An Analysis of B-Splines

10 January 2023

Victor Vella

First Published: 25 October 2019
Revised: 10 January 2023

Contents

This document presents an analysis of B-splines, especially uniform B-splines. It explains the nature of B-splines and how they are constructed, and also how B-splines can be merged, joined, and separated. The nature of B-splines is presented mathematically and geometrically. Methods for converting B-spline control points to Bezier control points are also presented, as well as emulating arbitrary curves by means of B-splines. A number of algorithms to implement B-splines is presented in an Appendix. The document also contains an appendix discussing Bezier curves and splines.

1. Introduction

In general, a mathematical spline is a piecewise polynomial function defined to "fit" a sequence of points in some way. If the spline passes through the points, it is said to be an "interpolation"; if it passes near the points it is said to be an "approximation". If a spline passes through some points but near others, it is proper to call it an approximation since passing through a point is also passing near that point, but passing through all the points is a special case of passing near the points, and in that case the spline is called an interpolation. It is also possible to define a one-piece polynomial spline of degree one less than the number of points. However, in practice, doing so is not a viable option because polynomials of high degree tend to oscillate under some conditions, and adding or removing points changes the degree of the polynomial and therefore the nature of the whole spline (adding or removing points associated with a spline should not change the nature of the whole spline). In addition, there is a limit to the degree of a polynomial that can be processed by computer systems. As a good compromise, therefore, piecewise polynomial splines of degree three are preferred.

By definition, a mathematical spline must be continuous at all points, but need not necessarily be smooth at all points. Typically, however, splines are at least once continuously differentiable ($C^{(1)}$).
B-splines are a particular set of general mathematical splines, and all (polynomial) splines can be converted to a series of connected Bezier curves (and their points) of corresponding degree.

2. Definition of a B-spline

A B-spline requires a number of defining points (**control points**) and an ordered set (**knot set**) of non-decreasing real ($\mathbb{R}$) numbers (**knots**). Also, the degree, d, of the polynomial involved in the spline needs to be specified. The first *control point* is traditionally numbered zero; the last *control point* is numbered n. The *control points* are represented by $\mathbf{P}_i$ $[i = 0, \cdots, n;\ i \in \mathbb{Z}]$. There are therefore $n + 1$ *control points*. Each *knot*, after the first d and before the last d *knots*, corresponds to a point (**knot point**) on the spline. *Knot points* are typically not identical to the *control points*. The first *knot* corresponding to the first *knot point* is represented by t_0, and the last *knot* corresponding to the last *knot point* is represented by t_{n+1-d}. There are d *knots* before t_0, and d *knots* after t_{n+1-d} that do not correspond to a *knot point*. Therefore, the *knot set* is $\langle t_{-d}, t_{1-d}, t_{2-d}, \cdots, \mathbf{t_0}, \mathbf{t_1}, \cdots, \mathbf{t_{n+1-d}}, t_{n+2-d}, \cdots, t_{n+1}\rangle$ (only the *knots* shown in **bold blue** correspond to a *knot point*). Note that a *knot set* is sometimes called a "knot vector" in other publications, even though it is not a mathematical vector — a vector has magnitude and direction, which does not apply to *knot sets*. The $n + 2 + d$ *knots* satisfy the following inequalities.

$$t_i \le t_{i+1} \quad [i = -d, \cdots, n;\ i \in \mathbb{Z}] \qquad (1)$$

The actual **B-spline curve** is a function $\mathbf{S}$ of t $[t_0 \le t \le t_{n+1-d};\ t \in \mathbb{R}]$ defined by a sequence of $(n + 1 - d)$ continuous piecewise polynomials (**spline segments**) of degree d $[d > 0]$, where each polynomial segment is defined between $\mathbf{S}(t_i)$ and $\mathbf{S}(t_{i+1})$ $[i = 0, \cdots, (n - d);\ i \in \mathbb{Z}]$. Each $\mathbf{S}(t_i)$ $[i = 0, \cdots, (n + 1 - d);\ i \in \mathbb{Z}]$ is a *knot point*. $\mathbf{S}$ must be continuous at each *knot point*, but need not be continuously differentiable. However, $\mathbf{S}$ is typically at least once continuously differentiable ($C^{(1)}$) at each *knot point*. Note that $n \ge d$. $\mathbf{S}$ typically approximates (passes near) the *control points*.

Each *spline segment* is derived from a **basis function** B of t, and requires $d + 1$ *control points* in its definition. A *basis function* is itself a continuous polynomial function of $d + 1$ segments.

The *basis functions*, $B_{i,d}$, for a d^{th} degree B-spline and a *knot set* $\langle t_{-d}, \cdots, t_{n+1}\rangle$ are defined by the Cox-deBoor recursive equations as follows.

$$B_{i,0}(t) = \begin{cases} 1 & [t_i \le t < t_{i+1}] \\ 0 & [\text{otherwise}] \end{cases} \qquad B_{i,d}(t) = \left(\frac{t - t_i}{t_{i+d} - t_i}\right) B_{i,d-1}(t) + \left(\frac{t_{i+d+1} - t}{t_{i+d+1} - t_{i+1}}\right) B_{i+1,d-1}(t) \quad [t \in \mathbb{R}]\ [i = -d, \cdots, (n + 1);\ i \in \mathbb{Z}] \qquad (2)$$

During the evaluation of equation (2), the result of a division by zero is deemed to be zero. Different definitions of *basis functions* define different kinds of splines (not only B-splines) — only the above definition of *basis functions* defines a B-spline. Note the inequality signs of the interval of t in the definition of $B_{i,0}(t)$. Some

publications incorrectly have $t_i \le t \le t_{i+1}$ for the interval, which can produce incorrect results in some cases if the *basis functions* are evaluated directly. However, the interval $t_i < t \le t_{i+1}$ is also correct.

Note that some publications incorrectly refer to the *basis functions* as "blending functions"; blending functions are derived from the segments of a *basis function* but are not necessarily the *basis functions* themselves.

Using the *basis functions* defined above, a *B-spline curve*, **S**, of degree d is defined as follows.

$$\mathbf{S}(t) = \sum_{i=0}^{n} B_{i-d,d}(t)\mathbf{P}_i \quad [t_0 \le t \le t_{n+1-d};\ t \in \mathbb{R}]\ [d \in \mathbb{Z}^+;\ n \ge d] \qquad (3)$$

Note that the subscript $i - d$ of $B_{i-d,d}$ can be a negative integer. However, if the *knot set* is relabelled so that the first *knot* is labelled t_0 rather than t_{-d}, then equation (3) is equivalent to

$$\mathbf{S}(t) = \sum_{i=0}^{n} B_{i,d}(t)\mathbf{P}_i \quad [t_d \le t \le t_{n+1};\ t \in \mathbb{R}]\ [d \in \mathbb{Z}^+;\ n \ge d] \qquad (3')$$

which is what is presented in most publications, sometimes without mentioning that the interval for t begins with t_d (rather than t_0) and ends with t_{n+1}.

Given equation (3) above, a third-degree ($d = 3$) *B-spline curve* determined by four ($n = 3$) *control points* with the *knot set* ⟨0, 1, 1.5, **2**, **3**, 4, 5, 5.7⟩ would be defined as

$$\mathbf{S}(t) = B_{-3,3}(t)\mathbf{P}_0 + B_{-2,3}(t)\mathbf{P}_1 + B_{-1,3}(t)\mathbf{P}_2 + B_{0,3}(t)\mathbf{P}_3 \quad [\mathbf{2} \le t \le \mathbf{3};\ t \in \mathbb{R}]$$

Note that if $n = d$, then **S** represents only one *spline segment*.

A second-degree ($d = 2$) *B-spline curve* determined by five ($n = 4$) *control points* with the *knot set* ⟨–3, –1.2, **0**, **3**, **3.6**, **4**, 5, 6⟩ would be defined as

$$\mathbf{S}(t) = B_{-2,2}(t)\mathbf{P}_0 + B_{-1,2}(t)\mathbf{P}_1 + B_{0,2}(t)\mathbf{P}_2 + B_{1,2}(t)\mathbf{P}_3 + B_{2,2}(t)\mathbf{P}_4 \quad [\mathbf{0} \le t \le \mathbf{4};\ t \in \mathbb{R}]$$

A *B-spline curve* of degree d has continuity at least $C^{(d-1)}$.

If $t_{i+1} - t_i = c \quad [c \in \mathbb{R}^+]\ [i = -d, \cdots, n;\ i \in \mathbb{Z}]$ then the B-spline is said to be **uniform**, otherwise it is said to be **non-uniform**.

2.1 Uniform B-splines

Uniform B-splines require further restrictions to the definition of a general B-spline. The restrictions are

$$t_{i+1} - t_i = c \quad [c \in \mathbb{R}]\ [i = -d, \cdots, n;\ i \in \mathbb{Z}] \qquad (4)$$

$$t_i < t_{i+1} \quad [i = -d, \cdots, n;\ i \in \mathbb{Z}] \qquad (5)$$

Typically, $c = 1$ and $t_0 = 0$, therefore the *knot set* $\langle t_{-d}, \cdots, t_{n+1} \rangle$ consists of consecutive integers, allowing calculations to be carried out more easily. The typical case will be assumed for *uniform* B-splines. This implies that the substitution $t_j = j$ is allowable, alleviating the need to explicitly specify a *knot set*. Also, because of the restrictions of *uniform* B-splines, the following equation can be deduced

$$B_{j,d}(t) = B_{0,d}(t - t_j) \quad [j \in \mathbb{Z}] \qquad (6a)$$

With $t_j = j$ (the typical case), the equation above becomes

$$B_{j,d}(t) = B_{0,d}(t - j) \quad [j \in \mathbb{Z}] \qquad (6b)$$

The equation above (6b) allows all the *basis functions* for a *uniform* B-spline to be expressed in terms of $B_{0,d}$. As a result, a *uniform B-spline curve*, **S**, of degree d can be defined as

$$\mathbf{S}(t) = \sum_{i=0}^{n} B_{0,d}(t - i + d)\mathbf{P}_i \quad [0 \le t \le (n + 1 - d);\ t \in \mathbb{R};\ n \ge d] \qquad (7)$$

which is deduced from equations (3) and (6b).

2.1.1 Basis Functions

The *basis functions* of a *uniform* B-spline of degree d can be deduced from equations (2) and $t_j = j$ as

$$B_{i,0}(t) = \begin{cases} 1 & [i \le t < i+1] \\ 0 & [\text{otherwise}] \end{cases} \qquad B_{i,d}(t) = \left(\frac{t-i}{d}\right)B_{i,d-1}(t) + \left(\frac{i+d+1-t}{d}\right)B_{i+1,d-1}(t) \quad [i \in \mathbb{Z};\, t \in \mathbb{R}] \qquad (8)$$

Equation (8) is further reduced by equation (6b) as follows

$$B_{i,0}(t) = \begin{cases} 1 & [i \le t < i+1] \\ 0 & [\text{otherwise}] \end{cases} \qquad B_{i,d}(t) = \left(\frac{t-i}{d}\right)B_{0,d-1}(t-i) + \left(\frac{i+d+1-t}{d}\right)B_{0,d-1}(t-i-1) \quad [i \in \mathbb{Z};\, t \in \mathbb{R}] \qquad (9)$$

Notice that, due to equation (7), only calculations where $i = 0$ for $B_{i,d}$ are required. Therefore, the following equation can be used instead of equation (9).

$$B_{0,0}(t) = \begin{cases} 1 & [0 \le t < 1] \\ 0 & [\text{otherwise}] \end{cases} \qquad B_{0,d}(t) = \left(\frac{t}{d}\right)B_{0,d-1}(t) + \left(\frac{d+1-t}{d}\right)B_{0,d-1}(t-1) \quad [t \in \mathbb{R}] \qquad (10)$$

It was mentioned earlier in this document that the *basis functions* are polynomials. To see this, the recursive function $B_{0,d}$ (equation (10)) needs to be expanded out to an explicit function. The expansion will be illustrated by an example where $d = 3$ (i.e. $B_{0,3}(t)$ will be expanded).

Since $B_{0,3}(t)$ requires $B_{0,2}(t)$ which requires $B_{0,1}(t)$, $B_{0,1}(t)$ will be expanded first.

$$B_{0,1}(t) = tB_{0,0}(t) + (2-t)B_{0,0}(t-1) = \begin{cases} t & [0 \le t < 1] \\ 2-t & [1 \le t < 2] \\ 0 & [\text{otherwise}] \end{cases} \qquad (11)$$

When expanding $B_{0,0}(t-1)$, it is important to substitute $t-1$ into the range $0 \le t < 1$ of $B_{0,0}(t)$, thus, $0 \le t < 1$ becomes $0 \le (t-1) < 1$ which, by adding 1 to all terms of the inequation, is equivalent to $1 \le t < 2$.

$$B_{0,2}(t) = \left(\frac{t}{2}\right)B_{0,1}(t) + \left(\frac{3-t}{2}\right)B_{0,1}(t-1) = \begin{cases} \frac{t^2}{2} & [0 \le t < 1] \\ \frac{t(2-t)}{2} & [1 \le t < 2] \\ \frac{(3-t)(t-1)}{2} & [1 \le t < 2] \\ \frac{(3-t)^2}{2} & [2 \le t < 3] \\ 0 & [\text{otherwise}] \end{cases} = \begin{cases} \frac{t^2}{2} & [0 \le t < 1] \\ \frac{t(2-t)+(3-t)(t-1)}{2} & [1 \le t < 2] \\ \frac{(3-t)^2}{2} & [2 \le t < 3] \\ 0 & [\text{otherwise}] \end{cases} \qquad (12)$$

Notice that the two middle ranges ($1 \le t < 2$) of the third term in the equation above have been combined in the fourth term. And finally,

$$B_{0,3}(t) = \left(\frac{t}{3}\right)B_{0,2}(t) + \left(\frac{4-t}{3}\right)B_{0,2}(t-1) = \begin{cases} \frac{t^3}{6} & [0 \le t < 1] \\ \frac{t^2(2-t)+t(3-t)(t-1)}{6} & [1 \le t < 2] \\ \frac{t(3-t)^2}{6} & [2 \le t < 3] \\ \frac{(4-t)(t-1)^2}{6} & [1 \le t < 2] \\ \frac{(4-t)((t-1)(3-t)+(4-t)(t-2))}{6} & [2 \le t < 3] \\ \frac{(4-t)^3}{6} & [3 \le t < 4] \\ 0 & [\text{otherwise}] \end{cases}$$

Expanding the equation above into polynomials and combining the same ranges results in

$$B_{0,3}(t) = \begin{cases} \dfrac{t^3}{6} & [0 \le t < 1] \\ \dfrac{-3t^3 + 12t^2 - 12t + 4}{6} & [1 \le t < 2] \\ \dfrac{3t^3 - 24t^2 + 60t - 44}{6} & [2 \le t < 3] \\ \dfrac{-t^3 + 12t^2 - 48t + 64}{6} & [3 \le t < 4] \\ 0 & [\text{otherwise}] \end{cases} \qquad (13)$$

Equation (13) is the explicit piecewise polynomial *basis function* mentioned previously, which can be used with equation (7) to define the desired *uniform B-spline curve*.

In this example, where $d = 3$, equation (7) becomes

$$\mathbf{S}(t) = \sum_{i=0}^{n} B_{0,3}(t - i + 3)\mathbf{P}_i \quad [0 \le t \le (n - 2);\ t \in \mathbb{R};\ n \ge d] \qquad (14)$$

where $B_{0,3}(t)$ is given by equation (13). Note that, for a *uniform B-spline curve*, equation (7) is equivalent to equation (3) which is the formal definition of a *B-spline curve*. Equation (7) can be used instead of equation (3) because the *basis functions*, $B_{i,d}$ $[i \in \mathbb{Z}]$, all have identical shape but are shifted one unit along the positive X-axis because of the uniformly spaced *knots*. Therefore, if the shifted *basis functions* are accounted for (as they are in equation (7)), only one *basis function* need be used.

The following diagram (**Figure 2-1**) is a plot of the *basis function* $B_{0,3}$ (solid curve). The four one-piece functions (broken curves) that combine to create the single piecewise *basis function* are shown for context.

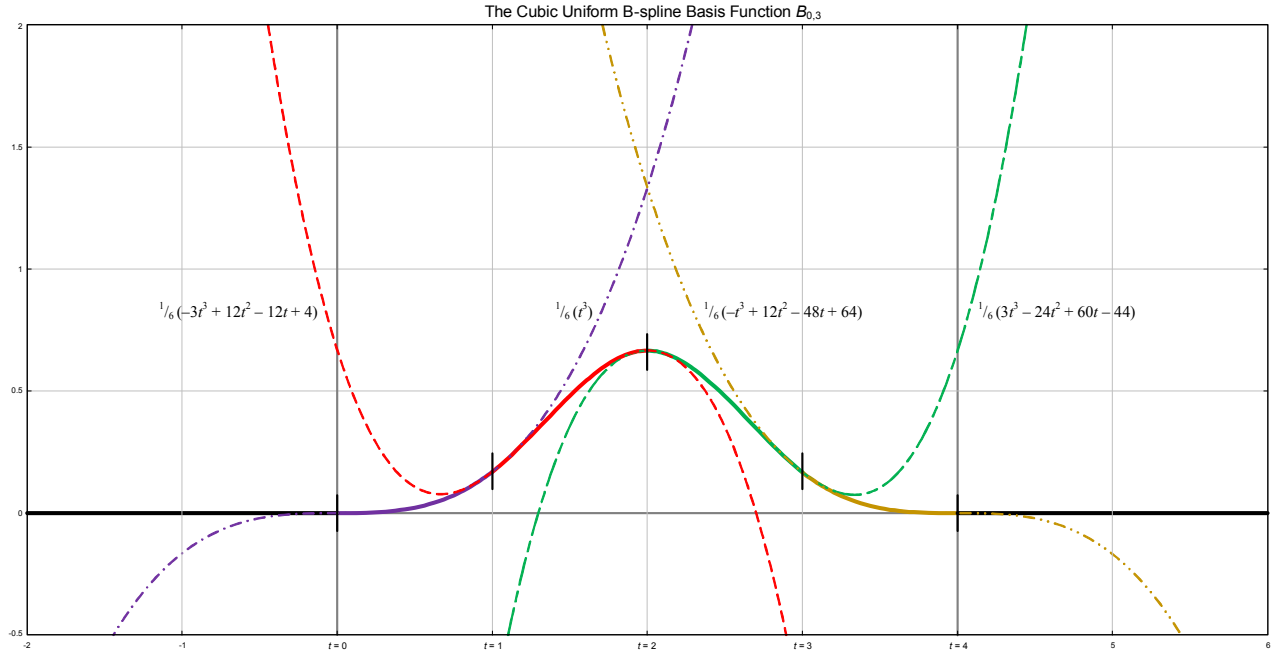

Figure 2-1

In general, a *basis function* of degree d will have $d + 1$ non-zero piecewise segments.

Figure 2-2, below, shows an example of how eight *basis functions* ($B_{-3,3}$, ⋯, $B_{4,3}$) are combined with eight *control points* ($\mathbf{P}_0$, ⋯, $\mathbf{P}_7$) to form a *uniform* third-degree *B-spline curve* using equation (3).

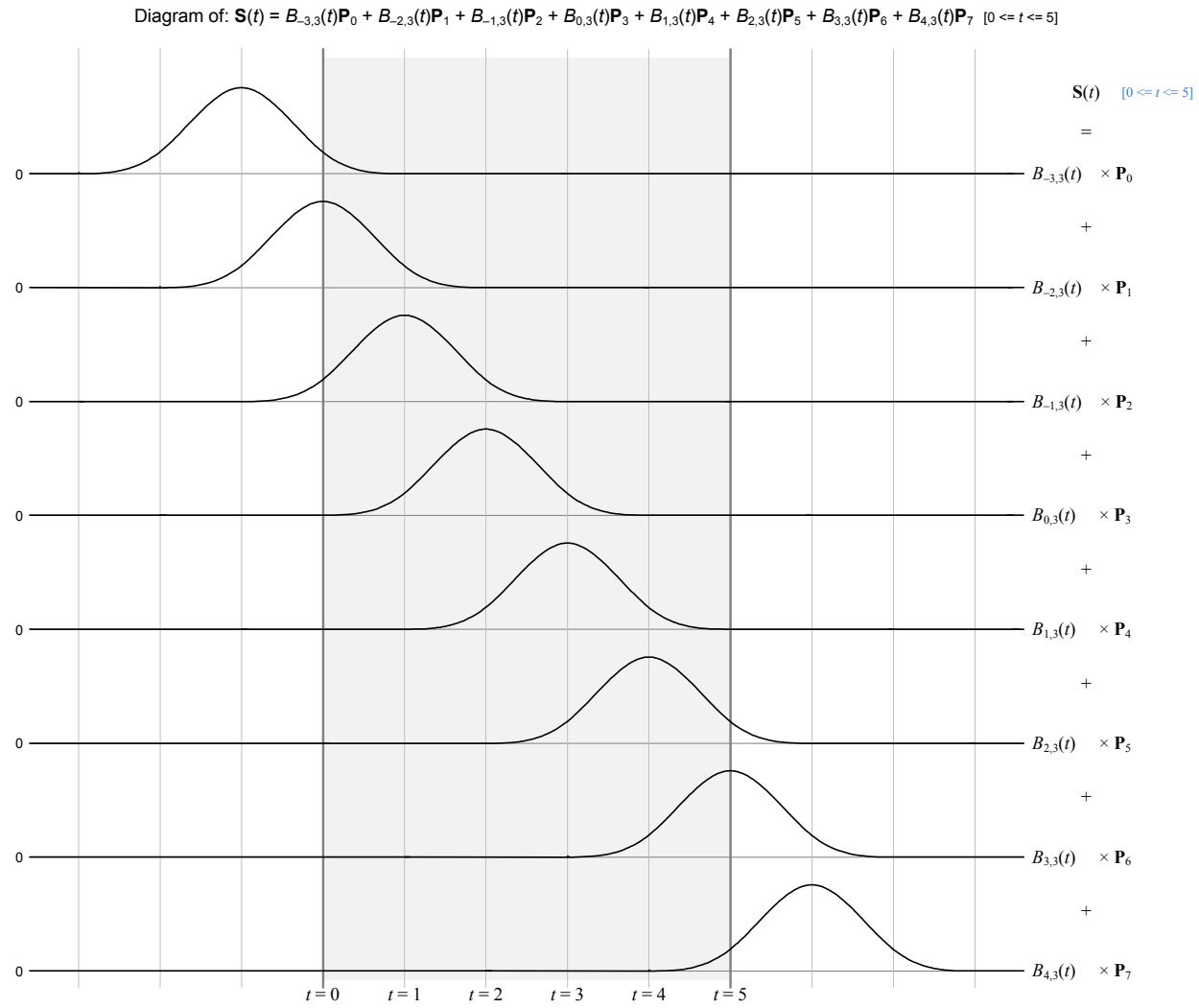

Figure 2-2

As shown in the diagram above, the various *basis functions* are the same curve shifted along the T-axis. The first shaded column, from $t = 0$ to $t = 1$, defines the first *spline segment* of the *B-spline curve*, the second column, from $t = 1$ to $t = 2$, defines the second *spline segment*, and so on. Altogether, the *B-spline curve* consists of five *spline segments*. Notice that for each *spline segment*, only four *control points* contribute towards the definition of that segment (in general, $d + 1$ points contribute); the other *control points* are multiplied by zero. So, for example, at $t = 2.5$, $\mathbf{S}(2.5) = 0\mathbf{P}_0 + 0\mathbf{P}_1 + \frac{1}{6}(0.125)\mathbf{P}_2 + \frac{1}{6}(2.875)\mathbf{P}_3 + \frac{1}{6}(2.875)\mathbf{P}_4 + \frac{1}{6}(0.125)\mathbf{P}_5 + 0\mathbf{P}_6 + 0\mathbf{P}_7$, which is equivalent to $\mathbf{S}(2.5) = \frac{1}{6}(0.125)\mathbf{P}_2 + \frac{1}{6}(2.875)\mathbf{P}_3 + \frac{1}{6}(2.875)\mathbf{P}_4 + \frac{1}{6}(0.125)\mathbf{P}_5$. Most of a large number of *control points* (relative to the degree of a *B-spline curve*) for a given t value will be multiplied by zero.

2.1.2 Blending Functions

Equation (3), or the equivalent equation (7) for a *uniform* B-spline, is sufficient to define the *B-spline curve* mathematically. However, because the *basis functions* turn out to be zero for most of a large number of *control points*, equation (3) (or equation (7)) is not an efficient way to implement a B-spline in a computer program; the multiplication of many *control points* by zero is an unnecessary overhead. The following paragraphs will present an alternate definition of a *uniform B-spline curve* suitable for implementing in a computer program.

As shown in section **2.1.1**, only $d + 1$ consecutive *control points* are needed to define each *spline segment* (d is the degree of the *B-spline curve*). Each point of a *spline segment* of a *uniform B-spline curve* is therefore a weighted combination (or "blending") of $d + 1$ consecutive *control points*; the weighting is defined by $d + 1$ functions, $\beta_1, \cdots, \beta_{d+1}$, called **blending functions**. It is desirable to have the domain of each *blending function* the same; a domain of real numbers between zero and one is preferred. Also, the $d + 1$ *blending functions* are the same for every $d + 1$ consecutive sequence of *control points*. In summary, the following is desired.

$$\mathbf{S}_{i+1}(u) = \beta_1(u)\mathbf{P}_i + \beta_2(u)\mathbf{P}_{i+1} + \cdots + \beta_{d+1}(u)\mathbf{P}_{i+d} \quad [0 \le u \le 1;\ u \in \mathbb{R}]\ [i = 0, \cdots, (n - d);\ i \in \mathbb{Z}] \qquad (15)$$

where $\mathbf{S}_{i+1}$ is the $(i+1)^{th}$ *spline segment* of a *uniform B-spline curve* of degree d with $n + 1$ *control points* (i begins with zero). The *blending functions* are derived from the *basis functions* as explained below.

Looking at **Figure 2-2**, above, notice that each *basis function* is composed of the same four one-piece curves shifted along the T-axis. Altogether, therefore, there are only four (in general, $d + 1$) distinct curves. The situation is illustrated in **Figure 2-3** below.

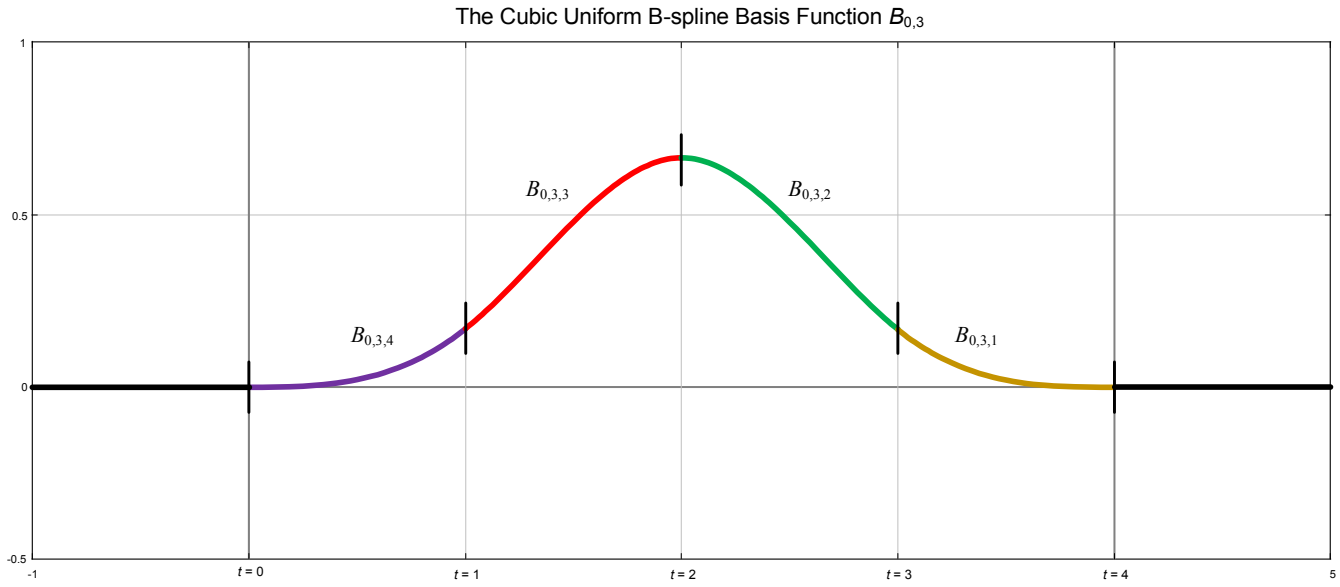

Figure 2-3

The diagram above depicts $B_{0,3}$ with its four non-zero segments, $B_{0,3,1}$, $B_{0,3,2}$, $B_{0,3,3}$, $B_{0,3,4}$, in different colours. From equation (13),

$$B_{0,3,1}(t) = \tfrac{1}{6}(-t^3 + 12t^2 - 48t + 64) = \tfrac{1}{6}(4 - t)^3 \quad [3 \le t \le 4] \qquad (16)$$

$$B_{0,3,2}(t) = \tfrac{1}{6}(3t^3 - 24t^2 + 60t - 44) \quad [2 \le t \le 3] \qquad (17)$$

$$B_{0,3,3}(t) = \tfrac{1}{6}(-3t^3 + 12t^2 - 12t + 4) \quad [1 \le t \le 2] \qquad (18)$$

$$B_{0,3,4}(t) = \tfrac{1}{6}(t^3) \quad [0 \le t \le 1] \qquad (19)$$

Note that it is desirable that the upper end of the intervals of t in the four equations above be closed at this stage. Closing the intervals is valid because the *basis functions* are continuous.

It is desired to define four *blending functions*, $\beta_1, \cdots, \beta_4$, based on the segments above (equations (16) to (19)) but with an interval between zero and one inclusive. Thus we have

$$\beta_{1,3}(u) = B_{0,3,1}(u + 3) = \tfrac{1}{6}(1 - u)^3 \quad [0 \le u \le 1] \qquad (20)$$

$$\beta_{2,3}(u) = B_{0,3,2}(u + 2) = \tfrac{1}{6}(3u^3 - 6u^2 + 4) \quad [0 \le u \le 1] \qquad (21)$$

$$\beta_{3,3}(u) = B_{0,3,3}(u + 1) = \tfrac{1}{6}(-3u^3 + 3u^2 + 3u + 1) \quad [0 \le u \le 1] \qquad (22)$$

$$\beta_{4,3}(u) = B_{0,3,4}(u) = \tfrac{1}{6}(u^3) \quad [0 \le u \le 1] \qquad (23)$$

The four *blending functions* are shown in the diagram below.

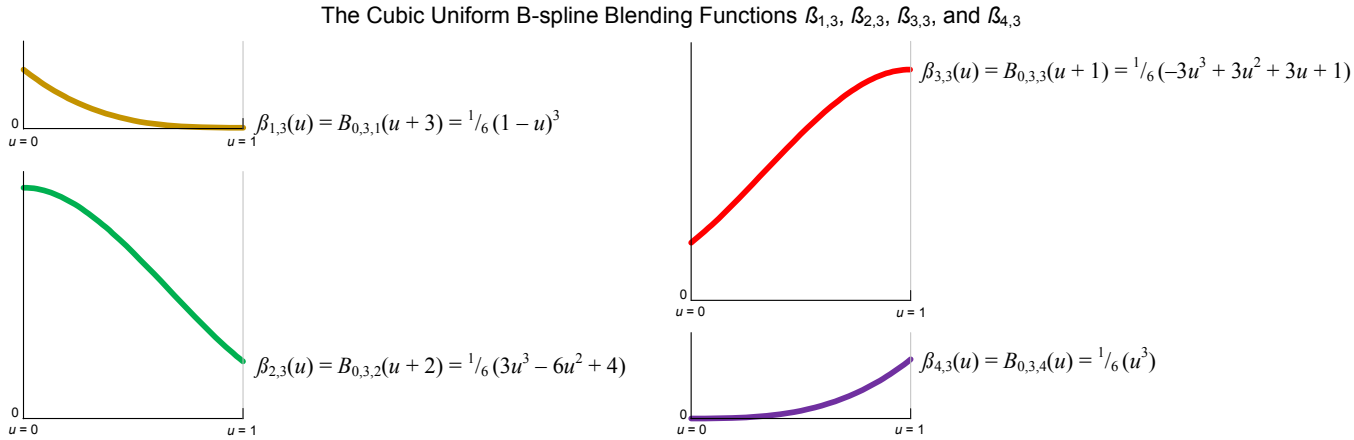

Figure 2-4

In general, the *blending functions* of a *uniform B-spline curve* of degree d are

$$\beta_{i,d}(u) = B_{0,d,i}(u + d - i + 1) \quad [0 \le u \le 1;\ u \in \mathbb{R}]\ [i = 1, \cdots, (d+1);\ i \in \mathbb{Z}] \qquad (24)$$

where $B_{0,d,i}(t) = B_{0,d}(t) \quad [d - i + 1 \le t \le d - i + 2;\ t \in \mathbb{R}]\ [i = 1, \cdots, (d+1);\ i \in \mathbb{Z}]$

In terms of *blending functions*, a *uniform* B-spline, S, of degree d is defined by

$$S = \bigcup_{i=0}^{n-d} \left\{ \mathbf{S}_{i+1}(u) : \mathbf{S}_{i+1}(u) = \beta_{1,d}(u)\mathbf{P}_i + \beta_{2,d}(u)\mathbf{P}_{i+1} + \cdots + \beta_{d+1,d}(u)\mathbf{P}_{i+d} \quad [0 \le u \le 1] \right\} \qquad (25)$$

Equation (25) can be written in matrix form. For a *uniform* third-degree B-spline, equation (25) evaluates explicitly to

$$S = \bigcup_{i=0}^{n-3} \left\{ \mathbf{S}_{i+1}(u) : \mathbf{S}_{i+1}(u) = \tfrac{1}{6}(1-u)^3\mathbf{P}_i + \tfrac{1}{6}(3u^3 - 6u^2 + 4)\mathbf{P}_{i+1} + \tfrac{1}{6}(-3u^3 + 3u^2 + 3u + 1)\mathbf{P}_{i+2} + \tfrac{1}{6}(u^3)\mathbf{P}_{i+3} \quad [0 \le u \le 1] \right\} \qquad (26)$$

which can be written in matrix form as

$$S = \bigcup_{i=0}^{n-3} \left\{ \mathbf{S}_{i+1}(u) : \mathbf{S}_{i+1}(u) = [u^3\ u^2\ u\ 1]\ \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_i \\ \mathbf{P}_{i+1} \\ \mathbf{P}_{i+2} \\ \mathbf{P}_{i+3} \end{bmatrix} \quad [0 \le u \le 1] \right\} \qquad (27)$$

Equation (26) can easily be implemented in software with minimal overheads (each value of i represents one *spline segment* of the *uniform B-spline curve*).

2.1.3 Basis vs. Blending Functions

As shown in the previous sections, the *basis functions* are not the same as the *blending functions*, even though the *blending functions* are derived from the *basis functions*. Many publications, unfortunately, confound *basis functions* and *blending functions*, regarding both terms as synonyms while failing to distinguish between the two types of functions.

The *basis functions* are suitably named as such; they are the basis of the definition of a particular type of spline. For example, B-splines are formally defined by their *basis functions*. Hermite splines are defined by their own basis functions. The *basis functions* determine the shape of the *B-spline curve* relative to its *control points*. *Basis functions* are less suited to be called "blending functions" because *blending functions* are actually mathematical weights applied to the *control points*, and the *basis functions* interpreted as weights result in zero weights for most of a large number of *control points* by definition. It is pointless having a weight which, by definition, is zero. *Basis functions* require a *knot set*. Note that the *basis functions* apply to all B-splines.

The *blending functions* are also suitably named as such; they are actually mathematical weights applied to a subsequence of *control points* that define a *curve segment* (each point of a *curve segment* is a "blend" of the subsequence of *control points*). The *control points* that are not involved in defining a particular *curve segment* are not part of the weighting, so there are no weights that are zero by definition for the *blending functions*.

Blending functions are not suitable to be called "basis functions" because the *blending functions* are derived from the *basis functions*, and so are not themselves the basis for defining different types of splines. *Blending functions* are not mathematically necessary for defining a spline. *Blending functions* do not require a *knot set*. Note that the *blending functions* apply only to *uniform* B-splines.

In any case, the two types of functions must be distinguished from each other — they are not the same set of functions and should never be referred to by the same name. In addition, *basis functions* are generally piecewise functions, and *blending functions* are generally one-piece functions.

2.1.4 Anchored Uniform B-splines

A *uniform* B-spline does not pass through any of its *control points*. However, there are situations where the B-spline is required to pass through the first and\or last *control points* ($\mathbf{P}_0$ and $\mathbf{P}_n$, respectively, where the number of *control points* is $n + 1$). Such a B-spline is called an **anchored** *uniform* B-spline because the first and\or last *knot point* of the B-spline is "anchored" to the first or last *control point*, respectively. Typically, an *anchored uniform* B-spline passes through both the first and last *control points*.

There are two ways to convert a non-*anchored uniform* B-spline to an *anchored uniform* B-spline. The first method is to alter the *knot set*; the second method is to alter the *control points*.

First Method: The *knot set* of a d^{th} degree B-spline with $n + 1$ *control points* is expressed as $\langle t_{-d}, t_{1-d}, t_{2-d}, \cdots, t_0, t_1, \cdots, t_{n+1-d}, t_{n+2-d}, \cdots, t_{n+1}\rangle$. For a *uniform* B-spline, the *knot set* will typically be $\langle -d, 1-d, 2-d, \cdots, 0, 1, \cdots, n+1-d, n+2-d, \cdots, n+1\rangle$. For example, the *knot set* of a *uniform* 3rd degree B-spline with eight *control points* ($n = 7$) is implicitly $\langle -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8\rangle$. To convert a *uniform* B-spline to an *anchored uniform* B-spline, simply replace the first d knots with zero (t_0), and the last d *knots* with *knot* values being the same as the last $(d + 1)^{\text{th}}$ *knot* (t_{n+1-d}). The new *knot set* will then be explicitly

$$\langle \underbrace{0, \cdots, 0}_{d \text{ times}}, 0, \cdots, n+1-d, \underbrace{n+1-d, \cdots, n+1-d}_{d \text{ times}} \rangle$$

For example, the *knot set* given in the example above would be altered to $\langle \mathbf{0}, \mathbf{0}, \mathbf{0}, 0, 1, 2, 3, 4, 5, \mathbf{5}, \mathbf{5}, \mathbf{5}\rangle$. Note that by altering the *knot set* as described in this method, the resulting B-spline is not a true *uniform* B-spline, and the *blending functions* no longer apply throughout. However, as long as it is understood what is meant by the term "anchored uniform B-spline" it is convenient to call it as such.

Second Method: The *knot set* of the *uniform* B-spline remains implicit. For a *uniform* d^{th} degree B-spline, the first and last *control points* are each repeated an additional $d - 1$ times. For example, if the *control points* of a *uniform* 3rd degree B-spline are $\mathbf{P}_0, \cdots, \mathbf{P}_7$, then the new *control points* will be $\mathbf{P}_0, \mathbf{P}_0, \mathbf{P}_0, \mathbf{P}_1, \cdots, \mathbf{P}_6, \mathbf{P}_7, \mathbf{P}_7, \mathbf{P}_7$. In this case, the B-spline remains a true *uniform* B-spline, and the *blending functions* apply throughout. Note that if this method is implemented in a computer program, then the first *curve segment* of the B-spline will be a single point ($\mathbf{P}_0$) repeated multiple times, and similarly for the last *curve segment*.

Either of the two methods above can also be used to create a sharp join at a *control point* by repeating a *knot* an additional d times or a *control point* an additional $d - 1$ times.

Note that the two methods separately applied to the same *control points* do not necessarily result in an identical *B-spline curve* when anchoring the end points or creating a sharp join at a *control point*. For example, anchoring the end points of a *uniform* B-spline using the first method (repeated *knots*) does not necessarily result in an identical *B-spline curve* as when using the second method (repeated end points) on the same *uniform* B-spline.

It may be useful to note that anchoring the end points of a *uniform* 3rd degree B-spline using the second method (repeated end points) results in a spline curve identical to an anchored Bezier spline that has continuity $C^{(2)}$ (the end points are not repeated for the Bezier spline); using the first method (repeated *knots*) on the *uniform* B-spline does not result in an identical spline curve. Also, an *anchored uniform* 3rd degree B-spline with four *control points* using the first anchoring method results in a 3rd degree Bezier curve (with the same four *control points*); using the second anchoring method does not typically produce a Bezier curve.

2.2 Joining Separate B-splines

Separate B-splines of a given degree can be joined together to form one continuous spline of the same degree in various ways. When a B-spline is joined to a given B-spline, some of the *control points* and *knots* of the joining B-spline are modified in a certain way, but the given B-spline remains unmodified.

When joining B-splines, a number of *control points* of the given B-spline is added to the beginning or end of the joining B-spline. The number of *control points* added is equal to the degree of the B-splines. The *knots* corresponding to the added *control points* are also added to the joining B-spline. Some of the original *knots* of the joining B-spline may need to be altered, resulting in the original shape of the joining *B-spline curve* potentially being altered near the joined end.

There are generally three methods for joining B-splines together. **(1)** The end of the joining B-spline is connected to the beginning of the given B-spline; **(2)** the beginning of the joining B-spline is connected to the end of the given B-spline; **(3)** the joining B-spline is connected to two given B-splines. The three methods are described in the following sections (d is the degree of the B-splines).

2.2.1 Joining the End of a B-spline to Another

This method extends the end of a joining B-spline, $\mathbf{Q}_0, \cdots, \mathbf{Q}_m$, to connect to the first *knot point* of a given B-spline, $\mathbf{P}_0, \cdots, \mathbf{P}_n$. The *control points* of the joining B-spline are modified to $\mathbf{Q}_0, \cdots, \mathbf{Q}_m, \mathbf{P}_0, \cdots, \mathbf{P}_{d-1}$. The given B-spline is unmodified.

Given B-spline:	$\mathbf{P}_0, \cdots, \mathbf{P}_n$	Knot set: $\langle p_0, \cdots, p_{n+d+1} \rangle$
B-spline to be joined:	$\mathbf{Q}_0, \cdots, \mathbf{Q}_m$	Knot set: $\langle q_0, \cdots, q_{m+d+1} \rangle$
Joined B-spline:	$\mathbf{Q}_0, \cdots, \mathbf{Q}_m, \mathbf{P}_0, \cdots, \mathbf{P}_{d-1}$	Knot set: $\langle q_0, \cdots, q_{m+1}, s_0, \cdots, s_{2d-1} \rangle$ where $s_i = p_{i+1} + (q_{m+2} - p_1) \quad [i = 0, \cdots, 2d-1]$

2.2.2 Joining the Beginning of a B-spline to Another

This method extends the beginning of a joining B-spline, $\mathbf{Q}_0, \cdots, \mathbf{Q}_m$, to connect to the last *knot point* of a given B-spline, $\mathbf{P}_0, \cdots, \mathbf{P}_n$. The *control points* of the joining B-spline are modified to $\mathbf{P}_{n-d+1}, \cdots, \mathbf{P}_n, \mathbf{Q}_0, \cdots, \mathbf{Q}_m$. The given B-spline is unmodified.

Given B-spline:	$\mathbf{P}_0, \cdots, \mathbf{P}_n$	Knot set: $\langle p_0, \cdots, p_{n+d+1} \rangle$
B-spline to be joined:	$\mathbf{Q}_0, \cdots, \mathbf{Q}_m$	Knot set: $\langle q_0, \cdots, q_{m+d+1} \rangle$
Joined B-spline:	$\mathbf{P}_{n-d+1}, \cdots, \mathbf{P}_n, \mathbf{Q}_0, \cdots, \mathbf{Q}_m$	Knot set: $\langle s_0, \cdots, s_{2d-1}, q_d, \cdots, q_{m+d+1} \rangle$ where $s_i = p_{i+n-d+1} + (q_{d-1} - p_{n+d}) \quad [i = 0, \cdots, 2d-1]$

2.2.3 Joining Two B-splines by Another

In this method, a new B-spline, $\mathbf{P}_{n-d+1}, \cdots, \mathbf{P}_n, *, \mathbf{Q}_0, \cdots, \mathbf{Q}_{d-1}$, is created which connects the last *knot point* of a given B-spline, $\mathbf{P}_0, \cdots, \mathbf{P}_n$, to the first *knot point* of another given B-spline, $\mathbf{Q}_0, \cdots, \mathbf{Q}_m$. * represents possibly new arbitrary control points, $\mathbf{R}_0, \cdots, \mathbf{R}_{d-3}$ $[d \geq 3]$. The two given B-splines are unmodified.

Given B-spline:	$\mathbf{P}_0, \cdots, \mathbf{P}_n$	Knot set: $\langle p_0, \cdots, p_{n+d+1} \rangle$
Given B-spline:	$\mathbf{Q}_0, \cdots, \mathbf{Q}_m$	Knot set: $\langle q_0, \cdots, q_{m+d+1} \rangle$
Joining B-spline:	$\mathbf{P}_{n-d+1}, \cdots, \mathbf{P}_n, *, \mathbf{Q}_0, \cdots, \mathbf{Q}_{d-1}$ where * is $\mathbf{R}_0, \cdots, \mathbf{R}_{d-3}$ if $d \geq 3$	Knot set: $\langle p_{n-d+1}, \cdots, p_{n+d}, s_0, \cdots, s_{2(d-1)} \rangle$ where $s_i = q_{i+2} + (p_{n+d} - q_1) \quad [i = 0, \cdots, 2(d-1)]$

If the degree, d, of the B-splines is greater than or equal to three then new arbitrary *control points*, $\mathbf{R}_0, \cdots, \mathbf{R}_{d-3}$, need to be inserted at * above, otherwise no arbitrary *control points* are inserted. Also, if the B-splines are *uniform*, and have implicit *knot sets*, then * is ignored and no arbitrary *control points* are inserted.

2.2.4 Creating a Closed B-spline

An open B-spline (a B-spline forming an open curve) can be closed (the ends of the B-spline are joined, thereby forming a closed curve) by letting m be identical to n, and letting $\mathbf{Q}_i$ be identical to $\mathbf{P}_i$ $[i = 0, \cdots, n]$ (thus only one B-spline is initially involved) in the three methods presented in the sections above.

2.2.5 Merging B-splines Together

Two or more B-splines can be merged into a given B-spline by using any of the three methods presented in the sections above. The two or more B-splines to merge would be the joining B-splines in the first two methods. For the third method, new joining B-splines need to be created. Any one of the joining B-splines can be joined to one end of the given B-spline, or to the middle (beginning of any intermediate *curve segment*) of the given B-spline. After joining, the original B-spline will branch into two or more B-splines.

To join a B-spline to the middle of a given B-spline, the *control points* and *knots* of the given B-spline not involved in the join are ignored. For example, if the given B-spline has *control points* $\mathbf{P}_0, \cdots, \mathbf{P}_k, \cdots, \mathbf{P}_n$, with *knot set* $\langle p_0, \cdots, p_k, \cdots, p_{n+d+1}\rangle$, and a joining B-spline is to be joined to the *curve segment* corresponding to the *control points* $\mathbf{P}_k, \cdots, \mathbf{P}_{k+d}$, then the *control points* $\mathbf{P}_0, \cdots, \mathbf{P}_{k-1}$, and *knots* $\langle p_0, \cdots, p_{k-1}\rangle$ can be ignored, and the *control points* $\mathbf{P}_k, \cdots, \mathbf{P}_n$, and *knots* $\langle p_k, \cdots, p_{n+d+1}\rangle$ of the given B-spline can be treated as *control points* $\mathbf{P}_0, \cdots, \mathbf{P}_{n-k}$ and *knots* $\langle p_0, \cdots, p_{n+d-k+1}\rangle$ in the methods presented in the sections above.

2.2.6 Joining Uniform B-splines

Uniform B-splines with implicit *knot sets* can be joined by any of the methods presented in the sections above by ignoring the *knot sets* presented in those methods. For example, to join the *uniform* B-spline, Q, with *control points* $\mathbf{Q}_0, \cdots, \mathbf{Q}_m$ to the given *uniform* B-spline defined by the *control points* $\mathbf{P}_0, \cdots, \mathbf{P}_n$, simply modify Q to $\mathbf{Q}_0, \cdots, \mathbf{Q}_m, \mathbf{P}_0, \cdots, \mathbf{P}_{d-1}$. To close a *uniform* B-spline defined by *control points* $\mathbf{P}_0, \cdots, \mathbf{P}_n$, modify the *control points* to $\mathbf{P}_0, \cdots, \mathbf{P}_n, \mathbf{P}_0, \cdots, \mathbf{P}_{d-1}$.

2.3 Separating B-splines

A single B-spline is easily separated into two independent B-splines, one with *control points* $\mathbf{P}_0, \cdots, \mathbf{P}_n$, and the other with *control points* $\mathbf{Q}_0, \cdots, \mathbf{Q}_m$. The method is as follows.

Given B-spline:	$\mathbf{P}_0, \cdots, \mathbf{P}_n, \mathbf{Q}_0, \cdots, \mathbf{Q}_m$	Knot set: $\langle p_0, \cdots, p_n, q_0, \cdots, q_{m+d+1}\rangle$
Separate B-spline 1:	$\mathbf{P}_0, \cdots, \mathbf{P}_n$	Knot set: $\langle p_0, \cdots, p_n, q_0, \cdots, q_d\rangle$
Separate B-spline 2:	$\mathbf{Q}_0, \cdots, \mathbf{Q}_m$	Knot set: $\langle q_0, \cdots, q_{m+d+1}\rangle$

3. B-splines in Terms of Bezier Control Points

The *control points* and *knot set* of a B-spline can be converted to the *control points* of a Bezier spline resulting in exactly the same *B-spline curve*. Such a conversion is useful for implementing B-splines in terms of Bezier *control points* for systems that implement only Bezier curves at the pixel level. In this document, a "Bezier curve" is a one-piece curve, not a piecewise curve; a "Bezier spline" consists of joined Bezier curves.

A **poly-bezier** is a sequence of points, where each adjacent group of points defines a Bezier curve, and only one point exists for the common end points of each adjacent pair of Bezier curves. For example, the points $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$, $\mathbf{P}_4$, $\mathbf{P}_5$, $\mathbf{P}_6$ define a 3rd degree *poly-bezier* consisting of two Bezier curves, the first of which is defined by the points $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$, and the second is defined by the points $\mathbf{P}_3$, $\mathbf{P}_4$, $\mathbf{P}_5$, $\mathbf{P}_6$. Notice that the common end point, $\mathbf{P}_3$, of the pair of Bezier curves is present only once in the sequence of the *poly-bezier* points. The *poly-bezier* points are the *control points* of a general Bezier spline.

3.1 Third Degree Uniform B-spline to Bezier

The following paragraphs show how to convert the *control points* of a *uniform* 3rd degree B-spline to the points of a 3rd degree *poly-bezier*. To develop the method, the matrix forms of a Bezier curve and a *spline segment* are used. The matrix form of one *spline segment* of a *uniform* 3rd degree B-spline (see equation (27)) is

$$\mathbf{S}(t) = \boldsymbol{T}(t)\boldsymbol{N}_3\boldsymbol{P} \quad [0 \le t \le 1] \qquad (28)$$

which is expanded to

$$\mathbf{S}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} \quad [0 \le t \le 1] \qquad (29)$$

Also, a 3rd degree Bezier curve can be represented in matrix form as

$$\mathbf{B}(t) = \boldsymbol{T}(t)\boldsymbol{M}_3\boldsymbol{B} \quad [0 \le t \le 1] \qquad (30)$$

which is expanded to

$$\mathbf{B}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{B}_0 \\ \mathbf{B}_1 \\ \mathbf{B}_2 \\ \mathbf{B}_3 \end{bmatrix} \quad [0 \le t \le 1] \qquad (31)$$

Equation (28) can be written as

$$\mathbf{S}(t) = \boldsymbol{T}(t)\boldsymbol{M}_3\boldsymbol{M}_3^{-1}\boldsymbol{N}_3\boldsymbol{P} = \boldsymbol{T}(t)\boldsymbol{M}_3(\boldsymbol{M}_3^{-1}\boldsymbol{N}_3\boldsymbol{P}) \quad [0 \le t \le 1] \qquad (32)$$

where $\boldsymbol{M}_3^{-1}$ is the inverse matrix of $\boldsymbol{M}_3$, which is

$$\boldsymbol{M}_3^{-1} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{3} & 1 \\ 0 & \frac{1}{3} & \frac{2}{3} & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad (33)$$

If we let

$$\boldsymbol{B} = \boldsymbol{M}_3^{-1}\boldsymbol{N}_3\boldsymbol{P} \qquad (34)$$

then equation (32) can be written as

$$\mathbf{S}(t) = \boldsymbol{T}(t)\boldsymbol{M}_3\boldsymbol{B} \quad [0 \le t \le 1] \qquad (35)$$

which is in the same form as equation (30).

Thus, equation (34) specifies the formula to convert the *control points* (***P***) of a *uniform* B-spline to the points (***B***) of a *poly-bezier* for a 3rd degree *B-spline curve*.

Noting that

$$\boldsymbol{M}_3^{-1}\boldsymbol{N}_3 = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix} \qquad (36)$$

To convert a *uniform* 3rd degree B-spline with $n + 1$ *control points* ($\mathbf{P}_i$) to a 3rd degree *poly-bezier* with $3n - 5$ points ($\mathbf{B}_i$), the following equation is used.

$$\begin{bmatrix} \mathbf{B}_{3i} \\ \begin{bmatrix} \mathbf{B}_{3i+1} \\ \mathbf{B}_{3i+2} \\ \mathbf{B}_{3i+3} \end{bmatrix} \end{bmatrix} = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & \begin{bmatrix} 4 & 2 & 0 \\ 2 & 4 & 0 \\ 1 & 4 & 1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \mathbf{P}_i \\ \begin{bmatrix} \mathbf{P}_{i+1} \\ \mathbf{P}_{i+2} \\ \mathbf{P}_{i+3} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 \\ 0 & \begin{bmatrix} \frac{2}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{2}{3} & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \mathbf{P}_i \\ \begin{bmatrix} \mathbf{P}_{i+1} \\ \mathbf{P}_{i+2} \\ \mathbf{P}_{i+3} \end{bmatrix} \end{bmatrix} \quad [i = 0, \cdots, (n-3)] \qquad (37)$$

In the equation above, the outer matrix is used if $i = 0$; the inner matrix is used if $i > 0$.

For an example of using equation (37), the *poly-bezier* points ($\mathbf{B}_0, \cdots, \mathbf{B}_6$) calculated from five *uniform* B-spline *control points* ($\mathbf{P}_0, \cdots, \mathbf{P}_4$) for a B-spline of third degree (where $n = 4$) are

$\mathbf{B}_0 = \frac{1}{6}\mathbf{P}_0 + \frac{2}{3}\mathbf{P}_1 + \frac{1}{6}\mathbf{P}_2$

$\mathbf{B}_1 = \frac{2}{3}\mathbf{P}_1 + \frac{1}{3}\mathbf{P}_2$

$\mathbf{B}_2 = \frac{1}{3}\mathbf{P}_1 + \frac{2}{3}\mathbf{P}_2$

$\mathbf{B}_3 = \frac{1}{6}\mathbf{P}_1 + \frac{2}{3}\mathbf{P}_2 + \frac{1}{6}\mathbf{P}_3$

$\mathbf{B}_4 = \frac{2}{3}\mathbf{P}_2 + \frac{1}{3}\mathbf{P}_3$

$\mathbf{B}_5 = \frac{1}{3}\mathbf{P}_2 + \frac{2}{3}\mathbf{P}_3$

$\mathbf{B}_6 = \frac{1}{6}\mathbf{P}_2 + \frac{2}{3}\mathbf{P}_3 + \frac{1}{6}\mathbf{P}_4$

3.2 General Uniform B-spline to Bezier

In general, for a *uniform* B-spline of degree d, equation (34) would be

$$\boldsymbol{B} = \boldsymbol{M}_d^{-1}\boldsymbol{N}_d\boldsymbol{P} \qquad (38)$$

which expands to

$$\begin{bmatrix} \mathbf{B}_{di} \\ \begin{bmatrix} \mathbf{B}_{di+1} \\ \mathbf{B}_{di+2} \\ \vdots \\ \mathbf{B}_{di+d} \end{bmatrix} \end{bmatrix} = \boldsymbol{M}_d^{-1}\boldsymbol{N}_d \begin{bmatrix} \mathbf{P}_i \\ \begin{bmatrix} \mathbf{P}_{i+1} \\ \mathbf{P}_{i+2} \\ \vdots \\ \mathbf{P}_{i+d} \end{bmatrix} \end{bmatrix} \quad [i = 0, \cdots, (n-d)] \qquad (39)$$

The outer matrix is used if $i = 0$; the inner matrix is used if $i > 0$. $\boldsymbol{M}_d^{-1}$ is the inverse of the matrix of a d^{th} degree Bezier curve (see **Matrix Form of a Bezier Curve** under **B.1 Bezier Curves** of **Appendix A: Bezier Splines** to calculate $\boldsymbol{M}_d$). The inner matrix of $\boldsymbol{M}_d^{-1}$ begins at the second row and second column. $\boldsymbol{N}_d$ is the matrix form of one *spline segment* of a *uniform* B-spline of degree d.

3.3 Non-uniform B-spline to Bezier

Converting *non-uniform* B-spline *control points* to Bezier *control points* is much more difficult than the method presented above for *uniform* B-splines. The method of inserting duplicate *control points* (via "Boehm's knot insertion algorithm) to obtain Bezier *control points* seems to fail in practice.

To convert a 3rd degree B-spline (not necessarily *uniform*) with $n + 1$ *control points* ($\mathbf{P}_i$) and *knots* $\langle t_0, \cdots, t_{n+4}\rangle$ to a 3rd degree *poly-bezier* with $3n - 5$ points ($\mathbf{B}_i$), the following equations can be used.

$$\mathbf{B}_{3i} = \frac{\Delta_{3,2}}{\Delta_{4,2}}\mathbf{B}_{3i+1} + \left(1 - \frac{\Delta_{3,2}}{\Delta_{4,2}}\right)\left(\frac{\Delta_{3,1}}{\Delta_{4,1}}\mathbf{P}_{i+1} + \left(1 - \frac{\Delta_{3,1}}{\Delta_{4,1}}\right)\mathbf{P}_i\right) \qquad (40a)$$

$$\mathbf{B}_{3i+1} = \frac{\Delta_{3,2}}{\Delta_{5,2}}\mathbf{P}_{i+2} + \left(1 - \frac{\Delta_{3,2}}{\Delta_{5,2}}\right)\mathbf{P}_{i+1} \qquad (40b)$$

$$\mathbf{B}_{3i+2} = \frac{\Delta_{4,2}}{\Delta_{5,2}}\mathbf{P}_{i+2} + \left(1 - \frac{\Delta_{4,2}}{\Delta_{5,2}}\right)\mathbf{P}_{i+1} \qquad (40c)$$

$$\mathbf{B}_{3i+3} = \frac{\Delta_{4,3}}{\Delta_{5,3}}\left(\frac{\Delta_{4,3}}{\Delta_{6,3}}\mathbf{P}_{i+3} + \left(1 - \frac{\Delta_{4,3}}{\Delta_{6,3}}\right)\mathbf{P}_{i+2}\right) + \left(1 - \frac{\Delta_{4,3}}{\Delta_{5,3}}\right)\mathbf{B}_{3i+2} \qquad (40d)$$

for $i = 0, \cdots, (n-3)$, where $\Delta_{a,b} =_{\text{df}} t_{i+a} - t_{i+b}$. Also, $\frac{\Delta_{a,b}}{\Delta_{c,d}}$ is deemed to be zero if $\Delta_{c,d} = 0$. Equation (40a) is used only when $i = 0$. Equations (40) are derived from the so-called "polar form" of the *control points* of a 3rd degree B-spline. Notice that for a *uniform* 3rd degree B-spline, equations (40) are identical to the expanded equation (37) ($\Delta_{a,b} =_{\text{df}} a - b$ for a *uniform* B-spline).

To convert the *control points* of any degree B-spline to Bezier *control points*, the methods involving the polar form of *control points*, introduced by Dr. Lyle Ramshaw, can be used (presented in the next section).

3.4 Polar Forms

The following is only a summary of the concept of polar forms (introduced by Dr. Lyle Ramshaw) of *control points* for B-splines and Bezier splines. Other publications on polar forms need to be referred to for more information. Using polar forms involves a heuristic method of converting a given sequence of B-spline *control points* to another sequence of points, given a (modified) *knot set*. Polar forms are typically used to convert the *control points* of any degree B-spline to Bezier *control points*. The term "polar forms" is unfortunate, since it has nothing to do with polar coordinates or complex numbers.

With the polar forms system, each *control point* of a B-spline or Bezier spline of degree d is associated with a d-tuple of *knots* called a **polar value**. If the *knot set* of a B-spline is represented by $\langle t_{-1}, t_0, \cdots, t_{n+d-1}, t_{n+d}\rangle$ (where $t_i \le t_{i+1}\ \ [i = 0, \cdots, n + d - 2]$), then the following four rules apply. In the polar forms system, knots t_{-1} and t_{n+d} are ignored because they do not partake in the resulting *B-spline curve*, therefore, a *knot set* will be understood to be represented as $\langle t_0, \cdots, t_{n+d-1}\rangle$ in this section (such a *knot set* will be called a **trimmed knot set** in this document).

The Four Rules of Polar Forms

1. The $d + 1$ *control points* ($\mathbf{B}_i\ [i = 0, \cdots, d]$) of a degree d Bezier curve between the interval $\mathbf{[}t_k, t_{k+1}\mathbf{]}$ of the *trimmed knot set* of a given B-spline are associated with *polar values*, $(\mathbf{\cdots})_\text{p}$, as follows.

   $$\mathbf{B}_i \equiv (u_1, \cdots, u_d)_\text{p}\ \ [i = 0, \cdots, d]$$

   where

   $$u_j = \begin{cases} t_k & [j \le d - i] \\ t_{k+1} & [\text{otherwise}] \end{cases}\ \ [j = 1, \cdots, d]$$

   The corresponding pair $\mathbf{B}_i$ and $(u_1, \cdots, u_d)_\text{p}$ can be denoted by $\mathbf{B}_i{:}(u_1, \cdots, u_d)_\text{p}$ (the subscript "p" outside the parentheses means "*polar value*").

   For example, the four *control point**polar value* pairs of a 3rd degree Bezier curve between the interval $\mathbf{[}3.5, 4.2\mathbf{]}$ of a B-spline *trimmed knot set* are denoted by:
   $\mathbf{B}_0{:}(3.5, 3.5, 3.5)_\text{p}$, $\mathbf{B}_1{:}(3.5, 3.5, 4.2)_\text{p}$, $\mathbf{B}_2{:}(3.5, 4.2, 4.2)_\text{p}$, $\mathbf{B}_3{:}(4.2, 4.2, 4.2)_\text{p}$.

2. The $n + 1$ *control points* ($\mathbf{P}_i$) of a degree d B-spline with *trimmed knot set* $\langle t_0, \cdots, t_{n+d-1}\rangle$ are associated with *polar values* as follows.

   $$\mathbf{P}_i \equiv (t_i, \cdots, t_{i+d-1})_\text{p}\ \ [i = 0, \cdots, n]$$

   For example, the *control point**polar value* pairs of a 3rd degree B-spline with *trimmed knot set* $\langle 2, 2.4, 3, 4, 5.5, 6.1\rangle$ are: $\mathbf{P}_0{:}(2, 2.4, 3)_\text{p}$, $\mathbf{P}_1{:}(2.4, 3, 4)_\text{p}$, $\mathbf{P}_2{:}(3, 4, 5.5)_\text{p}$, $\mathbf{P}_3{:}(4, 5.5, 6.1)_\text{p}$.

3. The elements of a *polar value* are symmetric

   $$(\text{perm}_1(u_0, \cdots, u_{d-1}))_\text{p} \equiv (\text{perm}_2(u_0, \cdots, u_{d-1}))_\text{p}$$

   where $\text{perm}(u_0, \cdots, u_{d-1})$ is any arbitrary permutation of $u_0, \cdots, u_{d-1}$.

   For example $(2, 2.4, 3, 7)_\text{p} \equiv (2.4, 3, 7, 2)_\text{p} \equiv (7, 2.4, 3, 2)_\text{p}$.

4. The following equation is used to compute a new *polar value* $(u_0, \cdots, u_{d-2}, u)_\text{p}$ for any desired value of u from two existing *polar values*, $(u_0, \cdots, u_{d-2}, u_b)_\text{p}$ and $(u_0, \cdots, u_{d-2}, u_a)_\text{p}$, and their respective associated points $\mathbf{P}_i$ and $\mathbf{P}_j$. The new *polar value* represents a new point $\mathbf{P}$.

   $$\mathbf{P}{:}(u_0, \cdots, u_{d-2}, u)_\text{p} = \alpha\mathbf{P}_i{:}(u_0, \cdots, u_{d-2}, u_b)_\text{p} + (1 - \alpha)\mathbf{P}_j{:}(u_0, \cdots, u_{d-2}, u_a)_\text{p}$$

   where

   $$\alpha = \begin{cases} \dfrac{u - u_a}{u_b - u_a} & [u_b - u_a \ne 0] \\ 0 & [\text{otherwise}] \end{cases}$$

   Note that $d - 1$ elements of the three *polar values* must be identical, but the d elements can be in any order.

   For example, given $\mathbf{P}_0{:}(2, 2.4, 3)_\text{p}$ and $\mathbf{P}_1{:}(2.4, 3, 4)_\text{p}$, along with a desired 3.7, then a new point, $\mathbf{P}$, is calculated by $\mathbf{P}{:}(3.7, 2.4, 3)_\text{p} = \alpha\mathbf{P}_1{:}(2.4, 3, 4)_\text{p} + (1 - \alpha)\mathbf{P}_0{:}(2, 2.4, 3)_\text{p}$ where $\alpha = (3.7 - 2) / (4 - 2) = 0.85$. In other words, the new point, $\mathbf{P}$, associated with the *polar value* $(3.7, 2.4, 3)_\text{p}$, is defined by $\mathbf{P} = 0.85\mathbf{P}_1 + 0.15\mathbf{P}_0$. The new point $\mathbf{P}$ is at a distance 85% along the line segment from $\mathbf{P}_0$ to $\mathbf{P}_1$. Notice that the two elements, 2.4 and 3, are common to the three *polar values*.

Among other purposes, polar forms can be used to convert B-spline *control points* to Bezier *control points* of the same degree spline. One *spline segment* is converted at a time. The following example shows how to determine the Bezier *control points*, $\mathbf{B}_0, \cdots, \mathbf{B}_3$, of one *spline segment* from the *control points* of a B-spline using the four rules stated above.

Example

Degree of B-spline and Bezier curve: 3

Given *control points* of B-spline: $\mathbf{P}_0$, ⋯, $\mathbf{P}_6$

Given *trimmed knot set*: ⟨0, 1.5, 4, 5, 5.5, 6, 7, 7.4, 12⟩

Spline segment intervals: **[**4, 5**)**, **[**5, 5.5**)**, **[**5.5, 6**)**, **[**6, 7**)**

Calculated *polar values* of B-spline: $\mathbf{P}_0$:$(0, 1.5, 4)_p$, $\mathbf{P}_1$:$(1.5, 4, 5)_p$, $\mathbf{P}_2$:$(4, 5, 5.5)_p$, $\mathbf{P}_3$:$(5, 5.5, 6)_p$, $\mathbf{P}_4$:$(5.5, 6, 7.0)_p$, $\mathbf{P}_5$:$(6, 7, 7.4)_p$, $\mathbf{P}_6$:$(7, 7.4, 12)_p$

Desired *spline segment* interval for calculating Bezier *control points*: **[**5, 5.5**]**

Calculated *polar values* of Bezier curve of desired *spline segment* interval: $\mathbf{B}_0$:$(5, 5, 5)_p$, $\mathbf{B}_1$:$(5, 5, 5.5)_p$, $\mathbf{B}_2$:$(5, 5.5, 5.5)_p$, $\mathbf{B}_3$:$(5.5, 5.5, 5.5)_p$

Calculation of $\mathbf{B}_1$: $\mathbf{B}_1$:$(5, 5, 5.5)_p = 0.5\mathbf{P}_3$:$(5, 5.5, 6)_p + (1 - 0.5)\mathbf{P}_2$:$(4, 5, 5.5)_p$

Calculation of $\mathbf{B}_2$: $\mathbf{B}_2$:$(5, 5.5, 5.5)_p = 0.75\mathbf{P}_3$:$(5, 5.5, 6)_p + (1 - 0.75)\mathbf{P}_2$:$(4, 5, 5.5)_p$

Calculation of temporary point $\mathbf{Q}_0$: $\mathbf{Q}_0$:$(4, 5, 5) = 0.875\mathbf{P}_2$:$(4, 5, 5.5)_p + (1 - 0.875)\mathbf{P}_1$:$(1.5, 4, 5)_p$

Calculation of $\mathbf{B}_0$: $\mathbf{B}_0$:$(5, 5, 5)_p = \frac{2}{3}\mathbf{B}_1$:$(5, 5, 5.5)_p + (1 - \frac{2}{3})\mathbf{Q}_0$:$(4, 5, 5)_p$

Calculation of temporary point $\mathbf{Q}_1$: $\mathbf{Q}_1$:$(5.5, 5.5, 6)_p = 0.25\mathbf{P}_4$:$(5.5, 6, 7)_p + (1 - 0.25)\mathbf{P}_3$:$(5, 5.5, 6)_p$

Calculation of $\mathbf{B}_3$: $\mathbf{B}_3$:$(5.5, 5.5, 5.5)_p = 0.5\mathbf{Q}_1$:$(5.5, 5.5, 6)_p + (1 - 0.5)\mathbf{B}_2$:$(5, 5.5, 5.5)_p$

Calculated $\mathbf{B}_0$: $\mathbf{B}_0 = \frac{2}{3}\mathbf{B}_1 + \frac{1}{3}(0.875\mathbf{P}_2 + 0.125)\mathbf{P}_1$

Calculated $\mathbf{B}_1$: $\mathbf{B}_1 = 0.5\mathbf{P}_3 + 0.5\mathbf{P}_2$

Calculated $\mathbf{B}_2$: $\mathbf{B}_2 = 0.75\mathbf{P}_3 + 0.25\mathbf{P}_2$

Calculated $\mathbf{B}_3$: $\mathbf{B}_3 = 0.125\mathbf{P}_4 + 0.375\mathbf{P}_3 + 0.5\mathbf{B}_2$

A similar procedure to the above can be used to obtain the four Bezier *control points* for each of the other *spline segment* intervals. As the degree of the B-spline increases, the calculations tend to be more complicated with more temporary point calculations.

The calculations for the Bezier *control points* follow the same pattern for each *spline segment*. Therefore, to develop a general formula for the Bezier *control points* for a particular degree B-spline, variable symbols, t_i, can be used for the *trimmed knot set* and *polar values* in the calculations involving one *spline segment* — the first *spline segment* interval is **[**t_{d-1}, t_d**)**. Equations (40) were developed in the said manner.

4. Emulating Curves Using B-splines

In software, a curve is typically emulated by a poly-line (joined straight line segments). A more accurate and smoother alternative can be achieved by using a cubic (3rd degree) B-spline. This method requires a sequence of sample points of the curve desired to be emulated, and results in a sequence of *control points* of a cubic B-spline (which can then be easily converted to the *control points* of a cubic Bezier spline (*poly-bezier*) if required).

4.1 Near Interpolation

The ideal way to use a cubic B-spline to emulate a curve is to use the sample points of the curve as the *control points* of an interpolated cubic B-spline. But, the calculations for emulating a given curve using an interpolated cubic B-spline on the sample points are an unacceptable overhead compared with emulating the same curve using a poly-line on the same sample points. However, there is a simple method to accurately approximate an interpolated cubic B-spline through given sample points of a curve, requiring far less overhead than a true interpolated cubic B-spline through the same sample points, but only slightly more overhead than a poly-line through those points. The method can be referred to as "**near interpolation**" in this document. (Recall that for an interpolated B-spline, the *knot points* are identical to the *control points*.)

The method of *near interpolation* is described as follows. The idea is to calculate the *control points* of a *uniform* B-spline approximation from a sequence of sample points of a given curve, such that the calculated *uniform B-spline curve* closely approximates a hypothetical *uniform* cubic *B-spline curve* that interpolates those sample points. For example, if the sample points are $\mathbf{S}_0$, ⋯, $\mathbf{S}_n$, then this method calculates *control points* $\mathbf{P}_0$, ⋯, $\mathbf{P}_n$, which are the *control points* of a *uniform* cubic B-spline approximation (ie: the B-spline does not necessarily go through the sample points) such that the said B-spline very closely approximates a hypothetical true interpolated *uniform* cubic *B-spline curve* through the same sample points with far less overheads than are required for calculating the true interpolated B-spline. The calculated *control points* $\mathbf{P}_0$, ⋯, $\mathbf{P}_n$ are then used in a computer program that would render the cubic B-spline approximation at the pixel level, thus producing a smooth curve approximation of the sample points at any zoom level (using a poly-line, instead, to interpolate the sample points can result in visible joined straight line segments at high zoom levels). If the computer program renders only Bezier splines (*poly-beziers*) at the pixel level, then the points $\mathbf{P}_0$, ⋯, $\mathbf{P}_n$ can be converted to Bezier points via equation (37).

The equation below calculates the *control points* ($\mathbf{P}_0$, ⋯, $\mathbf{P}_n$) of a *uniform* 3rd degree *B-spline curve* from sample points ($\mathbf{S}_0$, ⋯, $\mathbf{S}_n$). The *B-spline curve* is a *near interpolation* of the sample points.

$$\mathbf{P}_i = \begin{cases} k\mathbf{S}_i + \dfrac{1-k}{2}(\mathbf{S}_{i-1} + \mathbf{S}_{i+1}) & [\mathbf{S}_i \neq \mathbf{S}_{i-1} \text{ and } \mathbf{S}_i \neq \mathbf{S}_{i+1}] \\ \mathbf{S}_i & [\text{otherwise}] \end{cases} \quad [i = 1, \cdots, (n-1)] \qquad (41)$$

$$\mathbf{P}_0 = \mathbf{S}_0 \text{ and } \mathbf{P}_n = \mathbf{S}_n$$

where k determines the accuracy of the *near interpolated* B-spline with respect to a hypothetical true interpolated *uniform* B-spline through the sample points. A good value of k is 1.353 (obtained by trial-and-error). Note that, typically, the calculated B-spline is modified to be *anchored* at $\mathbf{P}_0$ and $\mathbf{P}_n$ (the first three points will be $\mathbf{P}_0$, and the last three points will be $\mathbf{P}_n$).

Geometric Interpretation

Equation (41) is interpreted, geometrically, as shown in **Figure 4-1** below.

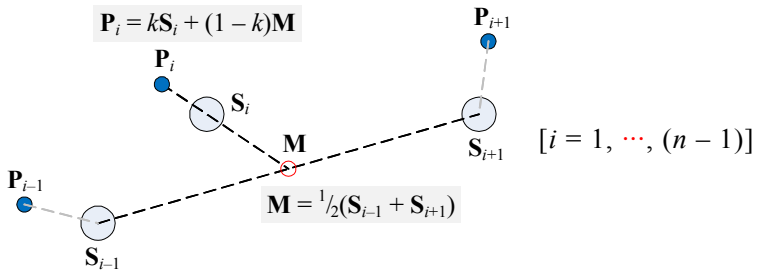

Figure 4-1

$\mathbf{P}_i$ $[i = 0, \cdots, n]$ are the *control points* of a (non-*anchored*) *uniform* 3rd degree *B-spline curve* passing near the sample points $\mathbf{S}_i$ $[i = 0, \cdots, n]$, thereby making the *B-spline curve* a *near interpolation*. The distance between $\mathbf{P}_i$ and $\mathbf{M}$ is k times the distance between $\mathbf{S}_i$ and $\mathbf{M}$. $\mathbf{M}$ is the midpoint between $\mathbf{S}_{i-1}$ and $\mathbf{S}_{i+1}$. The value of k is typically 1.353. Note that $\mathbf{P}_0 = \mathbf{S}_0$ and $\mathbf{P}_n = \mathbf{S}_n$.

The illustrations below show a comparison of a poly-line (green) and *near interpolated* B-spline (black) with a given B-spline (red). The centre of the circles represents the sample points of the given B-spline. The poly-line is typically used in graphics packages to emulate smooth curves. The *near interpolated* B-spline is a much more accurate emulation of smooth curves than the poly-line for the same sample points, but with only slightly more calculation overhead than for the poly-line. The *near interpolated* B-spline tends to have the most inaccuracy between the first two and last two sample points. (Note that, in the illustrations, the given B-spline (red) was emulated by a high-density poly-line, rather than a true interpolated B-spline.)

The illustration below, **Figure 4-2**, shows a given cubic B-spline (red) having three *spline segments* which is emulated by a poly-line (green) and a *near interpolated* B-spline (black). The centre of the circles (which are shown only for illustration) represent the sample points of the given B-spline (red) used for both emulations. The comparison is between the black and green curves relative to the red curve. The illustration shows how much more accurate the black curve (*near interpolated* B-spline) is compared to the green curve (a poly-line as typically used for emulating curves). In fact, the black curve is almost identical to the red curve, except at the ends.

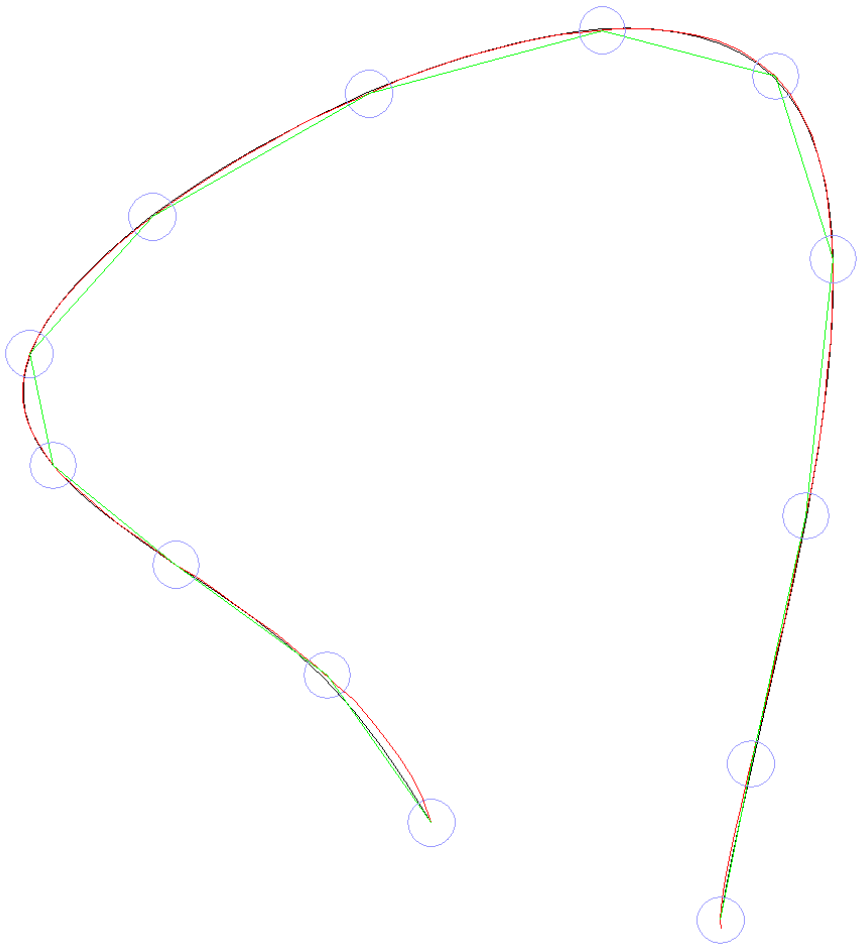

Figure 4-2

Figure 4-3 is a close-up of the top-right section of **Figure 4-2**, and shows that the *near interpolated* B-spline (black) does not necessarily pass through the sample points, but is close enough.

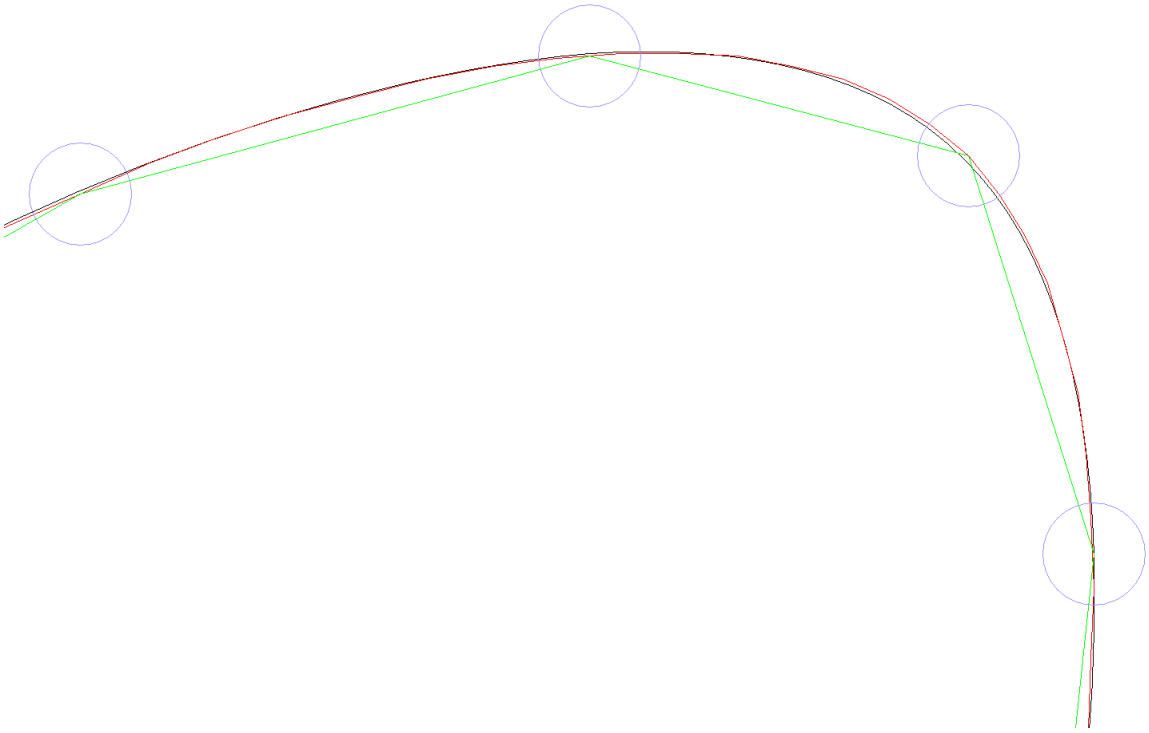

Figure 4-3

Figure 4-4 contains the same given B-spline as in **Figure 4-2**, but with fewer sample points. Even with such a low number of sample points, the *near interpolated* B-spline (black) is a much more accurate approximation of the red curve than is the poly-line (green), except between the end two sample points at each end of the B-spline.

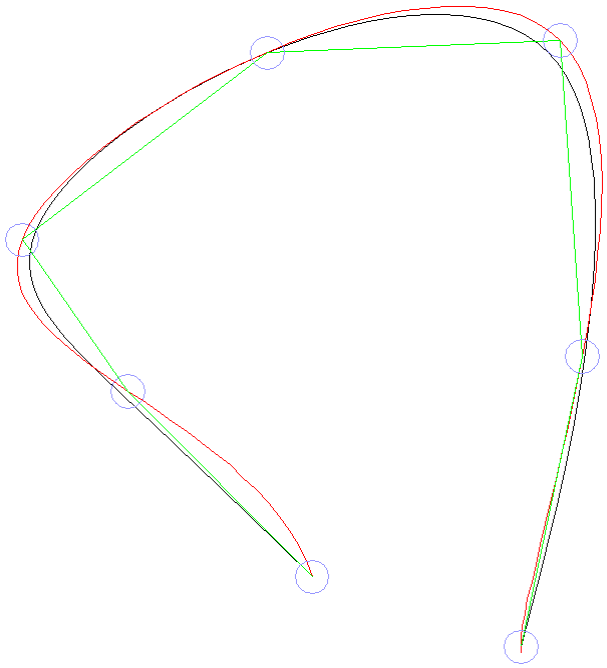

Figure 4-4

A *near interpolated* B-spline can be used not only for emulating any degree B-spline (or other splines) but also for emulating any curve that has continuity $C^{(1)}$. The illustration below, **Figure 4-5**, shows a portion of a given curve (red) defined by $y = {}^{1}\!/_{10}x\sin({}^{8}\!/_{25}x^2)$ emulated by a poly-line (green) and a *near interpolated* B-spline (black). Notice that, although the *near interpolated* B-spline is not extremely accurate at the left end of the figure, it nevertheless has a shape that resembles the given curve, whereas the poly-line has a very poor resemblance of the given curve.

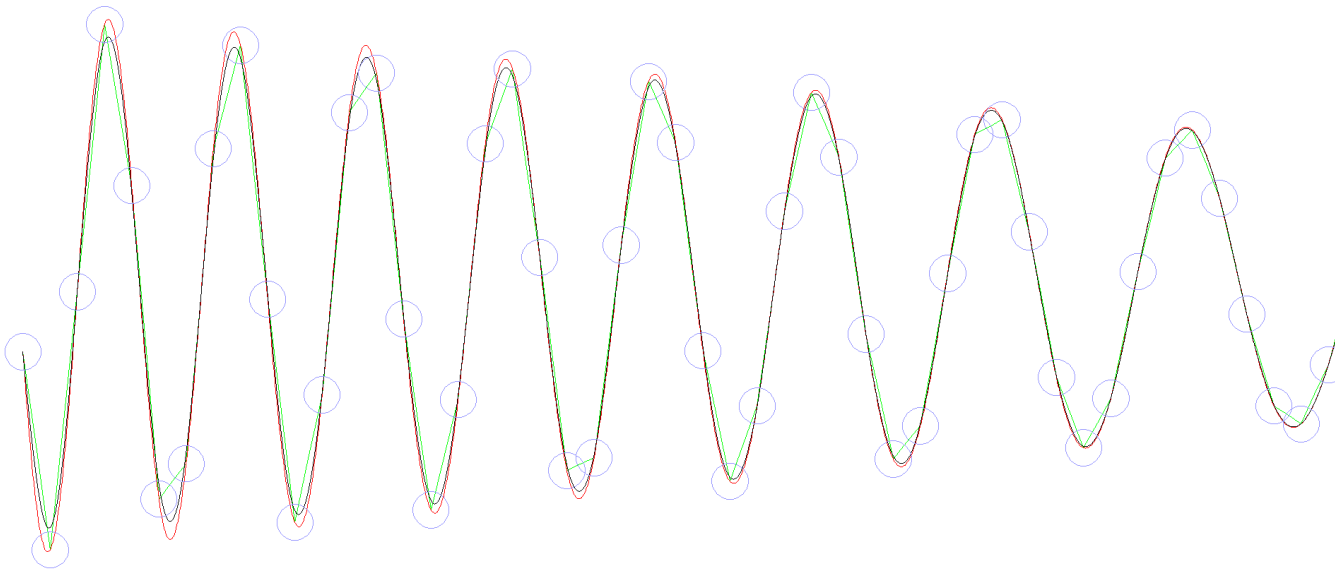

Figure 4-5

The greater the number of sample points the more closer a *near interpolated* B-spline will be to the true interpolated B-spline. However, if there are too many sample points, not only will there be more calculations to perform by the implementing software, but also the drawing canvas will need to be of higher density, otherwise the spline will appear rough.

A better approximation of a *near interpolated* B-spline to a true interpolated B-spline can be obtained if $\mathbf{S}_i$ is closer to **M** in **Figure 4-1**, and the sample points are more evenly distributed. Consequently, a *near interpolated* B-spline is not suitable for use with arbitrary *control points* to approximate a true interpolated B-spline through

those points. However, equation (41) can be used in its own right with the *control points* of any spline to flatten that spline by assigning a small positive value to k in the equation.

5. Misconceptions about B-splines

Unfortunately, there are a few common misconceptions about the comparison of B-splines with Bezier splines presented in some publications. Most of the misconceptions arise because a Bezier spline is inappropriately regarded as a one-piece curve of a varying degree that depends on the number of *control points* rather than a piecewise curve with each curve segment of the same degree (see **Appendix A: The Proper Definition of a Mathematical Spline**).

5.1 List of Misconceptions

The following is a list of some misconceptions about B-splines.

1. **Claim**: B-splines are more versatile than Bezier splines.

   B-splines and Bezier splines are inter-convertible — any *B-spline curve* can be converted to a Bezier spline (and vice versa), as described in chapter **3**, **B-splines in Terms of Bezier Control Points**. Therefore, B-splines are not more versatile than Bezier splines. However, direct Bezier splines are arguably more versatile than B-splines in practice because Bezier splines can be directly shaped to any polynomial spline via their *control points* without the need of additional information such as a *knot set*.

2. **Claim**: B-splines have local control and Bezier curves have global control.

   The justification offered is that modifying one *control point* of a B-spline affects at most $d + 1$ adjoining *spline segments* (where d is the degree of the B-spline), but that for a Bezier curve, a modified *control point* affects the whole "spline". The error made in the claim is to ignorantly compare a B-spline (which is a piecewise curve) of a particular degree with a one-piece Bezier curve of a degree that varies according to the number of *control points*. Comparing a B-spline (piecewise polynomial) with a Bezier curve (one-piece polynomial) is not a meaningful comparison; the comparison ought to be between a B-spline and a Bezier spline. A Bezier spline is actual a piecewise polynomial of a particular degree, not a one-piece polynomial with a degree depending on the number of *control points*.

   It turns out, in fact, that in general, direct Bezier splines have more local control than do B-splines. This is so because a modification of a Bezier spline *control point* affects at most only two adjoining polynomial segments for any degree Bezier spline.

3. **Claim**: B-splines are more general than Bezier splines.

   Such a claim needs to be qualified. The test for generality is that if A is more general than B, then there exist instances of A that are not instances of B, and all instances of B are also instances of A. However, all instances of B-splines are also instances of Bezier splines (since both B-splines and Bezier splines are piecewise polynomials and are inter-convertible), thus the test for generality fails. It could be said, however, that B-splines are more generic (not "general") with respect to their degree than are Bezier splines. The main reason for that is that a Bezier spline may pass through one *control point* for a particular degree, but not pass through that same *control point* for a different degree (for the same sequence of *control points*). Consequently, the shape of the Bezier spline can be drastically different from one degree to another (in fact the spline curve can become discontinuous from one degree to another with the same *control points*). With B-splines, however, the shape of the spline is only slightly different from one degree to another, but always remains a B-spline with the same *control points*.

   It could also be said that B-splines are more general than Bezier splines with respect to the same sequence of *control points* and degree, but not more general with respect to different *control points* and the same degree.

4. **Claim**: B-splines are composed of joined Bezier curves.

   B-splines, Bezier splines, and Bezier curves are not special types of curves — they are all polynomial curves. Each of the three terms signifies a particular relation between a sequence of *control points* and the shape of the corresponding polynomial curve. So, to say that a B-spline is "composed" of joined Bezier curves is to say that a certain sequence of *control points* (the B-spline *control points*) determining a sequence of certain joined polynomials can be expressed in terms of another certain sequence of *control points* (hypothetical Bezier *control points*) determining the same sequence of polynomials. The problem is with the word "composed" which does not seem appropriate. The claim should be modified to: B-splines can be converted to joined Bezier curves. Also, given that joined Bezier curves constitute a Bezier spline, the claim is equivalent to: "B-splines are composed of Bezier splines" — which is not particularly meaningful. The equivalent of the modified claim would be: "B-splines can be converted to Bezier splines" — which is both meaningful and correct.

Appendix A: The Proper Definition of a Mathematical Spline

Most publications verbal describe a mathematical spline as a "continuous piecewise polynomial curve", defined by

$$\mathbf{S}_{(d)}(t) = \sum_{i=0}^{n} \Phi_{i,d}(t)\mathbf{P}_i \quad [t \in \mathbb{R}] \qquad \text{(A1)}$$

where $\mathbf{P}_i$ $[i = 0, \cdots, n]$ are the *control points* of the spline, and d is the degree of the piecewise polynomials involved in the functions $\Phi_{i,d}$ $[i = 0, \cdots, n]$. However, equation (A1) is not consistent with the said verbal definition because the equation allows one-piece polynomial functions (for example, when $d = n$) to form a spline for a varying number of *control points*. A one-piece polynomial is not suitable for a spline that may have a varying number of *control points*; the implication is that such a polynomial would need to be of degree (say) 10000 for 10001 *control points*, which is impractical to implement, and the result may not look anything like a spline (for example the resulting curve may oscillate wildly in some places). The whole point of a spline is that its nature remains the same for any number of *control points* (above the minimum required number), rather than having different degree one-piece polynomials for a different number of *control points*.

The following ought to be the proper definition of a general mathematical spline for consistency with the verbal definition of a parametric polynomial spline (which is verbally defined to be piecewise). The equation below defines a family of spline curves of the same degree, d. There is one piecewise spline curve for each n.

$$\mathbf{S}_{(d)}(t) = \sum_{i=0}^{n} \Phi_{i,d}(t)\mathbf{P}_i \quad [t_0 \le t \le t_m;\ t \in \mathbb{R}]\ [d \in \mathbb{Z}^+;\ n = d,\ d + k,\ d + 2k, \cdots;\ k \in \mathbb{Z}^+] \qquad \text{(A2)}$$

In the definition above, the value of d is not determined by the value of n. That rules out using one-piece polynomials as splines where d is determined by the value of n (usually $d = n$). m is the number of polynomial segments of degree d. k is a particular value depending on the type of spline (eg: for a direct Bezier spline, $k = d$; for a B-spline, $k = 1$). Note that equation (A2) rules out having a one-piece Bezier curve as a Bezier spline where the degree of the curve depends on the value of n.

When comparing splines, it is nonsensical to compare splines of different degrees as is done in most publications when a B-spline of a particular degree is compared with one-piece Bezier curves of varying degrees as though each were a Bezier spline, with the false claim that B-splines have more local control than Bezier "splines".

Appendix B: Bezier Splines

Bezier splines are suitable for implementing geometric figures because they provide for precise control of the shape of the figures. *Uniform* B-splines are suitable for curve fitting with high order continuity.

Unfortunately, in the literature about Beziers, there is confusion between Bezier basis functions and blending functions, and also between Bezier splines and curves. This appendix includes recommendations to sort out the confusion.

B.1 Bezier Curves

In general, a Bezier curve refers to a one-piece polynomial function determined by a sequence of points called **control points**. For a degree d Bezier curve, there are $d + 1$ *control points*, $\mathbf{P}_0, \cdots, \mathbf{P}_d$. The curve passes through the two end *control points* ($\mathbf{P}_0$ and $\mathbf{P}_d$) but not through the inner *control points* ($\mathbf{P}_1, \cdots, \mathbf{P}_{d-1}$). The polynomial function of a Bezier curve can be defined in a number of equivalent ways as follows.

Standard form of Bezier Curve

The standard form of a Bezier curve is based on the Bernstein polynomials.

$$\mathbf{B}(t) = \sum_{i=0}^{d} \frac{d!}{i!(d-i)!} t^i (1-t)^{d-i} \mathbf{P}_i \quad [0 \le t \le 1] \qquad \text{(B1)}$$

where d is the degree of the desired polynomial function.

Polynomial form of Bezier Curve

The polynomial form of a Bezier curve is based on a polynomial function ($a_0 t^d + a_1 t^{d-1} + \cdots + a_{n-1} t + a_n$), and is

$$\mathbf{B}(t) = \sum_{k=0}^{d}\left(\frac{d!}{(d-k)!}\sum_{i=0}^{k}\frac{(-1)^{i+k}}{i!(k-i)!}\mathbf{P}_i t^k\right) \quad [0 \le t \le 1] \qquad \text{(B2)}$$

where d is the degree of the desired polynomial function.

Matrix Form of a Bezier Curve

The matrix form of a Bezier curve encapsulates both the standard and polynomial forms. An element, $m_{r,c}$, of a Bezier matrix $\boldsymbol{M}_d$ of degree d, where r is the r^{th} row, and c is the c^{th} column, and $m_{0,0}$ is the top-left element, is defined by

$$m_{r,c} = \begin{cases} \dfrac{(-1)^{c+d-r}\,d!}{r!\,c!\,(d-r-c)!} & [c \le d-r] \\ 0 & [\text{otherwise}] \end{cases} \quad [r, c \in \mathbb{N};\ r = 0, \cdots, d;\ c = 0, \cdots, d] \qquad \text{(B3)}$$

A d^{th} degree Bezier curve can be represented in matrix form as

$$\mathbf{B}(t) = \boldsymbol{T}_d(t)\boldsymbol{M}_d\boldsymbol{P} \quad [0 \le t \le 1] \qquad \text{(B4)}$$

where

$$\boldsymbol{T}_d(t) = [t^d \quad t^{d-1} \quad \cdots \quad t \quad 1]$$

and

$$\boldsymbol{P} = \begin{bmatrix} \mathbf{P}_0 \\ \vdots \\ \mathbf{P}_d \end{bmatrix}$$

Note that both the rows of $\boldsymbol{M}_d$ and the columns of $\boldsymbol{T}_d(t)$ can be reversed, resulting in

$$m_{r,c} = \begin{cases} \dfrac{(-1)^{c+r}\,d!}{(d-r)!\,c!\,(r-c)!} & [c \le r] \\ 0 & [\text{otherwise}] \end{cases} \quad [r, c \in \mathbb{N};\ r = 0, \cdots, d;\ c = 0, \cdots, d] \qquad \text{(B5)}$$

and

$$\boldsymbol{T}_d(t) = [1 \quad t \quad \cdots \quad t^{d-1} \quad t^d]$$

The inverse matrix, $\boldsymbol{M}_d^{-1}$, of $\boldsymbol{M}_d$ can be used to convert the *control points* of a *uniform* B-spline of degree d to Bezier *control points*.

B.2 Bezier Basis Functions

The proper definition of a mathematical spline was defined in **Appendix A: The Proper Definition of a Mathematical Spline** as

$$\mathbf{S}_{(d)}(t) = \sum_{i=0}^{n}\Phi_{i,d}(t)\mathbf{P}_i \quad [t_0 \le t \le t_m;\ t \in \mathbb{R}]\ [d \in \mathbb{Z}^+;\ n = d,\ d+k,\ d+2k, \cdots;\ k \in \mathbb{Z}^+] \qquad \text{(A2)}$$

where $\Phi_{i,d}$ $[i = 0, \cdots, n]$ are called the **basis functions**. In keeping with that definition, the *basis functions*, $B_{i,d}$, of a Bezier spline of degree d ought to be defined by

$$B_{i,d}(t) = \begin{cases} \mathrm{Bz}_{i,d}(t) & [i = 0] \\ \mathrm{Bz}_{k,d}\left(t - \left\lfloor \dfrac{(i-1)}{d} \right\rfloor\right) & \text{where } k = 1 + \mathsf{mod}_d(i-1) \quad [1 \le i \le n] \end{cases} \qquad \text{(B6)}$$

for $i = 0, \cdots, n$, where the value of $\mathsf{mod}_d(x)$ is equal to a such that $x \equiv a$ (modulo d), and where

$$\mathrm{Bz}_{k,d}(u) = \begin{cases} \dfrac{d!}{k!(d-k)!} u^k (1-u)^{d-k} \quad [0 \le k \le d] & [0 \le u < 1] \\ (2-u)^d \quad [k = d] & [1 \le u < 2] \\ 0 & [\text{otherwise}] \end{cases} \qquad \text{(B7)}$$

for $k = 0, \cdots, d$.

A general Bezier spline is therefore defined by

$$\mathbf{S}_{(d)}(t) = \sum_{i=0}^{n} B_{i,d}(t)\mathbf{P}_i \quad [0 \le t \le {}^{n}/_{d}] \; [d \ge 1;\, d \in \mathbb{Z}] \; [n = d, 2d, 3d, \cdots] \qquad \text{(B8)}$$

The *knot set* is implicitly fixed as $\langle 0, 1, 2, \cdots, {}^{n}/_{d} \rangle$. The *knot points* are $\mathbf{P}_0$, $\mathbf{P}_d$, $\mathbf{P}_{2d}$, $\cdots$, $\mathbf{P}_n$.

B.3 Bezier Blending Functions

As for B-splines, the *basis functions* of a Bezier spline turn out to be zero for most of a large number of *control points*. Each point of a Bezier spline segment is a weighted combination (or "blending") of $d + 1$ consecutive *control points*; the weighting is defined by $d + 1$ functions, $\mathrm{Bez}_{0,d}$, $\cdots$, $\mathrm{Bez}_{d,d}$, called **blending functions**.

$$\mathrm{Bez}_{k,d}(u) = \frac{d!}{k!(d-k)!} u^k (1-u)^{d-k} \quad [0 \le u \le 1] \; [k = 0, \cdots, d] \qquad \text{(B9)}$$

where d is the degree of the Bezier spline. Notice that the *blending functions* are the coefficients of the *control points* in equation (B1).

B.4 Types of Bezier Splines

Bezier curves can be joined in many different ways to form Bezier splines. Unlike B-splines, Bezier splines do not require a *knot set*. Instead, joining and other conditions may need to be specified to define particular types of Bezier splines. With Bezier splines, the Bezier *control points* may not all be identical to the given control points of the spline. Therefore, some Bezier *control points* may need to be calculated from the given control points. In this document, four categories of Bezier splines are defined.

Direct Bezier Spline

If the given spline control points are identical to the Bezier *control points*, and vice versa, then the spline is called a "direct Bezier spline". The *control points* may be restricted by joining conditions. A direct Bezier spline is the most versatile Bezier spline. A *poly-bezier* is a direct Bezier spline.

Controlled Bezier Spline

If all the given spline control points, with the exception of the two outer end control points of the given spline, are identical only to the inner *control points* of all the Bezier curve segments of the Bezier spline then the spline is called a "controlled Bezier spline". The outer *control points* of the Bezier curve segments are calculated from the given control points, except for the two outer end *control points* of the whole spline, which are identical to the two outer end control points of the given spline. The given control points may be restricted by joining conditions.

Interpolated Bezier Spline

If all the given spline control points are identical only to the outer *control points* of all the Bezier curve segments of the Bezier spline, then the spline is called an "interpolated Bezier spline". The inner *control points* of the Bezier curve segments are calculated from the given control points.

Indirect Bezier Spline

If none of the given spline control points are identical to any of the Bezier *control points* (except possibly the end *control points* of the spline), and vice versa, then the spline is called an "indirect Bezier spline". All of the *control points* of the Bezier curve segments are calculated from the given control points.

B.4.1 General Bezier Spline

A general Bezier spline is a direct Bezier spline, and is defined by equation (B1). There are no additional joining conditions other than those required for the spline to be continuous. Any polynomial spline can be represented as a general Bezier spline.

A general Bezier spline, including its *control points* ($\mathbf{P}_0$, ⋯, $\mathbf{P}_n$), is identical to a B-spline having the following *knot set* with that same sequence of *control points*.

$$\langle \underbrace{k_0, \cdots, k_0}_{d+1 \text{ times}}, \underbrace{k_1, \cdots, k_1}_{d \text{ times}}, \ldots, \underbrace{k_{\frac{n}{d}-1}, \cdots, k_{\frac{n}{d}-1}}_{d \text{ times}}, \underbrace{k_{\frac{n}{d}}, \cdots, k_{\frac{n}{d}}}_{d+1 \text{ times}} \rangle \quad [k_i \in \mathbb{R};\ k_i < k_{i+1};\ i = 0, \cdots, {}^{n}\!/_{d} - 1]$$

where d is the degree of the general Bezier spline, and k_i $[i = 0, \cdots, {}^{n}\!/_{d}]$ are arbitrary numbers as specified. In that sense, a general Bezier spline of a particular degree and sequence of *control points* can be said to be a special case of a B-spline of the same degree and sequence of *control points* (with the specified *knot set*).

B.4.2 Standard Cubic Bezier Spline

A default standard cubic Bezier spline is a controlled Bezier spline. The inner Bezier *control points* of all the Bezier curve segment are identical to the given control points, and vice versa (except for the two end control points of the whole spline). A non-default standard cubic Bezier spline is an indirect Bezier spline. In both cases, the common outer Bezier *control points* of each adjacent Bezier curve segment are midway between the two neighbouring inner *control points*. The two end Bezier *control points* of the whole spline are given.

The following is a mapping from a sequence of $n + 1$ given control points, $\mathbf{P}_i$ $[i = 0, \cdots, n]$, to a sequence of Bezier *control points*, $\mathbf{B}_i$ $[i = 0, \cdots, {}^{(n-3)}\!/_{2}]$ for a standard cubic Bezier spline. $n = 2j + 1$ $[j = 1, 2, \ldots]$.

$$\begin{bmatrix} \mathbf{B}_{3i} \\ \begin{bmatrix} \mathbf{B}_{3i+1} \\ \mathbf{B}_{3i+2} \end{bmatrix} \\ \mathbf{B}_{3i+3} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \begin{bmatrix} \frac{k}{2} & \frac{2-k}{2} \\ 0 & 0 \end{bmatrix} & & \begin{bmatrix} 0 & 0 \\ \frac{2-k}{2} & \frac{k}{2} \end{bmatrix} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \mathbf{P}_{2i} \\ \mathbf{P}_{2i+1} \\ \mathbf{P}_{2i+2} \\ \mathbf{P}_{2i+3} \end{bmatrix} \quad [i = 0, \cdots, \tfrac{n-3}{2}] \qquad \text{(B10)}$$

where k is a sharpness value (a real number), $0 \le k \le 1$; the default is $k = 0$. $k = 1$ results in a poly-line. For $i = 0$, the outer matrix is used; for $i > 0$, the inner matrix is used. If desired, adjustments can be made to *anchor* the end *control points*, as follows: $\mathbf{B}_0 = \mathbf{P}_0$; $\mathbf{B}_1 = (1 - k)\mathbf{P}_1 + k\mathbf{P}_0$; $\mathbf{B}_{(3n-5)/2} = (1 - k)\mathbf{P}_{n-1} + k\mathbf{P}_n$; $\mathbf{B}_{3(n-1)/2} = \mathbf{P}_n$.

A standard cubic Bezier spline is ideal for creating geometric figures with continuity $C^{(1)}$.

B.4.3 Cubic Bezier-B Spline

A cubic Bezier-B spline is an indirect Bezier spline, and is identical to an *anchored uniform* cubic B-spline (and therefore has continuity $C^{(2)}$). The mapping from a sequence of $n + 1$ given control points, $\mathbf{P}_i$ $[i = 0, \cdots, n]$, to a sequence of Bezier *control points*, $\mathbf{B}_i$ $[i = 0, \cdots, 3n - 6]$ for a cubic Bezier-B spline is defined by equation (37), reproduced below.

$$\begin{bmatrix} \mathbf{B}_{3i} \\ \begin{bmatrix} \mathbf{B}_{3i+1} \\ \mathbf{B}_{3i+2} \end{bmatrix} \\ \mathbf{B}_{3i+3} \end{bmatrix} = \begin{bmatrix} \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 \\ 0 & \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix} & & 0 \\ 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{bmatrix} \begin{bmatrix} \mathbf{P}_i \\ \begin{bmatrix} \mathbf{P}_{i+1} \\ \mathbf{P}_{i+2} \end{bmatrix} \\ \mathbf{P}_{i+3} \end{bmatrix} \quad [i = 0, \cdots, (n - 3)] \qquad (37)$$

In the equation above, the outer matrix is used if $i = 0$; the inner matrix is used if $i > 0$. The cubic Bezier-B spline is *anchored* to $\mathbf{P}_0$ and $\mathbf{P}_n$ as follows:

The following three *control points* are added before the first *control point* $\mathbf{B}_0$: $\mathbf{P}_0$, $^2/_3\mathbf{P}_0 + {}^1/_3\mathbf{P}_1$, $^1/_3\mathbf{P}_0 + {}^2/_3\mathbf{P}_1$.
The following three *control points* are added after the last *control point* $\mathbf{B}_{3n-6}$: $^2/_3\mathbf{P}_{n-1} + {}^1/_3\mathbf{P}_n$, $^1/_3\mathbf{P}_{n-1} + {}^2/_3\mathbf{P}_n$, $\mathbf{P}_n$.

The *control points* of a cubic Bezier-B spline would therefore be: $\mathbf{P}_0$, $^2/_3\mathbf{P}_0 + {}^1/_3\mathbf{P}_1$, $^1/_3\mathbf{P}_0 + {}^2/_3\mathbf{P}_1$, $\mathbf{B}_0$, ⋯, $\mathbf{B}_{3n-6}$, $^2/_3\mathbf{P}_{n-1} + {}^1/_3\mathbf{P}_n$, $^1/_3\mathbf{P}_{n-1} + {}^2/_3\mathbf{P}_n$, $\mathbf{P}_n$.

Appendix C: B-spline Algorithms

This appendix presents some B-spline pseudo-code algorithms that can be converted to a programming language. The algorithms are presented as an adaptation from mathematical notation, and can be used to implement various B-spline aspects.

The algorithms are designed for clarity rather than efficiency. Note that, in the algorithms, the symbols 'x +:= y' means appending the single element y to array x; 'x ++ y' means appending a copy of the elements of array y to a copy of the elements of array x to form a new array. Arrays and sequences are assumed to be one-based not zero-based.

C.1 Direct B-spline Function

This algorithm is a direct implementation of the B-spline function defined by equation (3′) for any B-spline. This algorithm is presented for demonstration purposes only; the algorithm is too slow to be used in practice. For an efficient implementation, see paragraph **C.2 deBoor Algorithm** below. Note that the function is defined recursively.

Given:
$s \in \mathbb{R}$ (desired density value, eg: 30); $d \in \mathbb{N}$ (desired degree of polynomial beginning with 0); $p =_{df} \langle \mathbf{P}_1, \cdots, \mathbf{P}_n \rangle$ (*control points*); $k =_{df} \langle t_1, \cdots, t_{n+d+1} \rangle$ (*knot set*).

Return:
$r =_{df}$ sequence of points on the *B-spline curve*.

Requirements:
Basis()

Algorithm:
Function: `BSplineDir`(s, d, p, k)

```
Inc := 1 / s;  T := t_{d+1};
while T ≤ t_{n+1}
{
    P := (0, 0)
    for I := 1 to n
    {
        P += Basis(T, I, d, k) × P_I
    }
    r +:= P;  T += Inc;
}

return r
```

Basis
The B-spline *basis function*. This algorithm is an implementation of the *basis function* as defined by equation (2).

Given:
$t \in \mathbb{R}$ (desired t value); $i \in \mathbb{Z}^+$ (desired interval number beginning with interval 1); $d \in \mathbb{N}$ (desired degree of polynomial beginning with 0); $k =_{df} \langle t_1, \cdots, t_{n+d+1} \rangle$ (*knot set*).

Return:
$r \in \mathbb{R}$ (return value of *basis function* at t).

Algorithm:
Function: `Basis`(t, i, d, k)

```
if d ≠ 0 then
{
    if  t_i ≤ t < t_{i+d+1}  then
    {
```

$$D_1 := \begin{cases} \dfrac{t - t_i}{t_{i+d} - t_i} & [t_{i+d} - t_i \neq 0] \\ 0 & [\text{otherwise}] \end{cases}; \quad D_2 := \begin{cases} \dfrac{t_{i+d+1} - t}{t_{i+d+1} - t_i} & [t_{i+d+1} - t_i \neq 0] \\ 0 & [\text{otherwise}] \end{cases};$$

$$P_1 := \begin{cases} 0 & [D_1 = 0] \\ D_1 \times \texttt{Basis}(t, i, d-1, k) & [\text{otherwise}] \end{cases}; \quad P_2 := \begin{cases} 0 & [D_2 = 0] \\ D_2 \times \texttt{Basis}(t, i+1, d-1, k) & [\text{otherwise}] \end{cases};$$

$r := P_1 + P_2$

```
    }
    else
    {
```

$r := 0$

```
    }
}
else
{
```

$$r := \begin{cases} 1 & [t_i \leq t < t_{i+1}] \\ 0 & [\text{otherwise}] \end{cases}$$

```
}

return r
```

C.2 deBoor Algorithm

This is an efficient implementation of the deBoor algorithm for displaying any B-spline. The algorithm returns a sample of points on the specified *B-spline curve*. For a direct (but inefficient) algorithm, see paragraph **D.2 The Algorithm** in **Appendix D: Derivation of the deBoor Recursive Function**.

The following algorithm was deduced from the deBoor recursive function (see **Appendix D: Derivation of the deBoor Recursive Function**). The density value specifies the number of returned points returned per interval.

Given:
$s \in \mathbb{N}^+$ (desired density value, eg: 10); $d \in \mathbb{N}$ (desired degree of polynomial beginning with 0); $p =_{df} \langle \mathbf{P}_1, \cdots, \mathbf{P}_n \rangle$ (*control points*); $k =_{df} \langle t_1, \cdots, t_{n+d+1} \rangle$ (*knot set* for *non-uniform* B-spline) or $k =_{df} 0$ (for *uniform* B-spline).

Return:
$r =_{df}$ sequence of points on the *B-spline curve*.

Requirements:
BSplineQSeq(), DeBoor(), UDeBoor()

Algorithm:
Function: BSplineAlgo(s, d, p, k)

```
for I := d + 1 to n
{
```

$$\texttt{T} := \begin{cases} t_I & [k \neq 0] \\ \texttt{I} & [\text{otherwise}] \end{cases}; \quad \texttt{NxtK} := \begin{cases} t_{I+1} & [k \neq 0] \\ \texttt{I}+1 & [\text{otherwise}] \end{cases}; \quad \texttt{Q} := \texttt{BSplineQSeq}(\texttt{I}, d, p);$$

$$\texttt{TDiff} := \begin{cases} \texttt{NxtK} - \texttt{T} & [\texttt{NxtK} - \texttt{T} \neq 0] \\ 1 & [\text{otherwise}] \end{cases}; \quad \texttt{Inc} := \texttt{TDiff} / s;$$

```
    repeat [* Calculate a sample of points between the current (T) and the next knot value (NxtK). *]
    {
```

$$r \mathrel{+}:= \begin{cases} \texttt{DeBoor}(\texttt{I}, d, \texttt{T}, k, \texttt{Q}) & [k \neq 0] \\ \texttt{UDeBoor}(\texttt{I}, d, \texttt{T}, \texttt{Q}) & [\text{otherwise}] \end{cases}; \quad \texttt{T} \mathrel{+}= \texttt{Inc};$$

```
    } until T ≥ NxtK
}
```

$r \mathrel{+}= \begin{cases} \texttt{DeBoor}(n, d, \texttt{NxtK}, k, \texttt{Q}) & [k \neq 0] \\ \texttt{UDeBoor}(n, d, \texttt{NxtK}, \texttt{Q}) & [\text{otherwise}] \end{cases}$ [* Calculate the last point. *]

`return` r

BSplineQSeq
Returns a subsequence of initialised *control points* corresponding to a given interval.

Given:
$i \in \mathbb{Z}^+$ (desired interval number beginning with interval $d + 1$); $d \in \mathbb{N}$ (desired degree of polynomial beginning with 0); $p =_{\text{df}} \langle \mathbf{P}_1, \cdots, \mathbf{P}_n \rangle$ (*control points*).

Return:
$r =_{\text{df}}$ subsequence of *control points*.

Algorithm:
Function: `BSplineQSeq`(i, d, p)
$r := \langle \mathbf{P}_i, \cdots, \mathbf{P}_{i-d} \rangle$

`return` r

DeBoor
The deBoor function for a particular interval.

Given:
$i \in \mathbb{Z}^+$ (desired interval number beginning with interval $d + 1$); $d \in \mathbb{N}$ (desired degree of polynomial beginning with 0); $t \in \mathbb{R}$ (desired t value within interval i to $i + 1$); $k =_{\text{df}} \langle t_1, \cdots, t_{n+d+1} \rangle$ (*knot set*); $q =_{\text{df}} \langle \mathbf{Q}_1, \cdots, \mathbf{Q}_{d+1} \rangle$ (subsequence of *control points*).

Return:
point.

Algorithm:
Function: `DeBoor`(i, d, t, k, q)
`Q` $:= q$
`for J :=` 1 `to` d
`{`
  `Idx :=` 1
  `for I :=` i `to` $i - d +$ `J step` -1
  `{`

$$\alpha := \begin{cases} \dfrac{t - t_{\texttt{I}}}{t_{\texttt{I}+d+1-\texttt{J}} - t_{\texttt{I}}} & [t_{\texttt{I}+d+1-\texttt{J}} - t_{\texttt{I}} \neq 0] \\ 0 & [\text{otherwise}] \end{cases} ; \quad \texttt{Q}_{(\texttt{Idx})} := (1 - \alpha)\texttt{Q}_{(\texttt{Idx}+1)} + \alpha \texttt{Q}_{(\texttt{Idx})};$$

    `Idx := Idx + 1`
  `}`
`}`

`return` $\texttt{Q}_{(1)}$

UDeBoor
The de Boor algorithm for a *uniform* B-spline with an implicit *knot set*.

Given:
$i \in \mathbb{Z}^+$ (desired interval number beginning with interval $d + 1$); $d \in \mathbb{N}$ (desired degree of polynomial beginning with 0); $t \in \mathbb{R}$ (desired t value within interval i to $i + 1$); $q =_{\text{df}} \langle \mathbf{Q}_1, \cdots, \mathbf{Q}_{d+1} \rangle$ (subsequence of *control points*).

Return:
point.

Algorithm:
Function: UDeBoor(i, d, t, q)
$\mathtt{Q} := q$
for $\mathtt{J} := 1$ to d
{
  $\mathtt{Idx} := 1$
  for $\mathtt{I} := i$ to $i - d + \mathtt{J}$ step -1
  {
    $$\alpha := \begin{cases} \dfrac{t - \mathtt{I}}{d + 1 - \mathtt{J}} & [d + 1 - \mathtt{J} \neq 0] \\ 0 & [\text{otherwise}] \end{cases}; \quad \mathtt{Q}_{(\mathtt{Idx})} := (1 - \alpha)\mathtt{Q}_{(\mathtt{Idx}+1)} + \alpha\mathtt{Q}_{(\mathtt{Idx})};$$
    $\mathtt{Idx} := \mathtt{Idx} + 1$
  }
}

return $\mathtt{Q}_{(1)}$

C.3 deBoor Algorithm for 3rd degree B-spline

This section describes an implementation of the deBoor algorithm for calculating 3rd degree B-spline values (with any *knot set*) as *poly-bezier* points. The function returns *poly-bezier* points defining the *B-spline curve*. This algorithm can be used for software systems that render Bezier curves directly for the returned *poly-bezier* points. If a software system renders B-splines directly, then this algorithm need not be used. The following algorithm was deduced from the so-called "polar forms" of B-splines (see **3.4 Polar Forms**).

Given:
$p =_{\text{df}} \langle \mathbf{P}_1, \cdots, \mathbf{P}_n \rangle$ (*control points*); $k =_{\text{df}} \langle t_1, \cdots, t_{n+d+1} \rangle$ (*knot set*).

Return:
$r =_{\text{df}}$ sequence of *poly-bezier* points defining the *B-spline curve*.

Algorithm:
Function: BSpline3Algo(p, k)
for $\mathtt{I} := 4$ to n
{
  $\mathtt{P} := \langle \mathbf{P}_{\mathtt{I}-3}, \mathbf{P}_{\mathtt{I}-2}, \mathbf{P}_{\mathtt{I}-1}, \mathbf{P}_{\mathtt{I}} \rangle$
  $$\alpha_1 := \begin{cases} \dfrac{t_{\mathtt{I}} - t_{\mathtt{I}-1}}{t_{\mathtt{I}+2} - t_{\mathtt{I}-1}} & [t_{\mathtt{I}+2} - t_{\mathtt{I}-1} \neq 0] \\ 0 & [\text{otherwise}] \end{cases}; \quad \mathbf{Q}_1 := \alpha_1 \mathtt{P}_{(3)} + (1 - \alpha_1)\mathtt{P}_{(2)};$$
  $$\alpha_1 := \begin{cases} \dfrac{t_{\mathtt{I}+1} - t_{\mathtt{I}-1}}{t_{\mathtt{I}+2} - t_{\mathtt{I}-1}} & [t_{\mathtt{I}+2} - t_{\mathtt{I}-1} \neq 0] \\ 0 & [\text{otherwise}] \end{cases}; \quad \mathbf{Q}_2 := \alpha_1 \mathtt{P}_{(3)} + (1 - \alpha_1)\mathtt{P}_{(2)};$$
  $$\alpha_1 := \begin{cases} \dfrac{t_{\mathtt{I}+1} - t_{\mathtt{I}}}{t_{\mathtt{I}+2} - t_{\mathtt{I}}} & [t_{\mathtt{I}+2} - t_{\mathtt{I}} \neq 0] \\ 0 & [\text{otherwise}] \end{cases}; \quad \alpha_2 := \begin{cases} \dfrac{t_{\mathtt{I}+1} - t_{\mathtt{I}}}{t_{\mathtt{I}+3} - t_{\mathtt{I}}} & [t_{\mathtt{I}+3} - t_{\mathtt{I}} \neq 0] \\ 0 & [\text{otherwise}] \end{cases};$$
  $\mathbf{Q}_3 := \alpha_1(\alpha_2 \mathtt{P}_{(4)} + (1 - \alpha_2)\mathtt{P}_{(3)}) + (1 - \alpha_1)\mathbf{Q}_2$

  if $\mathtt{I} = 4$ then [* Calculate first point. *]
  {
    $$\alpha_1 := \begin{cases} \dfrac{t_{\mathtt{I}} - t_{\mathtt{I}-1}}{t_{\mathtt{I}+1} - t_{\mathtt{I}-1}} & [t_{\mathtt{I}+1} - t_{\mathtt{I}-1} \neq 0] \\ 0 & [\text{otherwise}] \end{cases}; \quad \alpha_2 := \begin{cases} \dfrac{t_{\mathtt{I}} - t_{\mathtt{I}-2}}{t_{\mathtt{I}+1} - t_{\mathtt{I}-2}} & [t_{\mathtt{I}+1} - t_{\mathtt{I}-2} \neq 0] \\ 0 & [\text{otherwise}] \end{cases};$$
    $r +:= \alpha_1 \mathbf{Q}_1 + (1 - \alpha_1)(\alpha_2 \mathtt{P}_{(2)} + (1 - \alpha_2)\mathtt{P}_{(1)})$

```
    }

    r +:= Q1;    r +:= Q2;    r +:= Q3;
}

return r
```

Algorithm for Uniform 3rd degree B-spline

This section describes an algorithm for calculating *uniform* 3rd degree B-spline values as *poly-bezier* points. The function returns *poly-bezier* points defining the *uniform B-spline curve*. The following algorithm is a special case of the algorithm above.

Given:
$p =_{df} \langle \mathbf{P}_1, \cdots, \mathbf{P}_n \rangle$ (*control points*); $A_1 =_{df}$ boolean (`true` if curve is to be anchored at the beginning); $A_2 =_{df}$ boolean (`true` if curve is to be anchored at the end).

Return:
$r =_{df}$ sequence of *poly-bezier* points defining the *uniform B-spline curve*.

Algorithm:
Function: `BSplineBezPts`(p, A_1, A_2)
$r +:= \frac{1}{6}\mathbf{P}_1 + \frac{2}{3}\mathbf{P}_2 + \frac{1}{6}\mathbf{P}_3$

```
for I := 2 to n – 2
{
    r +:= ⅔P_I + ⅓P_I+1;    r +:= ⅓P_I + ⅔P_I+1;    r +:= ⅙P_I + ⅔P_I+1 + ⅙P_I+2;
}

if A1 then
{
    r := ⟨P1, ⅔P1 + ⅓P2, ⅓P1 + ⅔P2⟩ ++ r
}

if A2 then
{
    r := r ++ ⟨⅔P_n–1 + ⅓P_n, ⅓P_n–1 + ⅔P_n, P_n⟩
}

return r
```

C.4 Near Interpolated B-spline Function

The following *near interpolated* B-spline function accurately approximates an interpolated cubic B-spline through given sample points of a curve (See chapter **4**, **Emulating Curves Using B-splines**). An accuracy factor (f) of 1.353 gives a good approximation.

Given:
$p =_{df} \langle \mathbf{P}_1, \cdots, \mathbf{P}_n \rangle$ (*control points*); $f =_{df}$ accuracy factor.

Return:
$r =_{df}$ sequence of *control points* defining the *B-spline curve* that approximates the interpolated cubic B-spline through the given *control points* (sample points).

Algorithm:
Function: `NIBSpline`(p, f)

$C := \frac{1-f}{2}$; $r +:= \mathbf{P}_1$;

```
for I := 2 to n – 1
{
    Cur := P_I;    Prv := P_I–1;    Nxt := P_I+1;

    if not (Prv = Cur or Nxt = Cur) then
```

```
    {
        r +:= fCur + C(Prv + Nxt)
    }
    else
    {
        r +:= Cur
    }

    r +:= P_n
}

return r
```

To produce a good approximation of a *B-spline curve* of degree d, *control points* p, and *knot set* k, the function `NIBSpline(BSplineAlgo(8, d, p, k), 1.353)` can be called. The returned *control points*, z, of the called function are cubic B-spline *control points*, which can be rendered directly, or passed to `BSplineBezPts(z, true, true)` for rendering via Bezier curves.

Appendix D: Derivation of the deBoor Recursive Function

The deBoor algorithm is an efficient means by which to calculate a point on a *B-spline curve*. In this appendix, the said algorithm is derived mathematically from the definition of a B-spline.

D.1 The Mathematical Derivation

In this derivation, sequences begin with index one, rather than zero.

The standard definition of a *B-spline curve* (beginning with index one) with *knot set* $\langle t_1, \cdots, t_{n+d+1} \rangle$ is as follows.

$$\mathbf{S}(t) = \sum_{i=1}^{n} B_{i,d}(t)\mathbf{P}_i \quad [t_{d+1} \le t < t_{n+1};\ t \in \mathbb{R};\ n \ge d+1] \qquad \text{(D1)}$$

By expanding $B_{i,d}(t)$ via the *basis function* definition, equation (D1) becomes

$$\begin{aligned}
\mathbf{S}(t) &= \sum_{i=1}^{n}\left(\left(\frac{t-t_i}{t_{i+d}-t_i}\right)B_{i,d-1}(t) + \left(\frac{t_{i+d+1}-t}{t_{i+d+1}-t_{i+1}}\right)B_{i+1,d-1}(t)\right)\mathbf{P}_i \\
&= \sum_{i=1}^{n}\left(\frac{t-t_i}{t_{i+d}-t_i}\right)B_{i,d-1}(t)\mathbf{P}_i + \sum_{i=1}^{n}\left(\frac{t_{i+d+1}-t}{t_{i+d+1}-t_{i+1}}\right)B_{i+1,d-1}(t)\mathbf{P}_i \qquad \text{(D2)} \\
&= \left(\frac{t-t_1}{t_{1+d}-t_1}\right)B_{1,d-1}(t)\mathbf{P}_1 + \sum_{i=2}^{n}\left(\frac{t-t_i}{t_{i+d}-t_i}\right)B_{i,d-1}(t)\mathbf{P}_i + \sum_{i=1}^{n-1}\left(\frac{t_{i+d+1}-t}{t_{i+d+1}-t_{i+1}}\right)B_{i+1,d-1}(t)\mathbf{P}_i + \left(\frac{t_{n+d+1}-t}{t_{n+d+1}-t_{n+1}}\right)B_{n+1,d-1}(t)\mathbf{P}_n
\end{aligned}$$

The support for the first right-hand term in the last equation (D2) is $[t_1, t_{d+1})$. This interval is outside the interval for t, $[t_{d+1}, t_{n+1})$, defined for the whole *B-spline curve* in equation (D1), therefore, the first right-hand term of the last equation (D2) is zero for all t. Likewise, the support for the last right-hand term in the last equation (D2) is $[t_{n+1}, t_{n+d+1})$. This interval is outside the interval for t, $[t_{d+1}, t_{n+1})$, defined for the whole *B-spline curve* in equation (D1), therefore, the last right-hand term of the last equation (D2) is zero for all t. The last equation (D2) can therefore be simplified to its middle two right-hand terms, as follows.

$$\mathbf{S}(t) = \sum_{i=2}^{n}\left(\frac{t-t_i}{t_{i+d}-t_i}\right)B_{i,d-1}(t)\mathbf{P}_i + \sum_{i=1}^{n-1}\left(\frac{t_{i+d+1}-t}{t_{i+d+1}-t_{i+1}}\right)B_{i+1,d-1}(t)\mathbf{P}_i \qquad \text{(D3)}$$

If the sum of the last right-hand term of equation (D3) is altered to begin with 2 and end with n, the index of t will need to have 1 (one) subtracted from it for the value of the sum to remain the same. Equation (D3) is therefore equivalent to

$$\begin{aligned}
\mathbf{S}(t) &= \sum_{i=2}^{n}\left(\frac{t-t_i}{t_{i+d}-t_i}\right)B_{i,d-1}(t)\mathbf{P}_i + \sum_{i=2}^{n}\left(\frac{t_{i+d}-t}{t_{i+d}-t_i}\right)B_{i,d-1}(t)\mathbf{P}_{i-1} \qquad \text{(D4)} \\
&= \sum_{i=2}^{n}\left(\left(\frac{t_{i+d}-t}{t_{i+d}-t_i}\right)\mathbf{P}_{i-1} + \left(\frac{t-t_i}{t_{i+d}-t_i}\right)\mathbf{P}_i\right)B_{i,d-1}(t)
\end{aligned}$$

Note that, in the last line of equation (D4), $B_{i,d-1}$ has been factored out of the sum, and the remaining two terms swapped.

If the following assignment is made,

$$\mathbf{Q}_{1,i}(t) = \left(\frac{t_{i+d} - t}{t_{i+d} - t_i}\right)\mathbf{P}_{i-1} + \left(\frac{t - t_i}{t_{i+d} - t_i}\right)\mathbf{P}_i \qquad \text{(D5)}$$

then equation (D4) becomes

$$\mathbf{S}(t) = \sum_{i=2}^{n} B_{i,d-1}(t)\mathbf{Q}_{1,i}(t) \qquad \text{(D6)}$$

Equation (D6) is in a similar form as equation (D1), so the process above can be applied again to equation (D6). The final result is

$$\mathbf{S}(t) = \sum_{i=3}^{n} B_{i,d-2}(t)\mathbf{Q}_{2,i}(t) \qquad \text{(D7)}$$

where

$$\mathbf{Q}_{2,i}(t) = \left(\frac{t_{i+d-1} - t}{t_{i+d-1} - t_i}\right)\mathbf{Q}_{1,i-1}(t) + \left(\frac{t - t_i}{t_{i+d-1} - t_i}\right)\mathbf{Q}_{1,i}(t) \qquad \text{(D8)}$$

If the process is applied again with equation (D7), and repeated recursively, eventually the *basis function* of equation (D7) becomes $B_{i,0}$. The final result will be

$$\mathbf{S}(t) = \sum_{i=d+1}^{n} B_{i,0}(t)\mathbf{Q}_{d,i}(t) \quad [t_{d+1} \le t < t_{n+1};\ t \in \mathbb{R};\ n \ge d+1] \qquad \text{(D9)}$$

where

$$\mathbf{Q}_{j,i}(t) = \begin{cases} (1 - \alpha_{j,i}(t))\mathbf{Q}_{j-1,i-1}(t) + \alpha_{j,i}(t)\mathbf{Q}_{j-1,i}(t) & [j > 0] \\ \mathbf{P}_i & [j = 0] \end{cases} \qquad \text{(D10)}$$

and

$$\alpha_{j,i}(t) = \frac{t - t_i}{t_{i+d-j+1} - t_i} \quad [j > 0] \qquad \text{(D11)}$$

Note that

$$1 - \alpha_{j,i}(t) = 1 - \frac{t - t_i}{t_{i+d-j+1} - t_i} = \frac{t_{i+d-j+1} - t}{t_{i+d-j+1} - t_i}$$

which corresponds to the coefficient of $\mathbf{Q}_{1,i-1}(t)$ in equation (D8).

If $t \in \mathbf{[}t_k, t_{k+1}\mathbf{)}$ for some integral value of k inclusively between $d + 1$ and n, then equation (D9) reduces to

$$\mathbf{S}(t) = \mathbf{Q}_{d,k}(t) \quad [t_k \le t < t_{k+1};\ t \in \mathbb{R}]\ [k = d+1, \cdots, n] \qquad \text{(D12)}$$

This is because, in equation (D9), $B_{i,0}(t) = 1$ only in the interval $t \in \mathbf{[}t_i, t_{i+1}\mathbf{)}$ (where, in this case, $i = k$); $B_{i,0}(t) = 0$ outside the said interval. Note that the right hand side of equation (D12) is recursive via equation (D10).

The deBoor recursive function is equation (D12) in conjunction with equations (D10) and (D11).

D.2 The Algorithm

The deBoor recursive function can be implemented directly in software. However, a direct implementation is inefficient. For an efficient implementation, see **C.2 deBoor Algorithm** in **Appendix C: B-spline Algorithms**.

The direct algorithm is as follows. All arrays (sequences) are one-based.

Given:
$s \in \mathbb{R}$ (desired density value, eg: 30); $d \in \mathbb{N}$ (desired degree of polynomial beginning with 0); $p =_{df} \langle \mathbf{P}_1, \cdots, \mathbf{P}_n \rangle$ (*control points*); $k =_{df} \langle t_1, \cdots, t_{n+d+1} \rangle$ (*knot set* for *non-uniform* B-spline) or $k =_{df} 0$ (for *uniform* B-spline).

Return:
$r =_{df}$ sequence of points on the *B-spline curve*.

Requirements:
DeBoor(), UDeBoor()

Algorithm:
Function: `BSplineAlgoDir`(s, d, p, k)
`Inc` := $1 / s$
`for I` := $d + 1$ `to` n
{

$$\texttt{T} := \begin{cases} t_{\texttt{I}} & [k \neq 0] \\ \texttt{I} & [\text{otherwise}] \end{cases} ; \quad \texttt{NxtK} := \begin{cases} t_{\texttt{I}+1} & [k \neq 0] \\ \texttt{I}+1 & [\text{otherwise}] \end{cases} ;$$

```
    while T < NxtK
    {
```

$$r \mathrel{+}:= \begin{cases} \texttt{DeBoor}(d, \texttt{I}, d, \texttt{T}, p, k) & [k \neq 0] \\ \texttt{UDeBoor}(d, \texttt{I}, d, \texttt{T}, p) & [\text{otherwise}] \end{cases} ; \quad \texttt{T += Inc;}$$

```
    }
}
```

$$r \mathrel{+}:= \begin{cases} \texttt{DeBoor}(d, n, d, \texttt{NxtK}, p, k) & [k \neq 0] \\ \texttt{UDeBoor}(d, n, d, \texttt{NxtK}, p) & [\text{otherwise}] \end{cases}$$ [* Calculate the last point. *]

`return` r

DeBoor
The deBoor function for a t value of a particular interval between two adjacent *knots*. Note that the function is recursive.

Given:
$j \in \mathbb{N}$ (used internally); $i \in \mathbb{Z}^+$ (desired interval number beginning with interval $d + 1$); $d \in \mathbb{N}$ (desired degree of polynomial beginning with 0); $t \in \mathbb{R}$ (desired t value within interval i to $i + 1$); $p =_{df} \langle \mathbf{P}_1, \cdots, \mathbf{P}_n \rangle$ (*control points*); $k =_{df} \langle t_1, \cdots, t_{n+d+1} \rangle$ (*knot set*).

Return:
$r =_{df}$ point on the *B-spline curve* at t.

Algorithm:
Function: `DeBoor`(j, i, d, t, p, k)

$$\alpha := \begin{cases} \dfrac{t - t_i}{t_{i+d+1-j} - t_i} & [t_{i+d+1-j} - t_i \neq 0] \\ 0 & [\text{otherwise}] \end{cases}$$

`if` $j > 0$ `then`
{
    $r := (1 - \alpha)\texttt{DeBoor}(j - 1, i - 1, d, t, p, k) + \alpha\texttt{DeBoor}(j - 1, i, d, t, p, k)$
}
`else`
{
    $r := \mathbf{P}_i$
}

`return` r

UDeBoor

The de Boor algorithm for a *uniform* B-spline with an implicit *knot set*, and for a t value of a particular interval between two adjacent *knots*. Note that the function is recursive.

Given:

$j \in \mathbb{N}$ (used internally); $i \in \mathbb{Z}^+$ (desired interval number beginning with interval $d + 1$); $d \in \mathbb{N}$ (desired degree of polynomial beginning with 0); $t \in \mathbb{R}$ (desired t value within interval i to $i + 1$); $p =_{df} \langle \mathbf{P}_1, \cdots, \mathbf{P}_n \rangle$ (*control points*).

Return:

$r =_{df}$ point on the *B-spline curve* at t.

Algorithm:

Function: `UDeBoor`(j, i, d, t, p)

$$\alpha := \begin{cases} \dfrac{t - i}{d + 1 - j} & [d + 1 - j \neq 0] \\ 0 & [\text{otherwise}] \end{cases}$$

```
if j > 0 then
{
    r := (1 − α)UDeBoor(j − 1, i − 1, d, t, p) + αUDeBoor(j − 1, i, d, t, p)
}
else
{
    r := P_i
}

return r
```

Definitions

anchored
See page 8.

B-spline curve
See page 1.

basis function
See pages 1 and 20.

blending function
See pages 5 and 21.

control points
See pages 1 and 19.

knot points
See page 1.

knot set
See page 1.

knots
See page 1.

near interpolation
See page 14.

non-uniform
See page 2.

polar values
See page 13.

poly-bezier
See page 10.

spline segment
See page 1.

trimmed knot set
See page 13.

uniform
See page 2.