1.
 \end{aligned}$$

Therefore $D_1 = P_1$.

Let $j = 2$.

$$\begin{aligned}
 P_{3,2}^{(4)} &= [(t_3 + 1 - \tau_3)P_{3,2}^{(3)} + (\tau_4 - t_3)P_{2,2}^{(3)}] / (\tau_4 - \tau_3) \\
 &= [(0 - 0)P_{3,2}^{(3)} + (1 - 0)P_{2,2}^{(3)}] / 1 \\
 P_{2,2}^{(3)} &= [(t_4 - \tau_2)P_{2,2}^{(2)} + (\tau_4 - t_4)P_{1,2}^{(2)}] / (\tau_4 - \tau_2) \\
 &= [(1 - 0)P_{2,2}^{(2)} + (1 - 1)P_{1,2}^{(2)}] / 1 \\
 P_{1,2}^{(2)} &= [(t_5 - \tau_2)P_{1,2}^{(1)} + (\tau_5 - t_5)P_{0,2}^{(1)}] / (\tau_5 - \tau_2) \\
 &= [1.3P_{1,2}^{(1)} + 0.7P_{0,2}^{(1)}] / 2.
 \end{aligned}$$

Therefore $D_2 = 0.65P_2 + 0.35P_1$.

Let $j = 3$.

$$\begin{aligned}
 P_{3,3}^{(4)} &= [(t_4 - \tau_3)P_{3,3}^{(3)} + (\tau_4 - t_4)P_{2,3}^{(3)}] / (\tau_4 - \tau_3) \\
 &= [(1 - 0)P_{3,3}^{(3)} + (1 - 1)P_{2,3}^{(3)}] / 1 \\
 P_{2,3}^{(3)} &= [(t_5 - \tau_3)P_{2,3}^{(2)} + (\tau_5 - t_5)P_{1,3}^{(2)}] / (\tau_5 - \tau_3) \\
 &= [(1.3 - 0)P_{2,3}^{(2)} + (2 - 1.3)P_{1,3}^{(2)}] / 2
 \end{aligned}$$

$$\begin{aligned}
 &= 0.65P_3^{[2]} + 0.35P_2^{[2]} \\
 P_4^{[2]} &= [(t_6 - \tau_3)P_3^{[2]} + (t_6 - t_6)P_4^{[2]}]/(t_6 - \tau_3) \\
 &= [(2-0)P_3^{[2]} + (3-2)P_4^{[2]}]/3 \\
 &= \frac{2}{3}P_3^{[2]} + \frac{1}{3}P_4^{[2]} = \frac{2}{3}P_3 + \frac{1}{3}P_4 \\
 P_4^{[3]} &= [(t_6 - \tau_2)P_3^{[3]} + (t_6 - t_6)P_4^{[3]}]/(t_6 - \tau_2) \\
 &= [(2-0)P_3^{[3]} + (2-2)P_4^{[3]}]/2 \\
 &= P_3^{[3]} = P_3 \\
 &= 0.65(\frac{2}{3}P_3 + \frac{1}{3}P_4) + 0.35(P_3) = 0.434P_3 + 0.217P_4 + 0.35P_3 \\
 &= 0.434P_3 + 0.566P_2.
 \end{aligned}$$

Therefore $D_3 = 0.434P_3 + 0.566P_2$.

Let $j = 4$; then $\mu = 4$ since $\tau_4 = 1 \leq t_4 = 1 < \tau_5 = 2$.

$$\begin{aligned}
 P_4^{[4]} &= [(t_5 - \tau_4)P_3^{[4]} + (t_5 - t_5)P_4^{[4]}]/(t_5 - \tau_4) \\
 &= [(1.3-1)P_3^{[4]} + (2-1.3)P_4^{[4]}]/1 = 0.3P_3^{[4]} + 0.7P_4^{[4]} \\
 P_4^{[4]} &= [(t_6 - \tau_4)P_3^{[4]} + (t_6 - t_6)P_4^{[4]}]/(t_6 - \tau_4) \\
 &= [(2-1)P_3^{[4]} + (3-2)P_4^{[4]}]/2 \\
 &= \frac{1}{2}P_3^{[4]} + \frac{1}{2}P_4^{[4]} \\
 P_4^{[4]} &= [(t_7 - \tau_4)P_3^{[4]} + (t_7 - t_7)P_4^{[4]}]/(t_7 - \tau_4) \\
 &= P_4 = [(3-1)P_3^{[4]} + (3-3)P_4^{[4]}]/2 = P_4^{[4]} \\
 &= P_4^{[2]} = [(t_7 - \tau_3)P_3^{[4]} + (t_7 - t_7)P_4^{[4]}]/(t_7 - \tau_3) \\
 &= [(3-0)P_3^{[4]} + (3-3)P_4^{[4]}]/3 = P_3^{[4]} \\
 &= \frac{1}{3}P_4 + \frac{1}{3}P_3 \\
 P_4^{[4]} &= [(t_6 - \tau_3)P_3^{[4]} + (t_6 - t_6)P_4^{[4]}]/(t_6 - \tau_3) \\
 &= [(2-0)P_3^{[4]} + (2-2)P_4^{[4]}]/2 = P_3^{[4]} = P_4^{[4]} \quad (\text{as above}) \\
 &= P_3 \\
 &= 0.3(\frac{1}{3}P_4 + \frac{1}{3}P_3) + 0.7(P_3) = 0.15P_4 + 0.85P_3
 \end{aligned}$$

Therefore $D_4 = 0.15P_4 + 0.85P_3$.

Let $j = 5$; then $\mu = 4$ since $\tau_4 = 1 \leq t_5 = 1.3 < \tau_5 = 2$.

$$\begin{aligned}
 P_4^{[5]} &= [(t_6 - \tau_4)P_3^{[5]} + (t_6 - t_6)P_4^{[5]}]/(t_6 - \tau_4) \\
 &= [(2-1)P_3^{[5]} + (2-2)P_4^{[5]}]/1 \\
 &= P_3^{[5]} = [(t_7 - \tau_4)P_3^{[5]} + (t_7 - t_7)P_4^{[5]}]/(t_7 - \tau_4) \\
 &= [(3-1)P_3^{[5]} + (3-3)P_4^{[5]}]/2 \\
 &= P_4^{[5]} = [(t_6 - \tau_4)P_3^{[5]} + (t_6 - t_6)P_4^{[5]}]/(t_6 - \tau_4) \\
 &= [(3-1)P_3^{[5]} + (3-3)P_4^{[5]}]/2 = P_4^{[5]} \\
 &= P_4
 \end{aligned}$$

Therefore $D_5 = P_4$.

Let $j = 6$; then $\mu = 5$ since $\tau_5 = 2 \leq t_6 = 2 < \tau_6 = 3$.

$$\begin{aligned}
 P_4^{[6]} &= [(t_7 - \tau_5)P_3^{[6]} + (t_7 - t_7)P_4^{[6]}]/(t_7 - \tau_5) \\
 &= [(3-2)P_3^{[6]} + (3-3)P_4^{[6]}]/1 \\
 &= P_3^{[6]} = [(t_8 - \tau_5)P_3^{[6]} + (t_8 - t_8)P_4^{[6]}]/(t_8 - \tau_5) \\
 &= [(3-2)P_3^{[6]} + (3-3)P_4^{[6]}]/1 \\
 &= P_3^{[6]} = [(t_9 - t_5)P_3^{[6]} + (t_9 - t_9)P_4^{[6]}]/(t_9 - \tau_5) \\
 &= [(3-2)P_3^{[6]} + (3-3)P_4^{[6]}]/1 = P_4^{[6]} \\
 &= P_3
 \end{aligned}$$

Therefore $D_6 = P_3$. Thus we see $D = (P_0, P_1, 0.65P_2 + 0.35P_3, 0.434P_3 + 0.566P_2, 0.15P_4 + 0.85P_3, P_4, P_3)$. See Fig. 3.

(b) The original curve is a uniform cubic B-spline, either floating or periodic, with knot vector $\tau = (0, 1, 2, \dots, 20)$. A new polygon comes from the nonuniform refinement $t = (0, 1, \dots, 7, 8, 8.5, 9, 10, \dots, 20)$, where t has an extra knot at 8.5.

(c) The original curve is a periodic cubic B-spline of the knot vector $\tau = (0, 1, 2, 3, 4)$. The refinement is $t = (0, 0, 1, 2, 3, 4)$, which gives a multiple knot at 0.

APPLICATION 2. (a) Given an open B-spline curve as in (2.1), find two new open polygons $Q = Q_0Q_1 \dots Q_k$ and $R = R_0R_1 \dots R_k$ which define the same curve in two articulated pieces corresponding to a "curve split" at an interior point t_s , so that $\gamma(s) = \sum_{i=0}^k Q_i \theta_i(s)$ for $s \in \{\tau_0, t_s\}$ and $\gamma(s) = \sum_{i=0}^k R_i \theta_i(s)$ for $s \in \{t_s, \tau_k\}$.

This simple curve split can be effected by applying the Oslo algorithm to the original curve using the refinement vector

$$t = (\tau_0, \dots, \tau_k, t_s, \tau_{k+1}, \dots, \tau_k)$$

k times

assuming that t_s was not in the original knot vector τ . In case t_s already appeared in τ care must be taken to give it multiplicity exactly k , a task which could actually be performed as a postprocess on τ to the above as well. In either case the introduction of a knot of multiplicity k at t_s will result in a curve split at that value. Occasionally when working with curve splitting algorithms, it is desirable to divide

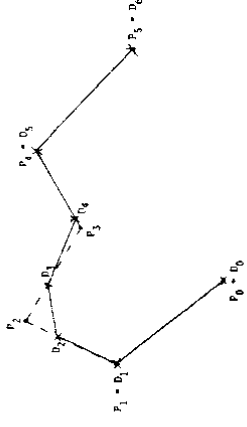


FIG. 3. Sample polygon for Application 1a.

the original curve $\gamma_1(s)$ into separate curves which are represented in a similar form. This can be achieved by dividing the refined knot vector t into two subpieces

$$t_l = (\tau_0, \dots, \tau_u, \underbrace{t_s, \dots, t_s}_{k \text{ times}}, \dots, t_s) \quad \text{and} \quad t_r = (\underbrace{t_s, \dots, t_s}_{k \text{ times}}, \tau_{u+1}, \dots, \tau_p)$$

for the left and right subpieces, respectively. Furthermore, since the relative spacing of the knots only is of importance in determining the shape of the B -splines, it is straightforward to normalize the pieces so that the new knot vectors both begin at a parameter value of 0. Division by the largest knot is also possible to normalize the parameter range, but this is usually not done in practice, since the benefits of doing so are not very important.

(b) Given a closed B -spline curve as in (2.1), split the curve into open B -spline curve points at t_i .

This problem is very similar to Part 2a above, and can be treated in an analogous manner by introducing a new knot of multiplicity k at the parameter value t_i , in the refinement vector t . This procedure is sufficient to specify an open polygon, if the knot vector is treated in a modulo fashion. Otherwise, it is simply a matter of rearrangement to string out the knots to form an ascending knot vector with appropriate multiplicities at the beginning and end knots.

(c) Divide a closed B -spline into two open B -spline curves.

This problem can be viewed as a combination of the process described in Application 2b followed by 2a.

APPLICATION 3. Determine the parametric values of the points of intersection of two B -spline curves $\gamma_1(s)$ and $\gamma_2(t)$ of orders k_1 and k_2 defined over knot vectors τ_1 and τ_2 , respectively.

This algorithm, in its most basic form, is a straightforward generalization of Algorithm V for Curve Intersections in Lane and Riesenfeld [23]. Using the Oslo algorithm, it is possible to split $\gamma_1(s)$ and $\gamma_2(t)$ at their parametric midpoints by adding new knots of multiplicities M_1 and M_2 at the parametric midpoints, in the case that the convex hulls for $\gamma_1(s)$ and $\gamma_2(t)$ overlap. If the convex hulls do not overlap, no curve intersection is possible. This basic procedure involving the above curve-splitting technique can now be applied recursively, adopting all of the refinements including tests for linearity of the subpieces described in [23] until all intersections are found to within a prespecified tolerance. Strategies for splitting at points other than midpoints are intriguing, but the pitfalls of such bolder strategies are many. A more complete analysis of these advantages and disadvantages is a topic for another paper. Here we shall only point out that the structure, but not the performance, of such variants is basically the same.

APPLICATION 4. Extend the Lane-Riesenfeld algorithms for rendering B -spline surfaces.

The power to split nonuniform B -spline curves leads very directly to the power to split tensor product B -spline surfaces in a manner completely analogous to the approach taken by Lane and Riesenfeld in [23]. Essentially they recursively

subdivide, say into four subpieces, until matters of intersections and visibility are reduced to unambiguous, piecewise linear cases. A simple frame buffer algorithm for generating a picture of a B -spline scene then consists of sending a parametric line through the eye and a pixel into the B -spline scene. Subdivision can be used to compute the nearest and subsequent intersections of the parametric line with the scene. Alternatively, one can invoke subdivision until the surface is determined to be sufficiently locally flat, and then resort to any standard polygon-rendering algorithm to compute the corresponding image for display. This, too, is a frame buffer algorithm, as the order of image generation is not generally in scan line order. All of the improvements that apply to the Lane-Riesenfeld algorithm can also be applied in this more general setting. The Oslo algorithm for subdividing arbitrary B -splines serves to extend the class of B -splines to which the Lane-Riesenfeld algorithms can be applied. The structure of the algorithms remains identical.

APPLICATION 5. Extend the Lane-Carpenter algorithm.

The Lane-Carpenter algorithm looks at all surfaces that intersect the "current scan line," and then progressively subdivides all of those "active surfaces" until they become either "nonactive" or sufficiently close to planar that they can be treated as polygons in a Watkins-like manner. Inasmuch as the Oslo algorithm allows for the splitting of nonuniform B -spline surfaces, it is clear that this can be used to extend the Lane-Carpenter algorithm to this wider class of surfaces. The Oslo algorithm can be used to split the active surfaces into four sub-surfaces by splitting at the parametric midpoints along each parameter domain. The remaining portions of the Lane-Carpenter algorithm can be applied without alteration.

APPLICATION 6. Generate curves and extend the deBoor-Cox algorithm.

In [20] it is shown that the polygon associated with each refinement converges to the B -spline curve as long as the mesh size of the refined knot vector goes to zero. Thus one can reasonably use a closely refined knot vector to generate a polygon which is useful as a representation of the curve on a graphics display, since a piecewise linear approximation is usually used for display purposes anyway. It was observed by Lane [18] that the Oslo Algorithm actually specializes to the deBoor-Cox algorithm when the refinement consists of adding a new knot of multiplicity k (the order). This observation further shows that the deBoor-Cox algorithm, in its intermediate recursive calculations, generates vertices corresponding to a split of the B -spline curve at the point of evaluation for the deBoor-Cox algorithm. These properties follow directly from observing when the two formulas become identical. One could consider this application as an extension to Chaikin's algorithm as well, since Chaikin's algorithm was shown to produce quadratic B -spline curves [35].

APPLICATION 7. Convert from B -spline to piecewise Bézier form and generalize constructions of Böhm and of Sablonnière.

As the Bézier curve form is purely polynomial and simpler to understand and use, it is occasionally desirable to convert to it from the B -spline representation. This problem has been studied for specialized cases and algorithms for these have been developed by Böhm [4] and by Sablonnière [36]. The Oslo algorithm is useful for providing the general construction for converting nonuniform B -splines to an

equivalent piecewise Bézier representation for any order k . This application of the Oslo algorithm follows immediately from the fact that the Bézier representation is actually a special case of the B -spline form in which there are knots of multiplicity k at each end of the curve and no interior knots. This necessary condition can be met for any B -spline curve simply by refining the knot vector so that each knot appears with multiplicity k .

APPLICATION 8. *Produce a contouring algorithm for B -splines.*

In many applications in engineering and other areas, it is required that a series of contour plots for a B -spline surface be made. These contours may represent fuel levels in an airplane wing, station curves of a ship's hull, or water lines on a cargo ship. In any case, contours can be computed as the intersection of two B -spline surfaces, one being the model for which the contours are desired and the other being a parametric plane. Stating it in this way, we have reduced the problem of producing contours to the problem of intersecting two B -spline surfaces, a problem to which the Oslo algorithm has already been successfully applied.

APPLICATION 9. *Generate refraction and shadow algorithms for B -spline surfaces.*

In order to produce very realistic pictures of B -spline surfaces representing transparent objects, it is helpful to include refraction calculations to simulate the proper distortion which would be present in a corresponding photograph. This calculation is an easy one involving only the normal to the surface at each pixel and a table of the indices of refraction for the surfaces in the scene being rendered. Certainly such information can be included in a rendering algorithm based on the Oslo algorithm, for the normal vector is easily calculated for a planar polygon produced by the Oslo algorithm. Similarly shadow algorithms require knowledge about the visibility of surfaces from the vantage point of the light source rather than the viewer. Anything that is not visible from the light source is shadowed. Seen in this way, a shadow algorithm is just an invocation of a hidden surface algorithm, an application which has already been discussed in Applications 4 and 5.

APPLICATION 10. *Find the zeros and extrema of a polynomial spline function.*

Normally the problem of finding these critical points of a spline are treated by viewing a spline function as piecewise polynomial and solving the problem for each polynomial span. Clearly subdivision techniques like those employed by Lane and Riesenfeld can be extended to finding the answer globally for the whole spline without decomposing it into constituent polynomial pieces. The formulation and analysis of this approach are a subject of further research by the authors.

APPLICATION 11. *Generate nontensor product surfaces.*

Recently some intriguing procedures for generating an approximating surface to a topologically nonrectangular array of control vertices have been proposed by Catmull and Clark [6], and Doo and Sabin [15]. These schemes are built on subdivision and produce a B -spline surface in the case in which the control points form a rectangular grid. Thus, in this special case, their approach can be viewed as an instance of the Oslo algorithm which produces tensor product B -splines. When the control points do not form a rectangular grid, it is felt that this theory might

shed further light on the behavior of such surfaces. Up to now, analysis of the surfaces has been difficult partly because of the lack of a theoretical basis for understanding subdivision algorithms. The issue is being pursued further by some of the authors.

APPLICATION 12. *Generate additional nodes for finite-element analysis of B -spline surfaces.*

In regions where stress gradients, thermal gradients, and the like are large, it is often required for an adequate finite-element analysis that additional nodes be added to get a better approximation in that region. When the model happens to be a B -spline surface, the addition of extra nodes might be achieved by refining the knot vectors and using the Oslo algorithm to generate the corresponding elements. The authors plan no work in this direction, but only suggest it as a possible application.

APPLICATION 13. *Prove the variation-diminishing property of B -splines.*

Traditionally the proof of the variation-diminishing property of B -spline approximation, which states that B -spline approximation produces approximants with no more undulations than the primitive itself, relies on the rather formidable and not commonly known theory of total positivity. In [22] Lane and Riesenfeld provided a geometric proof of this property based on subdivision of uniform B -splines which avoided the use of total positivity. They are now using the more general construction available through the Oslo algorithm to extend their proof to the general case [20]. This will significantly broaden the audience for whom this important concept of the variation-diminishing property is accessible.

6. CONCLUSION

In the paper we have developed a theoretical framework based on discrete splines for understanding subdivision techniques, with new results that specialize properly to previous work. The Oslo algorithm, in iterative and recursive form, provides the computational tool for widespread application of nonuniform subdivision of B -splines. Nonuniform subdivision greatly extends the uses of the subdivision technique because it is naturally suited in many situations where subdivision is a desirable approach to the problem. While it is evident that this topic can be pursued substantially further, the authors are moved to contain the length of this paper and yield to the increasing pressures for disseminating these results. It is hoped that this paper will provide the necessary results for immediate adoption in cases where they apply directly, as well as serve as a source of direction and reference for the many points of departure herefrom.

ACKNOWLEDGMENTS

E. Cohen and R. Riesenfeld are grateful to the Central Institute of Industrial Research and their hosts Even Mehlum and Frank Lillehagen in Oslo for support and encouragement of this work while they were visiting scientists there, and to NSF (MCS-76-80789 and MCS-78-01966, respectively) for partial support of this research. Riesenfeld is grateful to NITNF in Norway for a travel grant which helped bring about this visit.

REFERENCES

1. R. Barnhill and R. Riesenfeld (Eds), *Computer Aided Geometric Design*, Academic Press, New York, 1974.
2. P. Bézier, *Numerical Control—Mathematics and Applications*, Wiley, London, 1972.
3. J. Lane, L. Carpenter, T. Whitted, and J. Blinn, Scan line methods for displaying parametrically defined surfaces, *Comm. ACM* 23, 1980, 23–34.
4. W. Böhm, Parameterdarstellung, kubischer und bikubischer splines, *Computing* 17, 1976, 87–92.
5. E. Catmull, Computer display of curved surfaces, in *Proceedings, Conference on Computer Graphics, Pattern Recognition, and Data Structure*, (May 1975), pp. 11–17.
6. E. Catmull and J. Clark, Recursively generated B-spline surfaces on arbitrary topological meshes, *Comput. Aided Design* 10, 1978, 350–355.
7. G. Chaikin, An algorithm for high-speed curve generation, *Computer Graphics and Image Processing*, 3, 1974, 346–349.
8. J. H. Clark, Hierarchical geometric models for visible surface algorithms, *ACM* 19, 1976, 547–554.
9. M. G. Cox, The numerical evaluation of B-splines, *J. Inst. Math Appl.* 10, 1972, 134–149.
10. C. deBoor, *A Practical Guide to Splines*, Springer-Verlag, Berlin/New York, 1978.
11. C. deBoor and G. Fix, Spline approximation by quasi-interpolants, *J. Approximation Theory* 7, 1973, 19–45.
12. C. deBoor and A. Pinkus, Backward error analysis for totally positive linear systems, *Numer. Math.* 27, 1977, 485–490.
13. C. deBoor, Splines as linear combinations of B-splines: A survey, in *Approximation Theory II* (G. G. Lorentz, C. K. Chui, and L. L. Schumaker, Eds.), pp. 1–4, Academic Press, New York, 1976.
14. C. deBoor, On calculating with B-splines, *J. Approximation Theory* 6, 1972, 50–62.
15. D. Doo and M. Sabin, Behaviour of recursive division surfaces near extraordinary points, *Comput. Aided Design* 10, 1978, 356–360.
16. A. R. Forrest, Interactive interpolation and approximation by Bézier polynomials, *Comput. J.* 15, 1972, 71–79.
17. L. Knapp, *A Design Scheme Using Coors Surfaces with Nonuniform B-spline Curves*, Ph.D. thesis, Syracuse University, 1979.
18. J. Lane, Private communications, September 1979.
19. J. Lane and L. Carpenter, A generalized scan line algorithm for the computer display of parametrically defined surfaces, *Computer Graphics and Image Processing* 11, 1979, 290–297.
20. J. Lane and R. Riesenfeld, A geometric proof of the variation diminishing property of B-spline approximation, submitted for publication.
21. J. Lane and R. Riesenfeld, Bounds on a polynomial, *BIT*, in press.
22. J. Lane and R. Riesenfeld, *The Application of Total Positivity to Computer Aided Curve and Surface Design*, Computer Science Technical Report UUCS-79-115, October 1979.
23. J. Lane and R. Riesenfeld, A theoretical development for the computer generation of piecewise polynomial surfaces, *IEEE Trans. PAMI* 2, 1980, 35–46.
24. T. Lyche, *Discrete Polynomial Spline Approximation Methods*, Thesis, University of Texas at Austin, 1975. For a summary, see *Spline Functions, Karlsruhe 1975* (K. Bohmer, G. Meinhardt, and W. Schempp, Eds.), pp. 144–176, Lecture Notes in Mathematics No. 501, Springer-Verlag, Berlin/Heidelberg/New York, 1976.
25. T. Lyche, Discrete cubic spline interpolation, *BIT* 16, 1976, 281–290.
26. K. MacCallum, *The Application of Computer Graphics to the Preliminary Design of Ship Hulls*, Ph.D. thesis, London University, 1970.
27. M. A. Malcolm, On the computation of nonlinear spline functions, *Siam J. Numer. Anal.* 14, 1977, 254–282.
28. O. L. Mangasarian and L. L. Schumaker, Discrete splines via mathematical programming, *Siam J. Contr. & Optim.* 9, (1971), 174–183.
29. O. L. Mangasarian and L. L. Schumaker, Best summation formula and discrete splines, *Siam J. Numer. Anal.* 10, 1973, 448–459.
30. J. J. Marsten, An identity for spline functions with applications to variation-diminishing spline approximation, *J. Approximation Theory* 3, 1970, 7–49.
31. W. Newman and R. Sproull, *Principles of Interactive Graphics*, McGraw-Hill, New York, 1973; 2nd ed., 1979.
32. R. Nyddegger, *A Data Minimization Algorithm of Analytical Models for Computer Graphics*, Masters thesis, University of Utah, 1972.
33. U. Ramer, An iterative procedure for the polygonal approximation of plane curves, *Computer Graphics and Image Processing* 1, 1972.
34. R. Riesenfeld, *Applications of B-Spline Approximation to Geometric Problems of Computer-Aided Design*, University of Utah, Computer Science Technical Report, UTEC-CSc-73-126, March 1973.
35. R. Riesenfeld, On Chaikin's algorithm, *Computer Graphics and Image Processing* 4, 1975, 304–310.
36. P. Sablonniere, Spline and Bézier polygons associated with a polynomial spline curve, *Comput. Aided Design* 10, 1978, 257–261.
37. L. L. Schumaker, Constructive aspects of discrete polynomial spline functions, in *Approximation Theory* (C. C. Lorentz, Ed.), pp. 469–476, Academic Press, New York, 1973.
38. B. A. Barakly, End conditions for uniform B-spline curve and surface representations, submitted for publication.
39. W. Böhm, Inserting new knots into B-spline curves, *Comput. Aided Design* 12, 1980, 199–201.