Splines in Interactive Computer Graphics

Richard H. Bartels

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
N2L 3G1

Research Report CS-83-27

September 1983

Splines in Interactive Computer Graphics

Richard H. Bartels

University of Waterloo
Department of Computer Science
Waterloo, Ontario
Canada N2L 3G1

ABSTRACT

Computer graphics, particularly interactive computer graphics, is not, as the name might imply, concerned with drawing graphs, but rather with the broadest issues of manipulating, transforming, and displaying information in visual format. It is interactive in so far as operations can be carried out in real time – which requires algorithms of high computational efficiency and low complexity.

Splines are a valuable tool in graphics, but they are often applied in a way not used by the mathematician. This difference raises computational issues which the numerical analyst might otherwise never see. This talk will provide a brief introduction to such issues and follow with a study of two current developments.

We begin with a review of the graphics environment, mentioning the modelling and display process and pointing out some of the costly issues. The novel use of splines in interactive graphics comes through the construction of surfaces as weighted averages of selected points, called "control vertices" in which B-splines are taken as the weighting functions. Some examples will illustrate the characteristics of this use of B-splines.

With this background we consider two recent developments. The first is the control-vertex recurrence of Riesenfeld, Cohen, and Lyche; the second is Barsky's work on geometric vs. mathematical continuity, and his introduction of Beta-splines. We will close with some results on current research concerned with a synthesis of these two developments.

1. Introduction

The Computer Graphics Laboratory at the University of Waterloo has embarked on a programme to investigate techniques of potential use for the next generation of computer-aided design systems. The terms of reference are *interactive* and *surface modeling*. The context is not that of fitting curves or surfaces to data or objects which exist – plotting or approximation is not of interest. The context is that of (1) providing a mathematically naive industrial draftsman, sitting before a screen and control panel, with an easy way of creating the mathematical description of an object which does not yet exist, (2) displaying and manipulating images of that object on the screen, (3) modifying the object, and (4) ultimately generating machine-tool commands which will provide a means for producing the object. In design systems of this type splines have been very important in the past, and they are undergoing interesting developments for the future.

In this presentation we will look at a typical visual display environment in computer graphics to set the stage and provide a motivation for some of the things to be mentioned subsequently. There will be a brief, informal, intuitive review of the classical construction of B-splines, concentrating on simple knots, to provide a paradigm for some new developments. The use of B-splines in computer graphics to construct curves and surfaces is distinctly different from the use of B-splines in approximation and interpolation. This use will be presented, along with some of the reasons it is particularly appropriate from the point of view of computational efficiency and human interface. The work of Lyche, Riesenfeld, and Cohen for subdividing curves and surfaces, as a means of modification and display, will be outlined.

The author expresses his appreciation for the support he received in preparing this document from the sponsors of the 1983 Dundee Conference on Numerical Analysis. Some of the newer material derives from joint research by the author and John C. Beatty, and was done at the Computer Graphics Laboratory of the University of Waterloo. Funding has been received from the Natural Sciences and Engineering Research Council of Canada, from the National Science Foundation of the United States, from the Ontario Board of Industrial Leadership and Development, from the National Research Council of Canada, and from the Laidlaw Corporation. Appreciation is also expressed to the Mathematics Faculty Computing Facility of the University of Waterloo for its help in photo-printing.

This will end one of the thrusts of the presentation. A second will cover the concept of geometric (as opposed to mathematical) continuity which was explored by Barsky, and which provides a generalisation of B-splines to functions which can serve as generators for "tensed" and "biased" splines. Some examples of curves produced by these functions, called Beta-splines, will be given.

The third portion of this presentation will cover recent progress in expanding the notion of geometric continuity to a more general context and in adapting the classical B-spline construction methods to the production of Beta-splines. The goal of this work is to develop subdivision recurrences for Beta-splines.

The primary references for the spline material in this presentation are [2,5,8] and [9]. Good backgrounds on the graphics environment are to be found in [1,6] and [7].

2. The Graphics Display Environment

The following is not the only example of a visual display environment for a design system, but it is typical and will serve to motivate the later discussion. Figure 1 gives an overview of a *display pipeline*. In it, mathematical objects are defined separately as *templates*, each in its own *local coordinate system*, each arranged according to some canonical scaling and orientation. These objects a placed together in a *world coordinate system* using rotations, translations and scalings, all of which are rigid transformations. As such, they preserve the character of the objects; in particular, polynomials remain polynomials. The result of these *modelling transformations* is a composite scene to be displayed. Each time a new view of the scene is to be taken, *viewing transformations* must be carried out. In industrial design systems these may be no more complicated than a single orthographic projection. In graphic art systems however, a camera model is often used: an eye-point is specified, as are viewing direction, upward orientation, angle of view, aperture, depth of field, and image plane. The *viewing frustum* which results is often mapped to a canonical viewing configuration by further rigid transformations. This canonical configuration is then subjected to a distortion by a *perspective transformation*. Magically, straight lines are preserved, but polynomials become rational functions. Under this distortion, the viewing frustum is changed into a canonical *clipping box*. Up until now, objects in the scene which are not within the viewing volume may have been involved in the computations. Algorithms are now invoked to trim away all portions of the scene outside of this volume and project only what remains onto the image plane. The projected objects must then be discretised, if the display is a *raster device*, or be approximated in outline by simple, known curves, if the display is a *calligraphic device*.

Profound implications are hidden in the above. To do all of this exactly, transformations would have to be applied to infinite numbers of points or to functional descriptions of surfaces; extensive root-finding techniques would be required to determine the curves of intersection between pairs of surfaces, as well as between surfaces and the sides of the viewing frustum; information would also have to be extracted to determine which surfaces are obscured by which – requiring that additional root-finding operations be performed to trace the silhouettes of objects with respect to the eye-point – and finally, discretisation processes akin to differential-equation solvers would have to be applied to paint an image on the display. For the purposes of industrial design, a wire-frame rendering of each object in the scene may suffice, but for other purposes a more realistic rendering of the objects may be required. This may involve something as elaborate as using a mathematical model of the optical characteristics of some collection of materials together with computations requiring normal vectors to the various surfaces and the illuminant details of a number of light sources. If all of this is to be interactive, then it must take place as fast as the refresh rate of the display – that is: typically within a sixtieth to a thirtieth of a second. (It is small wonder that the Cray computer corporation has started to advertise in the graphics trade literature.) Finally, if machine-tool descriptions are needed, while they don't need to be performed with the speed of display computations, they will still involve determining how to track contours on a surface while holding a prescribed orientation to the normal – which could be a nontrivial problem in control theory.

The only salvation for owners of small computers lies in determining how little one can do, how efficiently and how approximately, without the disturbing the final result within visual or machine-tool tolerances. In graphics this is accomplished by "creative cheating", which frequently involves applying all of the above processes only to a small numbers of representers of the objects in question. For example, it is still usual to deal mostly with polyhedral bodies, plus a few additional primitive forms such as spheres and cylinders, and carry out the transformations of the pipeline merely on vertices, centres, radii, etc., using the images of these few representers at the bottom of the pipeline to recreate approximate pictures. One feature of the unique way in which spline surfaces are used in graphics derives from the fact it provides an efficient way for the surfaces to be approximated to arbitrary accuracy by polyhedral surfaces. It is to these polyhedra rather than to the splines themselves that the above computations in the pipeline can be applied.

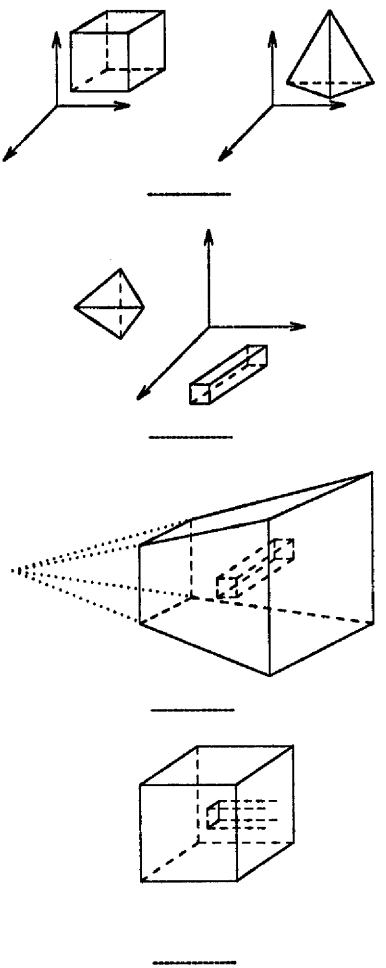


Figure 1. The graphics display pipeline.

3. Splines

For any $k > 0$, let $\mathbf{P}^k$ be the space of polynomials of order k with real coefficients:

$$p(u) = c_0 + c_1 u + \cdots + c_{k-1} u^{k-1} \ .$$

For any $m > 0$ let $\mathbf{U}^m$ be a sequence of m *knots*:

$$\mathbf{U}^m = \{ u_0, \ldots, u_m \} \ ,$$

where

$$u_0 \leq u_1 \leq \cdots \leq u_m \ .$$

Each knot u_i has *multiplicity* μ_i, the count of the knots in $\mathbf{U}^m$ having value equal to u_i. This includes u_i itself, By convention, $\mu_i \leq k$ for all i; hence, $u_i < u_{i+k}$ for all $i = 0, \ldots, m-k$. As a further convention, let $a < u_0$ and $b > u_m$, and agree that nothing outside of $[a, b)$ is of any interest.

The set of the M *distinct*, consecutive values in $\mathbf{U}^m$

$$u_{i_0} < \cdots < u_{i_M}$$

are the *breakpoints* of $\mathbf{U}^m$. ($\{ i_0, \ldots, i_m \} \subseteq \{ 0, \ldots, m \}$ is any conveniently chosen subsequence which picks out the breakpoints.) Each breakpoint is associated with the multiplicity of its corresponding knots, and the *breakpoint intervals* defined by $\mathbf{U}^m \bigcup [a, b)$ are the half-open intervals

$$\begin{aligned} \mathbf{I}_0 &= [a, u_{i_0}) \\ \mathbf{I}_j &= [u_{i_{j-1}}, u_{i_j}) \quad \text{for } j = 1, \ldots, M \ . \end{aligned}$$

and

$$\mathbf{I}_{M+1} = [u_{i_M}, b) \ .$$

Formally:

Definition: Assuming that $k \geq 1$, that $m \geq k-1$, and that $1 \leq \mu_i \leq k$ for all $i = 0, \ldots, m$, then $\mathbf{S}(\mathbf{P}^k, \mathbf{U}^m, a, b)$, the set of all splines of order k on $[a,b)$ with the knot sequence $\mathbf{U}^m \subset [a,b)$, breakpoints $\{ u_{i_0}, u_{i_1}, \ldots, u_{i_M} \} \subseteq \mathbf{U}^m$, and breakpoint intervals $\mathbf{I}_0, \ldots, \mathbf{I}_{M+1}$ is the set of all functions of the form $s(u)$ satisfying:

$$s(u) \in \mathbf{P}^k \quad \text{for each } \mathbf{I}_j \ (j = 0, \ldots, M+1)$$

and for any breakpoint u_{i_j} associated with multiplicity μ_{i_j} ($j = 0, \ldots, M$),
if

$$s(u) \equiv p_j \in \mathbf{P}^k \quad \text{on } \mathbf{I}_j$$

and

$$s(u) \equiv p_{j+1} \in \mathbf{P}^k \quad \text{on } \mathbf{I}_{j+1}$$

then

$$D_u^{(l)} p_j(u_{i_j}) = D_u^{(l)} p_{j+1}(u_{i_j}) \quad \text{for } l = 0, \ldots, k-1-\mu_{i_j} \ .$$

In the definition, $D_u^{(l)} p(u_i)$ stands for the l^{th} derivative with respect to u of $p(u)$ evaluated at u_i. We denote p_j, p_{j+1} as the *segment polynomials* which describe $s(u)$ in the interval $\mathbf{I}_j$ and $\mathbf{I}_{j+1}$ respectively. Notice that any issue of end conditions is left open. It is only necessary that a spline be a polynomial in the intervals $\mathbf{I}_0$ and $\mathbf{I}_{M+1}$. Imposing various end conditions will serve merely to isolate subsets of splines, and this issue will be left aside.

It is easily verified that $\mathbf{S}(\mathbf{P}^k, \mathbf{U}^m, a, b)$ is a vector space with $\mathbf{P}^k$ as a subspace.

4. The One-Sided Basis

The dimension of $\mathbf{S}(\mathbf{P}^k, \mathbf{U}^m, a, b)$ is

$$k + \mu_{i_0} + \cdots + \mu_{i_M} \ .$$

This can be made plausible by considering any $s(u) \in \mathbf{S}(\mathbf{P}^k, \mathbf{U}^m, a, b)$ as u moves from a rightwards to b. On the breakpoint interval $\mathbf{I}_0 = [a, u_{i_0})$, $s(u) = p_0(u)$ is a polynomial of order k; hence it can be represented as a linear combination of

$$(u-a)^0, (u-a)^1, \ldots, (u-a)^{k-1} \ . \tag{4.1}$$

In the next interval to the right $u_{i_0} \le u < u_{i_1}$, $s(u)$ changes to

$$\begin{aligned} s(u) &= p_0(u) + (p_1(u) - p_0(u)) \ . \\ &= p_0(u) + \Delta_{1_1}(u) \ , \end{aligned}$$

where $\Delta_1(u)$ "touches zero with $C^{k-1-\mu_{i_0}}$ continuity" at $u = u_{i_0}$. It proves true that $\Delta_1(u)$ can be represented as a linear combination of the truncated power functions

$$(u - u_{i_0})_+^{k-1}, (u - u_{i_0})_+^{k-2}, \ldots, (u - u_{i_0})_+^{k-\mu_{i_0}} \ . \tag{4.2}$$

Consequently $s(u)$ can be represented on the interval $a \le u < u_{i_1}$ as a linear combination of the functions (4.1) together with those of (4.2).

The same arguments apply as u crosses u_{i_j} and $p_j(u)$ changes into

$$p_{j+1}(u) \ = \ p_j(u) \ + \ \Delta_{j+1}(u)$$

for each $j = 0, \ldots, M$.

After further considerations:

Theorem: The functions

$$(u - a)^0, (u - a)^1, \ldots, (u - a)^{k-1} \ ,$$

together with

$$\begin{array}{ll} (u - u_{i_0})_+^{k-1}, (u - u_{i_0})_+^{k-2}, \ldots, (u - u_{i_0})_+^{k-\mu_{i_0}} & \text{for } u_{i_0} \\ (u - u_{i_1})_+^{k-1}, (u - u_{i_1})_+^{k-2}, \ldots, (u - u_{i_1})_+^{k-\mu_{i_1}} & \text{for } u_{i_1} \\ \quad \bullet \ \bullet \ \bullet \qquad\qquad \bullet \ \bullet \ \bullet & \qquad\qquad \bullet \ \bullet \ \bullet \\ (u - u_{i_M})_+^{k-1}, (u - u_{i_M})_+^{k-2}, \ldots, (u - u_{i_M})_+^{k-\mu_{i_M}} & \text{for } u_{i_M} \ . \end{array} \tag{4.3}$$

form a basis for $\mathbf{S}(\mathbf{P}^k, \mathbf{U}^m, a, b)$.

There are precisely $k + \mu_{i_0} + \cdots + \mu_{i_M}$ functions in (4.3). This is the *one-sided basis* of $\mathbf{S}(\mathbf{P}^k, \mathbf{U}^m, a, b)$.

5. Linear Combinations and Cancellation

Computing the coefficients in the representation of a spline with respect to this basis is often an ill-conditioned problem. This arises, roughly, as follows. Most splines with which we would want to deal in practice have moderate values throughout the interval $[a,b)$. The one-sided basis functions, on the other hand, blow up as u increases. Hence, if this basis is used to express "reasonable" spline curves and surfaces, the coefficients required for this could be expected to flip-flop between large positive and negative values in order to force *numerical cancellation* of the basis-function values as u increases.

A second shortcoming, from the point of view of graphics, is that the one-sided basis functions do not have *compact support*; they are all nonzero on half of the real line. If a curve or surface is represented in terms of the one-sided basis and some change is made to the representation to provide an adjustment of shape, then the change has an influence over the entire curve or surface. A complete recomputation of the curve or surface is necessary; no local updates are possible. The continual need engage in costly recomputations will all but rule out interactive graphical design. In graphics it is as significant that the B-splines have compact support as that they provide well-conditioned representations.

The key to constructing a desirable basis from the less desirable (but conceptually simple) one-sided basis is to recognise (1) that a basis with compact support will be an answer to the numerical objections above as well as (2) being desirable from the point of view of computational efficiency. Compact support can be achieved by a process of *symbolic cancellation*, before any numerical computations are begun. It is through this door that divided differences enter.

To illustrate, let $k = 4$ (cubic splines), and consider

$$u_i < u_{i+1} < u_{i+2} < u_{i+3} < u_{i+4} \ .$$

We have

$$\frac{(u-u_{i+1})_+^3-(u-u_i)_+^3}{(u_{i+1}-u_i)} = \begin{cases} 0 & u<u_i \\ -(u-u_i)^3 & u_i\le u<u_{i+1} \\ -3u^2+3u(u_{i+1}+u_i) & u_{i+1}\le u \\ \quad -(u_{i+1}^2+u_{i+1}u_i+u_i^2) & \end{cases} .$$

which is a "nicer" function than either $(u-u_i)_+^3$ or $(u-u_{i+1})_+^3$ in that it grows only quadratically for $u\to\infty$. Denote the result by

$$[u_i,u_{i+1}:t](u-t)_+^3 = \frac{(u-u_{i+1})_+^3-(u-u_i)_+^3}{u_{i+1}-u_i} .$$

Since this function is a linear combination of $(u-u_i)_+^3$ and $(u-u_{i+1})_+^3$, we may substitute it for one of these truncated power functions, e.g. for $(u-u_i)_+^3$. We may carry out a similar operation for the pairs

$$\{u_{i+1},u_{i+2}\}, \{u_{i+2},u_{i+3}\}, \text{ and } \{u_{i+3},u_{i+4}\} ,$$

to produce quadratic-growing substitutes for

$$(u-u_{i+1})_+^3, (u-u_{i+2})_+^3, \text{ and } (u-u_{i+3})_+^3 .$$

Going one stage further, let

$$[u_i,u_{i+1},u_{i+2}:t](u-t)_+^3 = \frac{[u_{i+1},u_{i+2}:t](u-t)_+^3-[u_i,u_{i+1}:t](u-t)_+^3}{u_{i+2}-u_i} ,$$

This function grows only linearly for $u\to\infty$. It may be used to replace $[u_i,u_{i+1}:t](u-t)_+^3$, and the differencing and replacement may be repeated pairwise, again. Ultimately we arrive at a function which "grows as zero" for $u\to\infty$.

When multiple knots appear, the construct

$$[u_i,u_{i+1}:t](u-t)_+^3 = \frac{(u-u_{i+1})_+^3-(u-u_i)_+^3}{u_{i+1}-u_i}$$

is regarded as being taken in the limit as $u_i\to u_{i+1}$. This provides the convention that the derivatives of the one-sided power function $(u-t)_+^r$ with respect to t for fixed u:

$$D_t^{(l)}(u-t)_+^r \quad \text{for } l,r=0,1,2,3,\cdots$$

are to be understood in the left-handed sense. The derivatives of $(u-t)_+^r$ with respect to u for fixed t are similarly to be understood in the right-handed sense. So long as we hold to this, the truncated power functions behave under differentiation just as the ordinary power functions do. This brings us to

---

Definition: For any values $z_i\le\cdots\le z_{i+l}$ the l-th *divided difference* with respect to the variable x of any function f depending upon x (and possibly other variables) is given
for $l=0$ by

$$[z_i:x]f(x) = f(z_i)$$

and for $l>0$ by

$$[z_i,\ldots,z_{i+l}:x]f(x) = \begin{cases} \dfrac{[z_{i+1},\ldots,z_{i+l}:x]f(x)-[z_i,\ldots,z_{i+l-1}:x]f(x)}{z_{i+l}-z_i} & \text{if } z_{i+l}>z_i \\ \dfrac{1}{l!}D_x^{(l)}f(x)\,|_{x=z_i} & \text{if } z_{i+l}=z_i . \end{cases}$$

When this general definition is applied to $(u-t)_+^r$, with t in the role of x and u replacing z, compact-support substitutes for the truncated power functions result, even for the case of multiple knots, and we have

Definition: Assuming that $i+k \leq m$, the *B-spline of order k associated with the knots* $u_i, \ldots, u_{i+k}$ is given by

$$B_{i,k}(u) \;=\; (-1)^k (u_{i+k} - u_i)[u_i, \ldots, u_{i+k} : t](u-t)_+^{k-1}$$

This function has support on $[u_i, u_{i+k})$. The factor $(-1)^k$ ensures nonnegativity, and the factor $(u_{i+k} - u_i)$ provides normalisation. These three observations will be stressed more formally somewhat below.

This does not, as such, provide a full replacement for the one-sided basis. If it had been our intention, for example, to construct a compact-support basis for $\mathbf{S}(\mathbf{P}^k, \mathbf{U}^m, a, b) = \mathbf{S}(\mathbf{P}^4, \{0,1,2,3,4\}, (-1), 5)$, then we would not have enough power functions to carry the differencing process to completion. The one-sided basis for the splines in question would be

$$(u-(-1))^0, \quad (u-(-1))^1, \quad (u-(-1))^2, \quad (u-(-1))^3,$$
$$(u-0)_+^3, \qquad (u-1)_+^3, \qquad (u-2)_+^3, \qquad (u-3)_+^3, \qquad (u-4)_+^3 \quad ,$$

and this provides us with only the one B-spline $(-1)^4(4-0)[0,1,2,3,4:t](u-t)_+^3$. There is nothing to the right of $u_4 = 4$ to use as a "partner" for u_1, u_2, u_3 and u_4 to produce a compact-support function which could substitute for $(u-u_1)_+^3$, for example. Furthermore, there is nothing which can form a compact-support substitute for any of the functions $(u-(-1))^j$, $j = 0, \ldots, 3$. To compensate for this, introduce additional knots:

$$u_{-4} \leq u_{-3} \leq u_{-2} \leq u_{-1} \leq a \equiv (-1)$$

and

$$b \equiv 5 \leq u_5 \leq u_6 \leq u_7 \leq u_8 \quad .$$

Then the differencing process will be able to produce B-splines with supports in the intervals

$$[u_{-4}, u_0), [u_{-3}, u_1), [u_{-2}, u_2), [u_{-1}, u_3), [u_0, u_4),$$
$$[u_1, u_5), [u_2, u_6), [u_3, u_7), [u_4, u_8) \quad .$$

These basis functions can represent any spline in $\mathbf{S}(\mathbf{P}^4, \{0,1,2,3,4\}, (-1), 5)$, but the representation is only good on the interval $[a,b) = [(-1),5)$. Since we are agreed that nothing outside of this interval is of interest, this is good enough. That this is true in general is the Curry-Schoenberg result:

Construction: To the sequence of knots in $\mathbf{U}^m$ add arbitrarily chosen knots

$$u_{-k} \leq \cdots \leq u_{-1} \leq a$$

and

$$b \leq u_{m+1} \leq \cdots \leq u_{m+k} \quad .$$

Then let

$$B_{i,k}(u) \;=\; (-1)^k (u_{i+k} - u_i)[u_i, \ldots, u_{i+k}](u-t)_+^{k-1}$$

for $i = -1, \ldots, m$.

Theorem: Any $B_{-1,k}(u), \ldots, B_{m,k}(u)$ constructed in this fashion is a basis for $\mathbf{S}(\mathbf{P}^k, \mathbf{U}^m, a, b)$.

It is usual in graphics to ignore the possibilities

$$u_{-1} < a \quad \text{and} \quad b < u_{m+1}$$

and to fix

$$u_{-1} = a \;, \; u_{m+1} = b \quad .$$

6. B-spline Properties

The differences of products of functions is governed by the *Leibniz rule*:

Theorem: For any $z_i \leq \cdots \leq z_{i+l}$ and any appropriately differentiable functions, $f_1(x)$ and $f_2(x)$:

$$[z_i, \ldots, z_{i+l}:x]\{f_1(x)f_2(x)\} = \sum_{r=0}^{l} \{[z_i, \ldots, z_{i+r}:x]f_1(x)\}\{[z_{i+r}, \ldots, z_{i+l}:x]f_2(x)\} \ .$$

This has relevance, since

$$B_{i,k}(u) = (-1)^k (u_{i+k} - u_i)[u_i, \ldots, u_{i+k}:t](u-t)_+^{k-1} \ .$$

And

$$(u-t)_+^{k-1} = (u-t)\cdot(u-t)_+^{k-2} \ ,$$

for $k \geq 2$. That is, that $B_{i,k}(u)$ is constructed by differencing a product. Therefore,

$$[u_i, \ldots, u_{i+k}:t](u-t)_+^{k-1} = \sum_{j=i}^{i+k} \{[u_i, \ldots, u_j:t](u-t)\}\cdot\{[u_j, \ldots, u_{j+k}:t](u-t)_+^{k-2}\}$$

It is found that

$$[u_i, \ldots, u_j:t](u-t)$$

is zero save for two of the terms in the summation, and these two terms can be rearranged to give

$$B_{i,k}(u) = (-1)^{k-1}(u_{i+k} - u)[u_{i+1}, \ldots, u_{i+k}:t](u-t)_+^{k-2} + (-1)^{k-1}(u-u_i)[u_i, \ldots, u_{i+k-1}:t](u-t)_+^{k-2} \ . \tag{6.1}$$

It is easy to recognise two lower-order B-splines in this expression. In other words, the B-splines satisfy a *recurrence relation.*

Theorem: For any $i \in \{-k, -k+1, \ldots, m\}$

$$B_{i,1}(u) = \begin{cases} 1 & u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

and

$$B_{i,r}(u) = (u_{i+r} - u)Q_{i+1,r-1}(u) + (u-u_i)Q_{i,r-1}(u) \ ,$$

for $r = 2, 3, \ldots, k$,

where

$$Q_{i,r}(u) = \begin{cases} B_{i,r}(u) \ / \ (u_{i+r} - u_i) & u_i \leq u < u_{i+r} \\ 0 & \text{otherwise} \ . \end{cases}$$

From this recurrence can be established the three properties of *nonnegativity*, *compact support*, and *partition of unity*:

Theorem:

$$B_{i,k}(u) > 0 \quad \text{for } u \in (u_i, u_{i+k}) \ ,$$

$$B_{i,k}(u) = 0 \quad \text{for } u \notin [u_i, u_{i+k}) \ ,$$

and

$$\sum_{i=-k}^{m} B_{i,k}(u) = 1 \quad \text{for each fixed } u \ .$$

(At $u = u_i$ we have: $B_{i,k}(u_i) = 1$ for $k = 1$ and $B_{i,k}(u_i) = 0$ for $k > 1$.) The proofs are inductive in each case, starting at the step functions $B_{i,1}(u)$, for which the properties obviously hold, and marching upward in k by means of the recurrence.

7. Parametric Curves and Surfaces

The above properties are the crux of the utility which B-splines offer to graphics. In graphics, the space $\mathbf{S}(\mathbf{P}^k, \mathbf{U}^m, a, b)$ is not at issue; it is the B-splines themselves which are important. The three properties above allow B-splines to be used as *weight functions* to produce curves and surfaces in *parametric* form.

The *parametric representation of a curve* is a mapping of some interval

$$a \leq u < b$$

onto a continuum of points

$$\mathbf{Q}(u) = [X(u), Y(u)]$$

or

$$\mathbf{Q}(u) = [X(u), Y(u), Z(u)] \ .$$

A surface is provided by a similar sort of mapping

$$\mathbf{Q}(u,v) = [X(u,v), Y(u,v), Z(u,v)]$$

for

$$a \leq u < b \quad \text{and} \quad c \leq v < d \ .$$

Consider the special cases of the above provided by

$$\mathbf{Q}(u) = \sum_{i=-k}^{m} \mathbf{V}_{i+k} B_{i,k}(u) \tag{7.1}$$

where

$$\mathbf{V}_0, \ldots, \mathbf{V}_{m+k}$$

is a collection of points in the plane (or in space), and

$$\mathbf{Q}(u,v) = \sum_{i=-k}^{m} \sum_{j=-k}^{n} \mathbf{V}_{i+k,j+k} B_{i,k}(u) B_{j,k}(v) \tag{7.2}$$

where

$$\mathbf{V}_{0,0}, \ldots, \mathbf{V}_{m+k,n+k}$$

are points in space. Note that (7.1) provides a curve in the plane or in space, and (7.2) gives a surface.

The points $\mathbf{V}$ are known as *control vertices*, and the curves and surfaces $\mathbf{Q}(u), \mathbf{Q}(u,v)$ are weighted averages of these points. In the case of cubics, any point on the curve, $\mathbf{Q}(u)$ for each u, rests in the *convex hull* of only 4 adjacent $\mathbf{V}_i$, because of the compact support of the B-splines. In like manner, each point of a surface, $\mathbf{Q}(u,v)$ for each u,v. lies in the convex hull of only 16 adjacent $\mathbf{V}_{i,j}$. It is this fact which causes the graphics community to refer to the nonnegativity, compact support and partition of unity properties of B-splines collectively as the *convex hull property*.

This method of constructing curves and surfaces has the desirable features that (1) only a small number of points are needed to determine quite extensive curves and surfaces, (2) it is easy for a mathematically unschooled

person to "sculpt" curves and surfaces by placing control vertices and moving them around using a graphics display, (3) these representations of curves and surfaces can be computed efficiently, (4) when any change to a curve and surface involving only a change in a single control vertex is made, these representations can be efficiently updated – the changes induced in the curve or surface are local to the vertex in question.

We close with three pictorial examples.

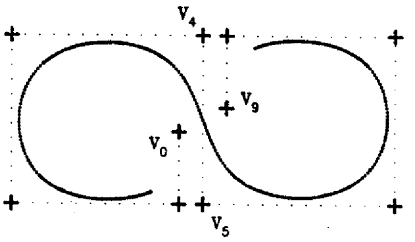
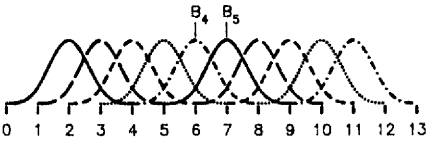


Figure 2. A B-spline generated curve with the B-splines shown for purposes of reference. The separate, pure-polynomial *segments* of the curve have been plotted in alternating solid and dotted lines to add insight. The *joints* between the segments are the images of the knots. Additionally, the lines between successive control vertices have been put in to show the *control graph*. It defines the curve and may be considered a rough approximation to it.



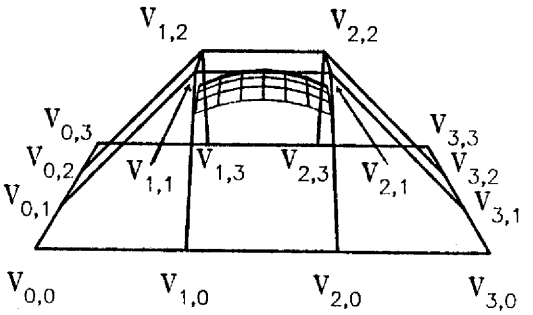


Figure 3. A B-spline generated surface patch. The surface patch itself has been represented as a wire frame mesh. The grid lines on the patch do not correspond to any feature of the splines. Again, the control graph has been made visible by joining the V points according to their index adjacencies in i and j.

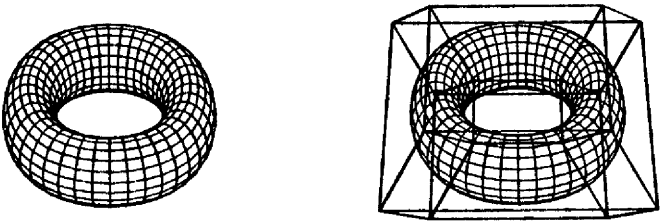


Figure 4. A B-spline generated "torus". (Actually not a torus. Control vertices have been chosen in such a way as to force periodicity in two directions. The cross section of the object is a closed cubic curve rather than a circle.) Again, a wire-frame rendering of the surface has been used, the grid lines do not correspond to any feature of the splines. The the control graph has been drawn in.

8. The Oslo Algorithm

Suppose we have constructed a curve

$$\mathbf{Q}(u) = \sum_{i=-k}^{m} \mathbf{V}_{i+k} B_{i,k}(u)$$

or a surface

$$\mathbf{Q}(u,\bar{v}) = \sum_{i=-k}^{m} \sum_{j=-k}^{n} \mathbf{V}_{i+k,j+k} B_{i,k}(u) B_{j,k}(\bar{v})$$

using control vertices, and we now wish to express the same curve or surface in terms of these control vertices plus some additional ones. There are two different reasons for wanting to do this. The obvious one is that we may wish to "fine tune" the curve or surface by adding some control vertices near a section which is being adjusted. The less

obvious reason is that we may wish to increase the number of control vertices as an intermediate step in displaying the curve or surface. It is a consequence of the convex hull property that this is a reasonable thing to do. To explain: consider any curve constructed from control vertices using cubic B-splines, so that each segment of the curve lies in the convex hull of no more than 4 control vertices. Let d stand for the minimum of the distances from the curve to any of the control vertices. Suppose the same curve can also be represented by double the number of control vertices (or a hundred times that number, or a thousand). It seems plausible that, if the number of control points is increased enough, and if they are distributed uniformly enough about the curve, they cannot all lie outside the minimum distance d. Each segment of the curve must still lie in the convex hull of only 4 vertices, so presumably there can be a large number of vertices in any region about the curve only at the cost of having them close to the curve. This is, indeed, the case. Similar results apply to surfaces constructed from products of B-splines. This suggests that vertices can be added to such an extent that the control graph "converges visually" to the curve or surface; i.e. the graph "clamps down on" the curve or surface as each of its facets is subdivided by additional vertices until each of the resulting subfacets span no more than a pixel or so. The graph rather than the curve or surface can then be subjected to all of the transformations needed to bring a presentable image onto a display device. Since each facet is a polygon (in fact, a quadrilateral for surfaces and a line segment for curves), and since polygons remain polygons throughout the display pipeline (as contrasted with parametric cubics, which are mapped into rational functions by perspective transformations), this adds a significant amount of computational simplicity through the length of the pipeline.

Using curves as the subject of our further discussion, we begin by noting the connection between control vertices and knots. Each new control vertex which we might like to add needs to be weighted by some new B-spline; each new B-spline which we might like to construct needs some knot at which to become nonzero. Thus, we can approach the problem in two ways. We want

$$\mathbf{Q}(u) = \sum_{i=-k}^{m} \mathbf{V}_{i+k} B_{i,k}(u) \equiv \sum_{j=-k}^{n} \mathbf{W}_{j+k} N_{j,k}(u) \ , \tag{8.1}$$

where

$$n > m \ .$$

We can attempt to find the vertices $\mathbf{W}$ explicitly, or we can attempt to find the new B-splines, $N_{j,k}(u)$, and use them to determine the $\mathbf{W}$ implicitly. The latter is the route which is taken. More deviously, everything can be accomplished merely by adding new knots to the existing knot sequence. This process will implicitly define the new control vertices. This approach is the one developed in [8] and is known as the "Oslo algorithm".

Add knots

$$s_1 \leq \cdots \leq s_f$$

to the existing sequence

$$u_{-k} \leq \cdots \leq u_{m+k}$$

to obtain a new sequence

$$\{ u_{-k}, \ldots, u_{m+k} \} \bigcup \{ s_1, \ldots, s_f \} \equiv \{ w_{-k}, \ldots, w_{n+k} \} \ .$$

We will denote the multiplicity of each w_j by ν_j.

Definition: The knot sequence

$$\{ w_{-k}, \ldots, w_{n+k} \}$$

formed in the above fashion will be called a *refinement* of the knot sequence

$$\{ u_{-k}, \ldots, u_{m+k} \} \ ,$$

provided that

$$\nu_j \leq k \quad \text{for all } j = -k, \ldots, n+k$$

and provided that

$$a < s_1 \text{ and } s_f < b \ .$$

The restrictions guarantee that the process of refinement does not lead to a knot sequence which would be illegal for constructing splines of order k and that the spline space we develop with the refined mesh is compatible with the original spline space in the sense that $B_{i,k}(u) \in \mathbf{S}(\mathbf{P}^k, \mathbf{W}^n, a, b)$ for all i,k.

The $N_{j,k}(u)$, of course, are given by

$$N_{j,k}(u) = (-1)^k (w_{j+k} - w_j)[w_j, \ldots, w_{j+k} : t](u-t)_+^{k-1} \tag{8.2}$$

for $j = -k, \ldots, n$.

The next major goal to be reached is the following

Theorem: For each $j = -k, \ldots, n$

$$\mathbf{W}_j = \sum_{i=1}^{m} \mathbf{V}_i \alpha_{i,k}(j) \quad , \tag{8.3}$$

for some collection of numbers $\alpha_{i,k}(j)$.

The proof of this resides in the "add-a-knot" result of the standard B-spline literature. The first step lies in finding a representation for the polynomial $(u-t)^{k-1}$, for any $u \in [a,b)$ and any t, in terms of the basis $N_{j,k}$. In fact, the representation is good for a slightly larger interval than $[a,b)$:

Theorem: For any t and any $u \in [w_{-1}, w_{n+1})$

$$(u-t)^{k-1} = \sum_{j=-k}^{n} \psi_{j,k}(t) N_{j,k}(u) \quad ,$$

where

$$\psi_{j,k}(t) = 1 \text{ if } k = 1 \quad ,$$

and

$$\psi_{j,k}(t) = \prod_{r=1}^{k-1} (w_{j+r} - t) \text{ if } k > 1 \quad .$$

This is known as Marsden's Lemma.

When t is restricted to the discrete set $\{w_{-1}, \ldots, w_{n+1}\}$, then the above representation of $(u-t)^{k-1}$ can be "chopped off" to obtain the representation of $(u-t)_+^{k-1}$ as follows:

Definition: Let

$$\phi_{j,k}(t) = (w_j + \varepsilon_j - t)_+^0 \psi_{j,k}(t) \quad ,$$

where the numbers ε_j satisfy

$$w_j \leq w_j + \varepsilon_j < w_{j+k} \quad .$$

Theorem: For any $t \in \{w_{-k}, \ldots, w_{n+1}\}$ and any $u \in [w_{-1}, w_{n+1})$,

$$(u-t)_+^{k-1} = \sum_{j=-k}^{n} \phi_{j,k}(t) N_{j,k}(u) \quad .$$

The proof of this result is given in [8]. The particular form of the $\phi_{j,k}(t)$, including the ε's, has been chosen to allow differentiation (left-handed) with respect to t, hence the divided-difference operator with respect to t can be applied to obtain

$$\begin{aligned} B_{i,k}(u) &= (-1)^k (u_{i+k} - u_i)[u_i, \ldots, u_{i+k} : t](u-t)_+^{k-1} \\ &= (-1)^k (u_{i+k} - u_i)[u_i, \ldots, u_{i+k} : t] \sum_{j=-k}^{n} \phi_{j,k}(t) N_{j,k}(u) \\ &= \sum_{j=-k}^{n} \left\{ (-1)^k (u_{i+k} - u_i)[u_i, \ldots, u_{i+k} : t]\, \phi_{j,k}(t) \right\} N_{j,k}(u) \quad . \end{aligned}$$

We put these together in the obvious way:

Definition:

$$\alpha_{i,k}(j) = (-1)^k (u_{i+k} - u_i)[u_i, \ldots, u_{i+k} : t]\phi_{j,k}(t) \quad .$$

Theorem:

$$B_{i,k}(u) = \sum_{j=-k}^{n} \alpha_{i,k}(j) N_{j,k}(u) \quad .$$

This completes the add-a-knot result, since it specifies how anything represented in terms of the functions $B_{i,k}(u)$ can be represented in terms of the $N_{j,k}(u)$, the B-spline basis functions for the added-knot mesh.

The functions

$$\phi_{j,k}(t) = (w_j - t)_+^0 \prod_{r=1}^{k-1} (w_{j+r} - t)$$

are similar to the truncated power functions

$$(u-t)_+^{k-1} = (u-t)_+^0 \prod_{r=1}^{k-1} (u-t) \quad .$$

The variable u has been restricted in a peculiar way to the discrete values in the sequence of knots $\mathbf{W}^n$. As a result, $\alpha_{i,k}(j)$ looks like a "discretised" version of $B_{i,k}(u)$. The alpha's are called *discrete B-splines*.

The final step in establishing the result (8.3) comes in observing in (8.1) that

$$\begin{aligned} \sum_{j=-k}^{n} \mathbf{W}_{j+k} N_{j,k}(u) &= \sum_{i=-k}^{m} \mathbf{V}_{i+k} B_{i,k}(u) \\ &= \sum_{i=-k}^{m} \mathbf{V}_{i+k} \sum_{j=-k}^{n} \alpha_{i,k}(j) N_{j,k}(u) \\ &= \sum_{j=-k}^{n} \left[\sum_{i=-k}^{m} \mathbf{V}_{i+k} \alpha_{i,k}(j) \right] N_{j,k}(u) \quad . \end{aligned}$$

We appeal to the linear independence of the functions $N_{j,k}(u)$ in order to equate coefficients on the left and the right.

As might be hoped, the discrete B-splines satisfy a recurrence, from which nonnegativity, compact support, and partition of unity can be established.

Theorem:

$$\alpha_{i,1}(j) = \begin{cases} 1 & u_i \leq w_j < u_{i+1} \\ & \text{and } u_i < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

and

$$\alpha_{i,r}(j) = (w_{j+r-1} - u_i)\gamma_{i,r-1}(j) + (u_{i+r} - w_{j+r-1})\gamma_{i+1,r-1}(j) \quad ,$$

for $r = 2, 3, \ldots, k$, where

$$\gamma_{i,r}(j) = \begin{cases} \alpha_{i,r}(j)/(u_{i+r} - u_i) & \text{if } u_{i+r} > u_i \\ 0 & \text{otherwise} \end{cases}$$

(The $\gamma_{i,r}(j)$ correspond to the Q-spline functions in the continuous case.)

Theorem:

$$\alpha_{i,k}(j) \geq 0$$

$$\sum_{i=-k}^{m} \alpha_{i,k}(j) = 1$$

and

For each j let δ be such that $u_\delta \leq w_j < u_{\delta+1}$.

Then $\alpha_{i,k}(j) = 0$ for $i \notin \{\delta-k+1, \ldots, \delta\}$.

The Oslo algorithm itself is simply the application of these results directly to the control vertices. Briefly, recall that

$$\mathbf{W}_j = \sum_{i=1}^{m} \mathbf{V}_i \alpha_{i,k}(j) \quad .$$

Equivalently,

$$\mathbf{W}_j = \sum_{i=\delta-k+1}^{\delta} \mathbf{V}_i \alpha_{i,k}(j) \quad ,$$

from the compact-support property of the discrete B-splines. This means that the **W** "depend locally" on the **V** in the sense that adding knots in a certain region of u will only change the control vertices being weighted by the B-splines whose nonzero intervals are touched by these new knots. Obviously, this has implications of computational efficiency. In the graphics environment it is possible to update spline curves and surfaces locally, rather than needing to recompute them globally, whenever control vertices are moved or added to.

Note that the recurrence for the alpha's can be applied to produce

$$\begin{aligned} \mathbf{W}_j &= \sum_{i=\delta-k+1}^{\delta} \mathbf{V}_i \alpha_{i,k}(j) \\ &= \sum_{i=\delta-k+1}^{\delta} \mathbf{V}_i \left[(w_{j+r-1} - u_i)\gamma_{i,r-1}(j) + (u_{i+r} - w_{j+r-1})\gamma_{i+1,r-1}(j) \right] \quad . \end{aligned}$$

This develops into a recurrence for the control vertices themselves. The third item in the theorem immediately above guarantees that

$$\gamma_{\delta-k+1,k-1}(j) = \gamma_{\delta+1,k-1}(j) = 0 \quad ,$$

which permits us to collect terms to obtain

$$\mathbf{W}_j = \sum_{i=\delta-k+2}^{\delta} \mathbf{V}_{i,2}(j) \alpha_{i,k-1}(j) \quad ,$$

where

$$\mathbf{V}_{i,2}(j) = \left[(w_{j+k-1} - u_i)\mathbf{V}_i + (u_{i+k-1} - w_{j+k-1})\mathbf{V}_{i-1} \right] / (u_{i+k-1} - u_i) \quad .$$

This may be repeated to yield

Control Vertex Recurrence:

Let

$$\mathbf{V}_{i,1}(j) = \mathbf{V}_i \quad ,$$

and

$$\mathbf{V}_{i,r}(j) = \left[(w_{j+k-r+1} - u_i)\mathbf{V}_{i,r-1}(j) + (u_{i+k-r+1} - w_{j+k-r+1})\mathbf{V}_{i-1,r-1}(j) \right] / (u_{i+k-r+1} - u_i)$$

for $r = 2, \ldots, k$, where ratios with zero denominators may be taken as zero.

Then

$$\mathbf{W}_j = \mathbf{V}_{\delta,k}(j) \quad .$$

This permits the direct computation of the **W** vertices from the **V** vertices using only the $\{w_j\}$ and the $\{u_i\}$ knots. This recurrence is exceedingly efficient.

Strictly speaking, if the display process is carried out by means of this refinement recurrence, using facets of

the control graph rather than patches of the spline curve or surface, the entire graphics environment could work purely in terms of the control vertices – a course graph "at the top of the pipeline" for the purposes of human interface and design, refined to a fine graph "at the bottom" for the purposes of display. The splines which are supposedly underlying this process could be ignored entirely!

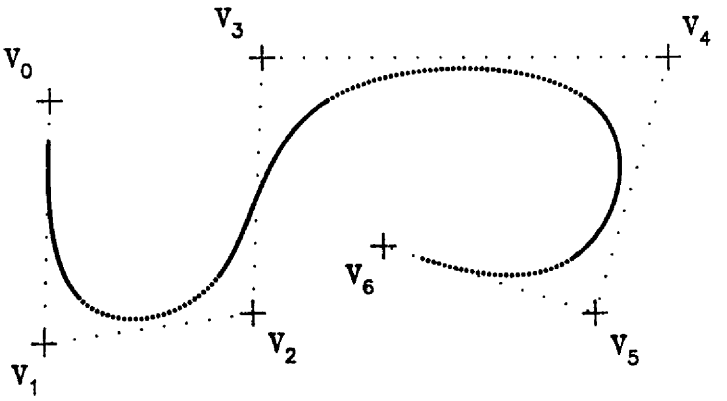
We close with one example of this process. The B-splines which weight the seven control vertices in the figure below are defined on the uniform knots $u_i = i$. The beginning and ending control vertices are repeated once. The coordinates x_i, y_i of the $\mathbf{V}_i$ are given below:

0.4568	1.3369	
0.4568	1.3369	
0.4122	0.2562	$\mathbf{V}_1$
1.3482	0.3788	$\mathbf{V}_2$
1.4100	1.5153	
3.2199	1.4930	
2.8746	0.3565	
1.9387	0.6685	
1.9387	0.6685	

We have flagged the second and third control vertices because, if we introduce a new knot at $u = 4.5$, the new control vertices $\mathbf{W}_j$ prove to be:

0.4568	1.3369	
0.4568	1.3369	
0.4196	0.4363	$\mathbf{V}_1'$
0.8802	0.3175	$\mathbf{P}$
1.3585	0.5682	$\mathbf{V}_2'$
1.4100	1.5153	
3.2199	1.4930	
2.8746	0.3565	
1.9387	0.6685	
1.9387	0.6685	

and the corresponding figure is:



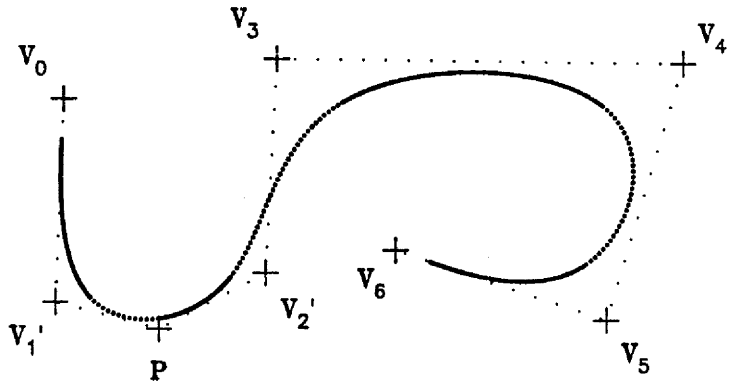


Figure 5. An example of the Oslo algorithm at work.

In particular, notice that the three new control vertices lie closer to the curve than did the two vertices which they replace.

9. Geometric Continuity

We shift gears now to consider an issue which is particularly important in the graphics context, that of the selection of the parameter mapping function. Up until now we have assumed that the interval $[a,b)$ is mapped directly onto a curve $\mathbf{Q}(u)$. In graphics it is often the case that each breakpoint interval $\mathbf{I}_j = [u_{i_j},u_{i_{j+1}})$ is mapped to the unit interval

$$s = s(u) = \frac{u - u_{i_j}}{u_{i_{j+1}} - u_{i_j}} \tag{9.1}$$

and the unit interval is then mapped to an individual curve segment

$$\mathbf{Q}(s) = \mathbf{Q}(s(u)) \quad .$$

This is done particularly when breakpoint intervals are of equal length (or when there are only a small number of possible lengths), for then the number of different segment polynomials is limited, and code can be written especially tuned to each distinct segment polynomial.

Why should we assume that the mapping (9.1) is the best one to use? The mapping given by

$$\bar{s} = \bar{s}(u) = \left[\frac{u - u_{i_j}}{u_{i_{j+1}} - u_{i_j}}\right]^2$$

might be preferred for some reason.

Unfortunately, if one is allowed to reparameterise curves and surfaces at will, some unpleasant and counter-

intuitive effects can arise. An example of what is possible is given in Figure 6 below. The curve shown has a mathematically continuous first derivative by virtue of a cleverly-chosen reparameterisation. It obviously does not have a continuous tangent vector.

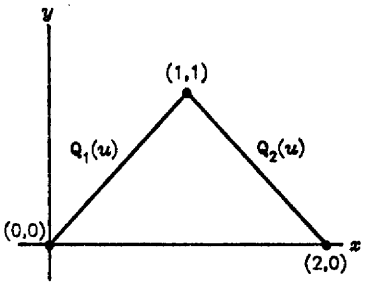


Figure 6. An example of a parametric curve which has a mathematically continuous derivative but does not have a continuously-varying tangent vector.

The parameterisations which have been chosen to make this happen are

$$\mathbf{Q}(u) = [X(u), Y(u)] = \begin{cases} [2u-u^2, 2u-u^2] & \text{for } 0 \leq u < 1 \\ [u^2-2u+2, 2u-u^2] & \text{for } 1 \leq u < 2 \end{cases}$$

Consequently,

$$\mathbf{Q}^{(1)}(u) = [X^{(1)}(u), Y^{(1)}(u)] = \begin{cases} [2-2u, 2-2u] & \text{for } 0 \leq u < 1 \\ [2u-2, 2-2u] & \text{for } 1 \leq u < 2 \end{cases}$$

and so

$$\mathbf{Q}^{(1)}(1) = \begin{cases} [0,0] & \text{from the left} \\ [0,0] & \text{from the right} \end{cases}$$

It is also easy to give examples of (1) a geometrically smooth curve whose tangent vector is mathematically discontinuous, (2) a curve which suffers an abrupt change in curvature but whose second derivative is mathematically continuous, and (3) a curve which is everywhere reasonably curved but whose second derivative changes abruptly. See [2] for these.

What is needed to avoid these anomalies is a definition of continuity which allows for a reasonable change of parametrisation between segments without permitting visual smoothness to be destroyed. Suppose

$$u \leq u^+ \; ; \; s \geq s^-$$

$$u = u(s) \text{ , monotone}$$

and

$$u^+ \text{ corresponds to } s^- \text{ ,}$$

giving the same point on Q.

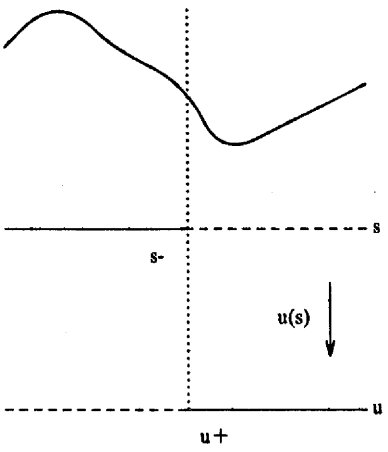


Figure 7. A diagram showing a typical reparameterisation.

This presents the situation in which the parameter mapping on the right is viewed as a reference, and the mapping on the left is viewed as newly introduced. We want the derivative of the curve on the reference side of the join, taken with respect to its parametric variable, to be consistent with the derivative on the other side, taken with respect to the new parametric variable. Moreover, the derivative of the curve should be consistent on both sides of the join, if we were to keep the reference parameterisation throughout, which is just what is required in the usual spline case. That is, we want

$$D_s^{(1)}\mathbf{Q}(u^+) \;=\; D_s^{(1)}\mathbf{Q}(s^-) \quad ,$$

or (by the chain rule)

$$[D_s^{(1)}u^+]\,D_u^{(1)}\mathbf{Q}(u^+) \;=\; D_s^{(1)}\mathbf{Q}(s^-) \quad .$$

That is:

$$[\beta^{(1)}]\,\mathbf{Q}^{(1)}(u^+) \;=\; \mathbf{Q}^{(1)}(s^-) \quad . \tag{9.2}$$

Similarly:

$$[D_s^{(1)}u^+]^2 D_u^{(2)}\mathbf{Q}(u^+) + [D_s^{(2)}u^+]\,D_u^{(1)}\mathbf{Q}(u^+) \;=\; D_s^{(2)}\mathbf{Q}(s^-) \quad .$$

That is:

$$[\beta^{(1)}]^2\mathbf{Q}^{(2)}(u^+) + [\beta^{(2)}]\,\mathbf{Q}^{(1)}(u^+) \;=\; \mathbf{Q}^{(2)}(s^-) \quad , \tag{9.3}$$

and so on. The β's are merely the derivatives of the function which relates the u-parameterisation to the s-parameterisation, the derivatives being evaluated at the joint.

Equations (9.2) and (9.3) are referred to as the conditions for *first geometric continuity*, or G^1 continuity, and *second geometric continuity*, or G^2 continuity, respectively. They were explored by Barsky in [2] on geometric grounds alone, using considerations involving the arc-length parameterisation of a curve, the unit tangent vector, and the curvature vector. In his investigations he dropped emphasis on the underlying parameter mappings and considered the equations (9.2) and (9.3) in their own right, regarding the β's as free (nonnegative) parameters which could be chosen. Clearly the choice of

$$\beta^{(1)} = 1 \text{ and } \beta^{(2)} = 0$$

correspond to the familiar conditions of mathematical continuity. The chain-rule approach above indicates that the notions of geometric continuity extend beyond those considered by Barsky, up to any order.

In the particular case of spline-generated curves, the conditions suggest that, since

$$D_u^{(l)}\mathbf{Q}(u) \;=\; \sum_{i=-k}^{m} \mathbf{V}_{i+k}\,[D_u^{(l)}B_{i,k}(u)] \quad ,$$

an easy way of ensuring that conditions (9.2) and (9.3) should hold for $\mathbf{Q}(u)$ is to insist that corresponding conditions hold for $B_{i,k}(u)$, namely

$$[\beta^{(1)}]\, B_{i,k}^{(1)}(u_j^-) \;=\; B_{i,k}^{(1)}(u_j^+)$$

and

$$[\beta^{(1)}]^2\, B_{i,k}^{(2)}(u_j^-) + [\beta^{(2)}]\, B_{i,k}^{(1)}(u_j^-) \;=\; B_{i,k}^{(2)}(u_j^+)$$

at every knot u_j in the interval of support for $B_{i,k}(u)$.

This suggests that we construct new versions of B-splines – geometrically continuous splines, of compact support, suitable for use as weighting functions. Such functions are known as *Beta-splines*. For example, consider quadratic splines and G^1 continuity, using the knots u_0, u_1, u_2, u_3, so that

$$B_{i,k}(u) \;=\; \begin{cases} p_1(u) & \text{on } [u_0, u_1) \\ p_2(u) & \text{on } [u_1, u_2) \\ p_3(u) & \text{on } [u_2, u_3) \\ 0 & \text{elsewhere} \end{cases} .$$

The conditions of G^1 continuity for this case are

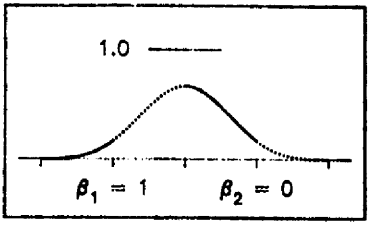
$$\begin{aligned} 0 &= p_1(u_0) \\ p_1(u_1) &= p_2(u_1) \\ p_2(u_2) &= p_3(u_2) \\ p_3(u_3) &= 0 \\ 0 &= p_1^{(1)}(u_0) \\ \beta^{(1)} p_1^{(1)}(u_1) &= p_2^{(1)}(u_1) \\ \beta^{(1)} p_2^{(1)}(u_2) &= p_3^{(1)}(u_2) \\ p_3^{(1)}(u_3) &= 0 \end{aligned} \tag{9.4}$$

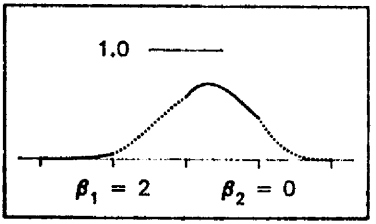
Also, to normalise the result so that it may be used as a weighting function

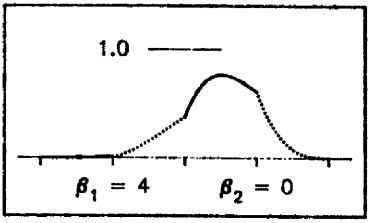
$$1 \;=\; p_2(u_1) + p_3(u_2) \quad .$$

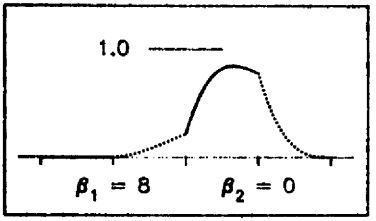
(Note, for future reference, that the value of $\beta^{(1)}$ does not depend upon its associated knot. The same value is used at u_1 and u_2. The result will be what is known as a *uniformly-shaped* quadratic Beta -spline.)

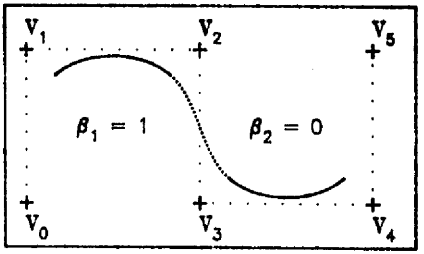
The result of "hand-crafting" such Beta-splines in the case of G^2 continuity and cubics is given below in the next sequence of figures. Note that geometric continuity is not continuity in the ordinary sense. The Beta-splines have obvious discontinuities in tangent, for example, for $\beta^{(1)} \neq 1$. But these discontinuities cancel out when the Beta-splines are used with control vertices to construct parametric curves. The same holds true when Cartesian products of Beta-splines are used to construct surfaces. In the sequence of pictures below giving examples of curves constructed from the beta splines, note how variations in the values of the β's serve to "tense" the curve. Barsky has explored the relationship between Beta-splines and the usual splines under tension, and some of this is to be found in [3]. Beta-splines are receiving enthusiastic interest in the graphics community, because they offer an obvious tool to the graphics designer in helping him "tune" the shape of a curve or surface. Until recently, Beta-spline weighting functions had to be constructed *ab initio* from the defining equations of geometric continuity.

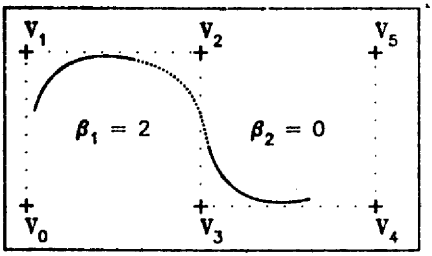

1.0
$\beta_1 = 1$
$\beta_2 = 0$

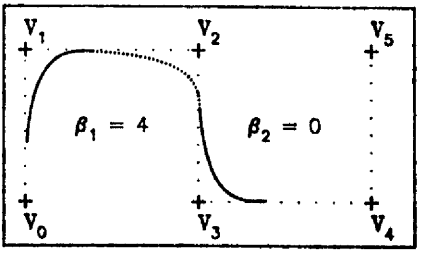

1.0
$\beta_1 = 2$
$\beta_2 = 0$

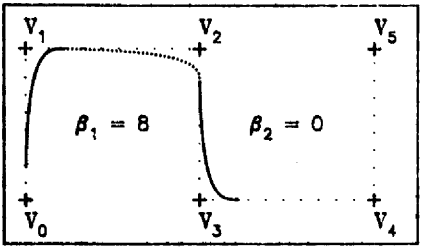

1.0
$\beta_1 = 4$
$\beta_2 = 0$

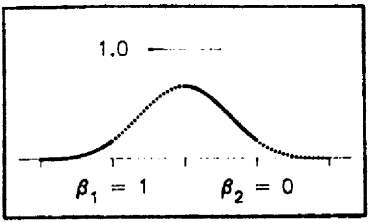

1.0
$\beta_1 = 8$
$\beta_2 = 0$

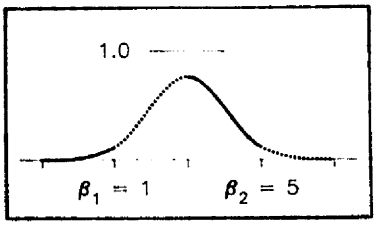

V_1
V_2
V_5
$\beta_1 = 1$
$\beta_2 = 0$
V_0
V_3
V_4

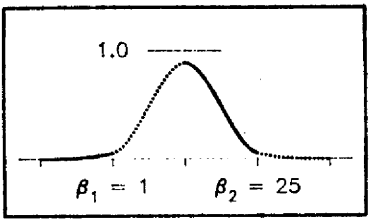

V_1
V_2
V_5
$\beta_1 = 2$
$\beta_2 = 0$
V_0
V_3
V_4

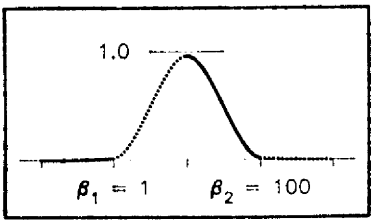

V_1
V_2
V_5
$\beta_1 = 4$
$\beta_2 = 0$
V_0
V_3
V_4

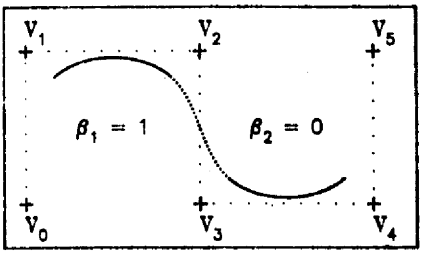

V_1
V_2
V_5
$\beta_1 = 8$
$\beta_2 = 0$
V_0
V_3
V_4

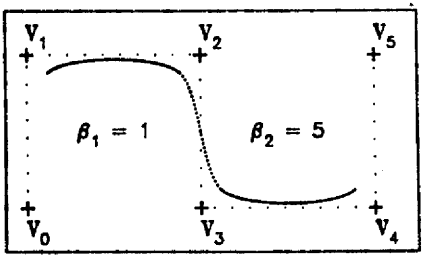

1.0
$\beta_1 = 1$
$\beta_2 = 0$

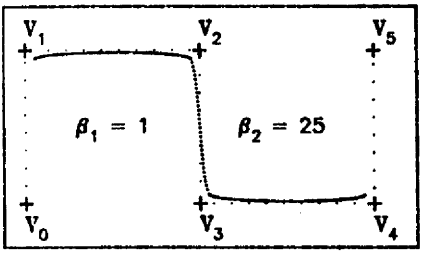

1.0
$\beta_1 = 1$
$\beta_2 = 5$

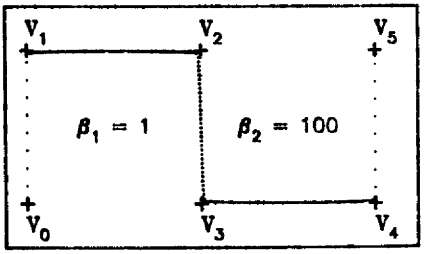

1.0
$\beta_1 = 1$
$\beta_2 = 25$


1.0
$\beta_1 = 1$
$\beta_2 = 100$


V_1
V_2
V_5
$\beta_1 = 1$
$\beta_2 = 0$
V_0
V_3
V_4


V_1
V_2
V_5
$\beta_1 = 1$
$\beta_2 = 5$
V_0
V_3
V_4


V_1
V_2
V_5
$\beta_1 = 1$
$\beta_2 = 25$
V_0
V_3
V_4


V_1
V_2
V_5
$\beta_1 = 1$
$\beta_2 = 100$
V_0
V_3
V_4

10. Extensions

We close by giving an overview of some work in progress.

Geometrically continuous splines on an interval $[a,b)$ with respect to a sequence of knots $\mathbf{U}^m$ form a vector space. This is true even when different values of the β's are associated with different knots; i.e.

$$\beta_1^{(1)} p_1^{(1)}(u_1) = p_2^{(1)}(u_1)$$
$$\beta_2^{(1)} p_2^{(1)}(u_2) = p_3^{(1)}(u_2) \quad .$$

If these were used instead of the corresponding conditions in (9.4), the result would be a *discretely-shaped* quadratic Beta-spline. (The term *continuously-shaped* might also be appropriate, but it has been used in [4] by Barsky and Beatty for a quite different function.) It should be clear that the ideal paradigm to be followed would be (1) find a one-sided basis for discretely-shaped geometric splines, (2) establish a differencing operation to obtain the discretely-shaped Beta-splines as a compact support basis, (3) extract a recurrence from the differencing operation, (4) establish the convex hull property from the recurrence, (5) use the recurrence to represent the one-sided basis in terms of the Beta-splines, (6) obtain the add-a-knot algorithm, an analogue of the discrete splines, and thereby the analogue of the Oslo algorithm. There are many interesting problems here. The remaining content of this presentation deals with the progress we have made on items (1) and (2) above. A remark about item (6), is in order, however. If knots are added to a sequence along with associated β values other than the ones corresponding to mathematical continuity, then the geometrically continuous splines on the refined mesh are not consistent with those on the coarse mesh. There is inclusion of spline spaces under knot refinement only within limited circumstances. This will probably make the discovery of an analogue to the Oslo algorithm quite challenging.

It is by no means clear what should correspond to the feature of a multiple knot in the case of geometric splines. Thus, all of the following discussion assumes simple knots.

The β-associated conditions of geometric continuity can be introduced into the simple truncated power functions by augmenting them with other truncated power functions of lower order. An example for the quadratic, G^1 case should suffice to give the idea.

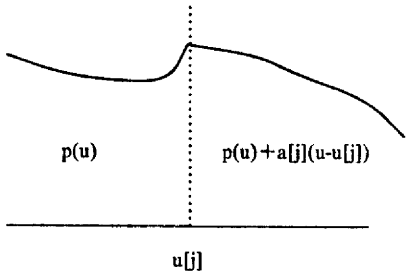


Figure 8. The G-1 jump achieved by a power-function.

With respect to Figure 8, note that, if we expect to have

$$[\beta_j^{(1)}] p^{(1)}(u_j^-) = p^{(1)}(u_j^+) + a_j \quad ,$$

then we can set

$$a_j = [\beta_j^{(1)} - 1] p^{(1)}(u_j)$$

and, as a result, the function

$$p(u) + a_j (u - u_j)_+^1$$

will satisfy the conditions for G^1 continuity at u_j. For cubics and G^2 continuity, we need to consider

$$p(u) + a_j (u - u_j)_+^1 + b_j (u - u_j)_+^2$$

for a similar result.

In the light of this, the one-sided basis for quadratic, G^1 splines proves to consist of functions of the form

$$g_i(u) = (u-u_i)_+^2 + \sum_{j=i+1}^{m} a_{i,j}(u-u_i)_+^1$$

where the $a_{i,j}$ can be determined by

for $i := 0$ **step** 1 **until** $m-1$ **do**

begin

$s := 0$;

for $j := i+1$ **step** 1 **until** m **do**

begin

$a_{i,j} := (\beta_j^{(1)}-1)[2(u_j-u_i)+s]$;

$s := s + a_{i,j}$

end

end .

The process for higher orders of geometric continuity is similar.

It is easy to collect terms in like powers of u; e.g. in the quadratic case:

$$g_i(u) = u^2 + [a_{i,i+1}+a_{i,i+2}+a_{i,i+3}-2u_i]u + \cdots$$
$$g_{i+1}(u) = u^2 + [a_{i+1,i+2}+a_{i+1,i+3}-2u_{i+1}]u + \cdots$$
$$g_{i+2}(u) = u^2 + [a_{i+2,i+3}-2u_{i+2}]u + \cdots$$
$$g_{i+3}(u) = u^2 + [-2u_{i+3}]u + \cdots$$

Higher orders are just as easy to deal with. As yet, however, a symbolic cancellation process has not been discovered which works for all orders. In the quadratic case, for instance, we find that

$$g_i(u) = (u-u_i)_+^2 + \sum_{j=i+1}^{m} a_{i_j}(u-u_j)_+^1$$

$$\Delta g_i(u) = \frac{g_{i+1}(u)-g_i(u)}{\left[\prod_{j=i+1}^{m-1}\beta_j^{(1)}\right](u_{i+1}-u_i)}$$

$$\Delta^2 p_i(u) = \frac{\left[\prod_{j=i+1}^{m-1}\beta_j^{(1)}\right](\Delta g_{i+1}(u)-\Delta g_i(u))}{(\beta_{i+1}^{(1)}u_{i+2}+[1-\beta_{i+1}^{(1)}]u_{i+1}-u_i)} \quad ,$$

but this becomes vastly more complicated for higher orders. However, a computational differencing process can be arranged quite easily for all orders. It is, trivially, the process illustrated here for quadratics:

Collect terms in the functions g_i as above to obtain

$$g_j(u) = u^2 + A_j u + B_j \quad \text{for } j = i, i+1, \cdots \quad .$$

Define

$$\Delta g_i(u) = \frac{(g_{i+1}(u)-g_i(u))}{(A_{i+1}-A_i)}$$
$$= u + \frac{(B_{i+1}-B_i)}{(A_{i+1}-A_i)}$$
$$= u + C_i$$

Then

$$\Delta^2 g_i(u) = \frac{(\Delta g_{i+1}(u)-\Delta g_i(u))}{(C_{i+1}-C_i)} = 1$$

And

$$\Delta^3 g_i(u) \;=\; B_{i,3}(u) \;=\; -(\Delta^2 g_{i+1}(u) - \Delta^2 g_i(u))$$

The functions found in this way are nonnegative, have compact support, and can be verified symbolically to have the partition of unity property. A general proof of these results has not been found.

The corresponding process has been carried out for the cubic case, in particular, and the final drawings of this presentation were all computed using Beta-splines defined by computational differencing.

Currently, work is progressing on the construction of a recurrence for discretely-shaped Beta-splines, starting from first principles and working up from the step functions.

11. References

[1] J. A. Adams and D. F. Rogers (1976), *Mathematical Elements for Computer Graphics*, McGraw-Hill.

[2] B. A. Barsky (1982), The Beta-spline: A Curve and Surface Representation for Computer Graphics and Computer Aided Geometric Design [submitted for publication].

[3] B. A. Barsky (1982), Exponential and Polynomial Methods for Applying Tension to an Interpolating Spline Curve [submitted for publication].

[4] B. A. Barsky and J. C. Beatty (1983), Local Control of Bias and Tension in Beta-splines, *Computer Graphics - SIGGRAPH '83 Conference Proceedings* **17**, 193-218.

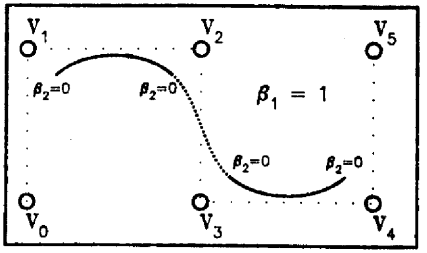
[5] C. de Boor (1978), *A Practical Guide to Splines*, Applied Mathematical Sciences Volume 27, Springer-Verlag.

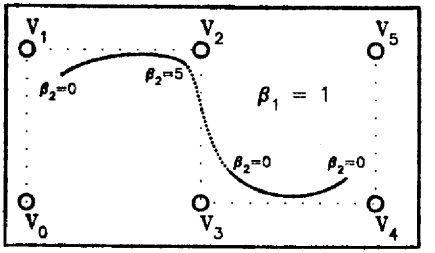
[6] I. D. Faux and M. J. Pratt (1979), *Computational Geometry for Design and Manufacture*, John Wiley & Sons.

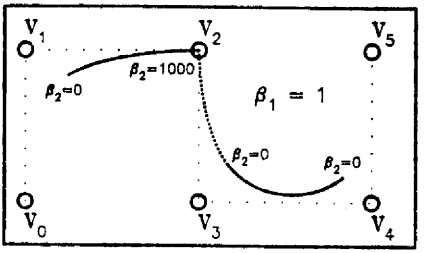
[7] J. D. Foley and A. van Dam (1982), *Fundamentals of Interactive Computer Graphics*, Addison Wesley.

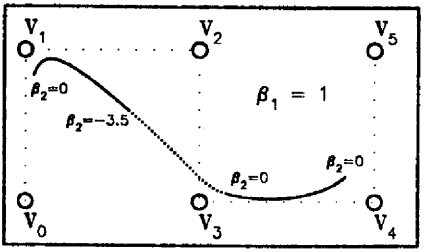
[8] R. Riesenfeld, E. Cohen, and T. Lyche (1980), Discrete B-splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics, *Computer Graphics and Image Processing* **14**(2), October, 87-111.

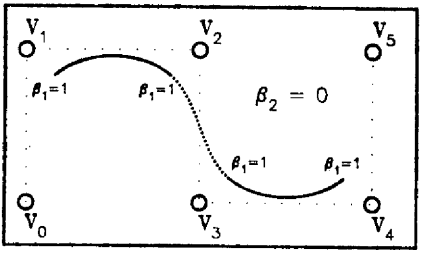
[9] L. L. Schumaker (1981), *Spline Functions: Basic Theory*, John Wiley & Sons.

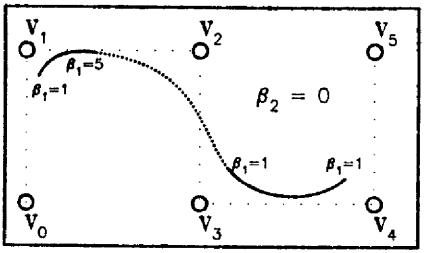

V1
V2
V5
V0
V3
V4
β2=0
β2=0
β1 = 1
β2=0
β2=0

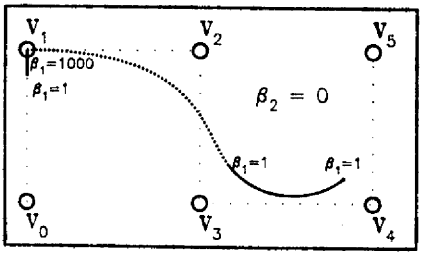

V1
V2
V5
V0
V3
V4
β2=0
β2=5
β1 = 1
β2=0
β2=0

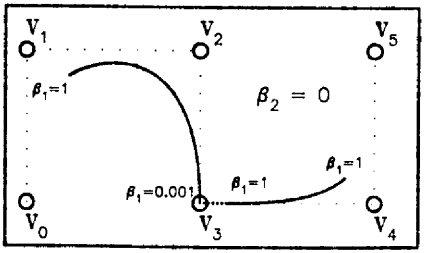

V1
V2
V5
V0
V3
V4
β2=0
β2=1000
β1 = 1
β2=0
β2=0


V1
V2
V5
V0
V3
V4
β2=0
β2=−3.5
β1 = 1
β2=0
β2=0


V1
V2
V5
V0
V3
V4
β1=1
β1=1
β2 = 0
β1=1
β1=1


V1
V2
V5
V0
V3
V4
β1=5
β1=1
β2 = 0
β1=1
β1=1


V1
V2
V5
V0
V3
V4
β1=1000
β1=1
β2 = 0
β1=1
β1=1


V1
V2
V5
V0
V3
V4
β1=1
β2 = 0
β1=0.001
β1=1
β1=1