Curves II: B-Splines

The issues arising in computer-aided curve and surface generation have been described in the previous two documents in this series. The first outlined the difficulties arising out of simply fitting a usually high order polynomial formula to a set of points. These difficulties were somewhat alleviated by the de Casteljau algorithm and development of Bezier curve formulas as described in the second document. The price paid was that the curve no longer passed through or interpolated all of the control points.

However, Bezier curves themselves still suffer from a number of disadvantages. They still lead to high order polynomial formulas when a large number of control points are involved – the parametric polynomials have a degree just one less than the number of control points. In fact, this means that the degree (and hence the continuity properties) of a Bezier curve are determined by the number of control points present, and not otherwise by the choice of the user.

Secondly, the entire curve is described by just one formula. While a change in the position of just one of the control points has the greatest effect on the shape of the curve in the vicinity of that control point, that change will have lesser effects on the shape of the entire curve. This non-local feature makes it difficult to fine-tune the shape of parts of the curve, since modifications at one end tend to result in changes in the shape at some distance along the curve – a little like the problem of trying to get rid of ripples in a carpet that's nailed down around the edges of the room. Bezier curves are said to lack "local control."

What is needed is a method that combines the good features of the Bezier approach (their invariance[1] under geometric, or more generally, affine, transformations; their convex hull property, variation diminishing property, and continuity properties -–perhaps even the fact that they interpolate the two end control points, and so on), while solving the problems posed by this lack of local control and the inability to restrict the mathematical representations of the curves to some desired low order polynomial functions.

Before this begins to sound too much like a stale radio commercial, we simply state that the so-called **B-splines** (for **B**asis splines) meet just these requirements, and some variations represent the state-of-the-art in curve and surface modelling methods.

Recall that it was possible to express Bezier curve formulas in the form:

$$\mathbf{p}^{(L)}(t) = \sum_{k=0}^{L} B_k{}^{(L)}(t) \cdot \mathbf{p}_k \qquad \text{(CURV2-1)}$$

where $\mathbf{p}^{(L)}(t)$ stands for coordinates of points along the Bezier curve, the $\mathbf{p}_k$ are the coordinates of the L+1 control points, and the $B_k^{(L)}$ are the so-called Bernstein-Bezier polynomials, which express how much the location of each control point influences the position (or shape, if you like) of the curve at each location. As the parameter t varies from 0 to 1 in value, formula (CURV2-1) produces the entire curve, starting from point $\mathbf{p}_0$, and ending up at $\mathbf{p}_L$. Here the index L reflects the number of control points involved and gives the highest power of t appearing in the formula for these blending functions (so that, if , say, you have 10 control points, or L+1 = 10, then L = 9, and the $B_k^{(L)}$ functions will have terms with powers of t up to and including t^9.). Further, the fact that the $B_k^{(L)}$ tend to have non-zero values for all values of t except possibly t = 0 and t = 1 is what gives rise to the non-locality problem.

[1] Oops! Apologies for the mathematical jargon that crept in. In this case "invariance" means simply that to carry out a translation, rotation, scaling, reflection, or whatever desired affine transformation (of which the geometric transformations mentioned are special cases), you don't have to transform the curve element-by-element. Just apply the desired transformation to the control points and redraw the curve. Not only does this probably save computational effort, but it means that the only information you need have available to create the curve in any geometric configuration is the coordinates of the control points – which is usually takes up a lot less memory than a pixel-by-pixel specification of the curve itself.

The formulas for the $B_k^{(L)}(t)$ can be obtained by noticing patterns that occur when the de Casteljau algorithm is applied to the set of points. This algorithm was described in some detail in the previous document.

B-Spline curves are based on a similar formula, often written

$$\mathbf{p}(t) = \sum_{k=0}^{L} N_{k,m}(t) \cdot \mathbf{p}_k \qquad \text{(CURV2-2)}$$

where again the $\mathbf{p}_k$ stand for the control points, and here $\mathbf{p}(t)$ stands for the coordinates of points along the B-spline curve. The $N_{k,m}(t)$ play the role of the blending functions in the Bezier formula, but in this context are more often referred to as **basis functions**. The index k obviously refers to specific control points, and the index m indicates the order of the function (you'll see shortly that the highest power of t occurring in a B-spline formula is always m – 1). Thus they differ from the Bernstein-Bezier functions in that the order of these "blending" or basis functions is not necessarily the same as the number of control points (but, if L = m, the B-spline formulas will just give us the Bezier curve!), but also, in the fact that the $N_{k,m}(t)$ functions are non-zero only for at most m control points. If you think back to the ideas presented in the introductory document in this series, this amounts to saying that B-splines have the greatest degree of localization possible since an order m curve cannot be determined by less than m points.

The most convenient and commonly used method for determining formulas for the $N_{k,m}(t)$ are the so-called **Cox-de Boor** recursion formulas, which we state shortly. We will make no attempt to present a derivation of the fundamental formula defining the $N_{k,m}(t)$ functions. Because of the generality built into the definition, the derivation or proof is rather involved and of little pedagogical value given the scope of the present course. In some sense, you can think it arising by performing the de Casteljau algorithm not on a succession of sets of control points, but on a succession of sets of increasing order blending functions.[2]

Relatively easy development of B-spline formulas requires consistency and precision of notation more than anything else. So,

> ❁ we will always number control points starting from zero. $\mathbf{p}_0$ is thus always the first control point. The last control point is $\mathbf{p}_L$.

Secondly,

> ❁ we will always use the symbol L to indicate the last control point. Thus, the number of control points is always L + 1.

Thirdly,

> ❁ we will use the symbol m to indicate the **order** of the B-spline curve. This means that the highest power of the parameter appearing in the basis functions will be m – 1

Thus, a cubic B-spline will have order m = 4, and the basis functions will contain terms involving the third power of the parameter. A quadratic B-spline has m = 3, and the highest power of the parameter in the basis functions will be the second power or the square.

---

[2] This is a rather gross over-simplification, but follows from an observation made by G. Farin in his book *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide"* [Academic Press] – see the sentence at the beginning of the paragraph midway down page 128 of the first edition (1988) of this book. Farin is one of the most comprehensive references on spline curves that is reasonably accessible. He takes about 10 pages to present his derivation the Cox-de Boor formula (based on other results developed earlier in his book), and yet earlier has referred to de Boor's original derivation as "mathematically involved," giving you the impression that the rigorous theory behind this formula is not trivial. B-spline formulas have a fairly extensive history, including what appears to be the break-through work of Cox and de Boor in the early 1970's. You can consult Farin for details of and references to the historical literature of the subject.

Now, the overall B-spline curve will be a number of m^{th} order **segments** pieced together. Since an m^{th} order polynomial is determined by m control points, we can easily work out how many segments the overall curve will have.

Figure 1 to the right gives an example. Here there are 8 control points, numbered 0, 1, 2, 3, 4, 5, 6, 7, and so L = 7. If we wish to construct a cubic B-Spline curve with these points, then m = 4. The first segment will be determined by points 0, 1, 2, and 3. Then the second segment is determined by points 1, 2, 3, and 4. Continuing this procedure, we see that the 8 control points will allow construction of 5 segments. This value 5 arises by starting with the 8 control points, and noticing that the first m – 1 = 3 control points are used essentially just to start the process. Points 3 = m - 1 through 7 = L can be matched one-for-one with one of the segments of the curve. The number of segments is 7 – (4 – 1) + 1= 5, or in symbols L – (m – 1) + 1 = L – m + 2[3].

Figure 1

❁ the B-spline curve will generally consist of L – m + 2 individual segments pieced together.

Associated with each B-spline curve is a set of values called **knots**, the entire set being referred to as the **knot vector**, and often written as

$$\mathbf{T} = \{ t_0, t_1, t_2, \ldots.\}$$

For the moment, we impose no additional conditions on these values except to require that

$$t_k \le t_{k+1} \text{ for all k} \qquad \text{(CURV2-3)}$$

That is, the knots must be non-decreasing. Two or more successive knots may be equal, but you may never have the values decrease as you proceed through the list from lower indices to higher indices. The successive values of the knots will be associated with the control points, but how these values are defined will depend on the situation and type of curve which is desired. The principles will become clear as we proceed.

After all this, we can now state the Cox – de Boor recursion relations defining the B-spline basis functions $N_{k,m}$:

$$N_{k,m}(t) = \frac{t - t_k}{t_{k+m-1} - t_k} N_{k,m-1} + \frac{t_{k+m} - t}{t_{k+m} - t_{k+1}} N_{k+1,m-1}, \quad k = 0, 1, 2, \ldots, L \qquad \text{(CURV2-4a)}$$

This expresses $N_{k,m}$ in terms of basis functions of one order lower, $N_{j,m-1}$. Terms in this formula which have denominators that evaluate to zero are ignored. To start the whole process off, we need a definition for the $N_{k,1}$, which is

$$N_{k,1}(t) = 1 \quad \text{for } t_k \le t \le t_{k+1}$$
$$= 0 \text{ otherwise.} \qquad \text{(CURV2-4b)}$$

[3] This tally of segments may be different if one has repeated knot values in the calculation, as you'll see from a couple of examples near the end of this document – in connection with Figures 4 and 5. It is certainly true when using the so-called standard open knot vector described shortly.

Formulas (CURV2-4a, b) are the starting point for all B-spline curve development. We will use them in this document to derive or develop some analytic results (that is, some general formulas for frequently encountered situations), but they can be programmed numerically more or less as they stand as well. The most efficient way to implement these relations numerically will depend quite a bit on the structure of the knot vector. Some brief discussions of this issue are given in Hill [*Computer Graphics,* MacMillan, 1990, page 519, exercise 14.30], and other references listed in the course outline.

The overall strategy is straightforward. First, you need to decide the order of B-spline you wish to obtain – that is, the value of m. In most applications, m = 3 or m = 4. Then, you need to develop an appropriate knot vector. Once this is done, (CURV2-4b) can be used to write down the $N_{k,1}$ functions. But, once you know the $N_{k,1}$ functions, they can be used in (CURV2-4a) to derive the $N_{k,2}$ functions. These $N_{k,2}$ functions can then be used in (CURV2-4a) to derive $N_{k,3}$ functions, and so on, until you arrive at the $N_{k,m}$ functions for the value of m originally chosen. Since m = 4 is the highest order usually used (producing cubic B-Splines), this recursion process is quite short, usually consisting of two or three stages. (Of course, two or three stages of ugly algebra is still plenty of ugly, but the end-in-view makes it a bit more tolerable perhaps.)

Formula (CURV2-2) hides a bit of the regularity of the individual segment formulas, particularly when many segments are involved. Although individual $N_{k,m}$ will be nonzero for a particular range of values of t covered by the knot vector, we can gain some simplification by redefining these functions so that all accept values of t in the range 0 to 1 (this process is called **reparameterization**). We can also rewrite (CURV2-2) in a matrix form. With these two refinements, we will arrive at some of the standard B-Spline formulas quoted in reference works.

Because Bezier curves are really special cases of B-spline curves, it is not surprising that B-spline curves possess all of the good features of Bezier curves, as well as a number of additional desirable properties. We will list these briefly after presenting some examples to make the discussion a bit more meaningful.

Additional Terminology

Before presenting some specific examples, we deal with a bit more terminology.

If the successive values of the knots increase in uniform increments (usually by 1), we obtain a **uniform B-spline**. Obviously then, if successive knot values do not all differ by the same amount, then we get a **non-uniform B-spline**. Allowing for non-uniform B-splines gives a finer control over the shape of the curve, but with some additional complexity in the working formulas.To achieve certain effects (such as mimicking circular or elliptical arcs with B-splines), it is necessary to adopt non-uniform knot vectors.

The formula (CURV2-2) is technically an example of a **non-rational B-spline**. So-called **rational B-splines** result from associating with each control point a weight value, w_k, and then replacing (CURV2-2) by the formula

$$\mathbf{p}(t) = \frac{\sum_{k=0}^{L} w_k \, \mathbf{p}_k N_{k,m}(t)}{\sum_{k=0}^{L} w_k N_{k,m}(t)} \qquad \text{(CURV2-5)}$$

The $N_{k,m}$ in this formula are still the ones determined by the Cox-de Boor relations (CURV2-4). The term “rational” in the name comes from the fact that this formula is a rational expression – it contains a numerator and an denominator. When all the weights, w_k, are 1, this formula simplifies to (CURV2-2). If you increase the value of a particular weight relative to the others, the curve will move closer to the corresponding control point, so this additional freedom gives you very fine control over the shape of the curve (at a price in complexity of course). Combining the rational form with a non-uniform knot vector gives **non-uniform rational B-splines** or **NURBs**, essentially the state-of-the-art in curve generation at present. **NURBs** are the basis of many high performance computer graphics systems,

and in some cases, have been so for many years. It is necessary to use a NURB formulation if you wish to have curves with circular or elliptical arc sections that blend smoothly into more general types of curves.

Cubic B-Splines: Standard Uniform Open Knot Vector

The heading is a mouthful, but you should be able to decode most of the words now. This is the B-spline curve which most closely mimics the open Bezier curve ("open" means that the two endpoints of the curve are distinct locations), but unlike the Bezier curve, is made up of third degree or cubic segments. The resulting curve will pass through the two endpoints, but in general, not through any of the other control points.

As mentioned earlier, the distinct numerical values in the knot vector associate in order with the control points. Furthermore, if such a value in the knot vector appears m times, then the B-spline curve must pass through the corresponding control point. So, to get a B-spline curve which interpolates the first and last control points, we just make sure that the first and last knot values repeat n times. For a uniform B-spline, all other knots increase in value by the same amount, usually taken to be 1. It turns out not to matter what you pick for the lowest knot value, so the choice is usually zero for simplicity. Thus, to get a uniform B-spline curve of order m which interpolates the first and last control points, we need a knot vector which starts out with m zeros, then in sequence 1, 2, 3, and so on, ending with m repetitions of the value L – m + 2, the number of segments in the curve.

As a specific example, we will present the example of a uniform open cubic B-spline based on 10 control points.[4] This means L = 9 and m = 4, so there will be L – m + 2 = 9 – 4 + 2 = 7 segments in this curve. The required knot vector will start with m = 4 zeros, end with m = 4 sevens, and in between have the values 1, 2, 3, 4, 5, and 6. In detail:

$$\begin{array}{lcccccccccccccc} \mathbf{T} = \{ & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 7 & 7 & 7 & \} \\ & t_0 & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} & t_{11} & t_{12} & t_{13} & \end{array}$$

(CURV2-6)

Now, we start the recursion process. The first stage is for the m = 1 functions. From (CURV2-4b) we can write:

k = 0:	$N_{0,1} = 1,$	$t = 0$ (since $t_0 \le t \le t_1$ means $0 \le t \le 0$ or $t = 0$)
1:	$N_{1,1} = 1,$	$t = 0$
2:	$N_{2,1} = 1,$	$t = 0$
3:	$N_{3,1} = 1,$	$0 \le t \le 1$ (ie. $t_3 \le t \le t_4$)
4:	$N_{4,1} = 1,$	$1 \le t \le 2$ (ie. $t_4 \le t \le t_5$)
5:	$N_{5,1} = 1,$	$2 \le t \le 3$
6:	$N_{6,1} = 1,$	$3 \le t \le 4$
7:	$N_{7,1} = 1,$	$4 \le t \le 5$
8:	$N_{8,1} = 1,$	$5 \le t \le 6$
9:	$N_{9,1} = 1,$	$6 \le t \le 7$
10:	$N_{10,1} = 1,$	$t = 7$
11:	$N_{11,1} = 1,$	$t = 7$
12:	$N_{12,1} = 1,$	$t = 7$

[4] You need only 8 control points to show all of the features of a uniform open cubic B-spline curve. However, that isn't obvious until you go beyond 8 control points and see that the formulas for the additional segments that occur are identical to formulas that have already appeared.

If we plugged these "functions" into (CURV2-2), the resulting "curve" would simply be the 10 control points – not too exciting. However, we can generate the next stage of functions, for m = 2. For this purpose, (CURV2-4a) becomes

$$N_{k,2} = \frac{t - t_k}{t_{k+1} - t_k} N_{k,1} + \frac{t_{k+2} - t}{t_{k+2} - t_{k+1}} N_{k+1,1} \qquad \text{(CURV2-7)}$$

and it will make sense to evaluate this for k = 0, 1, 2, …, 11 (since k = 12 requires a nonexistent t_{14} in the second term). The work is tedious but not difficult. First,

$$N_{0,2} = \frac{t - t_0}{t_1 - t_0} N_{0,1} + \frac{t_2 - t}{t_2 - t_1} N_{1,1} = \frac{t - 0}{0 - 0} N_{0,1} + \frac{0 - t}{0 - 0} N_{1,1} = 0$$

identically (for all values of t) because the denominators in both terms are zero.[5] You can verify that the same thing happens when we work out $N_{1,2}$. When we get to $N_{2,2}$, something new happens:

$$N_{2,2} = \frac{t - t_2}{t_3 - t_2} N_{2,1} + \frac{t_4 - t}{t_4 - t_3} N_{3,1} = \frac{t - 0}{0 - 0} N_{2,1} + \frac{1 - t}{1 - 0} N_{3,1} = (1 - t) N_{3,1}$$

The first term has been dropped because of the denominator with value zero. However, the second term has a denominator of 1, and so doesn't disappear for that reason. In fact, this indicates that $N_{2,2} = (1\text{-}t)\, N_{3,1}$. Now, from the table of $N_{k,1}$ functions above, we see that $N_{3,1} = 1$ for $0 \le t \le 1$, and so, substituting here, we can state that

$N_{2,2} = (1\text{-}t)$ $\quad 0 \le t \le 1$

Notice that the interval over which $N_{2,2}$ is defined follows from the interval over which $N_{3,1}$ has a nonzero value. Then

$$N_{3,2} = \frac{t - t_3}{t_4 - t_3} N_{3,1} + \frac{t_5 - t}{t_5 - t_4} N_{4,1} = \frac{t - 0}{1 - 0} N_{3,1} + \frac{2 - t}{2 - 1} N_{4,1} = t\, N_{3,1} + (2 - t) N_{4,1}$$

Since $N_{3,1} = 1$ for $0 \le t \le 1$, and $N_{4,1} = 1$ for $1 \le t \le 2$, we can write

$N_{3,2} = t$ $\quad$ for $0 \le t \le 1$

and

$N_{3,2} = 2 - t$ $\quad$ for $1 \le t \le 2$

This gives you an idea how the calculations are going to go. The following tables summarizes all of the m = 2 formulas:

k = 0:	$N_{0,2}$	= 0	identically
1:	$N_{1,2}$	= 0	identically
2:	$N_{2,2}$	= 1 - t	$0 \le t \le 1$
3:	$N_{3,2}$	= t	$0 \le t \le 1$

[5] This is definitely not standard mathematics where even the thought of division by zero is heresy in principle. Normally, when a formula indicates division by zero is necessary, it means some very serious error has occurred in developing the formula. In this particular case, however, terms showing up with a denominator of zero should never have been included in the first place and so we just ignore them. But, we don't ignore them because division by zero can be ignored, but because the terms which appear to give division by zero should never have been written down in the first place. So why did we write them down? Probably because it's easier to stick with the one standard formula (CURV2-4a) and weed out the offending terms as they arise than it is to come up with a correct but much more complicated recipe that will handle these cases automatically.

		= 2 - t	1 ≤ t ≤ 2
4:	$N_{4,2}$	= t - 1	1 ≤ t ≤ 2
		= 3 - t	2 ≤ t ≤ 3
5:	$N_{5,2}$	= t – 2	2 ≤ t ≤ 3
		= 4 – t	3 ≤ t ≤ 4
6:	$N_{6,2}$	= t – 3	3 ≤ t ≤ 4
		= 5 – t	4 ≤ t ≤ 5
7:	$N_{7,2}$	= t – 4	4 ≤ t ≤ 5
		= 6 – t	5 ≤ t ≤ 6
8:	$N_{8,2}$	= t – 5	5 ≤ t ≤ 6
		= 7 – t	6 ≤ t ≤ 7
9:	$N_{9,2}$	= t – 6	6 ≤ t ≤ 7
10:	$N_{10,2}$	= 0	identically
11:	$N_{11,2}$	= 0	identically

An Excel workbook called 10ptbasis.xls is included with this document, and gives plots of the basis functions in the above table and the two to follow. You can't really use these $N_{j,2}$ functions to produce a valid B-spline curve (the standard open knot vector is set up the way it is to give the $N_{k,4}$ basis functions from which we'll get the desired cubic B-spline curve – the $N_{k,2}$ and $N_{k,3}$ functions obtained here are more like intermediate work rather than useful in themselves). However, if we match the middle 10 of these $N_{k,2}$ functions up with the 10 control points, we get a polyline through all but the first and last of the control points (see the worksheet labelled "m = 2" in 10ptbasis.xls).

The working formula for the m = 3 cycle is

$$N_{k,3} = \frac{t - t_k}{t_{k+2} - t_k} N_{k,2} + \frac{t_{k+3} - t}{t_{k+3} - t_{k+1}} N_{k+1,2} \qquad \text{(CURV2-8)}$$

which will have to be applied in principle for k = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10. We will do just one calculation in detail before summarizing all of the results in a table. Take k = 4, for instance. According to (CURV2-8), we get

$$N_{4,3} = \frac{t - t_4}{t_6 - t_4} N_{4,2} + \frac{t_7 - t}{t_7 - t_5} N_{5,2} = \frac{t - 1}{2} N_{4,2} + \frac{4 - t}{2} N_{5,2}$$

Now, from the previous table, we see that $N_{4,2}$ is defined for 1 ≤ t ≤ 3 and $N_{5,2}$ is defined for 2 ≤ t ≤ 4. Thus, $N_{4,3}$ will be defined over three spans:

1 ≤ t ≤ 2: $$N_{4,3} = \frac{1}{2}(t-1) N_{4,2} = \frac{1}{2}(t-1)^2$$

2 ≤ t ≤ 3 $$N_{4,3} = \frac{1}{2}(t-1) N_{4,2} + \frac{1}{2}(4-t) N_{5,2} = \frac{1}{2}(t-1)(3-t) + \frac{1}{2}(4-t)(t-2)$$
$$= -\frac{11}{2} + 5t - t^2$$

3 ≤ t ≤ 4 $$N_{4,3} = \frac{1}{2}(4-t) N_{5,2} = \frac{1}{2}(4-t)^2$$

This represents a fair amount of work, but no particular difficulty in principle. Because many of the $N_{k,3}$ are sums of two of the $N_{k,2}$, and because many of the $N_{k,2}$ are non-zero over two adjacent spans of the curve, many of the $N_{k,3}$ will be nonzero over three adjacent spans of the curve. (This means that when we get to the m = 4 basis functions, the last step, many of them will be non-zero over four adjacent spans – just what we expect for a cubic spline formula. It is this process by which (CURV2-4a combines two lower order basis functions that is reminiscent of the de Casteljau algorithm on points which leads to Bezier curves.)

Carrying out the above procedure for each value of k that is relevant here, we come up with the following collection of m = 3 formulas:

k = 0:	$N_{0,3}$ =	= 0	identically
1:	$N_{1,3} = (1-t)\,N_{2,2}$	$= (1-t)^2$	$0 \le t \le 1$
2:	$N_{2,3} = t\,N_{2,2} + \frac{1}{2}(2-t)\,N_{3,2}$	$= \frac{1}{2}t(4-3t)$	$0 \le t \le 1$
		$= \frac{1}{2}(2-t)^2$	$1 \le t \le 2$
3:	$N_{3,3} = \frac{1}{2}t\,N_{3,2} + \frac{1}{2}(3-t)\,N_{4,2}$	$= \frac{1}{2}t^2$	$0 \le t \le 1$
		$= -\frac{3}{2} + 3t - t^2$	$1 \le t \le 2$
		$= \frac{1}{2}(3-t)^2$	$2 \le t \le 3$
4:	$N_{4,3} = \frac{1}{2}(t-1)\,N_{4,2} + \frac{1}{2}(4-t)\,N_{5,2}$	$= \frac{1}{2}(t-1)^2$	$1 \le t \le 2$
		$= -\frac{11}{2} + 5t - t^2$	$2 \le t \le 3$
		$= \frac{1}{2}(4-t)^2$	$3 \le t \le 4$
5:	$N_{5,3} = \frac{1}{2}(t-2)\,N_{5,2} + \frac{1}{2}(5-t)\,N_{6,2}$	$= \frac{1}{2}(t-2)^2$	$2 \le t \le 3$
		$= -\frac{23}{2} + 7t - t^2$	$3 \le t \le 4$
		$= \frac{1}{2}(5-t)^2$	$4 \le t \le 5$
6:	$N_{6,3} = \frac{1}{2}(t-3)\,N_{6,2} + \frac{1}{2}(6-t)\,N_{7,2}$	$= \frac{1}{2}(t-3)^2$	$3 \le t \le 4$
		$= -\frac{39}{2} + 9t - t^2$	$4 \le t \le 5$
		$= \frac{1}{2}(6-t)^2$	$5 \le t \le 6$
7:	$N_{7,3} = \frac{1}{2}(t-4)\,N_{7,2} + \frac{1}{2}(7-t)\,N_{8,2}$	$= \frac{1}{2}(t-4)^2$	$4 \le t \le 5$
		$= -\frac{59}{2} + 11t - t^2$	$5 \le t \le 6$
		$= \frac{1}{2}(7-t)^2$	$6 \le t \le 7$
8:	$N_{8,3} = \frac{1}{2}(t-5)\,N_{8,2} + \frac{1}{2}(7-t)\,N_{9,2}$	$= \frac{1}{2}(t-5)^2$	$5 \le t \le 6$

		$= -\frac{119}{2} + 19t - \frac{3}{2}t^2$	6 ≤ t ≤ 7
9:	$N_{9,3} = \frac{1}{2}(t-6)\,N_{9,2}$	$= (t-6)^2$	6 ≤ t ≤ 7
10:	$N_{10,3} = 0$		identically

Finally, we are ready for the m = 4 formulas, the goal of this first stage of the process. The prototype formula for this stage is

$$N_{k,4} = \frac{t - t_k}{t_{k+3} - t_k} N_{k,3} + \frac{t_{k+4} - t}{t_{k+4} - t_{k+1}} N_{k+1,3}, \qquad k = 0, 1, 2, 3, 4, 5, 6, 7, 8, \text{and } 9.$$

We won't take the space to work out even one of the functions in detail, since the procedure is the same as used in the earlier stages – all that changes is that the algebra is much more tedious. A software application such as *Mathematica* or *Maple* is very useful in checking the accuracy of this sort of work. The results are:

k = 0:	$N_{0,4} = (1-t)\,N_{1,3}$	$= (1-t)^3$	0 ≤ t ≤ 1
1:	$N_{1,4} = t\,N_{1,3} + \frac{2-t}{2}N_{2,3}$	$= \frac{t(12-18t+7t^2)}{4}$	0 ≤ t ≤ 1
		$= \frac{1}{4}(2-t)^3$	1 ≤ t ≤ 2
2:	$N_{2,4} = \frac{t}{2}N_{2,3} + \frac{3-t}{3}N_{3,3}$	$= \frac{t^2(18-11t)}{12}$	0 ≤ t ≤ 1
		$= \frac{-18+54t-36t^2+7t^3}{12}$	1 ≤ t ≤ 2
		$= \frac{1}{6}(3-t)^3$	2 ≤ t ≤ 3
3:	$N_{3,4} = \frac{t}{3}N_{3,3} + \frac{4-t}{3}N_{4,3}$	$= \frac{1}{6}t^3$	0 ≤ t ≤ 1
		$= \frac{2}{3} - 2t + 2t^2 - \frac{t^3}{2}$	1 ≤ t ≤ 2
		$= -\frac{22}{3} + 10t - 4t^2 + \frac{t^3}{2}$	2 ≤ t ≤ 3
		$= \frac{1}{6}(4-t)^3$	3 ≤ t ≤ 4
4:	$N_{4,4} = \frac{t-1}{3}N_{4,3} + \frac{5-t}{3}N_{5,3}$	$= \frac{1}{6}(t-1)^3$	1 ≤ t ≤ 2
		$= \frac{31-45t+21t^2-3t^3}{6}$	2 ≤ t ≤ 3
		$= \frac{-131+117t-33t^2+3t^3}{6}$	3 ≤ t ≤ 4
		$= \frac{1}{6}(5-t)^3$	4 ≤ t ≤ 5
5:	$N_{5,4} = \frac{t-2}{3}N_{5,3} + \frac{6-t}{3}N_{6,3}$	$= \frac{1}{6}(t-2)^3$	2 ≤ t ≤ 3

		$= \frac{50}{3} - 16t + 5t^2 - \frac{t^3}{2}$	$3 \le t \le 4$
		$= -\frac{142}{3} + 32t - 7t^2 + \frac{t^3}{2}$	$4 \le t \le 5$
		$= \frac{1}{6}(6-t)^3$	$5 \le t \le 6$
6:	$N_{6,4} = \frac{t-3}{3} N_{6,3} + \frac{7-t}{3} N_{7,3}$	$= \frac{1}{6}(t-3)^3$	$3 \le t \le 4$
		$= \frac{229 - 165t + 39t^2 - 3t^3}{6}$	$4 \le t \le 5$
		$= \frac{-521 + 285t - 51t^2 + 3t^3}{6}$	$5 \le t \le 6$
		$= \frac{1}{6}(7-t)^3$	$6 \le t \le 7$
7:	$N_{7,4} = \frac{t-4}{3} N_{7,3} + \frac{7-t}{2} N_{8,3}$	$= \frac{1}{6}(t-4)^3$	$4 \le t \le 5$
		$= \frac{997 - 579t + 111t^2 - 7t^3}{12}$	$5 \le t \le 6$
		$= \frac{(t-7)^2 (-59 + 11t)}{12}$	$6 \le t \le 7$
8:	$N_{8,4} = \frac{t-5}{2} N_{8,3} + (7-t) N_{9,3}$	$= \frac{1}{4}(t-5)^3$	$5 \le t \le 6$
		$= \frac{1603 - 789t + 129t^2 - 7t^3}{4}$	$6 \le t \le 7$
9:	$N_{9,4} = (t-6) N_{9,3}$	$= (t-6)^3$	$6 \le t \le 7$

(CURV2-9)

This looks like an enormous amount of work (which it is, I guess). However, it really involves just ordinary dreary algebraic substitution and simplification of basic polynomial expressions. Again, in obtaining the formulas above, and the ones to follow, a good symbolic mathematics application is very helpful.

Anyway, the formulas in this last table could be used in (CURV2-2) to generate the open cubic B-spline curve for a 10 control point problem. However, if one had an 11-point or even an 1100-point problem, more work would have to be done if we stopped with this formulation. Instead, it's helpful to plug these formulas into (CURV2-2), and then via simple algebraic substitution, obtain a new set of functions which will give the same values as the ones in the table above, but with the parameter values always ranging between 0 and 1, rather than the seven different intervals of values found in the tabulation above. We're still in algebraic-tedium land, folks, but there's a payoff at the end.

A sketch will help organize our work here. From the table above, we notice that $N_{0,4}$ is nonzero only for $0 \le t \le 1$, $N_{1,4}$ only for $0 \le t \le 2$, $N_{2,4}$ only for $0 \le t \le 3$, and so on. Since $N_{k,4}$ is multiplied by $\mathbf{p}_k$ in (CURV2-2), we can diagram the contributions to the overall B-spline curve in each interval of t-values present in the table above in the following way:

Figure 2

Each of the columns in Figure 2 represents one segment of our seven segment curve. We deal with each individually.

0 ≤ t ≤ 1:

Here, the parameter range is already zero to one, so no reparameterization is necessary (though we'll change the symbol for the parameter to u from t to get formulas consistent with what follows). On this interval, we have:

$$\mathbf{p}(t) = \mathbf{p}_0 N_{0,4} + \mathbf{p}_1 N_{1,4} + \mathbf{p}_2 N_{2,4} + \mathbf{p}_3 N_{3,4} \qquad \text{(CURV2-10)}$$

where, on the interval $0 \le t \le 1$, the basis functions have the formulas (after substituting u for t in the formulas from rows labelled $0 \le t \le 1$ in (CURV2-9) and building in a common denominator of 12 for all formulas in the set)

$$\begin{aligned} N_{0,4} &= (1-u)^3 = \frac{1}{12}(12 - 36u + 36u^2 - 12u^3) \\ N_{1,4} &= \frac{1}{12}(36u - 54u^2 + 21u^3) \\ N_{2,4} &= \frac{1}{12}(18u^2 - 11u^3) \\ N_{3,4} &= \frac{1}{12}(2u^3) \end{aligned} \qquad \text{(CURV2-11)}$$

Substituting (CURV2-11) into (CURV2-10) then gives the formula for the first segment of our B-spline curve. If you write the result out in full detail, it will be easy to verify that it is equivalent to the following matrix formula:

$$\mathbf{p}(t) = \frac{1}{12}\begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix}\begin{bmatrix} 12 & 0 & 0 & 0 \\ -36 & 36 & 0 & 0 \\ 36 & -54 & 18 & 0 \\ -12 & 21 & -11 & 2 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} \qquad \text{(CURV2-12)}$$

Remember, the bolded symbols really stand for a row of entries, the homogeneous coordinates of a point. What we are in the process of demonstrating here is that the formula for points along segment number j (now counting from j = 0) of the cubic B-spline curve can be written in the matrix form

$$\mathbf{p}_j(t) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \mathbf{N} \cdot \begin{bmatrix} \mathbf{p}_j \\ \mathbf{p}_{j+1} \\ \mathbf{p}_{j+2} \\ \mathbf{p}_{j+3} \end{bmatrix} \qquad \text{(CURV2-13)}$$

with the entire segment generated when u varies from zero to one. All we need to do then to generate the entire curve is to determine the elements of the 4 x 4 matrix **N** in each case. For the first segment (corresponding to j = 0 in formula (CURV2-13)), we've now found that

$$\mathbf{N} = \frac{1}{12} \begin{bmatrix} 12 & 0 & 0 & 0 \\ -36 & 36 & 0 & 0 \\ 36 & -54 & 18 & 0 \\ -12 & 21 & -11 & 2 \end{bmatrix} \qquad \text{(CURV2-14)}$$

<u>1 ≤ t ≤ 2</u> (the second segment):

Since t varies between 1 and 2 for this segment, we will get u to vary between zero and one if we make the following substitution:

u = t – 1 (which means: t = u + 1)

Then, referrring to Figure 2, we see that for this segment,

$\mathbf{p}(t) = \mathbf{p}_1 N_{1,4} + \mathbf{p}_2 N_{2,4} + \mathbf{p}_3 N_{3,4} + \mathbf{p}_4 N_{4,4}$

where, on the interval 1 ≤ t ≤ 2, we get the formulas for the $N_{k,4}$ to be

$$N_{1,4} = \frac{1}{4}(2-t)^3 = \frac{1}{4}[\,2 - (u+1)\,]^3 = \frac{1}{4}(1-u)^3 = \frac{1}{12}(3 - 9u + 9u^2 - 3u^3)$$

$$N_{2,4} = \frac{-18 + 54(u+1) - 36(u+1)^2 + 7(u+1)^3}{12} = \frac{1}{12}(7 + 3u - 15u^2 + 7u^3)$$

$$N_{3,4} = \frac{2}{3} - 2(u+1) + 2(u+1)^2 - \frac{(u+1)^3}{2} = \frac{1}{12}(2 + 6u + 6u^2 - 6u^3)$$

$$N_{4,4} = \frac{1}{6}(u+1-1)^3 = \frac{1}{12}(2u^3)$$

Substituting these formulas into the equation for **p**(t) for this segment gives

$$\mathbf{p}(t) = \frac{1}{12} \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} 3 & 7 & 2 & 0 \\ -9 & 3 & 6 & 0 \\ 9 & -15 & 6 & 0 \\ -3 & 7 & -6 & 2 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \mathbf{p}_4 \end{bmatrix}$$

so the basis matrix for this segment is

$$\mathbf{N} = \frac{1}{12}\begin{bmatrix} 3 & 7 & 2 & 0 \\ -9 & 3 & 6 & 0 \\ 9 & -15 & 6 & 0 \\ -3 & 7 & -6 & 2 \end{bmatrix} \qquad \text{(CURV2-15)}$$

We proceed in the same way for the remaining five segments of this particular curve. The results for all four seven segments are summarized in the following table:

$0 \le t \le 1$	$\mathbf{N}_0 = \frac{1}{12}\begin{bmatrix} 12 & 0 & 0 & 0 \\ -36 & 36 & 0 & 0 \\ 36 & -54 & 18 & 0 \\ -12 & 21 & -11 & 2 \end{bmatrix}$	$4 \le t \le 5$ (t = u + 4)	$\mathbf{N}_G = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$
$1 \le t \le 2$ (t = u + 1)	$\mathbf{N}_1 = \frac{1}{12}\begin{bmatrix} 3 & 7 & 2 & 0 \\ -9 & 3 & 6 & 0 \\ 9 & -15 & 6 & 0 \\ -3 & 7 & -6 & 2 \end{bmatrix}$	$5 \le t \le 6$ (t = u + 5)	$\mathbf{N}_{L-1} = \frac{1}{12}\begin{bmatrix} 2 & 8 & 2 & 0 \\ -6 & 0 & 6 & 0 \\ 6 & -12 & 6 & 0 \\ -2 & 6 & -7 & 3 \end{bmatrix}$
$2 \le t \le 3$ (t = u + 2)	$\mathbf{N}_G = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$	$6 \le t \le 7$ (t = u + 6)	$\mathbf{N}_L = \frac{1}{12}\begin{bmatrix} 2 & 7 & 3 & 0 \\ -6 & -3 & 9 & 0 \\ 6 & -15 & 9 & 0 \\ -2 & 11 & -21 & 12 \end{bmatrix}$
$3 \le t \le 4$ (t = u + 3)	$\mathbf{N}_G = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$		

(CURV2-16)

The most important observation from this table is that while the first two segments and the last two segments of the curve appear to require their own unique **N** matrices, all other internal segments have the same **N** matrix (labelled N_G for "generic") once the parameters have been adjusted so that their ranges of values are always zero to one for any given curve segment. Since a cubic curve requires four points (and hence covers three segments), this is more or less what you'd expect.

So, the information in the table (CURV2-16) is enough to construct any cubic B-spline curve that consists of five segments or more. All you do is use $\mathbf{N}_0$ and $\mathbf{N}_1$ to produce the first two segments, then use $\mathbf{N}_G$ to produce all the remaining segments except for the last two. Use $\mathbf{N}_{L-1}$ and $\mathbf{N}_L$, respectively, to produce the last two segments of the curve. For each segment, use formula (CURV2-13) with u varying from zero to one.

The Excel workbook, Cubbsp8.xls, included with this document, implements the formulas above for an example involving 8 control points.

Periodic Cubic B-Splines

Since the segment formula using $\mathbf{N}_G$ is the one that is repeated to produce most of a cubic B-spline curve when many segments are involved, this part of the curve is sometimes called a periodic cubic B-spline curve. Of course, one is free to use $\mathbf{N}_G$ for every segment in the cubic B-spline curve, but the resulting curve will not interpolate the end points (which may be quite all right in some circumstances).

More often, the periodic cubic B-spline formula is used to construct closed loop curves. In such a case, the loop would be generated by a sequence of control points, and by repeating the first three points a second time at the end, the sequence of L segments will close itself smoothly.

The Excel workbook, percbs.xls, implements this idea to produce a closed loop curve based on 5 control points.

Quadratic B-splines: Standard Open Knot Vector

It turns out that the smallest number of control points for which the open knot vector quadratic B-spline shows the "generic" segment is the 5-control point case (L = 4, m = 3). This curve has 3 segments, the middle one of which is the periodic segment. Since a quadratic curve (highest power of the parameter is the second) requires three points for determination, we would expect that only the single segment at each end of such a composite curve would have a distinct basis matrix.

For a quadratic B-spline, the standard open knot vector has the first and last knot values repeated three times. Thus, for the 5-control point case, the standard open knot vector is

$$\begin{array}{lcccccccccc} \mathbf{T} = \{ & 0, & 0, & 0, & 1, & 2, & 3, & 3, & 3 & \} \\ & t_0 & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & \end{array}$$

Going through the procedure illustrated earlier in some detail for the 10-control point cubic B-spline case, we end up with the following basis matrices (which are now 3 x 3, since each segment of the curve depends on the coordinates of only three points).

initial segment: $$\mathbf{N}_0 = \frac{1}{2}\begin{bmatrix} 2 & 0 & 0 \\ -4 & 4 & 0 \\ 2 & -3 & 1 \end{bmatrix}$$

periodic segment: $$\mathbf{N}_G = \frac{1}{2}\begin{bmatrix} 1 & 1 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix}$$

terminating segment: $$\mathbf{N}_L = \frac{1}{2}\begin{bmatrix} 1 & 1 & 0 \\ -2 & 2 & 0 \\ 1 & -3 & 1 \end{bmatrix}$$

The Excel workbook, qudbsp8.xls, implements these formulas to produce a quadratic B-spline curve based on the same 8 control points used early to illustrate the cubic B-spline formulas.

The "Shortest" B-splines

Actually, we're not referring to physical length here, but to the B-spline curves specified by the fewest possible control points. Since it takes k points to determine a curve of order k, we can never have a quadratic B-spline curve determined by fewer than 3 control points, a cubic B-spline curve determined by fewer than four points, and so on. In cases of the minimum number of control points, L = m - 1, and so, the number of segments is L – m + 2 = 1.

The 3-point quadratic B-spline will thus have the standard open knot vector

$$\mathbf{T} = \{ \begin{array}{cccccccc} 0 & 0 & 0 & 1 & 1 & 1 & \} \\ t_0 & t_1 & t_2 & t_3 & t_4 & t_5 \end{array}$$

In a very few steps, you can confirm that the equation of this single segment is

$$\mathbf{p}(t) = N_{0,3}\,\mathbf{p}_0 + N_{1,3}\,\mathbf{p}_1 + N_{2,3}\,\mathbf{p}_2$$

$$= (1-t)^2\,\mathbf{p}_0 + 2t(1-t)\,\mathbf{p}_1 + t^2\,\mathbf{p}_2 \qquad 0 \le t \le 1$$

or, in matrix form

$$\mathbf{p}(t) = \begin{bmatrix} 1 & t & t^2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} \qquad 0 \le t \le 1$$

This is just the Bezier curve resulting from three control points.

Similarly, when a cubic B-spline curve is created using just four control points, we need to use the knot vector given by

$$\mathbf{T} = \{ \begin{array}{cccccccccc} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \} \\ t_0 & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \end{array}$$

and the resulting one segment curve has the algebraic formula

$$\mathbf{p}(t) = N_{0,4}\,\mathbf{p}_0 + N_{1,4}\,\mathbf{p}_1 + N_{2,4}\,\mathbf{p}_2 + N_{3,4}\,\mathbf{p}_3$$

$$= (1-t)^3\,\mathbf{p}_0 + 3t(1-t)^2\,\mathbf{p}_1 + 3t^2\,(1\text{-}t)\,\mathbf{p}_2 + t^3\,\mathbf{p}_3 \qquad 0 \le t \le 1$$

or, in matrix form

$$\mathbf{p}(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} \qquad 0 \le t \le 1$$

This is easily recognized as the Bezier curve that results from four control points. In fact, in general, the one segment curve obtained by creating B-spline curve of order m using m control points and the standard open knot vector is the Bezier curve of order m. Thus, Bezier curves are a special case of a B-spline curve.

Other Non-Periodic Quadratic and Cubic B-Spline Cases

B-spline curves based on the minimal number or nearly the minimal number of required control points have a somewhat limited applicability because often even small curved figures need to use some "extra" control points to "line up" appropriately with other elements in the picture (see the discussion of B-spline properties in the next section). Nevertheless, for completeness sake, we just list formulas for four situations involving standard open knot vectors and quadratic or cubic B-splines that haven't been covered so far.

According to our previous discussion, quadratic B-spline curves generated by three control points is the 3-point Bezier curve, and quadratric B-spline curves generated by five or more control points

contain the generic or periodic basis matrix when all segments formulas are reparameterized to the interval [0, 1]. The only standard open knot vector quadratic B-spline curve not covered so far, then, is the one based on just four control points. This curve will have two segments, and using the methods illustrated above, we determine that the equations for these two segments in matrix form are:

$$\mathbf{p}_{first}(t) = \begin{bmatrix} 1 & t & t^2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -2 & 2 & 0 \\ 1 & -3/2 & 1/2 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} \qquad 0 \le t \le 1$$

and

$$\mathbf{p}_{last}(t) = \begin{bmatrix} 1 & t & t^2 \end{bmatrix} \begin{bmatrix} 1/2 & 1/2 & 0 \\ -1 & 1 & 0 \\ 1/2 & -3/2 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} \qquad 0 \le t \le 1$$

For cubic B-splines, the 4-control point case gives the cubic Bezier curve, and the 8-control point case already contains the periodic segment (the middle segment of the resulting five segment curve). This leaves the 5-, 6-, and 7- segment cases undetermined.

The standard open knot vector cubic B-spline curve determined by five control points has two segments, give by

$$\mathbf{p}_{first}(t) = \frac{1}{12} \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 12 & 0 & 0 & 0 \\ -36 & 36 & 0 & 0 \\ 36 & -54 & 18 & 0 \\ -12 & 21 & -12 & 3 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

and

$$\mathbf{p}_{last}(t) = \frac{1}{12} \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 3 & 6 & 3 & 0 \\ -9 & 0 & 9 & 0 \\ 9 & -18 & 9 & 0 \\ -3 & 12 & -21 & 12 \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \mathbf{p}_4 \end{bmatrix}$$

both defined for $0 \le t \le 1$. It's curious that the first two columns of the $\mathbf{p}_{first}$ basis matrix here is the same as the first two columns of the basis matrix for the first segment of the general open knot cubic B-spline curve for eight or more control points. Similarly, the last two columns of the basis matrix for $\mathbf{p}_{last}$ is the same as the last two columns of the final segment basis matrix in the more general case.

The six-point standard open knot vector cubic B-spline curve will have three segments, with basis matrix times geometry matrix given, respectively, by

$$\frac{1}{12} \begin{bmatrix} 12 & 0 & 0 & 0 \\ -36 & 36 & 0 & 0 \\ 36 & -54 & 18 & 0 \\ -12 & 21 & -11 & 2 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}, \qquad \frac{1}{12} \begin{bmatrix} 3 & 7 & 2 & 0 \\ -9 & 3 & 6 & 0 \\ 9 & -15 & 6 & 0 \\ -3 & 7 & -7 & 3 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \mathbf{p}_4 \end{bmatrix},$$

and

$$\frac{1}{12} \begin{bmatrix} 2 & 7 & 3 & 0 \\ -6 & -3 & 9 & 0 \\ 6 & -15 & 9 & 0 \\ -2 & 11 & -21 & 12 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_2 \\ \mathbf{p}_3 \\ \mathbf{p}_4 \\ \mathbf{p}_5 \end{bmatrix}$$

Comparison of these with previous formulas shows that the first and last are identical with the first and last segment formulas for the eight or more control point cases. The middle segment formula here is an interesting mix of the second and second-last segment formulas for the more general case.

The seven-control point cubic B-spline curve will have four segments. After seeing the way the formulas above correspond to the situation with eight or more control points, it is not surprising that the four segments of the seven-point cubic B-spline are given by the first two and the last two segment formulas in (CURV2-16) – that is, the seven point case is identical to the eight or more point cases, except that there is no generic segment(s) in the middle of the curve.

Properties of B-Spline Curves

Many of the mathematical properties of B-spline curves follow from the fact that they are formed by piecing together segments of order m in a fashion that makes them generalizations of Bezier curves (as noted above).

Each segment of an m^{th}-order B-spline curve is confined to the convex hull of the m control points on which it depends. This can be illustrated by the following 7-point example, which leads to a four segment cubic B-spline curve when the standard open knot vector is used:

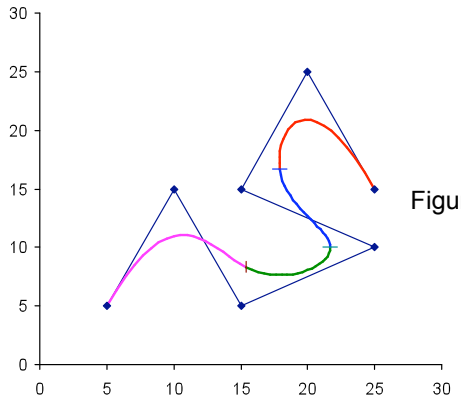


Figure 3

The individual segments are indicated by the small cross marks in the figure. The initial segment (segment #0 in our counting scheme) must be contained within the convex hull of the first four control points (counting from the lower left). The next segment is contained within the convex hull of the second through fifth control points. The next segment is contained within the convex hull of the third through sixth control point, and finally the last segment is contained within the convex hull of the last four control points. In pictures:

Segment #0

Segment #1

Segment #2

Segment #3

This property has a number of useful implications. For example, if m points fall along a straight line, then the resulting m^{th}-order B-spline curve segment will also be a straight line (since the convex hull of points along a straight line will be a straight line "region"). This provides a way to smoothly join curves and straight line segments in a figure.

Further, if a control point is repeated m times, then the convex hull of those m points is just the point itself. So, one way to force a B-spline curve of order m through a specific point is to repeat that control point m times. Repeating a control point more than once, but fewer than m times will tend to pull the curve closer to that location as well. This follows from the same property that repeating a control point tends to reduce the extent of the convex hull of the set of points in that vicinity.

The shape of the B-spline curve can also be affected by varying the values in the knot vector – in particular, by using repeated knot values. This sometimes has similar effects to repeating control points, but in general is a less intuitive (and less automatic) way of going about refining a curve. This is getting into the area of non-uniform B-splines, which is a bit beyond the scope of the present course. However, just to give an indication of the sort of thing that can be done, we present a couple of examples without a great deal of discussion.

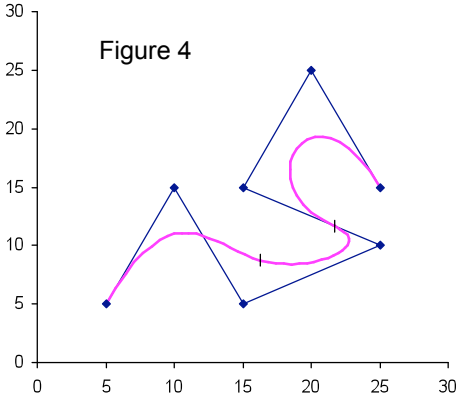
The curve in Figure 3 is a cubic B-spline based on seven control points and using the standard open knot vector

$$\mathbf{T} = \{0, 0, 0, 0, 1, 2, 3, 4, 4, 4, 4\}$$

If we change the knot value 3 to a 2, so that the third knot value is repeated,

$$\mathbf{T} = \{0, 0, 0, 0, 1, 2, 2, 4, 4, 4, 4\},$$

one of the immediate consequences is that the overall curve will consist of just three segments (since there are now only three gaps in knot values). It will still interpolate the first and last control point because of the m-fold repeated knot values at the two extremes.



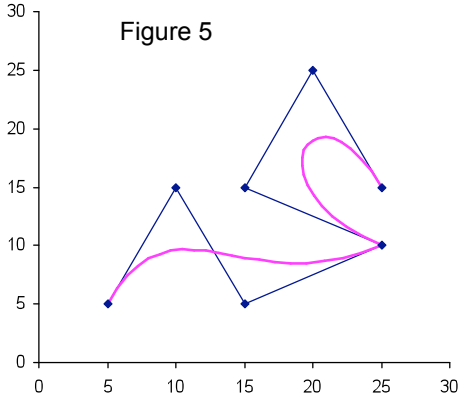


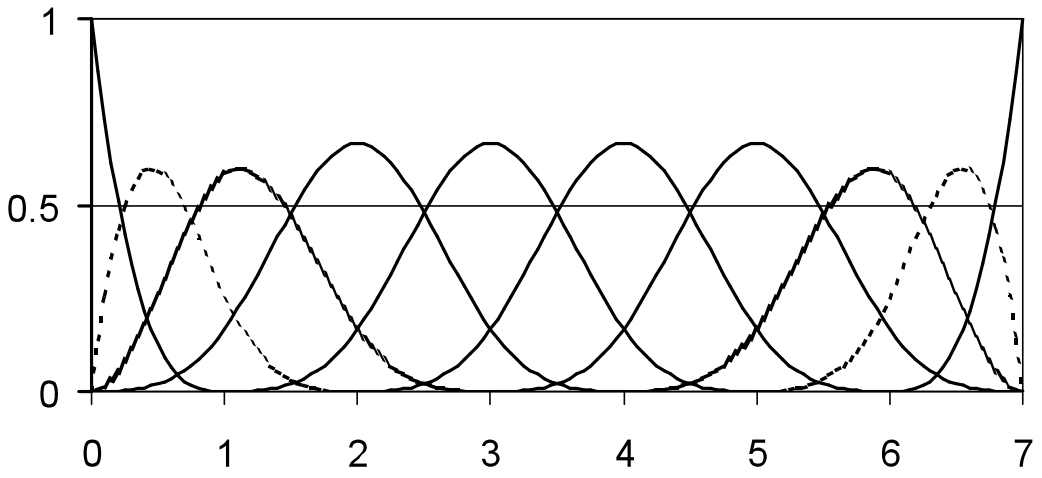
Figure 4 above right shows the resulting curve, with small cross ticks to show the segment joints. When you compare this with Figure 3, you see that the curve is pulled closer to control point number 3 (numbering from zero as the first one), and to some extent away from points 2, 4, and 5.

An even more drastic deformation of the curve in Figure 3 shows up in Figure 5 above, which is based on only three distinct knot values, and thus two segments. The knot vector used for Figure 5 is

T = {0, 0, 0, 0, 2, 2, 2, 4, 4, 4, 4},

The effect of the three-fold knot value at 2 is to reduce the overall curve to two quadratic segments which do not join smoothly at the middle point of the seven. This is probably not too useful an effect in most applications.

One final property needs some mention here. For this we refer back to the 10-control point, seven segment B-spline curve formulas worked out in some detail earlier, and in particular, the blending functions, $N_{k,m}$, given in table (CURV2-9). If these nine functions are plotted as a function of t, the following figure results:



The horizontal axis is t, and in order from left to right, the curves encountered are the graphs of $N_{0,4}$, $N_{1,4}$, $N_{2,4}$, …, $N_{9,4}$. You can see a fair degree of repetition and symmetry here. In particular, the four solid-line curves in the center ($N_{3,4}$, $N_{4,4}$, $N_{5,4}$, and $N_{6,4}$) are identical curves, just shifted rightwards by one unit in t in each case. In fact, if you take any of the unit intervals in t between 2 and 5, you can see that each contains the same four blending curve segments. This is why the reparameterized blending function formulas are the same for each of these "generic" intervals (and also why the two end segments of the cubic B-spline curve are different from the interior generic or periodic segments). While it is not absolutely obvious from the figure above, it is also possible to show that if you sum the graphs above any value of t, the blending functions sum to 1, which is a property these functions must have if they are to be true blending functions. Finally, all but $N_{04,}$ are zero at t = 0, and all but $N_{9,4}$ are zero at t = 7, which is why the B-spline curve passes through the first and last points.

The regularity seen in the above figure can be exploited in applications involving standard open knot vector B-spline curves with many periodic segments. You can set up a table of values for, say, $N_{4,4}$ for a sequence of values of t, and then not need to calculate any of the other periodic $N_{k,m}$ values, since they will just repeat those in the table for $N_{4,4}$.

NURBs

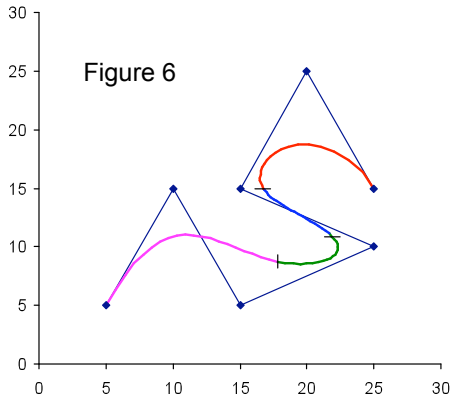
We finish off this document for the time being with a brief introduction to the most powerful and flexible of all B-Spline formulations, the so-called non-uniform rational B-splines or NURBs. In fact, most of the remarks here have to do with the "rational" part, leaving the consequences of adopting non-uniform knot vectors for follow-up in the technical literature.

The rational B-spline curve has already been defined and described briefly earlier in this document in formula (CURV2-5):

$$\mathbf{p}(t) = \frac{\sum_{k=0}^{L} w_k \, \mathbf{p}_k N_{k,m}(t)}{\sum_{k=0}^{L} w_k N_{k,m}(t)} \qquad \text{(CURV2-5)}$$

There are two ways to interpret this formula. The first is to consider the w_k's as weights associated with each control point. Those control points having relatively heavier weights tend to attract the B-spline curve closer to themselves, compared to those points with relatively lighter weights.

However, one could also begin by considering the control points to be given with homogeneous coordinates (x_k, y_k, z_k, w_k). Then, (CURV2-5) just represents the operation of restoring homogeneity to points generated by the usual non-rational B-spline formula, (CURV2-2). While this interpretation is useful in understanding some of the properties of rational B-splines, it also suggests an efficient method for implementing (CURV2-5) computationally.



We've formulated the computation of coordinates of points along various B-spline curves as matrix products earlier in this document. Formula (CURV2-13) is a general example. In those cases, the rows $\mathbf{p}_j$ of the rightmost matrix were view as the (x, y, z) coordinates of the control points in ordinary three-dimensional space. However, those formulas remain valid if the rows of the coordinate matrix are written as (x, y, z, w) coordinates of the points, incorporating the fourth homogeneous coordinate. The result will be (x, y, z, w) coordinates of points along the curve. To get (CURV2-5), you would simply divide each of x, y, and z by w on the left-hand side of (CURV2-13).

As a simple example of the sort of effect one can get, Figure 6 to the right is a B-spline curve based on the same seven controls points as in Figure 3, but with the fourth point given a weight of 3, and the fifth point a weight of 4 (compared to weights of 1 for all other points here, and for the seven points in Figure 3). You can see that the curve is pulled much more strongly towards control points three and four, giving it a much sharper swing back and forth than is visible in Figure 3. Once again, the small cross ticks mark the boundaries between the four segments forming the overall curve. Of course, the effects of non-unit weights are only seen in those segments which depend on the points with the non-unit weights. This means that one can effect fairly large changes to the shape of small regions of the overall curve without causing any changes in the shape of segments which don't depend on the specific control points for which non-unit weights are being imposed.

The Excel workbook, cub7pt.xls, contains the setup to produce Figures 3 and 6.